Python

Πτυχιακή εργασία:
Ορφέα Βουτσαρίδη (tl 3847)

Εισαγωγή

“A Byte of Python” είναι ένα βιβλίο προγραμματισμού για την γλώσσα Python.
Βοηθάει σαν οδηγός στην γλώσσα Python για τους αρχάριους προγραμματιστές. Αν το μονό που γνωρίζεις είναι να σώζεις αρχεία, τότε αυτό είναι το βιβλίο για σένα.

Ποιος διαβάζει το “A Base of Python”?
Εδώ είναι κάποια λόγια από ανθρώπους για το βιβλίο:
Απάντησής από αναγνώστες:

Είναι το καλύτερο βοήθημα για αρχάριους!Ευχαριστούμε για την προσπάθεια.
-Walt Michalik

Φτιάξατε το καλύτερο βοήθημα για την Python. Πολύ καλή δουλειά. Ευχαριστώ!
-Joshua Robin

Γεια σας, είμαι από την Dominican Republic. Το όνομα μου είναι Pavel, πρόσφατα διάβασα το βιβλίο 'A Byte of Python' και το βρίσκω υπέροχο. Έμαθα πολλά από τα παραδείγματα. Το βιβλίο σου είναι μεγάλη βοήθεια για αρχάριους σαν και έμενα.!
-Pavel Simo

Πρόσφατα τελείωσα το το βιβλίο “Byte of Python” και ένοιωθα ότι έπρεπε να σε ευχαριστήσω.
-Samuel Young

Αγαπώ το βιβλίο σου! είναι το καλύτερο βοήθημα για την Python και πολύ χρήσιμες αναφορές ένα πραγματικό αριστούργημα!Συνεχίστε την καλή δουλειά.
Για άλλη μια φορά, ευχαριστώ πάρα πολύ για μια τόσο καλά δομημένη και εύχρηστο οδηγό για την βασική γλωσσά προγραμματισμού στον ηλεκτρονικό ιστό. Με έσπρωξε μέσα και έξω στην OOP ενώ δυο άλλα βιβλία αποτύγχαναν.
-Matt Gallivan

Θα ήθελα να σε ευχαριστήσω για το βιβλίο σου 'A byte of Python' το οποίο το βρίσκω τον καλύτερο τρόπο για να μάθω την Python. Είμαι 15 χρονών και ζω στην Αιγύπτο. Το όνομα μου είναι Ahmed. Η Python είναι η δεύτερη γλωσσά που έμαθα,έμαθα visual basic 6 στο σχολείο άλλα δεν το ευχαριστήθηκα ,ενώ πραγματικά ευχαριστήθηκα την Pyhton. Έφτιαξα το βιβλίο διευθύνσεων και ήταν επιτυχία. Θα ήθελα να ξεκινήσω να φτιάχνω και άλλα προγράμματα και να διαβάσω προγράμματα Python. Θα ξεκινήσω να μαθαίνω Java και και θα ήθελα να μου πείτε που μπορώ να βρω το βοηθήματα καλά σαν τα δικά σας για Java. Αυτό θα με βοηθούσε πολύ. Ευχαριστώ.
-Ahmed Mohammed

Μια πολύ καλή πηγή για αρχάριους που θέλουν να μάθουν για την Python είναι το 110-σέλιδο PDF βοήθημα “A Byte dy Python” από τον Swaroop C H. Είναι ένα καλά γραμμένο βοήθημα. Εύκολο να το καταλάβεις και ίσως είναι η καλύτερη εισαγωγή στην Python γλωσσά προγραμματισμού που υπάρχει.
-Drew Ames

Χθες πέρασα το Byte of Python στο κινητό μου και είναι η ευκολότερη και πιο συνοπτική εισαγωγή στην Pyhton που έχω δει. Το συνιστώ για ξεκίνημα στην γλωσσά Python.
-Jason Delport

Ακαδημαϊκά μαθήματα

Αυτό το βιβλίο αρχίζει να χρησιμοποιείται σαν διδακτικό υλικό σε διάφορα εκπαιδευτικά ιδρύματα

1. 'Principles of programming languages' μαθημα στο Vriji Universiteit,Asterdam
2. 'Basic Concepts of Computing' μαθημα στο University of California,Davis
3. 'Progrmming With Pyhton' μαθημα στο Harvard University
4. 'Introduction to programming' μαθημα στο University of Leeds
5. 'Introduction to Application programming' μαθημα στο Bonston University
6. 'Information Technology Skills for Meteorology' μαθημα στο University of Oklahoma
7. 'Geoprocessin' μαθημα στο Michigan State University
8. 'Multi Agent Semantic Web System' μαθημα στο University of Edinburgh

Ακόμα και η NASA

Το βιβλίο χρησιμοποιείτε ακόμα και στην NASA! Άρχισε να χρησιμοποιείται στο εργαστήριο για την αεριωθούμενη προώθηση μαζί με το πρόγραμμα δικτύου στο διάστημα.

Επίσημη σύσταση

Αυτό το βιβλίο είναι στην λίστα της επίσημης σελίδας στο Python, στα ολοκληρωμένα βοηθήματα, διπλά στα επίσημα έγγραφα.

Άδειες

Διάβασε τώρα

Μπορείς να διαβάσεις το αυτό το βιβλίο στο ίντερνετ στην διεύθυνση Python_en:Table of contents

Αγόρασε το βιβλίο

Το αντίγραφο του βιβλίου μπορεί να αγοραστεί για την απολαύσει σας και να υποστηρίξετε την ανάπτυξη και βελτιώσει του βιβλίου.

Download

- PDF (631KB)
- Mediawiki XML dump (276KB) (για προχωρημένους χρήστες)

Αν ενδιαφέρεστε να στηρίξετε την συνεχεί ανάπτυξη του βιβλίου, παρακαλώ σκεφτείτε να κάνετε μια δωρεά ή αγοράσετε ένα εκτυπωμένο αντίγραφο.

Μετάφραση

Αν ενδιαφέρεστε να διαβάσετε ή να συνεισφέρεται σε μια μετάφραση αυτού του βιβλίου σε άλλες γλώσσες, παρακαλώ δείτε τις μεταφράσεις παρακάτω.

Αναφορές

1. http://www.ibiblio.org/swaroopch/byteeofpython/read/
2. http://www.ibiblio.org/swaroopch/byteofpython/files/120/byteofpython_120.pdf
3. http://www.swaroopch/com/files/byteofpython/byte_of_python_v191.pdf
4. http://www.swaroopch.com/buybook
5. http://www.ibiblio.org/swaroopch/byteofpython/files/120/
6. http://www.b-list.org/weblog/2008/dec/05/python-3000/
7. http://www.linux.com/feature/126522
8. http://www.paxmodept.com/telesto/blogitem.htm?id=627
9. http://www.few.vu.nl/~nsilvis/PPL/2007/index.html
10. http://www.cs.ucdavis.edu/courses/exp_course_desc/10.html
11. http://www.people.fas.harvard.edu/~preshman/python_winter.html
12. http://www.comp.leeds.ac.uk/acom1900/
13. http://www.cs.bu.edu/course/cs108/materials.html
14. http://gentry.metr.ou/byreofpython/
15. http://www.msu.edu/~ashton/classes/825/intex.html
16. http://homepages.inf.ed.ac.uk/ewan/masws
17. http://dsnra.jpl.nasa.gov/software/Python/byte-of-python/output/byteofpython_html
18. http://www.python.org/doc/intros
19. http://www.creativecommons.org/licenses/by-sa/3.0/
20. http://www.opensource.org/licenses/bsd-license.php
21. http://www.swaroopch.com/buybook
22. http://www.swaroopch.com/files/byteofpython/byte_of_python_v191.pdf
23. http://www.swaroopcg.com/files/byteofpython/byte_of_python v191.xml
24. http://www.paypal.com/cgi-bin/wesbscr?cmd=_donations&business=swaroop%40swaroopch%2ecom&item_name=A%20Byte%2020of%20Python&no_shipping=0&no_note=1&tax=0&currency_code=USD&lc=IN&bn=PP%2dDonationsBF&charset=UTF%2d8
25. http://www.swaroopch.com/buybook

Πηγή: http://www.swaroopch.com/mediawiki/index.php?oldid=1391
Συνεισφορές: Swaroop, 1 ανώνυμος

Python en: πινάκας περιεχομένων

Python en: Μεταφράσεις

Υπάρχουν πολλά μεταφρασμένα βιβλία από διάφορες γλώσσες, ευχαριστώ τους ακούραστους εθελοντές!
Αν θέλετε να βοηθήσετε παρακαλώ πρώτα δείτε τις λίστες παρακάτω και αποφασίστε αν θέλετε να ξεκινήσετε μια μετάφραση ή να βοηθήσετε σε μια υπάρχουσα μετάφραση.

Κινέζικα

Η Juan Shen μετέφρασε το βιβλίο στα Κινέζικα.

Παραδοσιακά Κινέζικα

Η Fred Lin μετέφρασε το βιβλίο.
Είναι διαθέσιμο στο http://code.google.com/p/zhpy/wiki/ByteOfZhpy
Μαζί με την μετάφραση περιέχεται ένα αρχείο Chinise python sources δίπλα από την αρχική πηγή της Python.

Ιταλικά

Ο Enrico Morelli και ο Massimo Lucci μετέφρασαν αυτό το βιβλίο.
Η Ιταλική έκδοση βρίσκετε στο http://www.gentoo.ti/programmazione.byteofpython.
Μια νέα έκδοση είναι σε εξέλιξη.

Γερμανικά

Ο Lutz Horn, ο Bernd Hengelein και ο Cristoph Zwerschke μετέφρασαν αυτό το βιβλίο.
Αυτή η μετάφραση βρίσκετε http://abop-german.berlios.de

Νορβηγία

Ο Eirik Vageskar είναι μαθητής Λυκείου στο Sandvika videregaende skole και μετέφρασε το βιβλίο.
Το βιβλίο βρίσκεται στο http://forbedre.blogspot.com

Ινδονησία

Ο Daniel μετέφρασε το βιβλίο βρίσκεται στο http://python.or.id./moin.cgi/ByteofPython

Πολωνικά

Ο Dominik Kozaczko μετέφρασε το βιβλίο στα Πολωνικά.
Και βρίσκεται στο http://wiki.mercury.lo5.bielsko.pl/index.php/UkA A_Pythona

Καταλονικά

Ο Moises Gomez μετέφρασε αυτό το βιβλίο. Η μετάφραση είναι σε εξέλιξη και ξεκινάει με το κεφάλαιο 'Taula de continguts'

Πορτογαλικά

Ο Fidel Viegas μετάφρασε το βιβλίο.

Ρουμάνικα

Ο Paul-Sebastian Manole μετάφρασε το βιβλίο.
Βρίσκεται στο http://www.swaroopch.com/notes/python_ro

Βραζιλιάνικα-Πορτογαλικα

Ο Ridrigo Amaral μετέφρασς αυτό το βιβλίο.
Βρίσκεται στο http://rodrigoamaral.net

Γαλλικά

Ο Gregory μετέφρασε αυτό το βιβλίο στα Γαλλικά

Δανία
Ο Lars Petersen μετέφρασε αυτό το βιβλίο στα Γερμανικά.

Ισπανικά
Ο Alfonso de la Gyarda Reyes και ο Gystavo Echeverria μετέφρασαν το βιβλίο στα Ισπανικά. Η μετάφραση είναι σε εξέλιξη.

Αραβικά
Ο Alaa Abadin μετέφρασε το βιβλίο στα Αραβικά ISA.

Σουηδικά
Ο Mikael Jacobsson μετέφρασε το βιβλίο σε Σουηδικά.

Ρώσικα και Ουκρανικά
Ο Averkiev Andrey μετέφρασε το βιβλίο σε Ρώσικα και μάλλον Ουκρανικά.

Τουρκία
Ο Turker Sezer και ο Bygra Cakir μετέφρασαν το βιβλίο σε Τούρκικα.

Μογγολία
Atiunsanaa Tunjin μετέφρασε το βιβλίο στα Μογγολικά.

Σημείωση
Αντικαταστάθηκαν 'at' με το '@', 'dot' με '.' και 'underscore' με '_' στης διεύθυνσης που αναφέρθηκαν σε αυτή την σελίδα. Η παύλες παρέμειναν ως έχουν.

Python en: πρόλογος

Η Python είναι πιθανόν από τις λίγες γλώσσες προγραμματισμού που είναι και απλή και πολύ δυνατή. Αυτό είναι κάλο και για τους αρχάριους και για αυτούς που έχουν προχωρημένο επίπεδο, και το πιο σημαντικό είναι διασκεδαστικό να προγραμματίζεις με αυτήν. Αυτό το βιβλίο στοχεύει στο να μάθεις μια πολύ δυνατή γλωσσά και να σου δείξει πόσο εύκολα και χωρίς κόπο να τη αποκτήσεις, είναι το καλύτερο αντίδοτο για τα προβλήματα που έχεις με την γλωσσά προγραμματισμού.

Για ποιον είναι το βιβλίο αυτό.

Αυτό το βιβλίο είναι ένας καλός οδηγός και βοήθημα στην γλώσσα της Python. Βοηθάει συγκεκριμένα τους αρχάριους. Άλλα είναι χρήσιμο και για τους υψηλού επίπεδου επίσης. Ο στόχος είναι ότι αν ξέρεις έστω και λίγο από ηλεκτρονικούς υπολογιστές τότε με αυτό το βιβλίο θα μάθεις τη γλωσσά προγραμματισμού Python. Αν έχεις γνώσεις προγραμματισμού τότε επίσης μπορείς να μάθεις την Python από αυτό το βιβλίο. Αν έχεις προηγούμενες γνώσεις στο προγραμματισμό θα ενδιαφερθείς για τις διαφορές μεταξύ της Python και της γλώσσας που σου αρέσει, έχω τονίσει πολλές τέτοιες διαφορές. Προσοχή!Η Python μπορεί να γίνει η νέα γλώσσα προγραμματισμού που θα προτιμάς!

Μάθημα Ιστορίας

Πρώτα άρχισα να χρησιμοποιώ την Python όταν θέλησα να εγκαταστήσω ένα λογισμικό που είχα γράψει με το όνομα 'Diamond' και ήθελα να κάνω την εγκαταστήσει πολύ εύκολα. Επρεπε να διαλέξω μεταξύ της Python και της Perl για την Qt βιβλιοθήκη.
Έκανα μια έρευνα στο διαδίκτυο και βρήκα ένα άρθρο του Eric S. Raymond, τον γνωστό και με σεβασμό hacker ,να λέει για την Python ότι έχει γίνει η αγαπημένη του γλώσσα προγραμματισμού. Επίσης βρήκα ότι το PyQt είναι πιο δεμένο μεταξύ τους σε σύγκριση με την Perl-Qt. Οπότε αποφάσισα ότι η Python είναι η γλώσσα για μένα. Μέτα άρχισα να ψάχνω για ένα κάλο βιβλίο για την Python. Και δεν μπορούσα να βρω κανένα! Βρήκα μερικά του O'Reilly άλλα ή ήταν πολύ ακριβά ή ήταν περισσότερο αναφορές παρά βοήθημα. Οπότε χρησιμοποίησα το εγγραφο που είχε μαζί με την Python. Όμως ήταν μικρο και σύντομο. Μου έδινε μια καλή ηδέα για την Python αλλά δεν ήταν ολοκληρωμένη. Τα κατάφερα γιατί είχα προηγούμενη γνώση στο προγραμματισμό, άλλα δεν έκανε για αρχάριους.
Για 6 μήνες αφού είχα ήδη ξεκινήσει με την Python, εγκατέστησα τα τελευταία Red Hat 9.0 Linux και έπαιζα με την Kword. Ήμουν πόλη χαρούμενος και ξαφνικά μου ήρθε η ηδέα να γράψω κάτι για την Python. Άρχισα να γράφω μερικές σελίδες άλλα γρήγορα έγιναν 30 σελίδες. Μέτα αποφάσισα να φτιάξω ένα βιβλίο το οποίο θα είναι χρήσιμο. Μέτα από πολύ γραφή, έφτασε στο σημείο όπου έγινε χρήσιμο για να την εκμάθηση της Python. Ελπίζω αυτό το βιβλίο να γίνει αφιέρωμα στης ανοιχτές κοινότητες.
Αυτό το βιβλίο ξεκίνησε σαν σημειωματάριο για την Python και ακόμα έτσι το θεωρώ, παρόλα αυτά έκανα πολύ προσπάθεια να το κάνω κατανοητό στους άλλους.

Κατάσταση του βιβλίου

Αλλάζει από τότε που δημιουργήθηκε στις 5 Μαρτίου και αναβαθμίζεται για την Python 3.0.
Από τότε η Python 3.0 δεν έχει τελειώσει ή σταματήσει σε αυτό το βιβλίο τις συνεχείς αλλαγές. Αλλά στην φιλοσοφία τον ανοιχτών πηγών της 'Release Early, Release Often', το ανανεωμένο βιβλίο είναι ελεύθερο και συνεχώς αναβαθμίζεται.
Το βιβλίο χρειάζεται την βοήθεια των αναγνωστών για να δείξουν όποιο κομμάτι του βιβλίου δεν είναι κάλο,δεν είναι κατανοητό ή άπλα λάθος. Σας παρακαλώ να γράψετε στον κεντρικό συγγραφέα (http://www.swaroopch.com/contact/) ή στους μεταφραστές τις σημείωσεις και τις προτάσεις σας. Είναι δύσκολο να κρατάς τις ισορροπίες σε ένα βιβλίο που είναι τόσο για αρχάριους όσο και για αναγνώστες σε προχωρημένο επίπεδο. Θα ήταν μεγάλη βοήθεια αν δίνατε απαντήσεις για το πόσο να βαθύνει αυτό το βιβλίο.

Επίσημη σελίδα

Η επίσημη σελίδα του βιβλίου είναι http://www.swaroopch.com/notes/Python όπου μπορείς να διαβάσεις όλο το βιβλίο, ή να κατεβάσεις την τελευταία έκδοση του βιβλίου, ή να αγοράσετε ένα αντιγράφο και επίσης να μου στείλετε ότι θέλετε να μου υποδείξετε για το βιβλίο.

Άδεια

1. Αυτό το βιβλίο έχει την άδεια του Creative Commons Attribustion-Noncommerxial-share Alike 3.0 βρίσκεται στη σελίδα http://creativecommons.org/licenses/by-nc-sa/3.0/

- Αυτό σημαίνει:
- Ότι είσαι ελεύθερος να μοιράζεις το αντίγραφο και να μεταφράσεις το βιβλίο.
- Ότι είσαι ελεύθερος να επεξεργαστείς το βιβλίο.
- Κάτω από αυτές τοις συνθήκες:
  - Να αναφέρεις την δουλειά του συγγραφέα για αυτό το βιβλίο με τον τρόπο που σου επιτρέπει και όχι με κανένα άλλο τρόπο.
  - Να το μοιράζεις. Αν αλλάξεις ή μεταποιήσεις το έγγραφο αυτό θα πρέπει να χρησιμοποίησής τους ειδικούς όρους.
  - Για κάθε νέο έγγραφο που θα φτιάχνεις πρέπει να δείχνεις καθαρά τους όρους του βιβλίου και του συγγραφέα
  - Όλοι οι προηγούμενοι οροί μπορεί να αλλάξουν μόνο με την άδεια του δικαιούχου.
  - Τίποτα από αυτούς τους ορούς δεν επηρεάζουν τις ηθικές αρχές του συγγραφέα.
- Το αφιέρωμα πρέπει να φαίνεται στην διεύθυνση http://www.swaroopch.com/notes/Python και να δείξετε καθαρά ότι το αυθεντικό αρχείο μπορούν να το βρουν σε αυτή την διεύθυνση.
- Όλα τα κείμενα και οι κώδικες που είναι σε αυτό το βιβλίο έχουν την άδεια του 3-clause BSD(http://www.opensource.org/licenses/bsd-licensed.php) εκτός κάποιας άλλης ειδοποίηση.
- Η συνεισφορά των εθελοντών σε αυτό το βιβλίο πρέπει να είναι κάτω από τις ίδιες άδειες και δικαιώματα με του συγγραφέα του βιβλίου.

Επικοινωνία

Έχω κάνει πολύ προσπάθεια για να γράψω αυτό το βιβλίο και να το κάνω ενδιαφέρον και ακριβές. Όμως, αν βρείτε κάποια λάθη ή κάποια βελτίωση, απλά ενημερώστε με, για να κάνω τις αλλαγές. Μπορείτε να επικοινωνήσετε μαζί μου μεσώ της σελίδας μου.

Αγορά του βιβλίου

Αν θέλετε να μας στηρίξετε στο να συνεχίσουμε να κάνουμε βελτιώσεις στο βιβλίο, παρακαλώ σκεφτείτε να αγοράζεται το αντίγραφο (http://www.swaroopch.com/buybook) ή να κάνετε μια δωρεά.

Κάτι για να σκεφτείτε

Υπάρχουν δυο τρόποι να φτιάξετε ένα λογισμικό: ένα από αυτά είναι να το κάνετε τόσο απλό ώστε να μην έχει ελλείψεις, ο άλλος τρόπος είναι να το κάνετε τόσο πολύπλοκο ώστε να μην είναι προφανείς οι ελλείψεις.
--C.A.R. Hoare
Η επιτυχία στην ζωή είναι θέμα όχι τόσο του ταλέντου και της ευκαιρίας άλλα της συγκέντρωση και της επιμονής.
--C.W. Wendte

Python en: Εισαγωγή

Εισαγωγή

Η Python είναι μια γλώσσα που μπορείς να ισχυρισθείς ότι είναι απλή και πολύ δυνατή. Θα ανακαλύψεις ευχάριστα πόσο εύκολα μπορείς να συγκεντρωθείς στη λύση του προβλήματος παρά στην σύνταξη και την δομή της γλώσσας που προγραμματίζεις.
Η επίσημη εισαγωγή στην Python είναι:

- Η Python είναι μια εύκολη για να τη μάθεις και δυνατή γλωσσά προγραμματισμού. Είναι εγκεκριμένα υψηλού επίπεδου κατασκευής δομή και απλή άλλα έχει και αποτελεσματική προσέγγιση στο αντικείμενο. Η Python είναι κομψή στην σύνταξη και δυνατή στην πληκτρολόγηση, μαζί με την φύση του ερμηνευτή, την κάνει ιδανική γλωσσά για γραφή και για γρήγορη ανάπτυξη εφαρμογών σε πολλές περιοχές και πλατφόρμες.
- Θα συζητήσω τα περισσότερα χαρακτηριστικά με περισσότερες λεπτομέρειες σε επόμενο κεφάλαιο.
- **Σημειώσεις**
  Ο Guido van Rossun, ο δημιουργός της γλώσσας Python, ονόμασε την γλώσσα από το πρόγραμμα του BBC “Monty Python's Flying Circus”. Δεν του άρεσαν τα φίδια που σκότωνα τα ζώα για φαγητό με το να τα σφίγγουν .

Τα χαρακτηριστικά της Python

Απλά

Η Python είναι μια απλή γλώσσα. Διαβάζοντας ένα πρόγραμμα της Python είναι σχεδόν σαν να διαβάζεις αγγλικά, αν και είναι αυστηρώς αγγλικά! Ψευδό-κώδικες είναι μια από τη φυσική δύναμη της Python. Σου επιτρέπει να επικεντρωθείς στην επίλυση του προβλήματος παρά στην γλώσσα από μόνη της.

Εύκολη στην Μάθηση

Όπως θα δεις, η Python είναι μια πολύ εύκολη γλώσσα να ξεκινήσεις. Η Python έχει μια πολύ απλή σύνταξη, όπως έχω ήδη αναφέρει.

Ελεύθερη και ανοιχτές πηγές

Η Python είναι ένα παράδειγμα της FLOSS(Free/Libre και Open Source Software). Με άπλα λόγια. Μπορείς ελεύθερα να μοιράζεις αντίγραφα από αυτό το λογισμικό, το FLOSS βασίζεται στην αντίληψη μιας κοινωνίας που μοιράζεται η γνώση. Αυτός είναι ένας από τους λόγους που η Python είναι τόσο καλή, γιατί η κοινωνία την βελτιώνει απλά για να κάνουν την Python καλύτερη.

Υψηλού επιπέδου γλώσσα

Όταν γράφεις προγράμματα στην Python, ποτέ δεν ασχολείσαι με τις λεπτομέρειες όπως το να κάνεις διαχείριση της μνήμης από το πρόγραμμα σου.

Φορητός

Λόγο της ανοιχτής πηγής, Η Python μπορεί να είναι λειτουργική σε πολλές πλατφόρμες. Όλα τα προγράμματα μπορούν να δουλέψουν σε οποιαδήποτε πλατφόρμα χωρίς να χρειάζεται καμιά αλλαγή, αν είσαι προσεχτικός αρκετά ώστε να αποφύγεις όποιο σύστημα εξαρτάται από τα χαρακτηριστικά του.
Μπορείς να χρησιμοποιήσεις την Python σε Linux, Windows, FreeBSD, Macintosh, OS.2, Amiga, Aros, AS/400, BeOS, OS/390 z.OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, Plastation, Sharp Zaurus, Windows CE και αοκμα και στο PocketPC !

Ερμηνευτής

Αυτό θέλει μια εξήγηση.
Το πρόγραμμα καταρτίζεται από την γλωσσά, όπως την C ή C++, και χρησιμοποιεί μετατροπείς από τις πήγες των γλωσσών. C ή C++ είναι οι γλώσσες που μιλάει ο ηλεκτρονικός υπολογιστής χρησιμοποιώντας ένα μετατροπέα με διάφορες σημαίες και επιλογές. Όταν τρέχεις το πρόγραμμα, το λειτουργικό, που φορτώνει τα αντίγραφα από το πρόγραμμα στον σκληρό δίσκο και στην μνήμη, ξεκινάει να τρέχει.
Η Python από την άλλη δεν χρειάζεται σύνταξη από άλλες βιβλιοθήκες. Απλά τρέχεις το

πρόγραμμα απευθείας από τον πηγαίο κωδικά. Επομένως, η Python μετατρέπει τον πηγαίο κωδικά σε ενδιάμεσες φόρμες που ονομάζεται Bytecodes και μεταφράζει αυτό στην ντόπια γλώσσα του υπολογιστή και μετά το τρέχει. Όλα αυτά πραγματικά κάνουν την Python πολύ πιο εύκολη αφού δεν χρειάζεται να ανησυχείς με το να καταρτίσεις το πρόγραμμα, να είσαι σίγουρος ότι η βιβλιοθήκες και οι συνδέσεις είναι φορτωμένες. Αυτό επίσης κάνει τα προγράμματα Python πολύ πιο φορητά, αφού μπορείς να κάνεις αντιγραφή του προγράμματος Python σε έναν άλλο υπολογιστή και να δουλέψει.

Αντικειμενοστραφής

Η Python υποστηρίζει τη διαδικασία προσανατολισμού του προγραμματισμού όπως και το αντικείμενο του προσανατολισμού προγράμματος. Στην διαδικασία προσανατολισμού για γλώσσες, το πρόγραμμα έχει φτιαχτεί γύρω από διαδικασίες ή λειτουργίες οπού δεν είναι τίποτα άλλο από επαναχρησιμοποιήσιμα κομμάτια από προγράμματα. Στα αντικείμενα προσανατολισμού γλώσσας, τα προγράμματα έχουν φτιαχτεί γύρω από αντικείμενα που έχουν συνδυάσει τα δεδομένα και την λειτουργικότητα. Η Python έχει ένα πολύ δυνατό αλλά απλοϊκό τρόπο να κάνει OOP, ειδικά όταν συνδυάζει μεγάλες γλώσσες όπως C++ ή java.

Επεκτάσιμη

Αν θες ένα σημαντικό κομμάτι κώδικα για να τρέξεις πολύ γρήγορα ή αν θες να έχεις ένα κομμάτι αλγόριθμου που να μην ανοίγει, μπορείς να κάνεις το κώδικα του προγράμματος στο C++ ή C και μετά να το χρησιμοποιήσετε στο Python πρόγραμμα.

Ενσωμάτωση

Μπορείς να ενσωματώσεις το Python μαζί με το C/C++ πρόγραμμα για να σου δώσει τη χωρητικότητα σεναρίου για τους χρήστες του προγράμματος.

Εκτενής Βιβλιοθήκες

Η στάνταρ βιβλιοθήκες της Python είναι πραγματικά μεγάλες. Μπορεί να σε βοηθήσει να κάνεις σοβαρά πράγματα που σχετίζονται με συνηθισμένες επεκτάσεις, δημιουργία εγγράφων, δοκιμασία ενότητας, σπείρωμα δεδομένων, προγράμματα περιήγησης στο Web, GCI, FTP, email, XML, XML-RPC, HTML, WAV αρχεία, κρυπτογραφία, GUI, Tk, και άλλα εφαρμοσμένα συστήματα. Να θυμάστε, όλα αυτά είναι πάντα προσιτά όπου είναι εγκατεστημένοι η Python. Αυτό λέγεται 'μπαταρίες συμπεριλαμβάνονται' η φιλοσοφία της Python.
Έκτος από τις στάνταρ βιβλιοθήκες, υπάρχουν και άλλες σημαντικές βιβλιοθήκες υψηλού επιπέδου όπως η wxPython(http://www.wxpython.org), Twisted(http://www.twistedmatrix.com/products/twisted), Python Imaging Library(http://www.pythonwera.com/products/pil/index.htm) και πολλές άλλες.
Η Python είναι πραγματικά μια ωραία και δυνατή γλώσσα. Έχει τα κατάλληλα συστατικά για την απόδοση της και χαρακτηριστικά που κάνουν την Python ευχάριστη και εύκολη.

Γιατί όχι Perl?

Αν δεν το ξέρατε ήδη, η Perl είναι μια άλλη γνωστή ανοιχτού κώδικα γλώσσα προγραμματισμού.
Αν έχετε προσπαθήσει να γράψετε ένα πρόγραμμα στην Perl, θα έχετε απαντήσει και εσείς στο ερώτημα.! Της Perl τα προγράμματα είναι εύκολα όταν είναι μικρά προγράμματα και υπερέχει σε μικρά hacks και σενάρια για να γίνει το έργο. Όμως γρήγορα γίνεται απρόθυμη όταν ξεκινάτε να γράφεται μεγάλα προγράμματα και το λέω αυτό από την εμπειρία μου να γράφω μεγάλα προγράμματα Perl στο Yahoo!
Όταν σύγκρινα την Perl με την Python ήταν απόλυτα απλή, καθαρή, εύκολη στην γράφει και ως εκ τούτου ποιο κατανοητή και διατηρήσιμη. Θαυμάζω την Perl και την χρησιμοποιώ καθημερινά σε διάφορα πράγματα αλλά όποτε θέλω να γράψω πρόγραμμα πάντα σκέφτομαι με τους όρους της Python. γιατί έχει γίνει τόσο φυσική σε μένα. Η Perl έχει υποστεί τόσες πολλές επιβαρύνσεις, που είναι πλέον σαν ένα μεγάλο πρόβλημα.

Δυστυχώς, η απερχόμενη Perl 6 δεν δείχνει να έχει κάνει καμιά βελτιώση σχετικά με αυτό. Το μόνο και πολύ σημαντικό πλεονέκτημα που έχει η Perl, είναι η μεγάλη CPAN (http://cpan.perl.org) βιβλιοθήκη- Το ολοκληρωμένο της Perl αρχείο Δικτύου. Όπως και το όνομα υποδηλώνει, αυτή είναι μια τεράστια συλλογή από μοντέλα της Perl και είναι απλό μυαλό - boggling γιατί έχει απόλυτο μέγεθος και βάθος – μπορείς να κάνεις πρακτικά τα πάντα με ότι αφορά την χρησιμοποίησή μοντέλων του ηλεκτρονικού υπολογιστή.
Ένας από αυτούς τους λόγους όπου η Perl έχει περισσότερες βιβλιοθήκες από την Python είναι γιατί η Perl είναι πιο παλαιά και λειτουργεί περισσότερο καιρό από την Python. Όμως αυτό δείχνει να αλλάζει με το μέγεθος των πακέτων δεδομένων για την Python που αναπτύσσονται.(http://pypi.python.org/pypi).

Γιατί όχι την Ruby

Αν δεν το ξέρατε ειδή, η Ruby είναι μια άλλη γνωστή ανοιχτού κώδικα γλώσσα προγραμματισμού. Αν χρησιμοποιείτε την Ruby, τότε σας προτείνω να συνεχίσετε να τη χρησιμοποιείτε. Για τους άλλους ανθρώπους που δεν έχουν αποφασίσει τη να χρησιμοποιούσουν την Python ή την Ruby, τότε σας συνιστώ την Python, από την άποψη ότι είναι πιο εύκολο να την μάθεις. Εγώ προσωπικά θεωρώ δύσκολο να κατανοήσω την Ruby, άλλα για τους ανθρώπους που καταλαβαίνουν την Ruby, όλοι επαινούν την ομορφιά της γλωσσάς. Δυστυχώς, δεν είμαι τόσο τυχερός.

Τι λένε οι προγραμματιστές

Μπορεί να το βρείτε ενδιαφέρον να δείτε τι λένε μεγάλοι hackers όπως ο ESR για την Python:

- Eric S. Raymond είναι ο συγγραφέας του “The Cathedral and has Bazaar” και είναι επίσης το άτομο που επινόησε τον όρο “ανοιχτές πήγες”. Λέει ότι η Python έχει γίνει η αγαπημένη του γλώσσα προγραμματισμού. (http://www.linuxjournal.com/article.php?sid=3882.). Αυτό το άρθρο ήταν έμπνευση για να γράψω την πρώτη βούρτσα στην Python.
- Bruce Eckel ειναι ο συγγραφέας του διάσημου βιβλίου “Thinking in Java and Thinking in C++” είπε ότι καμιά άλλη γλώσσα δεν τον έκανε πιο παραγωγικό από ότι η Python. Είπε ότι η Python μπορεί να είναι η μόνη γλωσσά η όποια εστιάζει στο να είναι εύκολη στο να προγραμματιστεί. Διαβάστε την ολοκληρωμένη συνέντευξη (http://www.artima.com/intv/aboutme.)
- Peter Norvig είναι γνωστός από το βιβλίο Director of Search Quality για την Google(ευχαριστώ τον Guido van Rossum που μου το υπόδειξε αυτό). Είπε ότι η Python είναι πάντα ένα ολοκληρωμένο κομμάτι της Google. Μπορείς να βρεις αυτές της δηλώσεις στην Google Jobs (http://www.google.com/jobs/index.html) σελίδα όπου μας κάνει γνωστό ότι η Python είναι ένα εξάρτημα για τους μηχανικούς λογισμικών.

Η Python

Η Python 3.0 είναι η νέα έκδοση της γλώσσας. Μερικές φόρες αναφέρεται σαν Python 3000 ή Py3K.
Ο λόγος της νέας έκδοσής της Python είναι να αφαιρέσουν τα μικρά προβλήματα και άλλες επιλογές που έχουν συσσωρευτή με το πέρας του χρόνου και για να κάνουν την γλώσσα ποιο κατανοητή. Αν έχετε ήδη πολλά από Python 2.x κώδικες, τότε για αυτό υπάρχει η χρησιμότητα του σκληρού δίσκου για να σας βοηθήσει να μετατρέψετε τους 2.χ σε 3.χ πηγές. (http://docs.python.org/dev/3.0/library/2to3.html)
Περισσότερες πληροφορίες εδώ:

- Guido van Rossum's παρουσίαση(http://www.artima.com/weblogs/viewpost.jsp?thread=208549)
- Τι νέα έχει η Python 2.6(http://docs.python.org/dev/3.0/whatnew/3.0.html) (χαρακτηριστικά που διαφέρουν σημαντικά από τις προηγούμενες εκδόσεις 2.x python και πιθανότατα θα συμπεριληφθούν στην Python 3.0)

- Τα νέα στην Python 3.0(http://docs.python.org/dev/3.0/whatsnew/3.0.html)
- Κυκλοφορία προγράμματος Python 2.6 και 3.0(http://www.python.org/dev/peps/pep-0361/)
- Python 3000 (επίσημη λίστα που προτείνει) (http://www.python.org/dev/peps/pep-3000/)
- Διάφορα σχέδια της Python 3.0 (http://www.python.org/dev/peps/pep-3100/)
- Νέα της Python (αναλυτική λίστα από αλλαγές) (http://www.python.org/download/releases/3.0/NEWS.txt)

Python en: Εγκατάσταση

Αν έχεις Python 2.χ εγκατεστημένοι ήδη, δεν χρειάζεται να αφιερώσετε χρόνο για να εγκαταστήσετε την Python 3.0. Μπορείς να έχεις και τις δυο την ίδια στιγμή.

Για Linux και BSD χρήστες

Αν χρησιμοποιείτε την Linux λειτουργικά όπως η Ubuntu, Fedora, OpenSUSE ή {βάλτε την επιλογή σας εδώ}, ή BSD λειτουργικό όπως και FreeBSD, τότε είναι πολύ πιθανόν να έχετε εγκαταστήσει την Python στο λειτουργικό σας.
Για να δείτε αν έχετε την Python ήδη εγκατεστημένοι στο Linux box, άνοιξε το τερματικό σας(οπός την κονσόλα ή το gnome-terminal) και βάλτε την εντολή python -v οπός δίνεται εδώ.

```
$ python -v
python 3.0b1
```

Σημείωση
Το “$” είναι εντολή του κέλυφους. Θα είναι διαφορετικά, εξαρτάται από το επιλογές του συστήματος OS, ως εκ τούτου θα υποδεικνύουν τις εντολές από το σύμβολο $.
Αν έχετε δει κάποια εκδοχή πληροφορίας όπως αυτή που δείχνετε από κάτω, τότε έχετε την Python εγκατεστημένοι ήδη.
Αλλά, αν έχετε ένα μήνυμα σαν αυτό:

```
$ python -v
bash: python: command noot found
```

τότε δεν έχετε εγκατεστημένη την Python. Αυτό είναι πολύ απίθανο αλλά και πιθανό.

Σημείωση
Αν έχετε την Python 2.χ εγκατεστημένη, τότε δοκιμάστε την python3 -v.
Σε αυτή την περίπτωση, έχετε δυο περιπτώσεις για να εγκαταστήσετε την Python στο λειτουργικού σας.

- Μπορείτε να εγκαταστήσετε την Python από(http://www.python.org/download/releases/3.0/) και να την εγκαταστήσετε. Οι οδηγίες δίνονται από το ίντερνετ.
- Αυτή η επιλογή θα είναι ελεύθερη μετά την τελική απελευθερώσει της Python 3.0. Εγκατάστησε τα πακέτα από την εφαρμογή διαχείρισης δεδομένων. Αυτή είναι ήδη εγκατεστημένη στο OS, όπως το apt-get στο Ubuntu/Debian και σε άλλες Debian-based Linux, yum στην Fedore Linux, pkg_add στα FreeBSD,
  σημείωση: θα χρειαστείτε σύνδεση ίντερνετ για κάνετε αυτή την μέθοδο. Εναλλακτικά, μπορείτε να κατεβάσετε τα εκτελέσιμα αρχεία άλλου και να τα αντιγράψετε στο PC και να το εγκαταστήσετε.

Για τους χρήστες Windows

Επισκεφτείτε την http://www.python.org/download/release/3.0/ και κατεβάστε την τελευταία έκδοση από αυτή την ιστοσελίδα, όπου υπάρχει η 3.0 beta 1 (http://www.python.org/ftp/python/3.0/python-3.0b1.msi) όπως αυτή γραφή. Αυτό είναι μόνο 12.8MB το οποίο είναι πιο συμπαγές αν αναλογιστείς τα λογισμικά ή τις γλώσσες σε σύγκρισή με άλλες. Η εγκατάσταση είναι σαν ένα άλλο Windows-based λογισμικό.

Προσοχή

Όταν σου δίνεται η επιλογή για να διαλέξεις προαιρετικά κάποια συστατικά, μην ξεδιαλέξετε τίποτα.! Μερικές από αυτά τα συστατικά μπορεί να είναι χρήσιμα για εσάς. Ιδικά το IDLE.
Ένα ενδιαφέρον γεγονός είναι η πλειοψηφία της χρήσης Python γίνετε από Windows χρήστες. Σίγουρα αυτό δεν δίνει την ολοκληρωμένη εικόνα αφού όσοι χρησιμοποιούν την Linux έχουν ήδη εγκαταστήσει την Python στο λειτουργικό τους.

DOS Prompt

Αν θέλετε να μπορείτε να χρησιμοποιείτε την Python από τη γραμμή εντολών τον Windows το DOS prompt, τότε πρέπει να βάλετε την διαδρομή για την μεταβλητή κατάλληλα.
Για τα Windows 2000, XP, 2003, κλικάρετε στο control panel > system > Advance > Environment Variables. κλικάρετε στην μεταβλητή στο όνομα PATH στο “System Variables” κομμάτι, μετά διαλέξτε Edit και Add; c:\Python30 στο τέλος σε ότι υπαρχή ήδη εκεί. Βέβαια, χρησιμοποιήστε το κατάλληλο όνομα αρχείου.

- Για παλαιότερες εκδόσεις της Windows, προσθέτεται στην παρακάτω γραμμή στο αρχείο (c:\AUTOEXECT.BAT:'PATH=%PATH%;C:\Python30') (χωρίς τις παρενθέσεις) και ξανά ξεκινήστε το σύστημα. Για Windows NT, χρησιμοποιήστε το AYTOEXEC.NT αρχείο.

Για Mac OS X χρήστες

Για Mac OS X οι χρήστες θα βρουν ήδη εγκατεστημένη στο σύστημα. Ανοίξτε το τερματικό .app και τρέξτε την εντολή python -v και μετά χρησιμοποιήστε τις εντολές από πριν για τους χρήστες της Linux.

Περίληψη

Για ένα σύστημα Linux, πολύ πιθανόν να έχετε ήδη εγκαταστήσει την Python. Αλλιώς, μπορείτε να την εγκαταστήσετε από την εφαρμογή για την διαχείριση δεδομένων αυτό είναι ήδη στο σύστημα.
Για ένα σύστημα Windows, η εγκαταστήσει της Python είναι εύκολη απλά κατεβάζετε και κάνετε διπλό κλικ στο Installer. Από τώρα θα θεωρούμε ότι έχετε εγκατεστημένη την Python στο σύστημα. Μετά, θα γράψουμε το πρώτο μας πρόγραμμα.

Python en: Τα πρώτα βήματα.

Εισαγωγή

Τώρα θα δούμε πως τρέχει το συνηθισμένο 'Hello World' πρόγραμμα στην Python. Αυτό θα μας διδάξει πως να γράφουμε, να αποθηκεύουμε και να τρέχουμε προγράμματα στην Python. Υπάρχουν δυο επιλογές να χρησιμοποιήσουμε την Python για να τρέξουμε ένα πρόγραμμα σε αυτήν. Χρησιμοποιώντας διαδραστικό διερμηνέα ή ένα πηγαίο αρχείο. Τώρα θα σας δείξουμε πως να χρησιμοποιείται και τα δυο.

Χρησιμοποιώντας των διερμηνέα εντολών

Ξεκινάμε τον διερμηνέα στην γραμμή εντολών βάζοντας την λέξη Python στο τερματικό.
Για χρήστες τον Windows, μπορείτε να τρέξετε τον διερμηνέα στη γραμμή εντολών αν έχετε βάλει σωστά το PATH. Αν χρησιμοποιείται IDLE, κάντε κλικ στο start > programs > Python 3.0 > IDLE(Python GUI).
Τώρα γράψτε: print ('Hello world') και μετά πατήστε το Enter. Θα δεις της λέξεις Hello World όπως παρακάτω.

```
 $python
        Python 3.0b2(r30b2:65106, jul 18 2008, 18:44:17) [MSC v.1500 32
bit (Intel)] on win32
        Type “help”, “copyright”, “credits” or “license” for more
information.
>>> print('Hello World')
Hello World
>>>
```

Παρατηρήστε ότι η Python σου δίνει στην έξοδο στην γραμμή αμέσως! Ότι έβαλες στην γραμμή σαν δήλωση. Χρησιμοποιούμε το 'print' για να μας εμφανίσει οποία τιμή του υποβάλλεις. Εδώ βάλαμε το κείμενο 'Hello World' και αμέσως το εμφανίζει στην οθόνη.

Πως να κλείσεις το διερμηνέα

Για να κλείσεις τον διερμηνέα, πατήστε 'ctrl-d' αν χρησιμοποιείτε IDLE ή χρησιμοποιείτε Linux/BSD κέλυφος. Στη περιπτώση των Windows για να κλείσετε της γραμμές εντολών, πατήστε 'ctrl-z και μετά πατήστε το 'enter'.

Διαλέγοντας έναν Συντάκτη

Πριν πάμε στο πως γράφουμε προγράμματα στην Python σε πηγές αρχείων, πρέπει να έχουμε έναν συντάκτη για να γράψουμε τα πηγαία αρχεία. Η επιλογή του συντάκτη είναι σημαντική. Πρέπει να διαλέξεις προσεκτικά, όπως αν αγόραζες ένα αμάξι. Ένας καλός συντάκτης θα σας βοηθήσει να γράψετε προγράμματα εύκολα, σου κάνει το ταξίδι πιο αναπαυτικό και σε βοηθάει να φτάσεις στον προορισμό σου με ένα πιο γρήγορο και ασφαλή δρόμο.
Ένα από τα ποιο σημαντικά προσόντα είναι ο **τονισμό της σύνταξης** όπου όλα τα διαφορετικά κομμάτια της Python στο πρόγραμμα είναι χρωματιστά ώστε να μπορείς να δεις το πρόγραμμα και να το φαντάζεσαι ότι τρέχει.
Αν χρησιμοποιείται Windows, τότε σας συνιστώ να χρησιμοποιείται IDLE. Η IDLE κάνει τονισμό της σύνταξης και πολλά άλλα όπως να σας αφήνει να τρέχετε το πρόγραμμα με IDLE μαζί με άλλα προγράμματα. Μια ξεχωριστή σημείωση: **μην χρησιμοποιείται το μπλοκ –** είναι μια κακή επιλογή γιατί δεν κάνει τονισμό της σύνταξης και επίσης σημαντικό δεν υποστηρίζει οδόντωση του κείμενου το οποίο είναι πολύ σημαντικό στην περίπτωση την δικιά μας, όπως θα δούμε πάρα κάτω. Καλές βιβλιοθήκες όπως IDLE(και επίσης VIM) θα σας βοηθήσουν αυτόματα. Αν χρησιμοποιείτε Linux/FreeBSD, τότε έχετε πολλές βιβλιοθήκες. Αν τώρα ξεκινάτε να προγραμματίζετε καλό θα ήταν να χρησιμοποιήστε την geany. Είναι για γραφική χρήση και έχει κουμπιά για να καταρτίζεις και να τρέχεις προγράμματα στην Python χωρίς φασαρία. Αν είστε έμπειρος προγραμματιστής, τότε πρέπει να χρησιμοποιείτε ήδη το Vim ή Emacs. Χρειάζεται να αναφέρω, ότι αυτά είναι δυο από τα πιο δυνατά συντακτικά και θα επωφεληθείς με το να τα χρησιμοποιήσεις για να γράψεις στην Python προγράμματα. Προσωπικά εγώ χρησιμοποιώ το Vim για τα περισσότερα προγράμματα. Αν είσαι αρχάριος, τότε πρέπει να χρησιμοποίησης το Kate είναι ένα από τα αγαπημένα μου. Στην

περιπτώση που θες να μάθεις την Vim ή Emacs, τοτε σας συνιστώ να μάθετε να χρησιμοποιείτε και τις δυο διότι θα σου είναι πολύ χρήσιμο στα μεγάλα προγράμματα.
Σε αυτό το βιβλίο, θα χρησιμοποιήσουμε IDLE, ή IDE και συντακτικό της επιλογή σας. IDLE είναι εγκατεστημένη μέσα στα Windows και Mac OS X Python. Επίσης είναι διαθέσιμη για εγκατάσταση των Linux (http://love-python.blogspot.com/2008/03/istall-idle-in-linux.html).
Θα εξερευνήσουμε πως να χρησιμοποιήσουμε την IDLE στο επόμενό κεφάλαιο. Για περισσότερες πληροφορίες, παρακαλώ πηγαίνετε στο αρχείο της IDLE (http://www.python.org/idle/doc/idlemain.html). Αν ακόμα θέλετε να εξερευνήσουμε για άλλους συντάκτες, κοιτάξτε την περιεκτική λίστα από συντάκτες της Python (http://www.python.org/cgi-bin/moinmoin/PythonEditors) και κάντε την επιλογή σας. Μπορείτε να επιλέξτε την IDE(Integrated Development Environment) για την Python. Δείτε την περιεκτική λίστα της IDES αυτή υποστηρίζει την Python (http://www.python.org/cgi-bin/moinmoin/integrateddevelopmentsenvironments) για περισσότερες πληροφορίες. Με το που ξεκινήσετε να γράφεται μεγάλα προγράμματα Python, IDEs θα σας είναι πολύ χρήσιμη.
Επαναλαμβάνω για άλλη μια φορά, παρακαλώ διαλέξτε ένα κατάλληλο συντάκτη – μπορεί να γράψει προγράμματα στην Python πιο ευχάριστα και εύκολα.

Για τους Vim χρήστες
Υπάρχει μια καλή εισαγωγή το πως να κάνουμε την Vim μια δυνατή Python IDE από τον John M. Anderson (http://blog.sontek.net/2008/05/11/python-with-a-modular-ide-vim/).

Για τους Emacs χρήστες
Υπάρχει μια καλή εισαγωγή το πως να κάνουμε την Emacs μια δυνατή Python IDE από τον Ryan McGuire (http://www.enigmacurry.com/2008/05/09/emacs-as-a-powerful-python-ide/).

Χρησιμοποιώντας ένα πηγαίο αρχείο

Τώρα ας επιστρέψουμε στον προγραμματισμό. Υπάρχει μια συνήθεια όταν μαθαίνεις μια νέα γλώσσα προγραμματισμού, το πρώτο πρόγραμμα που θα γράψεις και θα τρέξεις είναι το 'Hello World' πρόγραμμα – το μόνο που κάνει είναι να λέει 'Hello World' μετά το τρέχετε. Όπως ο Simon Cozens θέτει, είναι ένα παραδοσιακό ξόρκι προς τους θεούς του προγραμματισμού για να μας βοηθήσει να μάθουμε τη γλώσσα καλύτερα '.
Ξεκίνα την επιλογή του συντάκτη, βάλτε το ακόλουθο πρόγραμμα και σώστε το σαν Helloworlf.py. Αν χρησιμοποιείς IDLE, κάντε κλικ στο File > New Window και βάλτε το παρακάτω πρόγραμμα. Μετά κάντε κλικ στο File > Save.

```
#!/usr/bin/python
    #Filename: helloworld.py
    print('Hello world')
```

Τρέξτε αυτό το πρόγραμμα με το να ανοίξετε στο κέλυφος (Linux terminal ή DOS prompt) και βάλτε την εντολή python helloworld.py
Αν χρησιμοποιείτε IDLE, χρησιμοποιείστε το Run > Run Module ή την συντόμευση F5.
Η έξοδος είναι όπως αυτή:

```
    $ python helloworld.py
    Hello World
```

Αν βγάζεις και εσύ αυτό το αποτέλεσμα, συγχαρητήρια! Μόλις τρέξατε το πρώτο σας πρόγραμμα Python.
Σε περίπτωση που έχουμε σφάλμα, παρακαλώ πληκτρολογήστε το παρακάτω πρόγραμμα ακριβώς όπως φαίνεται από κάτω και τρέξτε το ξανά. Σημείωση ότι η Python είναι πεζών-κεφαλαίων οπότε το “print” δεν είναι το ίδιο με το “Print”. Παρατηρήστε τα πεζά “p” στην πρόταση και το κεφαλαίο “P” . Επίσης προσέξτε να μην έχετε κενά ή παύλες πριν το πρώτο χαρακτήρα σε κάθε γραμμή, θα δούμε πιο μετά γιατί αυτό είναι σημαντικό.

Πως δουλεύει

Ας σκεφτούμε τις δυο πρώτες γραμμές του προγράμματος. Ονομάζονται σχόλια – οτιδήποτε από τα δεξιά του σύμβολου “#” είναι σχόλια και είναι χρήσιμα για αυτόν που διαβάζει το πρόγραμμα.

Η Python δεν χρησιμοποιεί σχόλια εκτός από μερικές περιπτώσεις όπως εδώ στην πρώτη γραμμή. Ονομάζεται “shebang line”- όποτε οι δυο πρώτοι χαρακτήρες από τους πηγαίους φακέλους είναι “#!” που ακολουθούνται με την τοποθεσία του προγράμματος, αυτό λέει το σύστημα Linux/Unix ότι το πρόγραμμα πρέπει να τρέχει με διερμηνέα μόλις εκτελείται το πρόγραμμα. Αυτό εξηγείται στο επόμενο κεφάλαιο. Σημειώστε ότι μπορείτε να τρέξετε το πρόγραμμα σε οποιαδήποτε πλατφόρμα με το να προσδιορίσετε τον διερμηνέα ακριβώς στις γραμμές εντολών όπως η εντολή python helloworld.py.

Σημαντικό
Χρησιμοποιείστε τα σχόλια στο πρόγραμμα για να εξηγείτε κάποια σημαντικές λεπτομέρειες για το πρόγραμμα σας – αυτό είναι χρήσιμο για τους χρήστες του προγράμματος για να μπορούν εύκολα να καταλάβουν ακριβώς τι κάνει το πρόγραμμα. Θυμηθείτε, αυτό το άτομο μπορεί να είστε εσείς μετά από έξι μήνες! Τα σχόλια ακολουθούνται μετά από μια δηλώση στην Python. Εδώ καλούμε την εντολή “print function”, αυτό μόνο εκτυπώνει το κείμενο 'Hello World'. Θα μάθουμε για την Function σε ένα άλλο κεφάλαιο, αυτό που πρέπει να καταλάβετε τώρα είναι ότι και να βάλετε μέσα στης παρενθέσεις θα εκτυπωθεί στην οθόνη. Σε αυτή την περιπτώσει έχει βάλει την λέξη 'Hello World' το οποίο ονομάζεται string, μην ανησυχείτε θα ασχοληθούμε με τις ετοιμόλογιες πιο μετά.

Εκτελέσιμα προγράμματα της Python

Αυτό εφαρμόζεται μόνο για τους Linux/Unix χρήστες άλλα και για τους χρήστες τον Windows μπορεί να είναι περίεργη για την πρώτη γραμμή του προγράμματος. Πρώτα, πρέπει να δώσουμε στο πρόγραμμα άδεια εκτέλεσης χρησιμοποιώντας την εντολή “chmod” μετά τρέξτε το πηγαίο πρόγραμμα.

```
$ chmod a+x helloworld.py
$ ./helloworld.py
Hello World
```

Η εντολή chmod χρησιμοποιείται εδώ για να αλλάξει τη μορφή του αρχείου με το να δίνει άδεια εκτέλεσης σε όλους στους χρήστες του συστήματος. Μετά, εκτελούμε το πρόγραμμα προσδιορίζοντας την περιοχή του πηγαίου κώδικα. Χρησιμοποιούμε το “/.” για να περιγράψουμε ότι το πρόγραμμα βρίσκεται στον τρέχοντα κατάλογο. Για να κάνουμε αυτά πιο διασκεδαστικά, μπορείς να ξανά ονομάσεις το αρχείο σε helloWorld και να τρέξετε σαν ./helloworld και θα εξακολουθήσει να δουλεύει αφού το σύστημα ξέρει ότι θα τρέξει το πρόγραμμα χρησιμοποιώντας τον διερμηνέα που η θέση του προσδιορίζεται στην πρώτη γραμμή του προγράμματος.
Αν δεν ξέρετε που βρίσκεται η Python, Τότε, μπορείτε να χρησιμοποιείται την ειδική εντολή env στα Linux/Unix σύστημα. Απλά αλλάξτε την πρώτη γραμμή του προγράμματος στην ακόλουθη:

```
#!/usr/bin/env python
```

Το “env” πρόγραμμα θα κοιτάει τον διερμηνέα της Python, το οποίο θα τρέχει το πρόγραμμα. Μέχρι τώρα έχουμε την ικανότητα να τρέχουμε το πρόγραμμα όσο γνωρίζουμε την ακριβή διαδρομή. Και αν θέλουμε να τρέξουμε το πρόγραμμα από οπουδήποτε; Μπορούμε να το κάνουμε αυτό με το να αποθηκεύουμε το πρόγραμμα σε ένα από τους καταλόγους που είναι καταχωρημένους στο PATH περιβάλλον μεταβλητών. Όποτε τρέξετε οποιοδήποτε πρόγραμμα, το σύστημα ψάχνει το πρόγραμμα κάθε έναν από τους κατάλογους που είναι στη λίστα στο περιβάλλον PATH μεταβλητών και μετά τρέχει το πρόγραμμα. Μπορούμε να κάνουμε το πρόγραμμα διαθέσιμο παντού με το να αντιγράψουμε απλά το πηγαίο αρχείο σε ένα από τους καταλόγους που είναι καταχωρημένη στο PATH.

```
 $ echo $PATH
/usr/local/bin:usr/bin:/bin:/usr/X11R6/bin:/home/swaroop/bin
$ cp helloworld/py/home/swaroop/bin/helloworld
$ helloworld
Hello World
```

Μπορούμε να εμφανίσουμε της μεταβλητές της διαδρομής(PATH) χρησιμοποιώντας την εντολή echo και με το πρόθεμα το όνομα της μεταβλητής με το $ να το διευκρινίζει στο κέλυφος που χρειαζόμαστε την τιμή της μεταβλητής. Το βλέπουμε αυτό στο /home/swaroop/bin είναι στους καταλόγους στην διαδρομή όπου το swaroop είναι το όνομα του χρήστη που χρησιμοποιούμε στο σύστημα. Συνήθως θα υπάρχει μια παρόμοια βιβλιοθήκη για τον χρήστη του συστήματος. Εναλλακτικά, μπορείς να βάλεις κατάλογο της επιλογής σου στις μεταβλητές διαδρομής(PATH). Αυτό μπορεί να γίνει με το τρέξουμε το PATH=$:/home/swaroop/mydir όπου 'home/swaroop/mydir'. Είναι στον κατάλογο που θέλω να προσθέσω την μεταβλητή της διαδρομής. Αυτή η μέθοδος είναι πολύ χρήσιμη αν θέλεις να γράψεις χρήσιμα σενάρια όποτε και αν θες να τρέξεις το πρόγραμμα. Είναι σαν να δημιουργείς της δικές σου εντολές σαν cd ή σαν οποία άλλη εντολή χρησιμοποίησης στο τερματικό του Linux ή στο DOS prompt.

Προσοχή
W.r.t. Python, ένα πρόγραμμα ή ένα σενάριο ή λογισμικό όλα σημαίνουν το ίδιο πράγμα.

Δέχομαι βοήθεια

Αν χρειάζεσαι γρήγορα πληροφορίες για κάθε λειτουργία ή δήλωση στην Python, τότε μπορείτε να χρησιμοποιείται την ενσωματωμένοι βοήθεια – λειτουργικότητα. Είναι πολύ χρήσιμη ειδικά όταν χρησιμοποιείτε τον διερμηνέα εντολών. Για παράδειγμα, τρέξτε help(print) – αυτό θα μας δείξει τη βοήθεια για την εντολή 'print', η οποία χρησιμοποιείται για να μας εμφανίσει ότι θέλουμε στην οθόνη.

Σημείωση
Πατήστε q για να βγείτε από την βοήθεια.
Παρόμοια, μπορείτε να κατέχετε πληροφορίες για οτιδήποτε αφορά την Python. Χρησιμοποιείστε την εντολή help() για να μάθετε να χρησιμοποιείται την help. Σε περίπτωση που θέλετε βοήθεια για τους φορείς όπως το return, τότε χρειάζεσαι να βάλεις αυτά μέσα σε αγκύλες όπως την help('return') για να μην μπερδευτεί η Python για το τι θέλουμε να κάνουμε.

Περίληψη

Τώρα θα πρέπει να είστε σε θέση να γράψετε, να αποθηκεύσετε και να τρέξετε προγράμματα άνετα. Τώρα που είστε Python χρήστης, ας μάθουμε περισσότερες έννοιες της Python.

Python en: Βασικά

Απλά το να εμφανίζεις το 'Hello World' δεν είναι αρκετό, είναι; Θες να κάνεις περισσότερα από αυτό – θες να πάρεις κάποια δεδομένα να τα μετατρέψεις και να πάρεις κάποια αποτελέσματα από αυτά. Μπορείς να το πετύχεις αυτό στην Python χρησιμοποιώντας σταθερές και μεταβλητές.

Κυριολεκτικές σταθερές

Ένα παράδειγμα από κυριολεκτικές σταθερές είναι ένας αριθμός όπως 5,1.23,9.25e-3 ή ένα string όπως ' This is a string ' ή ' It's a string! '. Λέγονται κυριολεκτικές γιατί χρησιμοποιείς τις μεταβλητές κυριολεκτικά. Το νούμερο 2 πάντα αντιπροσωπεύει τον εαυτό του και τίποτα άλλο. Είναι σταθερά γιατί είναι τιμή που δεν μπορεί να αλλάξει. Ως εκ τούτου, όλα αυτά αναφέρονται σε μας ως κυριολεκτικές σταθερές.

Νούμερα

Τα νούμερα στην Python είναι τρία είδη – ακέραιων, κινητής υποδιαστολής και μιγαδικών αριθμών.

- Ένα παράδειγμα ακεραίων είναι το 2 γιατί είναι απλά ένα ολόκληρο νούμερο.
- Παράδειγμα κινητής υποδιαστολής είναι το 3.23 και 52.3E – 4. Η E σημειογραφία μας δείχνει την δύναμη του 10. Σε περίπτωση, 52.3E – 4 σημαίνει 52.3 *10-4.
- Παράδειγμα μιγαδικού αριθμού είναι (-5+4j) και (2.3 – 4.6j)

Σημείωση για έμπειρους προγραμματιστές

Δεν υπάρχει ξεχωριστός 'long int' τύπος. Ακέραιος τύπος μπορεί να γίνει οποιοσδήποτε με μεγάλη τιμή.

Χορδές(strings)

Ένα string είναι μια ακολουθία χαρακτήρων . Τα Strings είναι βασικά ένα μάτσο λέξεις . Οι λέξεις μπορεί να είναι Αγγλικές ή οποιαδήποτε άλλη γλώσσα η οποία υποστηρίζει την Unicode standard, που σημαίνει σχεδόν όλες της γλώσσες του κόσμου(http://www.unicode.org/faq/basic_1.html#16).

Σημείωση για έμπειρους προγραμματιστές

Δεν υπάρχουν “ASCII-only” strings επειδή Unicode είναι ένα υπερσύνολο από ASCII. Είναι αυστηρά ένα ASCII-encoded byte-stream χρειάζεται, μετά να χρησιμοποιείσετε str.encode(“ascii”). Για περισσότερες πληροφορίες, παρακαλώ δείτε την παρόμοια συζήτηση στο StackOverflow(http://stackoverflow.com/qouestion/175240/how-do-i-convert-a-files-format-from-uunicode-to-ascii-using-python#175270). Από την αρχή, όλα τα strings είναι Unicode.
Μπορώ να εγγυηθώ ότι θα χρησιμοποιείτε string σε κάθε Python πρόγραμμα το οποίο γράφετε, για αυτό δώστε προσοχή στα ακόλουθα κομμάτια στο πως θα χρησιμοποιείτε string στη Python.

Μονά εισαγωγικά

Μπορείς να προσδιορίσεις τα strings χρησιμοποιώντας μονά εισαγωγικά όπως 'Quote me on this'. Όλα τα λευκά είναι κενά π.χ. Κενά και καρτέλες είναι διατηρημένα ως έχει.

Διπλά εισαγωγικά

Τα strings σε διπλά εισαγωγικά δουλεύουν ακριβώς το ίδιο όπως τα strings με μονά εισαγωγικά. Ένα παράδειγμα είναι “What's your name?”

Τριπλά εισαγωγικά

Μπορείς να καθορίσετε πολλαπλές γραμμές strings χρησιμοποιώντας τριπλά εισαγωγικά - (“””ή'''). Μπορείς να χρησιμοποίησης μονά εισαγωγικά και διπλά εισαγωγικά ελεύθερα μαζί με τριπλά εισαγωγικά. Για παράδειγμα:

```
'''This is a multi – line string. This is the first line.
  This is the second line.
  “What's your name?,” I asked.
  He said “Bond, James Bond.”
'''
```

Ακολουθίες διαφυγής

Ας υποθέσουμε, ότι θέλετε να έχετε ένα “string” το οποίο περιεχέι μόνο μια εισαγωγή. Πως θα προσδιορίσετε το “string”? Για παράδειγμα, το string είναι “what's your name”?. Δεν μπορείτε να προσδιορίσετε 'what's your name?' επειδή η Python θα είναι μπερδεμένη για το που είναι η αρχή κ που το τέλος. Όποτε, θα πρέπει να προσδιορίσετε ότι τα μονά εισαγωγικά δεν υποδεικνύουν το τέλος του string. Αυτό μπορεί να γίνει με την βοήθεια αυτών που λέγονται ακολουθίες που ξεφεύγουν. Μπορείτε να προσδιορίσετε τα μονά εισαγωγικά με το \' - παρατηρήστε την κάθετη. Τώρα μπορείς να προσδιορίσετε το string όπως ' what\'s your name?'. Ένας άλλος τρόπος για να προσδιορίσετε το συγκεκριμένο string θα είναι “'what's your name?” π.χ. Χρησιμοποιώντας διπλά εισαγωγικά. Παρόμοια, πρέπει να χρησιμοποιήσεις τις ακολουθίες, που ξεφεύγουν για την χρησιμοποίηση διπλών εισαγωγικών, από μόνες τους μέσα σε διπλά εισαγωγικά string. Επίσης, πρέπει να υποδεικνύουν την κάθετη που από μονή της χρησιμοποιεί τις ακολουθίες διαφυγής \\. Και αν θέλετε να προσδιορίσετε δυο γραμμές string; Ένας τρόπος είναι να χρησιμοποιήσεις τριπλά – εισαγωγικά όπως δείξαμε πριν ή μπορείτε να χρησιμοποιήσετε την ακολουθία διαφυγής με την νέα γραμμή χαρακτήρων -\n για να ξεκινάει την νέα γραμμή. Ένα παράδειγμα είναι: This is the first line\nThis is the second line. Άλλη μια χρήσιμη ακολουθία διαφυγής είναι η καρτέλα -\t. Υπάρχουν πολλές ακολουθίες διαφυγής άλλα έχω αναφέρει μόνο τα πιο χρήσιμα εδώ. Ένα πράγμα να σημειώσουμε είναι ότι σε ένα string, μια απλή κάθετη γραμμή στο τέλος υποδηλώνει ότι το string συνεχίζεται στην επόμενη γραμμή, άλλα δεν προσθέτετε νέα γραμμή. Για παράδειγμα:

“This is the first sentence.\
This is the second sentence.”

είναι ισοδύναμο με το “This is the first sentence. This is the second sentence.”

Πρώτες χορδές(strings)

Αν θες να προσδιορίσεις μερικά strings δεν υπάρχουν ειδικές διαδικασίες όπως η ο χειρισμός της ακόλουθης διαφυγής, τότε χρειάζεται να προσδιορίσετε τα πρώτα string με το πρόθεμα r ή R στο string. Ένα παράδειγμα είναι r ”Newlines are indicated by \n”.

Τα strings είναι αμετάβλητα

Αυτό σημαίνει ότι αν μόλις έχετε χρησιμοποιήσει ένα string, δεν μπορείτε να τα αλλάξετε. Παρόλα αυτά αυτό μπορεί να είναι λάθος, πραγματικά είναι. Θα δούμε γιατί δεν είναι περιορισμός στα διάφορα προγράμματα που θα δούμε πιο μετά.

Strings κυριολεκτικής συνένωσης

Αν έχετε τοποθετήσει δυο strings κυριολεκτικά το ένα δίπλα στο άλλο, τότε αυτόματα συνενώνονται από την Python. Για παράδειγμα: 'What\'s' ' your name?' τότε αυτόματα συνενώνεται σε “What's your name?”.

Σημειώσει για c/c++ προγραμματιστές

Δεν υπάρχει ξεχωριστός char τύπου δεδομένα στην Python. Δεν είναι πραγματική ανάγκη για αυτό και είμαι σίγουρος ότι δεν θα μας λείψει.

Σημειώσει για Perl/PHP προγραμματιστές

Θυμηθείτε τα μονά – εισαγωγικά strings και τα διπλά – εισαγωγικά strings είναι τα ίδια – δεν είναι διαφορετικά σε καμιά περίπτωση.

Σημειώσει για απλούς χρήστες

Πάντα να χρησιμοποιείτε ακατέργαστο string όταν έχετε να κάνετε με κανονικές εκφράσεις. Αλλιώς, πολλά hacking μπορεί να χρειάζονται. Για παράδειγμα, μπορούν να ορισθούν σαν '\\1' ή '\1'.

Η μέθοδος Μορφοποίησης

Μερικές φορές μπορεί να θέλουμε να συντάξουμε ένα string από άλλες πληροφορίες. Εδώ είναι που η μέθοδος μορφοποίησης είναι χρήσιμη.

```
#!/ysr/bin/python
#Filename: str_format.py

age = 25
name = 'Swaroop'

print('{0} is {1} years old').format(name,age))
```

Παραγωγή:

```
$ python str_format.py
Swaroop is 25 years old
Why is Swaroop playing with that python?
```

Πως δουλεύει:
Ένα string μπορεί να χρησιμοποιεί ορισμένες προδιαγραφές και στη συνέχεια μπορεί να καλέσουμε τη μέθοδος μορφοποίησης σαν υποκατάστατο για αυτές τις προδιαγραφές με αντίστοιχα επιχειρήματα στην μέθοδο μορφοποίησης. Παρατηρήστε την πρώτη χρήση του {0} και αυτό αντίστοιχή στης μεταβλητές όνομα οι οποίες είναι τα πρώτα επιχειρήματα στη μέθοδο μορφοποίησης. Παρόμοια, οι δεύτερες προδιαγραφές είναι {1} αντιστοιχία στην ηλικία όπου είναι τα δεύτερα επιχειρήματα για την μέθοδο μορφοποίησης. Παρατηρήστε ότι θα μπορούσε να επιτευχθεί η ίδια αλληλουχία string: + ' is ' + str(age) + ' years old ' αλλά παρατηρήστε πόσο άσχημο και επιρρεπές σε λάθη είναι. Δεύτερον η συζήτηση σε string θα μπορούσε να είναι αυτόματη από τη μέθοδο μορφοποίησης αντί για τη ρητή συζήτηση εδώ. Τρίτον, όταν χρησιμοποιήσουμε την μέθοδο μορφοποίησης μπορούμε να αλλάξουμε το μήνυμα χωρίς να έχουμε να κάνουμε με την χρήση τον μεταβλητών και αντίστροφα. Την python την χρησιμοποιούμε στην μέθοδο μορφοποίησης και είναι υποκατάστατο για κάθε επιχείρημα σε τιμή ορίσματος στη θέση της προδιαγραφής. Υπάρχουν κι άλλες πληροφορίες όπως:

```
>>> '{0:.3}' .format(1/3 ) # decimal (.) precision of 3 for float '0.333'
>>> '{}:_^11}' .format('hello') # fill with underscore (_) with the text centered (^) to 11 width
'_hello_'
>>> '{name} wrote [book}'.format(name='Swaroop', book='A Byte of Python ')
# keyword-based
'Swaroop wrote Abyte of Python'
```

Πληροφορίες για την μέθοδο Format εξηγείτε στην Python No. 3101(http://www.python.org/dev/peps/pep-3101/)

Μεταβλητές

Χρησιμοποιώντας άπλα κυριολεκτικές σταθερές μπορούν να γίνουν βαρετές – χρειαζόμαστε κάποιο τρόπο για να αποθηκεύουμε οποία πληροφορία και να την χειριζόμαστε επίσης. Αυτό είναι όπου οι μεταβλητές γίνονται εικόνες. Μεταβλητές είναι ακριβώς ότι το όνομα τους υποδηλώνει – η τιμές τους μπορεί να ποικίλλουν, μπορείς να αποθηκεύεις οτιδήποτε χρησιμοποιώντας τις μεταβλητές. Μεταβλητές είναι απλά κομμάτια της μνήμης του υπολογιστή όπου αποθηκεύεις κάποιες πληροφορίες. Αντίθετα με τις κυριολεκτικές σταθερές, που χρειάζονται κάποιες μεθόδους που έχουν πρόσβαση στης μεταβλητές και ως εκ τούτου, σου δίνει τα ονόματα.

Αναγνώριση Ονόματος

Οι μεταβλητές είναι παραδείγματα αναγνώρισης. Η αναγνώριση ονόματος μας αναγνωρίζει κάτι. Υπάρχουν μερικοί κανόνες που πρέπει να ακολουθήσεις για την αναγνώριση ονόματος:

- Ο πρώτος χαρακτήρας της αναγνώρισης πρέπει να είναι ένα γραμμα από την αλφάβητα(κεφαλαία ASCII ή πεζά ASCII ή Unicode χαρακτήρες) ή κάτω παύλα('_').
- Τα υπόλοιπα από την αναγνώριση ονόματος μπορεί να θεωρηθούν γράμματα (κεφαλαία ASCII ή πεζά ASCII ή Unicode χαρακτήρες), η κάτω παύλα('_') ή τα ψηφία (0-9).
- Η αναγνώριση ονομάτων είναι ευαίσθητη υπόθεση. Για παράδειγμα, myname και myName δεν είναι τα ίδια. Παρατηρήστε το πεζό χαρακτήρα n και το N μέσα στο γραμμα.
- Παράδειγμα από έγκυρες αναγνώρισης ονόματος είναι i,_my_name,name_23,a1b2_c3 και resumAfA'A+AEAfa,-a,,cafe'aca,-aiafasA,_count.
- Παραδείγματα από μη έγκυρες αναγνώρισης ονομάτων είναι 2 πράγματα, είναι το κενώ, my-name, και “this_is_in_quotes”.

Τύποι Δεδομένων

Η μεταβλητές μπορούν να κρατούν τιμές από διάφορους τύπους ονομάζονται δεδομένα τύπων. Οι βασικοί τύποι είναι αριθμοί και strings, τα οποία τα έχουμε ήδη συζητήσει, σε προηγούμενο κεφάλαιο. Τώρα θα δούμε πως να δημιουργούμε τους δικούς μας τύπους χρησιμοποιώντας τμήματα.

Αντικείμενα

Θυμηθείτε, η Python αναφέρεται σαν αντικείμενα σε οτιδήποτε χρησιμοποιείτε στο πρόγραμμα. Αυτό συμβαίνει στην γενική έννοια. Αντί να λέμε 'the something', εμείς λέμε 'the object'.

Σημείωση για αντικειμενοστραφής χρήστες προγραμματισμού

Η Python είναι αντικειμενοστραφής με την έννοια ότι όλα είναι αντικείμενα που περιέχουν αριθμούς, strings και λειτουργίες. Τώρα θα δούμε πως να χρησιμοποιήσουμε μεταβλητές μαζί με κυριολεκτικές σταθερές. Σώστε το παρακάτω παράδειγμα και τρέξτε το πρόγραμμα.

Πως να γράψουμε τα Python προγράμματα

Πλέον, η σίγουρη διαδικασία για να σώσουμε και να τρέξουμε το πρόγραμμα Python είναι οι ακόλουθες:

1. Ανοίξτε το αγαπημένο συντάκτη.
2. Εισάγετε το κώδικα του προγράμματος που σας δίνετε από το παραδείγματα.
3. Σώστε το σαν αρχείο μαζί με το όνομα του αρχείου που αναφέρετε στα περιεχόμενα. Ακολουθούμε τη σύμβαση που έχουν όλα τα προγράμματα της Python αποθηκευμένα μαζί με την επέκταση .py.
4. Τρέξτε τον διερμηνέα μαζί με τις εντολές python program.py ή χρησιμοποιήστε IDLE για να τρέξει το πρόγραμμα. Μπορείς επίσης να χρησιμοποίησής την εκτελέσιμη μέθοδο όπως εξηγήθηκε πιο πριν.

Παράδειγμα: χρησιμοποιώντας μεταβλητές και κυριολεκτικές σταθερές

```
#Filename : var.py

i = 5
print(i)
i = i +1
print(i)
s = '''this is a multi-line string.
This is the second line.'''
print(s)
```

έξοδος:

```
$ python var.py
5
6
this is a multi-line string.
this is the second line.
```

Πως δουλεύει:
Εδώ είναι πως δουλεύει το πρόγραμμα. Πρώτα, αντιστοιχούμε τις κυριολεκτικές σταθερές τιμής 5 στις μεταβλητές i χρησιμοποιώντας τον τελεστή εκχώρησης(=). Αυτή η γραμμή ονομάζεται δήλωση γιατί δηλώνει ότι κάτι πρέπει να γίνει και σε αυτή την περίπτωση, ενώνουμε τις μεταβλητές ονόματος i στην τιμή 5. Μετά εμφανίζουμε την τιμή του i χρησιμοποιώντας την δήλωση print η οποία, όπως ήταν αναμενόμενο, απλά εμφανίζει την τιμή της μεταβλητής στην οθόνη. Μετά προσθέτουμε 1 στην τιμή που αποθηκεύτηκε στην i και την αποθηκεύει. Μετά εμφανίζουμε και αναμενόμενα, περνούμε την τιμή 6. Παρόμοια, αντιστοιχίζουμε το string στης μεταβλητές s και μετά το εμφανίζουμε.
Σημειώσει για σταθερές γλώσσες προγραμματισμού.
Η μεταβλητές χρησιμοποιούνται απλά για να αντιστοιχίσουμε τις τιμές. Καμία δήλωση ή στοιχεία τύπου ορίζονται σαν ανάγκη / χρήση.

Λογικές και φυσικές γραμμές

Μια φυσική γραμμή είναι ότι βλέπεις όταν γράφεις ένα πρόγραμμα. Μια λογική γραμμή είναι ότι βλέπει η Python σαν απλή δήλωση. Η Python εμμέσως θεωρεί ότι κάθε φυσική γραμμή αντιστοιχεί στην λογική γραμμή. Ένα παράδειγμα από λογική γραμμή είναι μια δήλωση όπως print('Hello World') – αν αυτό ήταν σε μια γραμμή από μόνο του(όπως το βλέπετε στον συντάκτη), τότε και αυτό αντιστοιχεί σε μια φυσική γραμμή. Σιωπηρά, η Python ενθαρρύνει την χρήση μιας απλής δήλωσης για κάθε γραμμή όπου κάνει τους κωδικές πιο αναγνωρίσιμους. Αν θέλετε να καθορίσετε περισσότερα από λογικές γραμμές σε μια απλή φυσική γραμμή, πρέπει να καθορίσετε ρητά ότι χρησιμοποιεί άνω τελεία (;) το οποίο υποδηλώνει το τέλος της λογικής γραμμής/δήλωση. Για παράδειγμα:

```
i = 5
print(i)
```

είναι παρόμοιο όπως

```
i = 5;
print(i);
```

και το ίδιο μπορεί να γραφτεί όπως

```
i = 5; print(i);
```

ή

```
i = 5; print(i)
```

Ωστόσο, σας συνιστώ να **γράφετε μια απλή λογική γραμμή σε μια απλή φυσική γραμμή μόνο.** Χρησιμοποιήστε περισσότερα από μια φυσική γραμμή για μια απλή λογική γραμμή μόνο αν η λογικές γραμμές είναι πραγματικά μεγάλες. Η ιδέα είναι να αποφύγουμε τις άνω κάτω τελείες όσο περισσότερο γίνετε αφού οδηγεί σε πολύ πιο αναγνώσιμο κώδικα. Στην πραγματικότητα, δεν έχω χρησιμοποιήσει ή έχω δει μια άνω κάτω τελεία σε ένα Python πρόγραμμα.
Ένα παράδειγμα για την γραφή λογικής γραμμής εκτείνεται σε πολλές φυσικές γραμμές ως εξής. Αυτό αναφέρεται στην **σαφή γραμμή που ενώνει.**

```
S= 'This is a string.\
This continues the string.'
print(s)
```

Αυτό δίνει το αποτέλεσμα:

This is a string. This continues the string.

Παρόμοια,

```
print\
(i)
```

Είναι το ίδιο με το:
print(i)
Μερικές φορές, υπαρχή μια σιωπηρή υπόθεση όπου δεν χρειάζεται να χρησιμοποιείς κάθετη γραμμή. Αυτή είναι η περίπτωση όπου η λογική γραμμή χρησιμοποιεί παρενθέσεις, αγκύλες ή εισαγωγικά. Αυτό λέγεται **σιωπηρή γραμμή που ενώνει.** Μπορείς να δεις ζωντανά όταν γράφεις ένα πρόγραμμα χρησιμοποιώντας λίστες μέσα σε λίστες θα το δούμε στα επόμενα κεφάλαια.

Οδόντωση

Τα κενά είναι σημαντικά στην Python. Στην πραγματικότητα, **τα κενά στην αρχή τον γραμμών είναι σημαντικά.** Αυτά λέγονται **οδόντωση.** Ξεκινώντας με κενά (κενά και καρτέλες) στην αρχή της λογικής γραμμής χρησιμοποιείτε για να ορισθεί το επίπεδο της οδόντωσης της λογικής γραμμής, ως αποτέλεσμα χρησιμοποιείτε για να ορισθούν οι ομαδικές δηλώσεις.
Αυτό σημαίνει ότι οι δηλώσεις που πάνε μαζί **πρέπει** να έχουν την ίδια οδόντωση. Έτσι κάθε τέτοιο σύνολο των δηλώσεων ονομάζεται **block**. Θα δούμε παράδειγμα για το πως η εμπλοκή είναι σημαντική στα επόμενα κεφάλαια. Ένα πράγμα που πρέπει να θυμάστε είναι ότι η λάθος οδόντωση μπορεί να δώσει την αύξηση των σφαλμάτων.
Για παράδειγμα:

```
i=5
print('Value is ', i) #Σφάλμα! Προσέξτε ένα διάστημα κατά την έναρξη της γραμμής
print('I repeat, the value is ', i)
```

Όταν τρέχεις αυτές τις γραμμές, θα έχεις τα παρακάτω λάθη:

```
 File “whitespace.py”, line 4
   print('Value is ',i) #Σφάλμα! Προσέξτε ένα διάστημα κατά την έναρξη της γραμμής
     ^
        IndentationErro: unexpected indent
```

Παρατηρήστε ότι υπάρχει μόνο ένα κενό στην αρχή της δεύτερης γραμμής. Το λάθος μας το υποδεικνύει η Python και μας λέει ότι η σύνταξη του προγράμματος είναι άκυρη το πρόγραμμα δεν είναι σωστά γραμμένο. Αυτό για εσάς σημαίνει ότι δεν μπορείς αυθαίρετα να ξεκινήσεις ένα νέο block για δηλώσεις(εκτός από το προεπιλεγμένο κύριο block που έχετε χρησιμοποιήσει όλα μαζί, φυσικά) Περιπτώσεις όπου μπορείς να χρησιμοποιήσεις νέα block θα είναι λεπτομερείς σε επόμενα κεφάλαια όπως να ελέγχεις την ροή του κεφαλαίου.

Πως να κάνω
Μην χρησιμοποιείτε ένα μείγμα από καρτέλες και κενά γιατί δεν δουλεύει σε διάφορες αξιόλογες πλατφόρμες. Σας συνιστώ να χρησιμοποιείτε μια απλή καρτέλα ή τέσσερα κενά για κάθε οδόντωση επιπέδου. Επιλέξτε είτε αυτά τα δυο είδη οδόντωσης. Ποιο σημαντικό, επιλέξτε μια και χρησιμοποιήστε με συνέπεια το είδος της οδόντωσης μόνο.
Σημειώστε τη στατική γλώσσα τον προγραμματιστών
Η Python πάντα χρησιμοποιεί οδόντωση για μπλοκ και δεν αφήνει πότε τιράντες. Τρέχει από _future_ εισάγει τιράντες για να μάθετε περισσότερα.

Περίληψη
Τώρα που έχουμε δει πολλές από τις λεπτομέρειες, μπορούμε να προχωρήσουμε σε πιο ενδιαφέροντα πράγματα όπως να ελέγχουμε τη ροή τον δηλώσεων. Να είστε σίγουρη ότι θα γίνει άνετο με ότι έχετε διαβάσει σε αυτό το κεφάλαιο.

Python en: Τελεστές και τις εκφράσεις
Εισαγωγή
Οι περισσότερες δηλώσεις (λογικές γραμμές) που γράφεις θα περιέχουν εκφράσεις. Ένα απλό παράδειγμα από εκφράσεις είναι το 2+3. μια έκφραση μπορεί να χωριστεί σε φορείς και τελεστές. Φορείς είναι λειτουργική για τους κάνεις και μπορούν να παρουσιαστούν με σύμβολα όπως τα δεδομένα λέγονται τελεστές. Στις περιπτώσεις 2 και 3 είναι οι φορείς.

Φορείς
Σύντομα θα δούμε τους φορείς και την χρήση τους: Σημειώστε ότι μπορείς να εκτιμήσεις τις εκφράσεις που σου δίνονται στα παραδείγματα χρησιμοποιώντας διαδραστικό διερμηνέα. Για παράδειγμα, για ελέγξουμε τοις εκφράσεις 2+3, χρησιμοποιούμε διαδραστικό διερμηνέα της Python:

```
>>>2 + 3
5
>>>3 * 5
15
>>>
```

Φορείς	Όνομα	Επεξήγηση	παράδειγμα
+	Συν	Πρόσθετη τα δυο αντικείμενα	3+5 μας δίνει 8. 'a' + 'b' μας δίνει "ab'.
-	πλην	Είτε δίνει αρνητικό αριθμό ή αφαιρεί τα δυο αντικείμενα	-5.2 μας δίνει αρνητικό αριθμό. 50 – 24 μας δίνει 26.
*	πολλαπλασιασμός	Πολλαπλασιάζει τα αντικείμενα ή μας επιστρέφει επανειλημμένα τα string.	2*3 μας δίνει 6. 'la' * 3 μας δίνει 'lalala'.
**	δύναμη	Επιστέφει x φορές την δύναμη του y	3**4 μας δίνει 81(3*3*3*3)
/	διαίρεση	Διαιρεί το χ με το y	4/3 μας δίνει 1.3333333333333.
//	Πηλίκο	Επιστρέφει το πηλίκον	4//3 μας δίνει 1
%	υπόλοιπο	Μας επιστρέφει το υπόλοιπο	8 %3 μας δίνει 2. -25,5 % 2,25 μας δίνει 1.5
<<	Αριστερή αλλαγή	Αλλάζει τα bits τον αριθμόν από τα αριστερά με τους αριθμούς τον bits που καθορίζονται(κάθε αριθμός αντικαθιστάτε στην μνήμη από bits ή δυαδικά ψηφία πχ. 0 και 1)	2<<2 μας δίνει 8.2 παρουσιάζεται σε bits από 10 σε bits. Αριστερή αλλαγή από 2 μας δίνει 1000 τα οποία μας παρουσιάζουν δεκαδικά 8.
>>	Δεξιά αλλαγή	Αλλάζει τα bits των αριθμών από τα δεξιά με τους αριθμούς τον bits που ορίζονται	11>>1 μας δίνει 5.11 μας παρουσιάζει σε bits με 1011 τα όποια όταν αλλάζει δεξιά από 1 bits μας δίνει 101 το οποίο είναι το δεκαδικό 5.

&	Bitwise and	Bitwise AND τους αριθμούς.	5&3 μας δίνει 1
|	Bit-wise or	Bitwise OR τους αριθμούς.	5 | 3 μας δίνει 7
^	Bitwise xor	Bitwise XOR τους αριθμούς.	5 ^ 3 μας δίνει 6
~	Bit-wise μετατροπή	Το bit-wise αντιστροφή του X είναι -(χ+1)	~5 μας δίνει -6
<	Λιγότερο από	Επιστρέφει όποτε χ είναι λιγότερο του y. Τελεστές σύγκρισης επιστρέφουν αληθινό και λάθος. Παρατηρήστε τα κεφάλαια τον ονομάτων.	x = 3; y = 6; x <= y μας δίνει το αληθινό.
>	Μεγαλύτερο	Μας επιστρέφει το χ είναι μεγαλύτερο y	5 > 3 μας δίνει το αληθινό. Αν και οι δυο συγκριτές είναι αριθμοί, το μετατρέπουν πρώτα σε τύπο. Αλλιώς, πάντα θα μας δίνει το λάθος
<=	Λιγότερο ή ίσο	Επιστρέφει το x όποτε είναι λιγότερο ή ίσο με το y	X= 3; y=6; x<= y επιστρέφει το αληθινό.
>=	Μεγαλύτερο ή ίσο	Επιστρέφει το χ όποτε είναι μεγαλύτερο ή ίσο με το y	X = 4; y = 3; x>= 3 επιστρέφει το αληθινό.
==	Ίσο με	Συγκρίνει αν τα δυο αντικείμενα είναι ίδια.	X=2; y=2; ==y επιστρέφει αληθινό. X= 'srt'; y= 'stR'; x== y επιστρέφει λάθος. x='srt'; y= 'srt'; x==y επιστρέφει αληθινό.
!=	Δεν είναι ίσο με το	Συγκρίνει αν τα δυο αντικείμενα δεν είναι ίδια.	X=2; y=3; x!=y επιστρέφει αληθινό.
not	Boolean NOT	Αν το χ είναι αληθινό, επιστρέφει το λάθος. Αν το χ είναι λάθος, επιστρέφει το αληθινό.	X= αληθινό; not x επιστρέφει λάθος.
and	Boolean AND	X και y επιστρέφει λάθος αν το x είναι λάθος, αλλιώς επιστρέφει την εκτίμηση του y.	X= λάθος; y = σωστό; χ και y επιστρέφει λάθος αφού το χ είναι λάθος. Στην περιπτώσει, της Python δεν θα εκτιμήσει το y αφού ξέρει ότι από το αριστερό μέρος της εκφράσεις 'and' είναι λάθος έτσι συνεπάγεται ότι όλοι η εκφράσει θα είναι λάθος ανεξάρτητα από τις άλλες τιμές. Αυτό λέγεται βραχυκύκλωμα αξιολόγησης.
Or	Boolean OR	Αν το χ είναι αληθινό, θα επιστρέφει αληθινό, αλλιώς θα επιστρέφει την αξιολόγηση του y	X= αληθινό; y= λάθος; x ή y επιστρέφει αληθινό. βραχυκύκλωμα αξιολόγησης εφαρμόζεται εδώ επίσης.

Συντόμευση για την χρήση μαθηματικών πράξεων και την εκχώρηση

Είναι σύνηθες να τρέχουμε μαθηματικές πράξεις σε μια μεταβλητή και μετά να αντιστοιχούμε τα αποτελέσματα των πράξεων πίσω στις μεταβλητές, ως εκ τούτου υπάρχει συντόμευση για τέτοιες εκφράσεις:
Μπορείς να γράψεις:

a = 2; a = a * 3

σαν:

a = 2: a *= 3

Σημειώστε ότι var = var έκφραση λειτουργίας γίνεται var =λειτουργία έκφρασης.

Για την αξιολόγηση

Αν έχετε μια έκφραση όπως αυτή 2+3*4, είναι η πρόσθεση που γίνεται πρώτη ή ο πολλαπλασιασμός ? Στα μαθηματικά του λυκείου μας λένε ότι ο πολλαπλασιασμός πρέπει να γίνεται πρώτος. Αυτό σημαίνει ότι ο πολλαπλασιασμός έχει υψηλότερη προτεραιότητα από τον τελεστή της πρόσθεσης.
Ο παρακάτω πίνακας μας δείχνει την προτεραιότητα στην Python, από την χαμηλότερη προτεραιότητα στην πιο υψηλή προτεραιότητα. Αυτό σημαίνει ότι σε μια επικείμενη έκφραση, η Python πρώτα θα εκτιμήσει τον χειριστή και της εκφράσεις χαμηλά στον πινάκα πριν από αυτά που είναι στην λίστα υψηλότερα στον πινάκα. Ο πινάκας που ακολουθεί, παίρνει από την Python το εγχειρίδιο αναφοράς(http://docs.python.org/dev/3.0/reference/expressions.html#evaluation-order), μας παρέχει για χάρη της πληρότητας. Είναι μακράν η καλύτερη να χρησιμοποιείτε παρενθέσεις στους διαχειριστές της ομάδας και τελεστές κατάλληλα προκειμένου να ορίσετε ξεκάθαρα την προτεραιότητα. Αυτό κάνει το πρόγραμμα πιο αναγνώσιμο. Δείτε αλλάζοντας τη σειρά αξιολόγησης παρακάτω για λεπτομέρειες.

Χείριστης	Περιγραφή
Λάμδα	Λάμδα εκφράσεις
Or	Boolean OR
And	Boolean AND
Not x	Boolean NOT
In, not in	Τεστ μέλους
Is, is not	Τεστ ταυτότητας
<,<=,>,>=,!=,==	Συγκρίσεις
|	Bitwise OR
^	Bitwise XOR
&	Bitwise AND
<<,>>	Αλλαγή
+,-	Πρόσθεση και αφαίρεση
*,/,//,%	Πολλαπλασιασμός, διαίρεση,διαίρεση όροφος,υπόλοιπο
+x,-x	Θετικό, αρνητικό
~x	Bitwise NOT
**	Υψωση σε δύναμη
x.attribute	Αναφορά χαρακτηριστικό
x[index]	Συνδρομή

x[index1:index2]	Τεμαχισμό
f(arguments...)	Κλήση της συνάρτησης
(expressions,...)	Δεσμευτική ή εμφάνιση πλειάδας
[expressions,...]	Οθόνη λίστας
{key:datum,...}	Λεξικό οθόνη

Τους φορείς που ακόμα δεν έχουμε συναντήσει θα τους εξηγήσουμε σε επόμενα κεφάλαια. Οι φορείς που έχουν την ίδια προτεραιότητα είναι στη λίστα στην ίδια σειρά στον παραπάνω πίνακα. Για παράδειγμα, + και – έχουν την ίδια προτεραιότητα.

Αλλάζοντας τη σειρά της αξιολόγησης

Για να κάνουμε τις εκφράσεις πιο ευανάγνωστες, μπορούμε να χρησιμοποιήσουμε παρενθέσεις. Για παράδειγμα, 2+(3*4) είναι πιο εύκολο να το καταλάβουμε από το 2+3*4 το οποίο χρειάζεται γνώση τις προτεραιότητας των τελεστών. Όπως με όλα τα άλλα, οι παρενθέσεις θα χρησιμοποιηθούν λογικά (μην το παρακάνετε) και να μην είναι περιττά (όπως π.χ. 2+(3+4)).
Υπάρχει ένα επιπλέον πλεονέκτημα με την χρήση των παρενθέσεων, βοηθάει στο να αλλάζουμε την σειρά της εκτίμησης. Για παράδειγμα, αν θέλετε επιπλέον να γίνει η πρόσθεση πριν τον πολλαπλασιασμό σε μια εκφράση, τότε μπορείτε να γράψετε κάτι σαν και αυτό (2+3)*4.

Προσαρμοστικότητα

Οι τελεστές συνήθως συνδέονται από τα αριστερά στα δεξιά, οι φορείς αξιολογούνται με την ίδια προτεραιότητα από τα αριστερά στα δεξιά. Για παράδειγμα, 2+3+4 αξιολογείται σαν (2+3)+4. Κάποιοι φορείς τους αρέσουν οι τελεστές έχουν δεξιά προς τα αριστερά ανάθεση a=b=c και αντιμετωπίζεται σαν a=(b=c).

Εκφράσεις

παράδειγμα:

```
#!/usr/bin/python
# Filename: expression.py

length = 5
breadth = 2

area = length * breadth
print('Area is',area)
print('Perimeter is',2 *(length + breadth))
```

Έξοδος:

```
$ python expression.py
Area is 10 Perimeter is 14
```

Πως δουλεύει:
Το μήκος και πλάτος του ορθογώνιο παραλληλόγραμμου αποθηκεύεται στις μεταβλητές με το ίδιο όνομα. Χρησιμοποιούμε αυτά για να υπολογίσουμε την περιοχή και την περίμετρο του ορθογώνιο παραλληλόγραμμου με την βοήθεια τον εκφράσεων. Αποθηκεύουμε τα αποτελέσματα από τις εκφράσεις length * breadth στην μεταβλητή area και την εμφανίζει χρησιμοποιώντας την εντολή print function. Στην δεύτερη περίπτωση, κατευθείαν χρησιμοποιεί τις μεταβλητές της εκφράσης 2* (length + breadth) στην print function.
Επίσης, παρατηρήστε πως η Python 'pretty-prints' στην έξοδο. Ακόμα και όταν δεν έχουμε καθορίσει το κενό μεταξύ 'Area is' και της μεταβλητή area, η Python μας την τοποθετεί για μας και έχουμε μια καλή και ωραία έξοδο και το πρόγραμμα να είναι πιο ευανάγνωστο με αυτόν τον τρόπο(αφού δεν χρειάζεται να ανησυχούμε για την απόσταση στα strings που χρησιμοποιούμε για έξοδο). Αυτό είναι ένα παράδειγμα για το πως η Python μας κάνει την ζωή πιο εύκολη για τους προγραμματιστές.

Περίληψη
Έχουμε δει πως να χρησιμοποιούμε τους φορείς, φορείς και εκφράσεις – αυτά είναι τα βασικά για να φτιάξεις ένα block για κάθε πρόγραμμα. Μετά, θα δούμε πως να χρησιμοποιούμε αυτά στα δικά μας προγράμματα χρησιμοποιώντας δηλώσεις.

Python en: Ελέγχου ροής
Στα προγράμματα που έχουμε δει μέχρι τώρα, υπάρχει πάντα μια σειρά από δηλώσεις και η Python πιστά εκτελεί αυτά με την ίδια σειρά. Και αν ήθελες να αλλάξεις την ροή για το πως δουλεύει? Για παράδειγμα, θες το πρόγραμμα να πάρει κάποιες αποφασίσεις και να κάνει διαφορετικά πράγματα ανάλογα με τις διαφορετικές καταστάσεις όπως το να εμφανίζει 'Good morning' ή 'Good evening' ανάλογα την ώρα της ημέρας?
Όπως θα έχεις φανταστεί, αυτό μπορεί να επιτευχθεί χρησιμοποιώντας τον έλεγχο ροής τον δηλώσεων. Υπάρχουν τρεις εντολές ελέγχου ροής στην Python -if, for και while.

Η “if” εντολή
Η if εντολή χρησιμοποιείται για να ελέγχει καταστάσεις και αν η κατάσταση είναι αληθινή , θα τρέξει ένα κομμάτι εντολών(ονομάζεται η if-block), αλλιώς θα εκτελέσει ένα άλλο κομμάτι εντολών (ονομάζεται else-block). Η πρόταση else είναι προαιρετική.
Παράδειγμα:

```
#!/usr/bin/python
#Filename: if.py

number = 23
guess = int(input('enter an integer: '))

if  guess == number:
        print ('congratulation,you guessed it.') #νέο μπλοκ ξεκινά εδώ
        print ('(but you do not win any prizes!)') #νέο μπλοκ τελειώνει εδώ
elif guess < number:
        print ('no, it is little higner than that') #άλλη μία ομάδα
#μπορείτε να κάνετε ό, τι θέλετε σε ένα μπλοκ....
else:
        print('no, it is a little lower than that')
#πρέπει να μαντέψεις > αριθμός να φτάσει εδώ
print('done')
#αυτή η τελευταία εντολή είναι πάντα εκτελέσιμη, μετά από την if  εντολή είναι εκτελέσιμη
```

Έξοδος:

```
$python if.py
Enter an integer : 50
no, it is little higner than that
done

$python if.py
enter an integer : 22
no, it is little higner than that
done

$python if.py
enter an integer : 23
congratulations, you gussed it.
(but you do not win any prizes!)
done
```

Πως δουλεύει:
Σε αυτό το πρόγραμμα, κάνουμε εικασίες από το χρήστη και τσεκάρουμε αν είναι ο αριθμός που έχουμε. Βάζουμε τον μεταβλητό αριθμό σε όποιον ακέραιο θέλουμε, ας που με 23. Τότε περνούμε τις εικασίες από τον χρήστη χρησιμοποιώντας την input() λειτουργία. Οι λειτουργίες είναι απλά επαναχρησιμοποιήσιμα κομμάτια από το πρόγραμμα. Λοιπόν θα διαβάσουμε περισσότερα για αυτά στα επόμενα κεφάλαια. Παρέχει μια σειρά με την ενσωματωμένη λειτουργία η οποία εμφανίζει στην οθόνη και περιμένει για εισαγωγή δεδομένων από το χρήστη. Όταν εισάγουμε κάτι και μετά πατάμε το enter κουμπί, η input() λειτουργία επιστρέφει ότι έχουμε εισάγει, σαν string. Μετά μετατρέπουμε αυτό το string σε έναν ακέραιο χρησιμοποιώντας την int και μετά αποθηκεύοντας τις μεταβλητές εικασίες. Πραγματικά, το int είναι μια τάξη άλλα το μόνο που χρειάζεται να ξέρουμε αυτή τη στιγμή είναι ότι μπορείτε να το χρησιμοποιήσουμε για να μετατρέπουμε ένα string σε έναν ακέραιο (ας υποθέσουμε ότι το string περιεχέι ένα έγκυρο ακέραιο αριθμό μέσα στο κείμενο).
Μετά, συγκρίνουμε την εικασία του χρήστη με τον αριθμό που έχουμε επιλέξει. Αν είναι ίδια, τότε εκτυπώνουμε ένα επιτυχημένο μήνυμα. Σημειώστε ότι χρησιμοποιούμε επίπεδα εσοχών για να δηλώσουμε στην python της δηλώσεις οι οποίες ανήκουν σε block. Για αυτό τα επίπεδα εσοχών είναι τόσο σημαντικά για την Python. Ελπίζω να έχετε κολλήσει με τον συνεπή κανόνα “εσοχή”. Είστε?
Σημειώστε πως η if δήλωση περιεχέι μια άνω κάτω τελεία στο τέλος – έτσι υποδεικνύουμε στην Python ότι ακολουθεί ένα block από δηλώσεις.
Μετά, ελέγχουμε αν η εικασία είναι μικρότερη από τον αριθμό, και αν είναι, ενημερώνουμε τον χρήστης κάνει μια εικασία λίγο μεγαλύτερη από αυτήν. Αυτό που έχουμε χρησιμοποιήσει εδώ είναι η εντολή elif ρήτρα η οποία στην πραγματικότητα συνδυάζει δυο if else-if else δηλώσεων μαζί με συνδυασμό if-elif-else δήλωση. Αυτό κάνει το πρόγραμμα ευκολότερο και μειώνει την ποσότητα της εσοχής που απαιτείται. Η elif και η else δήλωση πρέπει επίσης να έχουν μια άνω και κάτω τελεία στο τέλος της λογικής γραμμής που ακόλουθη από τα δικά της αντίστοιχα block από δήλωσης (με κατάλληλες εσοχές, βεβαίως). Μπορείς να έχεις μια άλλη if εντολή μέσα σε ένα if-block από if εντολές και ούτω καθεξής – αυτό ονομάζεται φωλιασμένα if εντολών. Να θυμάστε ότι όσον αφορά την elif και την else τα κομμάτια τους είναι προαιρετικά. Μια ελάχιστη έγκυρη if δήλωση είναι:

```
if True:
    print('Yes it is true')
```

Αφού η Python τελειώσει με την εκτέλεση ολόκληρης της if δήλωσης μαζί με τις συνδεδεμένες elif και else αίτιες, πηγαίνει στην επόμενη εντολή στο block που περιεχέι την if εντολή. Σε αυτήν την περίπτωση, είναι το κύριο κομμάτι του block που ξεκινά η εκτέλεση του προγράμματος και η επόμενη δήλωση είναι η print('Done') δήλωση. Μετά από αυτό, η Python βλέπει το τέλος του προγράμματος και απλά το τελειώνει.
Παρόλα αυτά είναι ένα απλό πρόγραμμα, Σας έχω πει πολλά πράγματα να προσέξετε ακόμα και σε αυτό το απλό πρόγραμμα. Όλα αυτά είναι αρκετά απλά (και εκπληκτικά απλή για όσους από εσάς που χρησιμοποιείτε C/C++) και χρειάζεται από εσάς να προσέχετε όλα αυτά τα αρχικά, αλλά μετά από αυτά, θα είστε άνετα με αυτή την γλωσσά και θα την αισθανθείτε φυσική προς σας.

Σημείωση για του προγραμματιστές της C/C++
Δεν υπάρχει switch δήλωση στην Python. Μπορείτε να χρησιμοποιείται if...elif...else δηλώσεις για να κάνει το ίδιο πράγμα (και σε μερικές υπόθεσης, χρησιμοποιήστε ένα λεξικό για να το κάνετε γρήγορα).

Η while εντολή

Η while εντολή σου επιτρέπει επανειλημμένα να εκτελείς ένα κομμάτι από εντολές όσο η συνθήκη είναι αληθινή. Η while εντολή είναι ένα παράδειγμα από αυτό που λέμε looping stetement. Η while εντολή μπορεί να έχει επιλεγμένα άλλη ρήτρα.
Παράδειγμα:

```
#!/usr/bin/python
#Filename: while.py

number = 23
running = True

while running:
        guess = int(input(' Enter an integer :'))

if guess == number:
        print ('Congratulations, you guessed it.')
        running = False # αυτό προκαλεί το βρόχο while να σταματήσει
elif guess < number:
        print('No ,it is a little higher than that.')
else:
        print('No ,it is a little lower than that.')
else:
        print('The while loop is over.')
#Κάντε οτιδήποτε άλλο θέλετε να κάνετε εδώ

print('Done')
```

Έξοδος:

```
$ python while.py
Enter an integer : 50
No,it is a little lower thn that.
Enter an integer : 22
No, it is a little higher than that.
Enter an integer : 23
Congratulations, you guessed it.
The while loop is over.
Done
```

Πως δουλεύει:
Στο πρόγραμμα, ακόμα παίζουμε το παιχνίδι με το να υποθέτουμε, αλλά το πλεονέκτημα είναι ότι ο χρήστης μπορεί να υποθέσει μέχρι να μαντέψει σωστά – δεν χρειάζεται να τρέχει ξεχωριστά το πρόγραμμα για κάθε υποθέσει που κάνει, όπως κάναμε και σε προηγούμενα τμήματα. Αυτά εφαρμόζονται με τη χρήση της δήλωσης while.
Αλλάζουμε την input και την if εντολή μέσα στην while loop και εγκαθιστούμε τις μεταβλητές να τρέξουν την αληθινή πριν την while loop. Πρώτα, ελέγχουμε αν η μεταβλητή που τρέχει είναι αληθινή και μετά προχωράμε στην εκτέλεση της αντίστοιχης while-block. Μετά αφού αυτό το block εκτελεστεί, η συνθήκη ελέγχεται όπου στην περίπτωση αυτή είναι η μεταβλητή running. Αν είναι αληθινή, εκτελείται η while-block ξανά, αλλιώς συνεχίζει να εκτελεί το προαιρετικό else-block μετά συνεχίζουμε στην επόμενη εντολή.
Η else block εκτελεί όταν η while loop συνθήκη γίνει λάθος - αυτό μπορεί να γίνει ακόμη και να είναι η πρώτη φορά που η κατάσταση ελέγχεται. Αν υπάρχει else ρήτρα για την while loop, πάντα εκτελεί εκτός αν ξεφύγουν από τον βρόχο με την εντολή break.
Τα αληθινά και λανθασμένα αποτελέσματα ονομάζονται Boolean και μπορείτε να τους εξετάσετε να είναι ισοδύναμο στο **1** και **0** αντίστοιχα.

Σημειώσει για τους χρήστες C/C++
Να θυμάστε ότι μπορείτε να έχετε μια πρόταση else για τον βρόχο while.

Η for βρόχος

Η for.. στην εντολή είναι μια άλλη εντολή για επανάληψη η οποία επαναλάβει πάνω από μια ακολουθία των αντικειμένων π.χ. περνούν από κάθε αντικείμενο σε μια ακολουθία. Θα δούμε περισσότερα για την ακολουθία με λεπτομέρειες στα επόμενα κεφαλαία. Ότι χρειάζεται να ξέρουμε τώρα είναι ότι μια ακολουθία είναι απλά μια διατεταγμένη συλλογή στοιχείων.

Παράδειγμα:

```
#!/usr/bin/python
#Filename: while.py

for  I  in ranger(1, 5):
        print(i)
else:
        print('The for loop is over')
```

Έξοδος:

```
$ python for.py
1
2
3
4
The for loop is over
```

Πως δουλεύει:
Σε αυτό το πρόγραμμα, εμφανίζουμε μια ακολουθία από αριθμούς. Παράγουμε αυτήν την ακολουθία από αριθμούς χρησιμοποιώντας την ενσωματωμένη συνάρτηση range.
Αυτό που κάνουμε εδώ είναι ότι παρέχουμε δύο αριθμούς και το εύρος μας επιστρέφει μια ακολουθία από αριθμούς ξεκινώντας από τον πρώτο αριθμό και ακολουθεί στον δεύτερο. Για παράδειγμα, το εύρος renge(1,5) μας δίνει την ακολουθία [1 2 3 4]. Από προεπιλογή, η σειρά παίρνει ένα μετρητή βημάτων από 1. Αν προμηθεύσουμε έναν τρίτο αριθμό στη σειρά, τότε αυτά γίνονται βήματα που μετράνε. Για παράδειγμα, εύρος(1,5,2) μας δίνει [1,3]. Να θυμάστε ότι το εύρος εκτείνεται μέχρι τον δεύτερο αριθμό. Δεν περιλαμβάνει τον δεύτερο αριθμό.
Η **for** βρόχος επαναλαμβάνει πέρα από το εύρος – για **I εύρος** (1,5) είναι ισοδύναμο για το **I** μέσα στο [1,2,3,4] το οποίο είναι σαν ανάθεση για κάθε αριθμό (η αντικείμενο) στην ακολουθία του **I,** ένα κάθε φορά, και μετά εκτελεί το block από τις εντολές για κάθε τιμή του **I.** Σε αυτή την περιπτώσει, απλά εκτυπώνει την τιμή μέσα στο block από της εντολές. Να θυμάστε ότι το else κομμάτι είναι προεπιλογή. Όταν περιέχονται, να εκτελείται πάντα μόλις τερματιστεί ο βρόχος for, εκτός κι αν εντολή break απαντατε.
Να θυμάστε ότι η for...in βρονχος δουλεύει για κάθε ακολουθία. Εδώ έχουμε μια λίστα από αριθμούς που παράγονται από την built-in λειτουργία, αλλά σε γενικές γραμμές μπορούμε να χρησιμοποιήσουμε οποιαδήποτε ακολουθία οποιοδήποτε αντικείμενου! Θα εξερευνήσουμε αυτήν την ιδέα με λεπτομέρειες σε επόμενα κεφάλαια.

Σημείωση για προγραμματιστές C/C++/java/C#
Η Python για την loop είναι ριζικά διαφορετική από την C/C++ για την loop. Η C# προγραμματιστές θα παρατηρήσουν ότι η for βρονχος στην Python είναι παρόμοια για κάθε for στον βρονχο της C#. Η java προγραμματιστές θα παρατηρήσουν ότι παρομοιάζετε με την for (int I : IntArry) της java 1.5.
Στην C/C++, αν θες να γράψεις την for (int I = 0; I < 5; i++), ενώ στην Python γράφεις απλά for in range(0,5). Όπως μπορείς να δεις, η for βρόχος είναι παρόμοια, πιο εκφραστική και λιγότερη επιρρεπής σε λάθη στην Python.

Η break εντολή

Η break εντολή χρησιμοποιείται για να διακόψουμε μια βρόχου εντολή π.χ. σταματά την εκτέλεση για μια βρόχου εντολή, ακόμα και όταν η βρόχου συνθήκη δεν έχει γίνει ψευδής ή η ακολουθία των στοιχείων έχει επαναληφθεί ξανά.
Μια σημαντική σημείωση είναι ότι αν διακόψουν μια for ή while βρόχο, οποιαδήποτε αντίστοιχη loop else το block **δεν** εκτελείτε.

Παράδειγμα:

```
#!/usr/bin/python
#Filename: if.py

while True:
        s = (input('Enter something :'))
        if s == 'quit':
                break
        print('Length of the string is', len(s))
print('Done')
```

Έξοδος:

```
$ python for.py
Enter something : Programming is fun
Length of the string is 18
Enter something : when the work is done
Length of the string is 21
Enter something : if you wanna make your work also fun :
Length of the string is 37
Enter something : use Python!
Lenth of the string is 12
Enter something : quit
Done
```

Πως δουλεύει:
Σε αυτό το πρόγραμμα, επανειλημμένα παίρνουμε τις εισόδους των χρηστών και εκτυπώνουμε το μήκος για κάθε μια εισαγωγή κάθε φορά. Παρέχουμε μια εξαιρετική συνθήκη για να σταματήσουμε το πρόγραμμα με το να ελέγξουμε αν οι εισαγωγή τον χρηστών είναι 'quit'. Σταματάμε το πρόγραμμα με το να βγαίνουμε από τον βρόχο και να φθάσουν στο τέλος του προγράμματος.
Το μήκος από την εισαγωγή string μπορεί να βρεθεί χρησιμοποιώντας την ενσωματωμένη συνάρτηση len.
Να θυμάστε ότι η break εντολή μπορεί να χρησιμοποιηθεί μαζί με τον βρόχο επίσης.

Swaroop's Poetic Python

Για την εισαγωγή έχω χρησιμοποιήσει εδώ ένα μικρό ποίημα αυτό που έχω γράψει ονομάζεται **Swaroop's Poetic Python**:

```
Programming is fun
When the work is done
if you wonna make your work alse fun:
        use Python!
```

Η continue εντολή

Η continue εντολή χρησιμοποιείται για να πει στην Python να παραβλέψει τα υπόλοιπα από τις εντολές στον τρέχον βρόχο block και στην continue και στην επόμενη επανάληψη του βρόχου.

Παράδειγμα:

```
#!/usr/bin/python
#Filename: if.py

while True:
        s = input('Enter something :')
        if s == 'quit':
                break
        if len(s)< 3:
                print('Too small')
                continue
        print('Input is of sufficien length')
#Κάντε άλλα είδη της επεξεργασίας εδώ...
```

Έξοδος:

```
$ python for.py
Enter something : a
Too small
Enter something : 12
Too small
Enter something : abc
Input is of sufficient length
Enter something : quit
```

Πως δουλεύει:
Σε αυτό το πρόγραμμα, δεχόμαστε εισαγωγές δεδομένων από τον χρήστη, αλλά τις επεξεργαζόμαστε μόνο αν είναι τουλάχιστον 3 χαρακτήρες. Οπότε, χρησιμοποιούμε την len λειτουργία για να δούμε το μήκος και αν το μήκος είναι μικρότερο από 3 χαρακτήρες, παραλείπουμε τις υπόλοιπες εντολές μέσα στο block με το να χρησιμοποιούμε την εντολή continue. Αλλιώς, οι υπόλοιπες εντολές μέσα στην loop εκτελούνται και μπορούμε να κάνουμε οποιαδήποτε επεξεργασία θέλουμε να κάνουμε εδώ.
Σημειώστε ότι η εντολή continue δουλεύει μαζί με τον for βρόχο επίσης.

Περίληψη

Μέχρι τώρα έχουμε δει πως να χρησιμοποιούμαι τις 3 εντολές ελέγχου τον καταστάσεων – **if**,**while** και **for** μαζί με αυτές συνδέονται οι break και continue. Αυτές είναι μερικές από τις πιο συχνές χρήσεις κομματιών στην Python ως εκ τούτου, να γίνεται άνετη με αυτές μέχρι το τέλος.
Μετά, θα δούμε πως να δημιουργήσουμε και να χρησιμοποιήσουμε λειτουργίες της.

Python en: Λειτουργίες

Εισαγωγή
Οι λειτουργίες είναι επαναχρησιμοποιημένα κομμάτια από προγράμματα. Σε αφήνουν να δώσεις ένα όνομα στο block με τις εντολές και μπορείς να τρέξεις αυτό το block χρησιμοποιώντας αυτό το όνομα οπουδήποτε μέσα στο πρόγραμμα και όσες φορές θες. Αυτό είναι γνωστό ως καλώντας την function. Έχουμε ήδη χρησιμοποιήσει πολλές ενσωματωμένες function όπως η len και renge.
Η ιδέα function είναι πιθανών η πιο σημαντική για την δημιουργία block σε οπουδήποτε μη τετριμμένο λογισμικό(σε οποιαδήποτε γλώσσα προγραμματισμού), οπότε θα εξερευνήσουμε διάφορες πτυχές της function σε αυτό το κεφάλαιο.
Οι λειτουργίες **def**ined χρησιμοποιούναι τη **def** λέξη κλειδί. Αυτό ακολουθείτε από ένα αναγνωριστικό όνομα για την λειτουργία που ακολουθείτε μαζί με ένα ζευγάρι από παρενθέσεις οι οποίες μπορεί να επισυνάπτουν μερικά ονόματα από μεταβλητές και οι γραμμές τελειώνουν με άνω και κάτω τελεία. Μέτα ακολουθούν οι δηλώσεις στο block τα οποία είναι κομμάτια από αυτές τις λειτουργίες. Ένα παράδειγμα θα σου δείξει ότι αυτό είναι πολύ απλό:

Παράδειγμα:

```
#!/usr/bin/python
#Filename: if.py

def sayHello()
        print('Hello World') #το block ανήκει στην function
#τέλος της function
sayHello() # καλέστε την function
sayHello() # καλέστε την function ξανά
```

Έξοδος:

```
$ python function.py
Hello World!
Hello World!
```

Πως δουλεύει:
Θα καθορίσουν μια function που ονομάζεται sayHello χρησιμοποιώντας την σύνταξη όπως εξηγήθηκε πιο πριν. Αυτή η function δεν παίρνει παρενθέσεις και ως εκ τούτου, δεν υπάρχουν μεταβλητές μέσα στις παρενθέσεις.
Οι παραμέτροι για να λειτουργήσει είναι απλά η είσοδος στη λειτουργία έτσι ώστε να μπορεί να περάσει σε διαφορετικές τιμές για να δεχτούμε και τα αντίστοιχα αποτελέσματα. Παρατηρήστε ότι μπορούμε να καλέσουμε μια function δυο φορές το οποίο σημαίνει ότι δεν θα χρειαστεί να το γράψεις τον ίδιο κώδικα ξανά.

Παραμέτρους function

Μια function μπορεί να πάρει παραμέτρους, οπού οι τιμές που διοχετεύεις στην function ώστε η function να μπορεί να κάνει κάτι αξιοποιώντας αυτές τις τιμές. Αυτές οι παράμετροι είναι ακριβώς όπως οι μεταβλητές εκτός από το ότι οι τιμές αυτών των μεταβλητών ορίζονται όταν καλούμε τη συνάρτηση και έχουν ήδη εκχωρηθεί τιμές όταν τρέχει η συνάρτηση.
Παράμετροι καθορίζονται μέσα στο ζευγάρι των παρενθέσεων στον ορισμό της συνάρτησης, χωρίζονται με κόμμα. Όταν καλούμε την function, παρέχουμε τις τιμές με τον ίδιο τρόπο.
Σημειώστε την ορολογία που χρησιμοποιείται – τα ονόματα που μας δίνονται στον ορισμό της συνάρτησης ονομάζεται **parameters** όπου οι τιμές που παρέχουμε στην function call ονομάζονται επιχειρήματα (arguments).

Παράδειγμα:

```
#!/usr/bin/python
#Filename: func_param.py

def printMax(a,b):
        if a>b:
                print(a, 'is maximum')
        elif a == b:
                print(a, 'is eqal to', b)
        else:
                print(b, 'is maximum')
printMax(3,4) # άμεσα μας δίνει τις κυριολεκτικές τιμές

x = 5
y = 7

printMax(x, y) #δινει μεταβλητές σαν ορίσματα
```

Έξοδος:

```
$ python func_param.py
4 is maximum
7 is maximum
```

Πως δουλεύει:
Εδώ, ορίζουμε την function που ονομάζεται printMax οπού περνούμε δυο παραμέτρους που ονομάζεται a και b. Ανακαλύψαμε τα μεγαλύτερα νούμερα χρησιμοποιώντας απλά την if...else εντολή και μετά εμφανίζουμε τον μεγαλύτερο αριθμό.
Για αρχή χρησιμοποιούμε την printMax, προμηθεύουμε κατευθείαν τους αριθμούς π.χ. επιχειρήματα. Για την δεύτερη χρήση, την ονομάζουμε function με την χρήση τον μεταβλητον. printMax(x,y) προκαλει την τιμή του ορίσματος x για να δωθεί στην παράμετρο a και η αξία του επιχειρήματος y για να ανατεθεί η παράμετρος b. Η printMax function δουλεύει το ίδιο και στις δυο περιπτώσεις.

Τοπικές μεταβλητές

Όταν δηλώνω μεταβλητές μέσα σε ένα ορισμό function, δεν είναι συγγενεις με οποιοδήποτε τρόπο με άλλες μεταβλητές με τα ίδια ονόματα που χρησιμοποιούνται έξω από την function π.χ. τα ονόματα των μεταβλητών χρησιμοποιούνται μόνο τοπικά στη function. Αυτό ονομάζεται πεδίο της μεταβλητής. Όλες οι μεταβλητές έχουν πεδίο του block που έχουν δηλωθεί, αρχίζοντας από το σημείο του ορισμού του ονόματος.
Παράδειγμα:

```
#!/usr/bin/python
#Filename: func_local.py

x = 50

def func(x):
        print('x is', x)
        x = 2
        print('Changed local x to', x)

func(x)
print('x is still', x)
```

Έξοδος:

```
$ python func_local.py
x is 50
Changed local x to 2
x is still 50
```

Πως δουλεύει:

Στην function, την πρώτη στιγμή που χρησιμοποιούμε την τιμή του ονόματος x , Η Python χρησιμοποιεί την τιμή τον παραμέτρων που δηλώνονται στην function.
Μετά, αντιστοιχούμε την τιμή 2 με το x. Το όνομα x είναι τοπικό στην δική μας function. Οπότε, όταν αλλάζουμε την τιμή του x στην function, το x που ορίζεται στο κεντρικό block παραμένει ανεπηρέαστο.
Στο τελευταίο κάλεσμα print function, εμφάνιζουμε την τιμή του x στο κεντρικό block και επιβεβαιώνει ότι στην πραγματικότητα είναι ανεπηρέαστη.

Χρησιμοποιώντας την παγκόσμια δήλωση

Αν θέλετε να αντιστοιχίσετε μια τιμή σε ένα όνομα το ορίζετε στην αρχή του προγράμματος (π.χ. Όχι μέσα σε οποιοδήποτε είδος του πεδίου εφαρμογής όπως μια function ή classes), μετά πρέπει να πείτε στην Python ότι το όνομα δεν είναι τοπικό, άλλα είναι παγκόσμια. Και το κάνουμε αυτό με την χρήση της παγκόσμιας δήλωσης. Είναι πιθανόν για να αντιστοιχίσετε μια τιμή σε μια μεταβλητή για να ορίζεται έξω από την function χωρίς την παγκόσμια δήλωση.
Μπορείτε να χρησιμοποιήσετε τιμές τέτοιων μεταβλητών που ορίζονται έξω από την συνάρτηση (ας υποθέσουμε ότι δεν υπάρχει μεταβλητή με το ίδιο όνομα μέσα στην συνάρτηση). Ωστόσο, αυτό δεν ενθαρρύνεται και θα πρέπει να αποφεύγεται δεδομένου ότι καθίσταται σαφές για τον αναγνώστη του προγράμματος ως προς το πού είναι ο ορισμός μεταβλητών. Χρησιμοποιώντας την παγκόσμια δήλωση γίνεται ξεκάθαρο ότι η μεταβλητή αυτή ορίζεται σε ένα εξωτερικό τμήμα εντολών.

Παράδειγμα:

```
#!/usr/bin/python
# Filename: func_global.py

x = 50

def func():
    global x

    print ('x is', x)
    x =2
    print('Change global x to ', x)

func()
print('Value of x is', x)
```

Έξοδος:

```
$ python func_global.py
x is 50
Changed global x to 2
Value of x is 2
```

Πως δουλεύει:
Η παγκόσμια δήλωση χρησιμοποιείται για να δηλώσουμε ότι x είναι παγκόσμια μεταβλητή - ως εκ τούτου, όταν αντιστοιχίσετε μια τιμή του x μέσα στην συνάρτηση, η αλλαγή αυτή αντικατοπτρίζεται όταν χρησιμοποιούμε την τιμή του x στο κύριο κομμάτι του block.
Μπορείτε να προσδιορίσετε περισσότερες από μια παγκόσμια μεταβλητές χρησιμοποιώντας την ίδια παγκόσμια δήλωση. Για παράδειγμα, παγκόσμια x, y, z.

Με τη χρήση της εντολής nonlocal

Έχουμε δει πως να έχουμε πρόσβαση σε μεταβλητές στην τοπική και στην παγκόσμια εμβέλεια πιο πριν. Υπάρχει ένα άλλο πεδίο εφαρμογής που ονομάζεται “nonlocal” τα οποία είναι μεταξύ των δυο τύπων των πεδίων. Το Nonlocal πεδίο είναι παρατηρητής όπου ορίζεις την λειτουργία μέσα στην λειτουργιά.
Εφόσον όλα στην Python είναι εκτελεστικός κώδικας, μπορείς να ορίσεις την λειτουργιά οπουδήποτε.
Ας κάνουμε ένα παράδειγμα:

```
#!/usr/bin/python
#  Filenema: func_nonlocal.py

def func_outer():
        x = 2
        print('x is',x)

def func_inner():
        nonlocal x
        x = 5
func_inner()
print('Change local x to ', x)

func_inner()
```

Έξοδος:

```
$ python func nonlocal.py
x is 2
Change local x to 5
```

Πως δουλεύει:
Όταν είμαστε μέσα στην func_inner, το 'x' ορίζετε στην πρώτη γραμμή του func_outer σχετικά ούτε στην τοπική ούτε σε παγκόσμια εμβέλεια. Δηλώνουμε ότι χρησιμοποιούμε αυτό το x με nonlocal x και ως εκ τούτου, έχουμε πρόσβασή στην μεταβλητή.
Δοκιμάστε να αλλάξετε την nonlocal x στην παγκόσμια x και επίσης αφαιρέστε τις δηλώσεις itself και παρατηρήστε την διάφορα της συμπεριφοράς σε αυτές τις δυο υποθέσεις.

Προεπιλεγμένες τιμές ορίσματος

Για κάποιες function, μπορεί να χρειαστεί να κάνεις κάποιες από τις παραμέτρους προαιρετικές και να χρησιμοποιήσεις προεπιλεγμένες τιμές αν ο χρήστης δεν θέλει να παρέχει τιμές για αυτές τις παραμέτρους. Αυτό γίνετε με την βοήθεια του ορίσματος των προεπιλεγμένων τιμών. Μπορείτε να καθορίσετε προεπιλεγμένες τιμές ορίσματος για τις παραμέτρους με το να ακολουθήσεις τα ονόματα των παραμέτρων στον ορισμό της function μαζί με τον φορέα ανάθεσης (=) που ακολουθείτε από την προκαθορισμένη τιμή.
Σημειώστε ότι η προκαθορισμένη τιμή ορίσματος θα είναι μια σταθερά. Ακριβέστερα, η προκαθορισμένη τιμή ορίσματος θα είναι αμετάβλητη – αυτό εξηγείται σε επόμενα κεφάλαια. Προς το παρόν άπλα να θυμάστε αυτό.

Παράδειγμα:

```
#!/usr/bin/python
#Filename: func_default.py

def say(massage, times = 1):

say('Hello')
say('World',5)
```

έξοδος:

```
$ python finc_default.py
Hello
WorldWorldWorldWorldWorld
```

Πως δουλεύει:
Στην function το όνομα 'say' χρησιμοποιείται για να εμφανίσει το string όσες φόρες έχει ορισθεί. Αν δεν του καταχωρήσουμε μια τιμή, από παράλειψη, τότε το string θα εμφανιστεί μια φορά. Το πετυχαίνουμε αυτό με το να προσδιορίσουμε μια προεπιλογή με την τιμή **1** στους παραμέτρους. Στην πρώτη χρήση του 'say', έχουμε προμηθεύσει μόνο το string και εμφανίζει μόνο μια φορά το string. Στην δεύτερη χρήση του 'say', έχουμε προμηθεύσει και τα δυο string και ένα επιχείρημα 5 ξεκινώντας ότι θέλουμε το say string μήνυμα να εμφανιστεί 5 φορές.
Σημαντικό
Μονό αυτές οι παράμετροι οι οποίες είναι στο τέλος της λίστας των παραμέτρων μπορούν να δωθούν με προεπιλεγμένες τιμές ορίσματος π.χ. Δεν μπορείς να έχεις παραμέτρους μαζί με προεπιλεγμένες τιμές ορισμάτων πριν την παράμετρο χωρίς μια προεπιλεγμένη τιμή ορίσματος κατά σειρά παραμέτρων που δηλώνονται στις παραμέτρους λειτουργίας.
Και αυτό γίνεται γιατί η τιμή έχει ανατεθεί στην παράμετρο με την θέση. Για παράδειγμα, def func(a, b=5) είναι έγκυρη, άλλα def func(a=5, b) δεν είναι έγκυρη.

Ορίσματα με λέξεις κλειδιά

Αν έχεις κάποιες function με πολλές παραμέτρους και θες να καθορίσεις μόνο μερικές από αυτές, τότε μπορείς να δώσεις τιμές για αυτές τις παραμέτρους με το να τις ονοματίσεις – αυτό ονομάζεται keyword arguments(ορίσματα με λέξεις κλειδιά) – χρησιμοποιούμε το όνομα (keyword) αντί της θέσης(που χρησιμοποιούσαμε ανέκαθεν) για να προσδιορίσουμε τα ορίσματα στην function. Υπάρχουν δυο πλεονεκτήματα – το ένα είναι να χρησιμοποιούμε την function είναι πιο εύκολο αφού δεν χρειαζόμαστε να ανησυχούμε σχετικά με τη σειρά των ορισμάτων. Δεύτερο, μπορούμε να δώσουμε τιμές μόνο σε εκείνες τις παραμέτρους τις οποίες θέλουμε, υπό την προϋπόθεση ότι οι άλλες παράμετροι έχουν προκαθορισμένες τιμές-επιχειρήματα.
Παράδειγμα:

```
#!/usr/python
# Filename: func_key.py

def func(a, b=5, c=10):
        print('a is', a, 'and b is', b, 'and c is', c)
func(3, 7)
func(25, c=24)
func(c=50, a=100)
```

Έξοδος:

```
$ python func_key.py
a is 3 and b is 7 and c is 10
a is 25 and b is 5 and c is 24
a is 100 and b is 5 and is c is 50
```

Πως δουλεύει:
Η function ονομάζεται func και έχει μια παράμετρο χωρίς προεπιλεγμένες τιμές ορίσματος, που ακολουθούνται από δυο παραμέτρους μαζί με προεπιλεγμένες τιμές ορίσματος.
Στην πρώτη χρήση, func(3, 7), η παράμετρος a παίρνει την τιμή 3. η παράμετρος b παίρνει την τιμή 7 και το c παίρνει την τιμή του 10.
Στην δεύτερη χρίση func(25, c=24), οι μεταβλητές a περνούν την τιμή του 25 λόγο της θέσης τον ορισμάτων. Τότε, οι παράμετροι c περνάει την τιμή του 24 λόγο τον ονομάτων π.χ. Kayword arguments. Οι μεταβλητές b περνούν την τιμή του 5.
Στην τρίτη χρίση func(c=50, a=100), χρησιμοποιούμε keyword arguments αποκλειστικά για να ορίσουμε της τιμές. Παρατηρήστε, ότι προσδιορίζουμε την τιμή για την παράμετρο c πριν από αυτήν για μια ακόμα και αν η a είναι ορισμένη πριν από τη c στον ορισμό της συνάρτησης.

Παράμετροι VarArs

Να κάνω

Θα έπρεπε να γράψω για αυτό σε επόμενο κεφάλαιο αφού δεν έχουμε μιλήσει για τις λίστες και για τα λεξιλόγια ακόμα.
Κάποιες φόρες μπορεί να θες να ορίσεις μια function ότι μπορεί να πάρει οποιοδήποτε νούμερο από παραμέτρους, αυτό μπορεί να γίνει εφικτό με το να χρησιμοποιείς το αστέρι(*).

```
#!usr/bin/python
# Filename:total.py

def total(initial=5, *numbers, **keywords):
        count = initial:
        for number in numbers:
                count += number
        for key  in keywords:
                count += keywords[key]
        return count

print(total (10, 1, 2, 3, vagetables=50, fruits=100))
```

Έξοδος:

```
$ python total.py
166
```

Πως δουλεύει:
Όταν δηλώνουμε μια πρωταγωνιστική παράμετρο όπως *param, τότε όλα τα ορίσματα από εκείνο το σημείο μέχρι το τέλος συλλέγονται σαν μια λίστα που ονομάζεται 'param'.
Παρόμοια, όταν δηλώνουμε μια διπλή πρωταγωνιστική παράμετρο όπως **param, τότε όλες οι λέξεις κλειδιά ορισμάτων από εκείνο το σημείο μέχρι το τέλος συλλέγονται σαν λεξικό που ονομάζεται "param”.
Θα τα εξερευνήσουμε τις λίστες σε επόμενα κεφάλαια.

Λέξεις-κλειδιά μόνο παραμέτρους

Αν θέλουμε να προσδιορίσουμε ορισμένες λέξεις-κλειδιά παραμέτρους για να είναι διαθέσιμες σαν λέξεις κλειδιά μόνο και όχι σαν ορίσματα, μπορεί να δηλώνονται μετά από μια αρχική παράμετρο:

```
#!/usr/bin/python
# Filename: keyword_only.py

def total(initial=5, *numbers, vegetables):
        count = initial:
        for number in numbers:
```

```
            count += number
        count += vegetables
        return count
print (total(10, 1, 2, 3, vegetables= 50))
print(total (10, 1, 2, 3))
# Είναι λάθος γιατί δεν έχουν προμηθεύσει μια προεπιλεγμένη τιμή ορίσματος
for 'vegatables'
```

Έξοδος:

```
$ Python keyword_omly.py
66
Traceback (most recent call last):
File “test.py”, line 12, in <mudule>
print(total(10, 1, 2, 3))
TypeError: total() needs keyword-only argument vegetables
```

Πως δουλεύει:
Δηλώνοντας παραμέτρους μετά την αρχική παράμετρο αποτελεσμάτων σε λέξεις κλειδιά ορισμάτων. Αν αυτά τα επιχειρήματα δεν παρέχονται από προκαθορισμένη τιμή, μετά καλεί την function που θα δείχνει το λάθος αν η λέξεις-κλειδιά των επιχειρημάτων δεν παρέχονται, όπως έχουμε δει παραπάνω.
Αν θες να έχεις λέξεις-κλειδιά μόνο τις παραμέτρους άλλα δεν έχεις ανάγκη για μια αρχή παραμέτρων, τότε απλά θα χρησιμοποιήσουμε ένα άδειο αστέρι χωρίς τη χρήση οποιουδήποτε ονόματος όπως def total(initial = 5, *, vegetables).

Η δήλωση return

Η δήλωση επιστροφής χρησιμοποιείτε για να επιστρέφει από μια function π.χ σπάζοντας από μια function. Μπορούμε προαιρετικά να επιστρέψουμε μια τιμή στην function επίσης.
Για παράδειγμα:

```
#!/usr/bin/python
#Filename: func_return.py

def maximum(x,y):
        if x > y :
                return x
        else:
                return y
print(maximum(2, 3))
```

Έξοδος:

```
$python func_return.py
3
```

Πως δουλεύει:
Το maximum function επιστρέφει την μεγίστη τιμή από τις παραμέτρους, σε αυτή την περίπτωση οι αριθμοί προμηθεύονται στην function. Χρησιμοποιεί μια απλή if..else δήλωση για να βρει την μεγαλύτερη τιμή και να την επιστρέψει.
Σημειώστε ότι η return δήλωση χωρίς τιμή είναι ισοδύναμη με την return None. None είναι ένας ειδικός τύπος στην Python ο οποίος αντιπροσωπεύει την ανυπαρξία. Για παράδειγμα, συνηθίζεται να υποδεικνύουν μια μεταβλητή ότι δεν έχει τιμή αν έχει μια τιμή None.
Οποιαδήποτε function σιωπηρά περιεχέι μια δήλωση return None στο τέλος έκτος αν έχεις γράψει την δικιά σου retyrn δήλωση. Αυτό μπορείς να το δεις με το να τρέξεις την running

print(someFunction()) οπού η function someFunction δεν χρησιμοποιεί την return δήλωση όπως:

```
def someFunction():
        pass
```

Η pass δήλωση χρησιμοποιείτε στην python για να υποδείξει ένα άδειο block από δήλωσης.
Σημείωση
Υπάρχει μια built-in function που ονομάζεται max όπου ήδη υλοποιεί την 'find maximum' λειτουργικότητα της, όποτε χρησιμοποιείται αυτήν την built-in function όποτε αυτό είναι δυνατό.

DOCStrings

Η Python έχει ένα φίνο χαρακτηριστικό ονομάζεται documentation strings, επακριβώς αναφέρεται με το μικρότερο του όνομα docstrings. Το DocStrings είναι ένα σημαντικό εργαλείο το οποίο πρέπει να το χρησιμοποιείτε εφόσον βοηθάει για να τεκμηριώνει το πρόγραμμα καλύτερα και να το κάνει ποιο εύκολο για να το καταλάβουμε. Καταπληκτικά, μπορούμε επίσης να έχουμε το docstring, λέγοντας function, όταν το πρόγραμμα πραγματικά τρέχει!
Παράδειγμα:

```
#!/usr/bin/python
# Filename: func_doc.py

def printMax(x, y):
'''Prints the maximum of two numbers.

The two values muat be integers.'''
x= int(x) # convert to integers, if possible
y= int(y)

if x > y:
        print(x, 'is maximum')
else:
        print(y, 'is maximum')
printMax(3,5)
print(printMax._doc_)
```

Έξοδος:

```
$ Python func_doc.py
5 is maximum
Print the maximum of two numvers.

        The two values must de integers.
```

Πως δουλεύει:
Ένα string στην πρώτη γραμμή λογικής σε ένα function είναι το docstrings για μια function. Σημειώστε ότι το docstrings επίσης εφαρμόζεται σε ενότητες και σε κατηγορίες που θα μάθουμε στα αντίστοιχα κεφάλαια. Η σύμβαση που ακολουθεί ένα docstring είναι ένα multi-line string όπου είναι οι πρώτες γραμμές που ξεκινάνε με κεφάλαιο γραμμα και τελειώνουν με μια τελεία. Μετά η δεύτερη γραμμή είναι λευκή όπου ακολουθείτε από κάθε λεπτομερή εξήγηση ξεκινώντας από την τρίτη γραμμή. Σου συνιστώ να ακολουθήσεις αυτή την συνθήκη για όλα τα docstrings για όλα τα non-trivial functions.
Μπορείτε να έχετε πρόσβαση στο docstrings για την printMax function χρησιμοποιώντας the__doc__(παρατηρήστε την διπλή κάτω παύλα) χαρακτηριστικό (το όνομα ανήκει σε) της function. Απλά να θυμάστε ότι η Python τα μεταχειρίζεται όλα σαν να είναι αντικείμενο και αυτό περιλαμβάνει και την function. Λοιπόν μάθαμε περισσότερα για τα αντικείμενα σε κεφάλαια για τις κλάσης. Αν έχετε χρησιμοποίησή το help() στην Python, τότε έχετε δει ήδη την χρίση τον

docstrings! Αυτό που κάνει είναι να φέρνει το the_doc_attribute της function και να το δείχνει με ένα τακτοποιημένο τρόπο για εσένα. Μπορείς να το χρησιμοποίησής στην function πιο πάνω – απλά προσθέτοντας το help(printMax) στο πρόγραμμα σας. Να θυμάστε να πατήσετε το q κουμπί για να βγείτε από την βοήθεια. Αυτοματοποιημένα εργαλεία μπορούν να ανακτήσουν τεκμηρίωσης από το πρόγραμμά σας με αυτόν τον τρόπο. Και για αυτό, συνιστώ να χρησιμοποιείται τα docstrings για κάθε μη τετριμμένη function που γράφεις. Η pydoc εντολή που είναι μαζί με την Python δουλεύει παρόμοια με την help() χρησιμοποιώντας τα docstrings.

Σχολιασμός

Η Function έχει ένα προχωρημένο χαρακτηριστικό που ονομάζεται annotations η οποία είναι ένας ωραίος τρόπος στο να προσθέτεις πληροφορίες για κάθε παράμετρο όπως η return τιμή. Εφόσον η γλώσσα της Python από μονής της δεν ερμηνεύει αυτές της annotations σε οποιοδήποτε δρόμο (αυτή η functionality έχει αφήσει τις τρίτες βιβλιοθήκες που ερμηνεύει με οποιοδήποτε τρόπο.) θα παραλείψουμε αυτό το χαρακτηριστικό από τη συζήτησή μας. Αν ενδιαφέρεστε να διαβάσετε για τις annotetions, παρακαλώ δείτε την πρόταση της Python No. 3107 (http://www.python.otg/dv/peps/pep-3107).

Περίληψη

Έχετε δει τόσες πλευρές τις function άλλα σημειώστε ότι ακόμα δεν έχουμε καλύψει όλες τις πτυχές. Ωστόσο, έχουμε καλύψει πολλές από ότι θα χρησιμοποιήσουμε για την Python function σε καθημερινή βάση. Μετά θα δούμε πως να χρησιμοποιούμε και να δημιουργούμε τα πρότυπα της Python.

Python en: πρότυπα

Εισαγωγή

Έχετε δει πως να επαναχρησιμοποιείτε τον κώδικα μέσα στο πρόγραμμα με το να ορίσετε την function μια φορά. Και αν θέλουμε να επαναχρησιμοποιείσουμε έναν αριθμό από την function σε άλλα προγράμματα που γράφουμε ? Όπως θα έχετε ήδη μαντέψει, η απάντηση είναι με τα πρότυπα. Υπάρχουν διάφορες μεθόδους για την γράφει πρότυπον, αλλά είναι ο πιο απλός τρόπος για την δημιουργία ενός αρχείου .py επέκτασης που περιέχει function και μεταβλητών. Άλλη μέθοδος είναι να γράψουμε τα πρότυπα στην ντόπια γλώσσα στην οποία η Python ερμηνεύει από μόνη της τη γραφή. Για παράδειγμα, μπορείτε να γράψετε πρότυπα μέσα στη C γλωσσά προγράμματος (http://docs.python.org/extending/) και όταν καταρτίζονται, μπορούμε να χρησιμοποιήσουμε από τον κώδικα Python όταν χρησιμοποιούμε τον κλασσικό διερμηνέα Python. Ένα πρότυπο μπορεί να εισάγεται από ένα άλλο πρόγραμμα για να κάνει τη χρήση της λειτουργικότητάς του. Έτσι μπορούμε να χρησιμοποιήσουμε την Python από την βασική βιβλιοθήκη επίσης. Πρώτα θα δούμε πως να χρησιμοποιήσουμε την κλασική βιβλιοθήκη των προτύπων.
Παράδειγμα:

```
#!/usr/bin/python
#Filename: using_sys.py

import sys

print('the command line arguments are:')
for i in sys.argv:
        print(i)
print('\n\n the PYTHONPATH is', sys.path,'\n')
```

Έξοδος:

```
$ python using_sys.py ew are arguments
The command line arguments are:
using_sys.py
```

we
are
arguments

The PYTHONPATH is ['', ' C:\\windows\\system32\\python30.zip','
C:\\python30\\DLLs', 'C:\\Python30\\lib',
'C:\\Python30\\lib\\plat-win' 'C:\\python30',
'C:\\python30\\lib\\site-packages']

Πως δουλεύει:
Πρώτα, εισάγουμε τα sys πρότυπα χρησιμοποιώντας τις δηλώσεις εισαγωγής. Βασικά, αυτό μεταφράζει σε εμάς, λέγοντας στην Python ότι θέλουμε να χρησιμοποιήσουμε αυτό το πρότυπο. Αυτό το sys πρότυπο περιεχέι λειτουργικότητα παρεμφερή με αυτή του διερμηνέα της Python και του περιβάλλοντος π.χ. Του συστήματος. Όταν η Python εκτελεί την import sys δήλωση, ψάχνει για το sys πρότυπο. Σε αυτή την περίπτωση, είναι μια από τα ενσωματωμένα πρότυπα, και αυξάνει τη γνώση της Python για να την βρει. Αν δεν καταρτίζονται από πρότυπα π.χ. Ένα πρότυπο γράφει στην python, μετά ο διερμηνέας της python θα ψάξει μέσα στης λίστες των καταλόγων στις μεταβλητές sys.path. Αν τα πρότυπα βρεθούν, μετά οι δηλώσεις στο κορμό των προτύπων τρέχουν και μετά τα πρότυπα είναι έτοιμα για εσένα να τα χρησιμοποιήσεις. Σημειώστε ότι η αρχικοποίηση είναι έτοιμη μόνο τον πρώτο χρόνο που εισάγουμε ένα πρότυπο. Η argv μεταβλητή στα sys πρότυπα χρησιμοποιούν τον συμβολισμό με τελείες. π.χ. Sys.argv. Ξεκάθαρα μας δηλώνει ότι το όνομα είναι η διαδρομή των sys προτύπων. Άλλο ένα πλεονέκτημα από αυτήν την προσέγγιση είναι ότι το όνομα δεν συγκρούεται με καμιά argv μεταβλητή χρησιμοποιώντας το μέσα στο πρόγραμμα. Η sys.argv μεταβλητή είναι μια λίστα από strings(η λίστα εξηγείτε σε επόμενα κεφάλαια.) συγκεκριμένα, η sys.argv περιεχέι την λίστα τον command line arguments π.χ. Η arguments πέρασε στο πρόγραμμα χρησιμοποιώντας την command line. Αν χρησιμοποιείτε το IDE για να γράψετε και να τρέξετε αυτό το πρόγραμμα, κοιτάει ένα τρόπο να καθορίσετε την command line επιχειρήματα στο μενού του προγράμματος. Εδώ, όταν εκτελούμε την python χρησιμοποιώντας_sys.py έχουμε επιχειρήματα, που τρέχουνε τα πρότυπα χρησιμοποιώντας_sys.py με την python εντολή και άλλα πράγματα που ακολουθούν τα επιχειρήματα περνάνε στο πρόγραμμα. Η Python αποθηκεύει την εντολή γραμμή επιχειρημάτων μέσα στην sys.argv μεταβλητή για εμάς για χρήση. Να θυμάστε, το όνομα του σεναρίου που τρέχει πάντα στα πρώτα επιχειρήματα μέσα στην λίστα sys.argv. Όποτε, σε αυτή την περίπτωση μπορούμε να έχουμε 'using_sys.py' σαν sys.argv[0], 'we' σαν sys.argv[1], 'are' σαν sys.argv[2] και 'arguments' σαν sys.argv[3]. Σημειώστε ότι η Python ξεκινά την μέτρηση από το 0 και από το 1. Η sys.path περιεχέι τη λίστα με τον κατάλογο με τα ονόματα από όπου τα πρότυπα εισάγονται. Παρατηστε ότι τα πρώτα string στα sys.path είναι άδεια – αυτά τα άδεια string υποδεικνύουν ότι το ρεύμα των καταλόγων είναι επίσης η διαδρομή της sys.path η οποία είναι ίδια με την PYTHONPATH περιβάλλον των μεταβλητών. Αυτό σημαίνει ότι μπορείτε απευθείας να εισάγετε πρότυπα που βρίσκονται στην ροή του κατάλογου. Αλλιώς, θα πρέπει να τοποθετήσετε τα πρότυπα σε ένα από τους κατάλογους στην sys.path.
Σημειώστε ότι ο πρόσφατος κατάλογος είναι ο κατάλογος από όπου ξεκινάει το πρόγραμμα. Τρέξτε import os; print(os.getcwd()) για να βρούμε τον τωρινό κατάλογο του προγράμματος.

Αρχεια Byte-compiled.pyc

Η εισαγωγή των προτύπων είναι σχετικά δαπανηρή υπόθεση, για αυτό η Python κάνει κάποια κόλπα για να το κάνει πιο γρήγορο. Ο ένας τρόπος είναι να δημιουργήσει ένα αρχείο byte-compiled με επέκταση .pyc που είναι ενδιάμεση μορφή που η Python μετατρέπει το πρόγραμμα σε (να θυμάστε το κομμάτι της εισαγωγής για το πως η python δουλεύει?) αυτά τα αρχεία .pyc είναι χρήσιμα όταν εισάγεις το πρότυπο την επόμενη φορά από ένα πρόγραμμα – έτσι θα είναι πιο γρήγορο εφόσον η μερίδα της επεξεργασίας της εισαγωγής ενός πρότυπου έχει γίνει. Επίσης, αυτά τα αρχεία byte-compiled είναι ανεξάρτητα από την πλατφόρμα.

Σημείωση

Αυτά τα αρχεία .pyc συνήθως δημιουργούνται στον ίδιο κατάλογο ως το αντίστοιχο αρχείο .py. Η python δεν έχει την άδεια να γράφει αρχεία σε αυτόν το κατάλογο, μετά τα αρχεία .pyc δεν θα είναι δημιουργικά.

Η from...import... δήλωση

Αν θέλεις να εισάγεις άμεσα την argv μεταβλητή μέσα στο πρόγραμμα (για να αποφύγετε να πληκτρολογείτε την sys. Κάθε φορά για αυτό), μετά μπορείτε να χρησιμοποιείτε την μορφή sys import argv δήλωση. Αν θέλετε να εισάγετε όλα τα ονόματα χρησιμοποιείστε τα sys πρότυπα, μετά μπορείτε να χρησιμοποιήσετε την sys import * δήλωση. Αυτό δουλεύει για οποιοδήποτε πρότυπο. Γενικά, θα μπορούσατε να αποφεύγετε να χρησιμοποιήσετε αυτή την δήλωση και να χρησιμοποιήσετε την import δήλωση εφόσον το πρόγραμμα σας θα αποφύγει τις συγκρούσεις ονομάτων και θα είναι πιο αξιόπιστο.

Τα ονόματα του πρότυπου

Κάθε πρότυπο έχει ένα όνομα και με μια δήλωση ένα πρότυπο μπορεί να βρει τα ονόματα από τα πρότυπα. Αυτό είναι βολικό σε μια συγκεκριμένη κατάσταση για την εξεύρεση αν το πρότυπο ξεκινά να τρέχει αυτόνομο ή εισάγονται. Όπως αναφέρθηκε προηγουμένως, όταν ένα πρότυπο εισάγεται για πρώτη φορά, ο κώδικας στο πρότυπο εκτελείται. Μπορείτε να χρησιμοποιείται την ιδέα να μεταβάλει τη συμπεριφορά του πρότυπου αν το πρόγραμμα χρησιμοποιείτε από μόνο του και όχι όταν εισάγουμε από ένα άλλο πρότυπο. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας the__name__attridute του πρότυπου.

Παράδειγμα:

```
#!/usr/bin/python
# Filename: using_name.py

if__name__== '__main__':
        print('this program is being run by itself')
else:
        print('i am being imported from another module')
```

Έξοδος:

```
$ python using_name.py
this program is being run by itself

$python
>>> import using_name
I am being imported from another module
>>>
```

Πως δουλεύει:
Οπότε το Python πρότυπο έχει ορίσει το__όνομα__ και αυτό αν είναι '__main__', τότε αυτό συνεπάγεται ότι το πρότυπο ξεκινά να τρέχει αυτόνομα από τον χρήστη και μπορούμε να λαμβάνουν τα κατάλληλα μέτρα.

Κάνοντας τα δικά σας πρότυπα

Η δημιουργία των δικών σας προτύπων είναι εύκολο, το κάνετε όλη την ώρα! Αυτό γίνεται γιατί κάθε πρόγραμμα python είναι και αυτό πρότυπο. Απλά πρέπει να είστε σίγουρη ότι έχει την επέκταση .py. Τα επόμενα παραδείγματα θα σας βοηθήσουν να το καταλάβετε.

```
#!/usr/bin/python
#Filename: mymodule.py

def sayhi():
        print('hi, this is mymodule speaking.')

__version__='0.1'
# το τέλος του πρότυπου .py
```

Το παραπάνω ήταν ένα παράδειγμα προτύπου. Όπως μπορείτε να δείτε, δεν υπάρχει κάτι ιδιαίτερο για να συγκρίνει με το συνηθισμένο πρόγραμμα Python. Μέτα θα δούμε πως να χρησιμοποιείτε αυτό το πρότυπο και σε άλλα προγράμματα της python.
Να θυμάστε ότι τα πρότυπα θα πρέπει να τοποθετηθούν στον ίδιο κατάλογο όπως στο πρόγραμμα που εισάγουμε εμείς, ή το πρότυπο πρέπει να είναι μια από τους κατάλογους στην λίστα sys.path.

```
#!/usr/bin/python
# Filename: mymodule_demo.py

import mymodule

mymodule.sayhi()
print('version',mymodule.__version__)
```

Έξοδος

```
$ python mymodule_demo.py
hi, this is my module spaeking.
Vesion 0.1
```

Πως δουλεύει:
Παρατήστε ότι χρησιμοποιούμε τον ίδιο συμβολισμό με τελείες για την πρόσβαση της ενότητας. Η Python κάνει επαναχρησιμοποίηση της ίδιας σημειογραφία για να δώσει το διακριτικό 'Pythonic' έχει την αίσθηση σε αυτό, έτσι ώστε δεν έχουμε να μαθαίνουμε νέους τρόπους για να κάνουμε τα πράγματα.
Εδώ είναι η έκδοση που χρησιμοποιεί η from...import syntax:

```
#!/usr/bin/python
# Filename: mymodule demo2.py

from mymodule import sayhi,__version__

sayhi()
print('versin',__version__)
```

Η έξοδος του mymodule_demo2.py είναι ίδια με την έξοδο mymodule_demo.py.
Να σημειώσετε ότι αν υπάρχει μια__εκδοσή__ όνομα που δηλώνονται στο πρότυπο ότι εισάγεις στο mymodule, θα υπάρξει σύγκρουση. Αυτό είναι επίσης ίδιο γιατί είναι κοινή πρακτική για κάθε πρότυπο να δηλώνουν τον αριθμό έκδοσης του χρησιμοποιώντας το όνομα του. ως εκ τούτου, πάντα συνιστάτε να προτιμάτε να εισάγετε την δήλωση έστω και αν κάνει το πρόγραμμα πιο μεγάλο.
Θα μπορούσατε να χρησιμοποιείτε:

```
from mymodule import
```

Αυτό θα εισάγει όλα τα δημόσια ονόματα όπως το sayhi άλλα δεν θα μπορείτε να εισάγετε __έκδοση__ γιατί θα ξεκινά με διπλή κάτω παύλα.

Zen of Python
Ένα από της κατευθυντήριες αρχές της Python είναι ότι η “σαφής είναι καλύτερη από ότι σιωπηρή”. Τρέξτε το import για να μάθετε παραπάνω και θα δείτε αυτή την συζήτηση (http://stackoverflow.com/questions/228181/zen-of-python)

H dir function

Μπορείτε να χρησιμοποιήσετε το ενσωματωμένο dir function στην λίστα με αναγνωριστικά που ορίζει ένα αντικείμενο. Για παράδειγμα, για ένα πρότυπο, τα αναγνωριστικά περιλαμβάνουν τη function, κλάσεις και τις μεταβλητές που ορίζονται σε αυτό το πρότυπο.
Όταν προμηθεύετε ένα πρότυπο όνομα στο dir() function, επιστρέφει την λίστα με τα ονόματα που ορίζονται σε αυτό το πρότυπο. Όταν καμία διαφωνία εφαρμόζεται σε αυτό, επιστρέφει τη λίστα με τα ονόματα που ορίζονται σε αυτό το πρότυπο.

Παράδειγμα:

$ python

>>> **import sys** #get list of attributes, in this case, for the sys module

>>> dir(sys)
['__displayhook__','__doc__','__excethook__'.'__name__','__package__','__s
tderr__','__stdin__','__stdout__','__clear_type_cache','__compact__freelist','_current_frames','_getfr
ame','api_version','argv','builtin_module_name','byteorder','call_tracing','callstats','callstats','copyrig
ht','displayhook','dllhandle','dont_write_bytecode','exc_onfo','excepthook','exec_pthook','exec_prefi
x,'executable','exit','flags','float_info','getcheckinterval','getdefaultencoding','getfil','esystemencoding
','getprofile','getrecursionlimit','getrefcouunt','getsizeof','gettrace','getwindowsversion','hexversion','i
ntern','maxsize','maxunicode','meta_path','module','path','path_hooks','path_importer_cache','platfor
m','prefix','ps1','ps2','setcheckinterval','setprofile','setrecursionlimit','settrace','stberr','stdin','stdout','s
ubversion','version','version_info','warnoptions','winver']

>>>dir()# πάρετε τον κατάλογο των χαρακτηριστικών για τα τρέχουσα πρότυπα
['__builtins__','__dor__','__name__','__package__', sys']

>>> a=5 # βρείτε μια μεταβλητή 'a'

>>> dir()
['__builtins__','__doc__','__name__','package__','a',sys']

>>> **del** a # διαγραφή / αφαιρέσει ένα όνομα

>>> dir()
['__builtins__','__doc__','__name__','__package__',' sys']

>>>

Πως δουλεύει:
Πρώτα, βλέπουμε την χρήση του dir στην εισαγωγή του sys προτύπου. Μπορούμε να δούμε την μεγάλη λίστα από ιδιότητες που περιέχει. Μετά, χρησιμοποιούμε την dir function χωρίς να περνάμε τις παραμέτρους. Από προεπιλογή επιστρέφει την λίστα χαρακτηριστικών για το παρόν πρότυπο. Σημειώστε ότι η λίστα από τα εισαγόμενα πρότυπα είναι επίσης από αυτήν την λίστα.
Στην διαδικασία να παρατηρήσουμε την dir σε δράση, ορίζουμε μια καινούρια μεταβλητή a και αντιστοιχίζουμε μια τιμή και μετά ελέγχουμε την dir και παρατηρούμε ότι υπάρχει μια πρόσθετη αξία στην λίστα με τα ίδια ονόματα. Μεταθέτουμε τις μεταβλητές / χαρακτηριστικά από το παρόν πρότυπο χρησιμοποιώντας την del δήλωση και η αλλαγή αντανακλάται πάλι στο αποτέλεσμα της συνάρτησης dir.
Μια σημείωση για την del - αυτη η δήλωση χρησιμοποιείται για να διαγράψει μια μεταβλητή / όνομα και μετά την δήλωση που έχει τρέξει, σε αυτή την περίπτωση del a, δεν μπορείτε να έχετε πρόσβαση στην μεταβλητή a – είναι σαν να μην υπήρχε πότε.
Σημειώστε ότι η dir() function δουλεύει σε οποιοδήποτε αντικείμενο. Για παράδειγμα, τρέξτε την dir(print) για να μάθετε για τα χαρακτηριστικά της print λειτουργίας, ή dir(str) για τα χαρακτηριστικά της str τάξης.

Πακέτα

Μέχρι τώρα, θα πρέπει να έχετε παρατήσει την ιεραρχία της οργάνωσης του προγράμματος.
Η μεταβλητές συνήθως μπαίνουν μέσα στην function. Η Function και οι παγκόσμιες μεταβλητές συνήθως πάνε στα πρότυπα. Και αν θέλετε να οργανώσετε τα πρότυπα? Εκεί είναι όπου τα πακέτα μπαίνουν στην εικόνα. Τα πακέτα είναι απλά φάκελοι πακέτων από πρότυπα με ιδιαίτερα αρχεία __int__.py που δηλώνουν στην python ότι αυτοί οι φάκελοι είναι ιδιαίτερη γιατί περιέχουν πρότυπα της python. Ας πούμε ότι θέλετε να δημιουργήσετε ένα πακέτο που ονομάζεται 'world' με υπό πακέτα 'asia','africa',' και κ.λ.π. και αυτά τα υπό πακέτα με τη σειρά τους περιέχουν πρότυπα όπως το 'india','madagascar' κ.λ.π.
Έτσι είναι ο τρόπος να δομήσετε αυτά τα αρχεία:

```
-<some folder present in the sys.path>/
        -world/
                -__init__.py
                -asia/
                -__init__.py
                -andia/
                        -__init__.py
                        -foo.py
                -africa/
                        -__init__.py
                        -madagascar/
                        -__init__.py
                        -bar.py
```

Τα πακέτα είναι απλά μια ευκολία να οργανώνονται ιεραρχικά τα πρότυπα. Θα δείτε πολλές περιπτώσεις από αυτά στην βασική βιβλιοθήκη.

Περίληψη

Όπως και η function έχει επαναχρησιμοποιούμενα μέρη προγραμμάτων, τα πρότυπα είναι επαναχρησιμοποιούμενα προγράμματα. Τα πρότυπα είναι μια άλλη ιεραρχία για να οργανώνετε τα πρότυπα. Οι βασικές βιβλιοθήκες που είναι στην python είναι ένα παράδειγμα από τέτοια πακέτα και πρότυπα. Έχουμε δει πως να χρησιμοποιήσουμε αυτά τα πρότυπα και να δημιουργήσουμε τα δικά μας πρότυπα. Μετά, θα μάθουμε για κάποιες έννοιες που ονομάζονται δομή.

Python en: Δομή Δεδομένων

Οδηγίες

Δομή δεδομένων σημαίνει απλά ότι - υπάρχουν δομές όπου μπορούν να κρατήσουν κάποια δεδομένα μαζί. Με άλλα λόγια, έχουν συνηθίσει να αποθηκεύουν μια συλλογή από παρόμοια δεδομένα. Υπάρχουν τέσσερις ενσωματωμένες δομές δεδομένων της Python – λίστα, πλειάδα, λεξικό και σετ. Θα δούμε πως να χρησιμοποιούμε κάθε ένα από αυτά και πως μας κάνουν την ζωή μας ευκολότερη.

Λίστα

Η λίστα είναι μια δομή δεδομένων όπου κρατάει μια διατεταγμένη συλλογή από αντικείμενα, μπορείτε να αποθηκεύεται μια σειρά από αντικείμενα στην λίστα. Αυτό είναι εύκολο να το φανταστείτε αν μπορείτε να σκεφτείτε μια λίστα αγορών όπου έχεις μια λίστα από αντικείμενα να αγοράσεις, να περιμένετε ότι μάλλον έχετε κάθε αντικείμενο σε ξεχωριστή γραμμή στην λίστα αγορών ενώ στην Python βάζεις κόμμα μεταξύ στα αντικείμενα. Η λίστα από αντικείμενα θα πρέπει να περικλείονται με αγκύλες έτσι ώστε η Python να κατανοεί προσδιορίζοντας την λίστα. Όταν έχεις δημιουργήσει μια λίστα, μπορείς να προσθέσεις, αφαιρέσεις ή ψάξεις για αντικείμενα στην λίστα. Εφόσον μπορούμε να προσθέσουμε και να αφαιρέσουμε αντικείμενα, λέμε ότι η λίστα είναι ευμετάβλητη τύποι δεδομένων. Αυτός ο τύπος μπορεί να αλλοιωθεί.

Μια γρήγορη εισαγωγή σε αντικείμενα και τις κατηγορίες

Παρόλο που γενικά έχω καθυστερήσει την συζήτηση για τα αντικείμενα και τις κατηγορίες μέχρι τώρα, μια μικρή εξήγηση χρειάζεται τώρα ώστε να καταλάβετε την λίστα καλύτερα. Θα εξερευνήσουμε αυτό το θέμα με λεπτομέρειες πιο μετά στο δικό του κεφάλαιο. Η κατηγορίες είναι ένα παράδειγμα χρήσης των αντικειμένων και των τάξεων, όταν χρησιμοποιούμε μια μεταβλητή I και αντιστοιχίσουμε μια τιμή σε αυτή, ας πούμε έναν ακέραιο 5 σε αυτή, μπορείτε να σκεφτείτε όταν δημιουργείτε έναν αντικείμενο i από κλάσης int. όντως, μπορείτε να διαβάσετε help(int) για να καταλάβετε αυτό καλύτερα. Μια κατηγορία μπορεί επίσης να έχει μεθόδους π.χ. Fuction ορίζετε για χρήση με σεβασμό σε αυτή την κατηγορία μόνο. Μπορείτε να χρησιμοποιήσετε αυτά τα κομμάτια από την λειτουργικότητα μόνο όταν έχετε ένα αντικείμενο από αυτές τις κατηγορίες. Για παράδειγμα, η Python παρέχει μια μέθοδο προσάρτησης για την κατηγορία της λίστας που σας επιτρέπει να προσθέτετε ένα αντικείμενο στο τέλος της λίστας. Για παράδειγμα, mylist.append('an item') θα προσθέσει αυτό το string στην λίστα mylist. Σημειώστε την χρήση για τον συμβολισμό με τελείες για την πρόσβαση της μεθόδου των αντικειμένων. Μια κατηγορία μπορεί να έχει επίσης **πεδία** τα οποία δεν είναι τίποτα άλλο από ορίσματα μεταβλητών για την χρήση με σεβασμό για αυτήν την κατηγορία μόνο. Μπορείτε να χρησιμοποιήσετε αυτές τις μεταβλητές/ονόματα μόνο όταν έχετε ένα αντικείμενο από αυτήν την κατηγορία. Τα πεδία επίσης δέχονται από τον συμβολισμό με τελείες, για παράδειγμα, mylist.field.

Παράδειγμα:

```
#!/usr/bin/python
# File name: using_list.py

# Αυτή είναι η λίστα με τα ψώνια μου
shoplist = ['apple','mango','carrot','banana']
print('i have', len(shoplist), 'items to purchase,')

print('these items are:', end=' ')

for item  in shoplist:
        print(item, end=' ')
print('\n I also have to buy rice.')
shoplist.append('rice')
print('my shopping list is now', shoplist)
```

```
print('i will sort my list now')
shoplist.sort()
print('sorted shopping list is', shoplist)
print('the first item I will buy is',shoplist[0])
olditam = shoplist[0]
del shoplist[0]
print('i bought the',olditem)
print('my shopping list is now', shoplist)
```

Έξοδος:

```
$ python using_list.py
I have 4 items to purchase.
These items are: apple mango carrot banana
I also have to buy rice.
My shopping list is now ['apple','mango','carrot','banana','rice']
I will sort my list now
sorted shopping list is ['apple','banana','carrot','mango','rice']
the first item I will buy is apple
I bought the apple
my shopping list is now ['banana','carrot','mango','rice']
```

Πως δουλεύει:
Η μεταβλητή shoplist είναι η λίστα αγορών για κάποιον που πάει να αγοράσει κάτι από το super market. Στην shoplist, αποθηκεύει μόνο string με τα ονόματα από τα αντικείμενα που αγοράζεις αλλά μπορείς να προσθέσεις οποιοδήποτε αντικείμενο ακόμα και λίστες ακόμα σε και άλλες λίστες. Επίσης έχουμε χρησιμοποίησή την for... σε επανάληψη για να επανεκλέγω μέσα από τα αντικείμενα από την λίστα. Μέχρι τώρα θα πρέπει να έχετε συνειδητοποίησε ότι η λίστα είναι μια ακολουθία. Η ειδικότητα της ακολουθίας θα την συζητήσουμε σε επόμενο κεφάλαιο. Σημειώστε τη χρίση του επιχειρήματος end στην print function για να δείχνει αυτό που θέλουμε στο τέλος στην έξοδο χωρίς κενά ενδιάμεσα τους από της συνηθισμένες αλλαγές γραμμής. Μέτα, προσθέτουμε ένα αντικείμενο στην λίστα χρησιμοποιώντας την append μέθοδο από την λίστα τον αντικείμενον, όπως έχουμε ήδη συζητήσει πριν. Τότε, ελέγχουμε αν όντως το αντικείμενο έχει προστεθεί στην λίστα με το να εκτυπώνουμε το περιεχόμενο της λίστα και απλά να περάσουμε την λίστα στην print δήλωση που εκτυπώνει νοικοκυρεμένα. Μέτα, μικραίνουμε την λίστα με το να χρησιμοποιούμε την sort μέθοδο από την λίστα. Είναι σημαντικό να καταλάβουμε ότι αυτή η μέθοδος επηρεάζει τη λίστα από μόνη της και δεν επιστρέφει μια τροποποιημένη λίστα – αυτό είναι διαφορετικό από το πως δουλεύουν τα string. Αυτό ενώνουμε με το να λέμε αυτή η λίστα είναι ευμετάβλητη και αυτά τα string είναι αμετάβλητα. Μετά, όταν τελειώσουμε με το να αγοράσουμε ένα αντικείμενο από το super market, θέλουμε να το αφαιρέσουμε από την λίστα. Και αυτό το πετυχαίνουμε με το να χρησιμοποιούμε την del δήλωση. Εδώ, αναφέρουμε ποια από τα αντικείμενα από την λίστα που θέλουμε να τα αφαιρέσουμε και η del δήλωση το αφαιρεί για εμάς. Καθορίζουμε αυτό που θέλουμε να αφαιρέσουμε με το πρώτο αντικείμενο από την λίστα και ως εκ τούτου, χρησιμοποιούμε την del shoplist[0] (να θυμάστε ότι η python ξεκινά την μέτρηση από το 0).
Αν θέλετε να ξέρετε όλες της μεθόδους που ορίζονται από την λίστα τον αντικείμενον, δείτε το help(list) για λεπτομέρειες.

Πλειάδα

Η πλειάδα χρησιμοποιείται για να κρατάει μαζί πολλαπλά αντικείμενα. Σκεφτείτε τα αυτά σαν παρόμοια με την λίστα, άλλα χωρίς την εκτενή λειτουργικότητα που η κλάση της λίστας σου δίνει. Ένα σημαντικό χαρακτηριστικό τις πλειάδας είναι ότι είναι αμετάβλητη όπως στα string π.χ. Δεν μπορείτε να τροποποιήσετε τις πλειάδες. Οι πλειάδες ορίζονται προσδιορίζοντας αντικείμενα χωρισμένα με κόμμα μέσα σε ένα προαιρετικό ζευγάρι από παρενθέσεις. Οι πλειάδες συνήθως χρησιμοποιούνται σε περιπτώσεις όπου μια δήλωση ή μια οριζόμενη από το χρήστη συνάρτηση μπορεί με ασφάλεια να υποθέσει μια συλλογή από τιμές π.χ. Η πλειάδα από τιμές χρησιμοποιείτε χωρίς αλλαγές.

Παράδειγμα:
```
#!/usr/bin/python
# Filename: using_tuple.py

zoo = ('python','elephant','penguin') # να θυμάστε ότι η παρενθέσεις είναι προαιρετικά
print('number of animals in the zoo is', len(zoo))

new_zoo = ('monkey','camel', zoo)
print('number of cages in the new zoo is', len(new_zoo))
print('all animals in new zoo are', new_zoo)
print('animals brought from old zoo are', new_zoo[2])
print('last animal brought from old zoo is', new_zoo[2][2])
print('number of animals in the new zoo is',len(new_zoo)-1+len(new_zoo[2]))
```

Εξοδος:
```
$ python using_tuple.py
number of animals in the zoo is 3
number of cages in the new zoo is 3
all animals in new zoo are ('monkey','camel',('python','elephant','penguin'))
animals btought from old zoo are('python','elephant','penguin')
last animal btought from old zoo is penguin
number of animals in the new  zoo is 5
```

Πως δουλεύει:
Οι μεταβλητές zoo αναφέρεται σε μια πλειάδα στοιχείων. Το βλέπουμε αυτό στην len function που μπορείτε να χρησιμοποιείτε για να πάρει το μήκος της πλειάδας. Αυτό επίσης δείχνει ότι η πλειάδα είναι μια ακολουθία επίσης. Τώρα μπορούμε να μεταφέρουμε αυτά τα ζώα σε ένα νέο ζωολογικό κήπο εφόσον ο παλαιός κλείνει. Για αυτό ο new_zoo πλειάδα περιεχέι μερικά ζώα που είναι ήδη εκεί μαζί με τα ζώα που έφεραν από τον παλαιό ζωολογικό κήπο. Πίσω στην πραγματικότητα, σημειώστε ότι μια πλειάδα μέσα σε μια πλειάδα δεν χάνει την ταυτότητα της. Μπορούμε να έχουμε προσβάσεις στα αντικείμενα μέσα στην πλειάδα με το να προσδιορίζουμε την θέση τον αντικείμενον μέσα σε ένα ζεύγος από αγκύλες όπως κάναμε και με τις λίστες. Αυτό ονομάζεται τελεστή ευρετηρίασης. Έχουμε πρόσβαση στο τρίτο αντικείμενο μέσα στο new_zoo με το να προσδιορίζουμε το new_zoo[2] και έχουμε πρόσβαση στο τρίτο αντικείμενο μέσα στο τρίτο αντικείμενο στο new_zoo πλειάδα με το να προσδιορίζουμε new_zoo[2][2]. Αυτό είναι πολύ απλό εφόσον καταλάβουμε το ιδίωμα.

Παρενθέσεις
Παρόλο που οι παρενθέσεις είναι προαιρετικές, προτιμάω πάντα να τις έχω για να κάνω προφανές ότι έχω μια πλειάδα, ειδικά επειδή αποφεύγεται η ασάφεια. Για παράδειγμα, print(1,2,3) και print((1,2,3)) σημαίνει δυο διαφορετικά πράγματα - το μεν τυπώνει τρεις αριθμούς, αντίθετα το δε τυπώνει μια πλειάδα (όπου περιεχέι τρεις αριθμούς).

Πλειάδα με 0 ή 1 αντικείμενα
Μια άδεια πλειάδα κατασκευάζεται με δυο άδεια ζευγάρια από παρενθέσεις όπως το myempty = (). Ωστόσο, μια πλειάδα με μόνο ένα αντικείμενο δεν είναι τόσο απλό. Πρέπει να ορίσετε ότι θα χρησιμοποιήσει με ένα κόμμα που ακολουθείτε το πρώτο (και μόνο) αντικείμενο ώστε η python να μπορεί να διακρίνει μεταξύ μια πλειάδας και ενός ζευγαριού από παρενθέσεις περιβάλλεται από αντικείμενα σε μια έκφραση π.χ. Πρέπει να προσδιορίσετε singleton = (2,) αν εννοείτε ότι θέλετε μια πλειάδα να περιέχει το αντικείμενο 2.

Σημειώσει για τους προγραμματιστές της perl
Μια λίστα μέσα σε μια λίστα δεν χάνει την ταυτότητα της π.χ. Η λίστα δεν είναι ισοπεδωμένη όπως στην perl. Το ίδιο ισχύει στην πλειάδα μέσα στην πλειάδα, ή μια πλειάδα μέσα στην λίστα, ή μια λίστα μέσα στην πλειάδα, και ου το κάθεξεις. Εφόσον η python εκφράζει την ανησυχία της, ότι είναι άπλα πράγματα που αποθηκεύονται χρησιμοποιώντας ένα άλλο αντικείμενο, αυτά.

Λεξικό

Το λεξικό είναι σαν ένα βιβλίο διευθύνσεων όπου μπορείτε να βρείτε της διεύθυνσης ή πληροφορίες επαφών για ένα πρόσωπο γνωρίζοντας μόνο το όνομά του / της π.χ. Συνδυάζουμε κλειδιά (ονόματα) με τιμές (πληροφορίες). Σημειώστε ότι το κλειδί πρέπει να είναι μοναδικό σαν να μην μπορούμε να βρούμε τις σωστές πληροφορίες αν έχεις δυο πρόσωπα με το ίδιο όνομα. Σημειώστε ότι μπορείτε να χρησιμοποιήσετε μόνο αμετάβλητα αντικείμενα (όπως τα string) για κλειδιά για το λεξικό άλλα μπορείτε να χρησιμοποιήσετε είτε αμετάβλητα ή ευμετάβλητα αντικείμενα για τις τιμές του λεξικού. Αυτό βασικά μεταφράζεται για να λέει ότι μπορούμε να χρησιμοποιούμε μόνο τα άπλα αντικείμενα για κλειδιά. Ζευγάρια από κλειδιά και τιμές ορίζονται σε ένα λεξικό με το να χρησιμοποιείται ο συμβολισμός d = {key 1: value1, key2: value2}. Σημειώστε ότι τα ζευγάρια κλειδιά-τιμές χωρίζονται από άνω κάτω τελεία και τα ζεύγη χωρίζονται από μονά τους με κόμμα και όλα αυτά περικλείονται σε ένα ζευγάρι με αγκύλες. Να θυμάστε ότι τα ζευγάρια κλειδιά – τιμές στο λεξικό δεν ταξινομούνται με οποιοδήποτε τρόπο. Αν θέλετε ειδικότερα συγκεκριμένη σειρά, τότε θα πρέπει να το μικρύνετε για τον εαυτό σας πριν την χρησιμοποιήσετε. Το λεξικό που θα χρησιμοποιήσετε είναι υποστάσεις / αντικείμενα της κατηγορίας dict.

Παράδειγμα:

```
#!/usr/bin/python
# Filename: using_dict.py
# 'ab' ειναι η συντόμευση για το 'a'ddress'b'ook
ab = { 'swaroop' : 'swaroop@swaroopch.com',
         'larry' : 'larry@wall.org',
         'matsumoto' : 'matz@ruby.org',
         'spammer' : 'spammer@hotmail.com'
         }

print(“swaroop's address is”, ab['swaroop'])
#διαγραφή ενός ζευγαριού κλειδιού – τιμή

del ab['spammer']
print('\n there are {0} contacts in the address – book \n'.format(ab)))

for name, address in ab.items():
        print('contacts {0} at {1}'. format (name,address))

# προσθέτοντας ενα ζευγάρι κλειδιού – τιμή

ab['guido'] = 'guido@python.org'
if  'guido' in ab: # ή ab.has_key('guido')
        print(“\nguido's address is”, ab['guido'])
```

Έξοδος:

```
$ python using_dict.py
swaroop's address is swaroop@swaroopch.com

there are 3 contacts in the address – book

contact swaroop at swaroop@swaroopch.com
contact matsumoto at matx@ruby-lang.org
contact larry at larry@wall.org

guido's address is guido@python.org
```

Πως δουλεύει:

Δημιουργούμε το λεξικό ad χρησιμοποιώντας τη σημειογραφία που ήδη έχουμε συζητήσει. Μέτα έχουμε πρόσβαση στις λέξεις – τιμές με το να ορίζουμε τις λέξεις χρησιμοποιώντας το ευρετήριο

όπως συζητείται στο πλαίσιο της λίστας και τον πλειάδων. Παρατηρήστε την απλή σύνταξη. Μπορούμε να διαγράψουμε τα ζευγάρια λέξεις – τιμές χρησιμοποιώντας τους παλαιούς φίλους – της del δήλωσης. Απλά ορίζουμε το λεξικό και ο τελεστή ευρετηρίασης για το κλειδί για να αφαιρούνται και να περάσει η del δήλωση. Δεν υπάρχει ανάγκη να ξέρουμε την τιμή που αντιστοιχεί στο κλειδί για αυτή τη λειτουργία. Μετά, έχουμε πρόσβαση σε κάθε ζευγάρι κλειδί – τιμή από το λεξικό χρησιμοποιώντας την items μέθοδο από το λεξικό το οποίο γύρνα την λίστα με πλειάδες όπου κάθε πλειάδα περιέχει ένα ζευγάρι από items - το κλειδί ακολουθείται από την τιμή. Ανακτούμε αυτό το ζευγάρι και τον συσχετισμό με τις μεταβλητές name και address αντίστοιχα για κάθε ζευγάρι που χρησιμοποιούμε την for...in loop και μετά print αυτές τις μεταβλητές for-block. Μπούμε να προσθέσουμε καινούρια ζευγάρια κλειδιά – τιμές απλά χρησιμοποιώντας τον τελεστή ευρετηρίασης για να έχουμε πρόσβαση στο κλειδί και να εκχωρήσετε αυτή την τιμή, όπως έχουμε κάνει για το Guido στην προηγούμενη περίπτωση.

Ορίσματα με Λέξεις Κλειδιά και Λεξικά
Μια διαφορετική σημείωση, αν έχετε χρησιμοποιήσει επιχειρήματα λέξεις κλειδιά στην function, έχετε ήδη χρησιμοποιήσει λεξικά! Απλά σκεφτείτε ότι – τα ζευγάρια κλειδιά – τιμές ορίζονται από εσένα με την λίστα παραμέτρων του ορισμού της function και έχουμε πρόσβαση στης μεταβλητές με την function, είναι απλά ένα κλειδί πρόσβασης του λεξικού (το οποίο ονομάζεται ο πίνακας σύμβολο ορολογία του σχεδίασης μεταγλωτιστών).

Ακολουθίες

Λίστες, πλειάδες και string είναι παραδείγματα από ακολουθίες, άλλα τι είναι ακολουθίες και τι είναι το τόσο ειδικό για αυτές?
Τα κύρια χαρακτηριστικά είναι ότι έχουν δοκιμή συμμετοχής (π.χ. Στη in και όχι στην in εκφράσεις) και τις λειτουργίες ευρετηρίασης. Οι λειτουργίες ευρετηρίασης οι οποίες μας επιτρέπει να φέρουμε το συγκεκριμένο αντικείμενο άμεσα στην ακολουθία. Τα τρία ήδη από ακολουθίες αναφέρονται παρακάτω – λίστες, πλειάδα και string, επίσης έχουμε λειτουργία τεμαχισμού που μας επιτρέπουν να ανακτήσουμε τμήμα της ακολουθίας π.χ. Ένα κομμάτι της ακολουθίας.

Παράδειγμα:

```
#!/usr/bin/python
# Filename: seq.py

shoplist = ['apple','mango','carrot','banana']
name = 'swaroop'

# ευρετηρίαση ή 'sybscription' λειτουργία

print('item 0 is', shoplist[0])
print('item 1 is', shoplist[1])
print('item 2 is', shoplist[2])
print('item 3 is', shoplist[3])
print('item -1 is', shoplist[-1])
print('item -2 is', shoplist[-2])
print('character  0 is', name[0])

# τεμαχισμό σε λίστα

print('item 1 to 3 is', shoplist[1:3])
print('item 2 to and is', shoplist[2: ])
print('item 1 to -1 is', shoplist[1:-1])
print('item start to end is', shoplist[:])
```

```
# τεμαχισμό σε μια σειρά

print('characters 1 to 3 is', name[1:3])
print('characters 2 to and is', name[2: ])
print('characters 1 to -1 is', name[1:-1])
print('characters starts to and is', name[ : ])
```

Έξοδος:

```
$ python seq.py
item 0 is apple
item 1 is mango
item 2 is carrot
item 3 is banana
item -1 is banana
item -2 is carrot
character 0 is s
item 1 to 3 is ['mango','carrot']
item 2 to end is ['carrot','banana']
item 1 to -1 is ['mango','carrot']
item start to and is ['apple','mango','carrot','banana']
characters 1 to 3 is wa
characters 2 to and is aroop
characters 1 to -1 is waroo
characters start to and is swaroop
```

Πως δουλεύει:
Πρώτα, θα δούμε πως να χρησιμοποιήσουμε ευρετήρια για να βρούμε μεμονωμένα στοιχεία μιας ακολουθίας. Αυτό επίσης αναφέρεται σε εμάς σαν λειτουργία εγγραφής. Όταν προσδιορίζετε έναν αριθμό σε μια ακολουθία σε αγκύλες όπως είδαμε πιο πάνω, η python θα φέρει το αντικείμενο που είναι αντίστοιχο στην θέσει στην ακολουθία. Να θυμάστε ότι η Python ξεκινά να μετρά αριθμούς από το 0. Ως εκ τούτου, shoplist[0] προσκομίζει το πρώτο αντικείμενο και η shoplist[3] προσκομίζει το τέταρτο αντικείμενο στην ακολουθία της shoplist. Το περιεχόμενο μπορεί να είναι και αρνητικός αριθμός, στην οποία περίπτωση, η θέση υπολογίζεται από το τέλος της ακολουθίας. Για αυτό, η shoplist[-1] αναφέρετε στα τελευταία αντικείμενα στην ακολουθία και η shoplist[-2] φέρνει το δεύτερο τελευταίο αντικείμενο στην ακολουθία. Η λειτουργία τεμαχισμού χρησιμοποιείται με το να προσδιορίζουμε το όνομα της ακολουθίας που ακολουθείτε με προαιρετικό ζευγάρι αριθμών που χωρίζονται από την άνω και κάτω τελεία μέσα σε αγκύλες. Σημειώστε ότι αυτό είναι παρόμοιο με τη λειτουργία ευρετηρίασης, που έχουν χρησιμοποιηθεί μέχρι τώρα. Να θυμάστε ότι οι αριθμοί είναι προαιρετικοί αλλά η διπλή τελεία δεν είναι. Ο πρώτος αριθμός (πριν την αγκύλη) στην λειτουργία κοπής αναφέρεται στην θέση από που ξεκινά να κόβει και ο δεύτερος αριθμός (με την αγκύλη) δείχνει όπου η κοπή θα σταματήσει. Στον πρώτο αριθμό δεν ορίζεται, η python θα ξεκινήσει την ακολουθία. Αν ο δεύτερος αριθμός μένει έξω, η Python θα σταματήσει στο τέλος της μιας ακολουθίας. Σημειώστε ότι η κοπή επέστρεψε το starts στην start θέση και θα τελειώσει μόλις πριν την end θέση π.χ. Η start θέση περιλαμβάνετε αλλά η end θέση αποκλείεται από την ακολουθία κοπίς. Έτσι, η shoplist[1:3] επιστρέφει ένα κομμάτι από την ακολουθία που ξεκινά στην θέση 1, περιεχέι την θέση 2 άλλα σταμάτα στην θέση 3 και για αυτό ένα κομμάτι από δυο αντικείμενα επιστρέφονται. Παρόμοια, η shoplist[:] επιστρέφει ένα αντίγραφο από όλη την ακολουθία. Μπορείτε επίσης να κάνετε την κοπή και με αρνητικές θέσεις. Οι αρνητικοί αριθμοί χρησιμοποιούνται για τις θέσεις από το τέλος της ακολουθίας. Για παράδειγμα, shoplist[:1] θα επιστρέφει ένα κομμάτι από την ακολουθία, πράγμα το οποίο αποκλείει τα τελευταία αντικείμενα από την ακολουθία άλλα περιεχέι όλα τα υπόλοιπα. Μπορείτε επίσης να δώσετε ένα τρίτο όρισμα για το κομμάτι, το οποίο είναι το βήμα για τον τεμαχισμό (από προεπιλογή το μέγεθος βήματος είναι 1):

```
>>>shoplist = [' apple','mango','carrot','banana']
>>>shoplist[ : :1]
['apple','mango','carrot','banana']
>>>shoplist[ : :2]
['apple','carrot']
>>>shoplist[ : :3]
['apple','banana']
>>>shoplist[ : :-1]
['banana','carrot','mango','apple']
```

Σημειώστε ότι όταν το βήμα είναι 2, τότε έχουμε τα αντικείμενα με θέση 0,2,... όταν το μέγεθος του βήματος είναι 3, τότε το αντικείμενο με θέση 0,3, και ου τω κάθεξής. Δοκιμάστε διάφορους συνδυασμούς από τέτοιους προσδιορισμούς τμημάτων χρησιμοποιώντας της Python τον διαδραστικό διερμηνέα π.χ. η προτροπή ώστε να μπορείτε να δείτε τα αποτελέσματα αμέσως. Το μεγάλο πράγμα για τις ακολουθίες είναι ότι μπορείτε να έχετε πρόσβαση στις πλειάδες, λίστες και strings σε όλα την ίδια ακριβώς στιγμή.!

Ρυθμίσεις

Η ρυθμίσεις επιτυγχάνονται με την εντολή των απλών αντικειμένων. Αυτά χρησιμοποιούνται όταν η ύπαρξη ενός αντικειμένου σε μια συλλογή είναι πιο σημαντική από τη διάταξη ή πόσες φορές εμφανίζεται. Χρησιμοποιούμε τις ρυθμίσεις, μπορείτε να ελέγξετε για μέλη, αν είναι ένα υποσύνολο ενός άλλου συνόλου, βρίσκουμε τη διασταύρωση μεταξύ δύο.

```
>>> bri = set(['brazil','russia','india'])
>>> 'india' in bri
true
>>> 'usa' in bri
false
>>> bric = bri.copy()
>>> bric.add('china')
>>> bric.issuperset(bri)
true
>>> bri.remove('russia')
>>> bri & bric # ή bri.intersection(bric)
{'brazil','india'}
```

Πως δουλεύει:
Το παράδειγμα είναι λίγο πολύ αυτονόητο, διότι περιλαμβάνει βασική θεωρία μαθηματικών συνόλων του.

Αναφορές

Όταν δημιουργείτε ένα αντικείμενο και το αντιστοιχίσετε σε μια μεταβλητή, η μεταβλητή μόνο αναφέρει το αντικείμενο και δεν αντιπροσωπεύει το ίδιο το αντικείμενο! Αυτό είναι, οι μεταβλητές είναι σημεία ονόματα που περνούν κομμάτια στην μνήμη των υπολογιστών όπου το αντικείμενο αποθηκεύετε. Αυτό ονομάζεται δέσμευση της ονομασίας με το αντικείμενο.
Γενικά, δεν χρειάζεται να ανησυχείτε για αυτό, άλλα είναι μια λεπτή επίδραση εξαιτίας των παραπομπών τις οποίες πρέπει να γνωρίζετε:

Παράδειγμα:

```
#!/usr/bin/python
# Filename: referemce.py

print('simple assignement')
```

```
shoplist = ['apple','mango','carrot','banana']
mylist = shoplist # mylist είναι ένα ακόμα όνομα που συγκλίνουν προς το ίδιο αντικείμενο

del shoplist[0] # Αγόρασα το πρώτο στοιχείο έτσι μπορώ να το αφαιρέσετε από τη λίστα
print('shoplist is', shoplist)
print('mylist is',mylist)
# να σημειώσετε ότι και τα δυο shoplist και mylist εκτυπώνονται στην ίδια λίστα χωρίς
# το 'apple ' επιβεβαιώνοντας ότι δείχνουν στο ίδιο αντικείμενο

print('copy by making a full slice')
mylist = shoplist[:] # κάντε ένα αντίγραφο με το να κάνετε μια πλήρης κοπή
del mylist[0] # αφαιρέστε το πρώτο αντικείμενο
print('shoplist is', shoplist)
print('mylist is,'mylist)
# σημειώστε ότι τώρα και οι δυο λίστες είναι διαφορετικές
```

Έξοδος:

```
$ python reference.py
simple assignment
shoplist is ['mango','carrot','banana']
mylist is ['mango','carrot','banana']
copy by making a full slice
shoplist is ['mango','carrot','banana']
mylist is ['carrot','banana']
```

Πως δουλεύει:
Πολλές από τις εξηγήσεις είναι προσβάσιμες στα σχόλια. Να θυμάστε ότι αν θέλετε να κάνετε αντίγραφο της λίστας ή τέτοιου είδους ακολουθίες ή συγκρότημα αντικειμένων (όχι άπλα τα αντικείμενα όπως ένας ακέραιος), μετά μπορείτε να χρησιμοποιήσετε την λειτουργία τεμαχισμού για να κάνετε αντίγραφο. Αν εκχωρήσετε την ονομασία της μεταβλητής σε μια άλλη ονομασία, και οι δυο από αυτές που θα αναφέρετε στο ίδιο αντικείμενο και αυτό θα μπορούσε να είναι το πρόβλημα, αν δεν είστε προσεκτικοί.

Σημείωση για τους προγραμματιστές perl
Να θυμάστε ότι μια δήλωση εκχώρησης για της λίστες δεν δημιουργεί αντίγραφο. Θα πρέπει να χρησιμοποιήσετε την λειτουργία τεμαχισμού για να κάνει ένα αντίγραφο για μια ακολουθία.

Περισσότερα για τα strings

Έχουμε συζήτηση για τα strings με λεπτομέρειες νωρίτερα. Τι περισσότερο μπορεί να υπάρχει? Λοιπόν ξέρατε ότι τα strings είναι και αυτά αντικείμενα και έχουν και αυτά μεθόδους όπου κάνουν τα πάντα εκτός από το να ελέγχουν κομμάτια των strings για την απογύμνωση χώρων! Τα strings που χρησιμοποιούμε στα προγράμματα είναι όλα αντικείμενα της κλάσης srt. Κάποιες χρήσιμες μεθόδους αυτής της κλάσης δείχνονται στο επόμενο παράδειγμα. Για μια ολοκληρωμένη λίστα για τέτοιους μεθόδους, δείτε το help(srt).

Παράδειγμα:

```
#!usr/bin/python
# Filename: srt_methods.py

name = 'swaroop' # αυτό είναι ένα string αντικείμενο

if name.startswith('swa'):
        print('yes, the string starts with “swa”')

if 'a' in name:
```

```
        print('yes, it contains the string “a”')
if name.find('war') !=-1:
        print('yes, it contains the string “war”')

delimiter ='_*_'
mylist = ['brazil','russia','india','india','china']
print(delimiter.join(mylist))
```

Έξοδος:
```
$ python srt_methods.py
yes,the string starts with “swa”
yes, it contains the string “a”
yes, it contains the string “war”
brazil_*_russia_*_india_*_china
```

Πως δουλεύει:
Εδώ, θα δούμε πολλές μεθόδους για τα string σε δράση. Η startwith μέθοδος χρησιμοποιείτε για να βρούμε αν τα string που ξεκινάνε με τα δοσμένα string. Ο in χειριστής χρησιμοποιείτε για να ελέγξουμε αν ένα δοσμένο string είναι κομμάτι του string. Η find μέθοδος χρησιμοποιείτε για να βρούμε την θέση του δοσμένου string στο string ή επιστρέφει το -1 αν δεν είναι επιτυχείς για να βρει το substring. Η κλάση srt επίσης έχει μια ωραία μέθοδο για να ενώσει τα αντικείμενα της ακολουθίας με το string ενεργώντας ως οριοθέτη μετάξυ κάθε αντικειμένου της ακολουθίας και επιστρέφει ένα μεγαλύτερο string που παράγεται από αυτό.

Περίληψη
Έχουμε εξερευνήσει διάφορες ενσωματωμένες δομές δεδομένων της Python με λεπτομέρειες. Αυτές οι δομές δεδομένων θα είναι απαραίτητες για να γράψουμε προγράμματα του λογικού μεγέθους.
Τώρα που έχουμε πολλά από τα βασικά της Python στη γνώση μας, θα δούμε μετά πως να σχεδιάζουμε και να γράψουμε ένα πραγματικό – κόσμο προγράμματος της Python.

Python en: Επίλυση Προβλημάτων
Έχουμε εξερευνήσει διάφορα κομμάτια της γλώσσας της Python και τώρα θα δούμε πως όλα αυτά τα κομμάτια ταιριάζουν μεταξύ τους, με το να σχεδιάσουμε και να γράψουμε ένα πρόγραμμα το όποιο να κάνει κάτι χρήσιμο. Η ιδέα είναι να μάθουμε να γράφουμε ένα σενάριο Python από μόνη μας.

Το πρόβλημα
Το πρόβλημα είναι “θέλω ένα πρόγραμμα το οποίο να δημιουργεί ένα αντίγραφο ασφαλείας όλων των σημαντικών αρχείων”. Παρόλα αυτά, είναι ένα απλό προβλήματα, δεν υπάρχουν αρκετές πληροφορίες για εμάς για να ξεκινήσουμε με την λύση. Μια μικρή ανάλυση χρειάζεται. Για παράδειγμα, πως καθορίζετε ποιο αρχείο είναι για να γίνει backed up? Πως αποθηκευται? Που αποθηκεύεται? Αφού αναλύσουμε σωστά το πρόβλημα, σχεδιάζουμε το δικό μας πρόβλημα. Κάντε μια λίστα από πράγματα πως το πρόγραμμα μας πρέπει να δουλεύει. Σε αυτή την περίπτωση, έχω δημιουργήσει την ακόλουθη λίστα στο πως θέλω να δουλεύει. Αν κάνετε το σχεδιασμό, μπορεί να κάνετε την ίδια ανάλυση αφού κάθε άνθρωπος είναι διαφορετικός, οπότε αυτό είναι κάλο.

1. Τα αρχεία και οι κατάλογοι για να κάνουμε αντίγραφο ασφαλείας προσδιορίζονται σε μια λίστα
2. Το αντίγραφο ασφαλείας πρέπει να αποθηκεύετε στον κεντρικό κατάλογο αντιγράφων ασφαλείας.
3. Τα αρχεία των αντιγράφων ασφάλειας αποθηκεύονται σε .zip αρχεία.
4. Το όνομα του συμπιεσμένου αρχείου είναι η τρέχουσα ημερομηνία και ώρα.
5. Χρησιμοποιούμε τη γνωστή εντολή zip που είναι διαθέσιμη σε κάθε πρότυπο Linux/Unix. Η χρήστες windows καλούν την εγκατάσταση

(http://gnuwin32.sourceforge.net/bownlinks/zip.php) από το GnuWin32project σελίδα(**http://gnuwin32.sourceforge.net/packages/zip.htm**) και προσθέτουμε **c:\program** Files\GnuWin32\bin στο δικό μας σύστημα περιβάλλον PATH μεταβλητών, παρόμοια με το ότι έχουμε κάνει για να αναγνωρίζει την Python εντολές από μόνο του. Σημειώστε ότι μπορείτε να χρησιμοποιήσετε οποιαδήποτε αρχειοθέτηση θέλετε όσο έχει μια εντολή γραμμών διεπαφή έτσι ώστε να μπορεί να περάσει τα επιχειρήματα για το σενάριο από μας.

Η λύση

Όπως ο σχεδιασμός του προγράμματος μας είναι πλέον αρκετά σταθερός, μπορούμε να γράψουμε τον κώδικα η οποία είναι μια υλοποίηση της λύσης μας.

```
#!/usr/bin/python
# Filename: backup_ver1.py

import os
import time

#1. Τα αρχεία και οι κατάλογοί για να γίνουν αντίγραφα ασφαλείας προσδιορίζονται σε μια λίστα.
Source = ['"C:\\my documents"', 'c:\\code']
# σημειώστε ότι πρέπει να χρησιμοποιήσουμε δίπλες αγκύλες μέσα σε ένα string για ονόματα με κενά ανάμεσα του.

#2. τα αντίγραφα ασφαλείας πρέπει να αποθηκεύονται στο κύριο αντίγραφο καταλόγων
target_dir = 'E:\\Backup'  # να θυμάστε για να αλλάξετε αυτό σε αυτό που θα χρησιμοποιήσετε

#3. Αυτά τα αρχεία αντιγράφων ασφαλείας είναι σε αρχεία zip.
#4. Το όνομα του zip αρχείου είναι η τρέχουσα ημερομηνία και ώρα.
Target = target_dir + os.sep +time.strftime('%Y%m%d%H%M%S') + ' .zip'

#5. Χρησιμοποιούμε την zip εντολή για βάλουμε τα αρχεία σε ένα αρχείο zip
zip_command = "zip -qr {0} {1}". format(target, ' '.join(source))

# Τρέχουμε το αντίγραφο ασφάλειας
if os.system(zip_command)== 0:
        print('successful backup to', target)
else:
        print('backup FAILED')
```

Έξοδος:

```
$ python backuo_ver1.py
Successful backup to E:\backup\20080702185040.zip
```

Τώρα, βρισκόμαστε στην δοκιμαστική φάση όπου βλέπουμε αν το πρόγραμμα δουλεύει κατάλληλα. Αν δεν συμπεριφέρεται όπως περιμένουμε, τότε πρέπει να κάνουμε debug στο πρόγραμμα π.χ. Αφαιρέστε τα λάθη από το πρόγραμμα.
Αν το παραπάνω πρόγραμμα δεν δουλέψει για εσένα, βάλτε το print(zip_command) πριν το os.system καλέστε και τρέξτε το πρόγραμμα. Τώρα κάντε αντιγραφή/επικόλληση και εκτυπώστε zip_command στην προτροπή του κελύφους και δείτε αν δουλεύει σωστά από μόνο του. Αν αυτή η εντολή αποτύχει, ελέγξτε την zip command εσείς για να δείτε τι είναι λάθος. Αν αυτή εντολή είναι σωστή, τότε ελέγξτε το πρόγραμμα της Python αν ταιριάζει ακριβώς με το πρόγραμμα που γράφεται παραπάνω.

Πως δουλεύει:
Θα παρατηρήσατε πως έχουμε μετατρέψει το δικό μας σε κώδικα με ένα τρόπο βήμα-βήμα. Κάνουμε χρήση της os και ενότητες από την πρώτη στιγμή στην εισαγωγή τους. Τότε, προσδιορίζουμε τους φακέλους και τους κατάλογους για αντίγραφα ασφαλείας στην πηγή λίστα. Ο κατάλογος στόχος είναι εκεί που αποθηκεύετε όλα τα αρχεία των αντιγράφων ασφάλειας και αυτό προσδιορίζεται στην target_dir μεταβλητή. Το όνομα του zip αρχείο που θα δημιουργήσουμε θα ειναι στην τωρινή ήμερα και ώρα όπου βρήκα να χρησιμοποιούμε την function time.strftime(). Επίσης θα έχει την zip. επέκταση και θα αποθηκεύει στην trget_dir κατάλογο. Σημειώστε την χρήση της μεταβλητής os.sep – αυτό δίνει στον κατάλογο διαχωριστή, σύμφωνα με το λειτουργικό σας σύστημα π.χ. Θα γίνει '/' στην Linux,Unix, θα είναι '\\' στα Windows και ':' στην Mac OS. Χρησιμοποιώντας την os.sep σε αντίθεση αυτών των χαρακτήρων κατευθείαν θα κάνουν το πρόγραμμα να ανακοινώνεται στο χώρο του και δουλεύει σε αυτά τα συστήματα. Το time.strftime() function παίρνει μια προδιαγραφή όπως αυτή που έχουμε χρησιμοποιήσει στο παραπάνω πρόγραμμα. Ο προσδιορισμός %Y θα αντικατασταθεί από το έτος χωρίς τον αιώνα. Ο προσδιορισμός %m θα αντικατασταθεί με τον μηνά σε ένα δεκαδικό αριθμό μεταξύ του 01 και 12 και ούτω κάθεξής. Η ολοκληρωμένη λίστα από τέτοιους προσδιορισμούς μπορεί να βρεθεί στην Python Reference Manual(http://docs.python.org/dev/3.0/library/time.html#time.strftime). Δημιουργούμε ένα όνομα για το target zip αρχείο χρησιμοποιώντας τον τελεστή της πρόσθεσης όπου συνενώνει τα string π.χ. Ενώνει τα δυο string μαζί και επιστρέφει ένα νέο. Μετά, δημιουργούμε ένα string zip_command όπου περιέχει την εντολή που θα εκτελέσουμε. Μπορείτε να δείτε αν δουλεύει η εντολή με το να τρέξεις στο κέλυφος (Linux terminal ή DOC prompt). Η zip εντολή όπου χρησιμοποιούμε έχει κάποιες επιλογές και παραμέτρους. Η επιλογή -q χρησιμοποιείτε για να υποδείξει ότι η εντολή zip θα πρέπει να δουλέψει **q**uietly. Η επιλογή -r προσδιορίζει ότι η εντολή zip θα πρέπει να δουλέψει **r**ecursively για τους κατάλογους π.χ. θα πρέπει να συμπεριλάβει όλους τους υπό φακέλους και τα αρχεία. Δυο επιλογές συνδυάζονται και προσδιορίζονται με συντομεύσεις όπως -qr. Οι επιλογές ακολουθούνται με το όνομα του zip αρχείου για να δημιουργήσει ακολουθούμενο από την λίστα των φακέλων και των καταλογών για την δημιουργία αντιγράφων ασφάλειας. Μετατρέπουμε την πηγή λίστα σε ένα string χρησιμοποιώντας την μέθοδο join τον strings που έχουμε ήδη δει πώς να τα χρησιμοποιούμε. Μέτα, θα τρέξουμε στο τέλος την εντολή χρησιμοποιώντας os.system function η οποία τρέχει την εντολή σαν να είχε εκτελεστεί από το σύστημα π.χ. Στο κέλυφος – επιστρέφει 0 αν η εντολή είναι επιτυχής, αλλιώς επιστρέφει ένα αριθμό error. Ανάλογα με το αποτέλεσμα της εντολής, εμφανίζουμε το κατάλληλο μήνυμα ότι τα αντίγραφα ασφάλειας είναι επιτυχής ή αποτυχημένα. Αυτό είναι, έχουμε δημιουργήσει ένα κείμενο για να κάνουμε ένα αντίγραφο ασφάλειας για τα σημαντικά μας αρχεία.!

Σημείωση για τους χρήστες Windows
Αντίθετα με τις διπλές κάθετους ακολουθία διαφυγής, μπορείτε επίσης να χρησιμοποιήσετε μια σειρά από strings. Για παράδειγμα, χρησιμοποιείστε 'C:\\Documents' ή '**C:\Documents**'. Ωστόσο, μην χρησιμοποιήσετε το '**C:\Documents**' εφόσον έχετε καταλήξει να χρησιμοποιήσετε μια άγνωστη ακολουθία διαφυγής \D. Τώρα που έχουμε ένα λειτουργικό αντίγραφο ασφάλειας, μπορούμε να το χρησιμοποιήσουμε όποτε θελήσομε για να κάνουμε αντίγραφα ασφάλειας από αρχεία. Οι Linux/Unix χρήστες συνιστάται να χρησιμοποιήσετε την εκτελέσιμο μέθοδο όπως έχουμε συζητήσει πριν έτσι ώστε να μπορεί να τρέξει το σενάριο δημιουργίας αντιγράφων ασφαλείας οποτεδήποτε, οπουδήποτε. Αυτό ονομάζεται φάση λειτουργίας ή ανάπτυξη φάσης του λειτουργικού. Το προηγούμενο πρόγραμμα δουλεύει κατάλληλα, άλλα συνήθως τα πρώτα προγράμματα δεν δουλεύουν όπως ακριβώς θέλουμε. Για παράδειγμα, θα υπάρχουν προβλήματα αν δεν έχετε σχεδιάσει το πρόγραμμα κατάλληλα ή αν έχετε κάνει λάθος στην πληκτρολόγηση του κώδικα. Κατάλληλα, θα πρέπει να ανατρέξετε πίσω στην σχεδιαστική φάση ή θα πρέπει να κάνουμε debug στο πρόγραμμα σας.

Δεύτερη έκδοση

Η πρώτη εκδοσή του κείμενου μας δουλεύει. Ωστόσο, μπορείτε να κάνετε κάποιες βελτιώσεις για να δουλεύει καλύτερα ή σε καθημερινή εκδοσή. Αυτό λέγετε φάση συντήρησης του λειτουργικού. Μια από τις βελτιώσεις που πιστεύω πως είναι χρήσιμη είναι ένας καλύτερος μηχανισμός ονομασία φακέλων – χρησιμοποιώντας τον χρόνο σαν όνομα του φακέλου μέσα στον κατάλογο μαζί με την τωρινή ημερομηνία σαν κατάλογος μέσα στο κεντρικό κατάλογο αντιγράφων

ασφάλειας. Πρώτο πλεονέκτημα είναι ότι τα αντίγραφα ασφάλειας αποθηκεύονται με ιεραρχικό τρόπο και ως εκ τούτου είναι πιο εύκολο να τα ελέγχετε. Δεύτερο πλεονέκτημα είναι ότι το μήκος του ονόματος του αρχείου είναι πιο μικρο. Τρίτο πλεονέκτημα είναι ότι η ξεχωριστή καταλογή θα σας βοηθήσουν πιο εύκολα να ελέγχετε αν έχετε κάνει αντίγραφα ασφάλειας για κάθε μέρα εφόσον ο κατάλογος θα δημιουργείται μόνο αν έχετε κάνει ένα αντίγραφο ασφάλειά για κάθε μέρα.

```
#!/usr/bin/python
# Filename: backup_ver2.py

import os
import time

#1. Τα αρχεία και τους καταλόγους να είναι αντίγραφο ασφαλείας που καθορίζονται σε μια λίστα
source = ['"C:\\My Documents"', 'C:\\Code']
# παρατηρήστε είχαμε τη χρήση διπλών εισαγωγικών εντός τα string για ονόματα αρχείων με κενά διαστήματα σε αυτήν.
#2. Τα αντίγραφα ασφαλείας πρέπει να αποθηκευτούν σε ένα κεντρικό φάκελο
target_dir = 'E:\\Backup' # θυμηθείτε να το αλλάξετε σε αυτό που θα πρέπει να χρησιμοποιούν

#3. Τα αρχεία αντιγράφων ασφαλείας είναι σε ένα αρχείο zip
#4. Η σημερινή ημέρα είναι το όνομα του υπό φακέλου στον κεντρικό κατάλογο
today = target_dir + os.sep + time.strftime('%Y%m%d')
# η τρέχουσα ώρα είναι το όνομα του συμπιεσμένου αρχείου
now = time.strftime('%H%M%S')

# Δημιουργήσετε το υποκατάλογο, αν δεν είναι ήδη εκεί
if not os.path.exists (today):
        os.mkdir(today) # κάντε ένα κατάλογο
        print ('succesfull created directory', today)
# Το όνομα του αρχείου zip
target = today + os.sep + now + '.zip'

#5. Χρησιμοποιούμε την εντολή zip για να βάλει τα αρχεία σε ένα αρχείο zip
zip_command = "zip -qr {0} {1} ".format(target, ' '.join(source))
# Εκτελέσετε το αντίγραφο ασφαλείας
if os.system(zip_command)== 0:
        print('successful backup to', target)
else:
        print('Backup FAILED')
```

Έξοδος:

```
$ python backup_ver2.py
successfully created directory E:\backup\20080702
successful backup to E:\backup\20080702\202311.zip

$ python backup_ver2.py
successful backup to E:\Backup\20080702\202325.zip
```

Πως δουλεύει:
Το πιο σημαντικό πρόγραμμα παραμένει το ίδιο. οι αλλαγές είναι ότι ελέγχουμε αν υπάρχει ένας κατάλογος με την τρέχουσα ημέρα ως όνομα μέσα στο κεντρικό φάκελο χρησιμοποιώντας το os.path.exists finction. Αν δεν υπάρχει, μπορούμε να δημιουργήσουμε χρησιμοποιώντας το os.mkdir function.

Τρίτη έκδοση

Η δεύτερη έκδοση δουλεύει μια χαρά όταν κάνω πολλά αντίγραφα ασφάλειας, άλλα όταν υπάρχουν πλήθη από αντίγραφα ασφάλειας, το βρίσκω δύσκολο να διαφοροποιήσω τα αντίγραφα ασφάλειας! Για παράδειγμα, μπορεί να έχω κάνει μεγάλες αλλαγές στο πρόγραμμα ή στην παρουσίασή, μετά θέλω να συνδυάσω τι είναι αυτές η αλλαγές μαζί με το όνομα αυτού του zip αρχείου. Αυτό μπορεί εύκολα να γίνει με το να επισυναψω ένα παρεχόμενο από τον χρήστη σχόλιο για το όνομα του zip αρχείου.

Σημείωση

Το ακόλουθο πρόγραμμα δεν δουλεύει, γι 'αυτό μην ανησυχήσετε, παρακαλώ παρακολουθείστε γιατί υπάρχει ένα μάθημα εδώ.

```
#!/usr/bin/python
# Filename: backup_ver3.py

import os
import time

#1. τα αρχεία και τους καταλόγους που είναι αντίγραφα ασφαλείας καθορίζονται σε μια λίστα
source = ['"C:\\My Documents"', 'C:\\Code']
# Σημείωση πρέπει να χρησιμοποιούμε διπλά εισαγωγικά μέσα σε ένα string για ονόματα με κενά ενδιάμεσα τους.

#2. Τα αντίγραφα ασφαλείας πρέπει να αποθηκευτούν σε ένα κεντρικό κατάλογο
target_dir = 'E:\\Backup' # να θυμάστε να αλλάζετε αυτό με αυτό που θα χρησιμοποίησή

#3. Τα αρχεία αντιγράφων ασφαλείας είναι σε ένα αρχείο zip

#4. Η σημερινή ημέρα είναι το όνομα του υπο φακέλου στον κεντρικό κατάλογο
today = target_dir + os.sep + time.strdtime('%Y%m%d')
# Η τωρινή ώρα είναι το όνομα του zip αρχείου.
Now = time.strftime('%H%M%S')
#  Κάντε ένα σχόλιο για τον χρήστη ώστε να δημιουργήσει το όνομα του zip αρχείου.
Comment = input('enter a comment-->')
if len (comment) == 0 : # ελέγξτε αν το σχόλιο έχει συμπεριφερθεί
        target = today + os.sep + now + '.zip'
else:
        target = today + os.sep + now + '_' +
        comment.replace(' ','_') + ' .zip'
# δημιουργήστε υποκατάλογο αν δεν υπάρχει ήδη.
If not os.path.exists(today):
        os.mkdir(today) # κάντε ένα κατάλογο
        print('successfully created directory', today)

#5. Χρησιμοποιούμε την εντολή zip για να βάλει τα αρχεία σε ένα αρχείο zip
zip_command = "zip -qr {0} {1} ".format(target,' ' .join(source))

#τρέξτε τα αντίγραφα ασφάλειας
if os.system(zip_command) ==0:
        print('successful backup to',target)
else:
        print('backup FAILED')
```

Έξοδος:

```
$ python backup_ver3.py
File "backup_ver3.py",line 25
```

```
        target = today + os.sep + now + '_' +
```

SyntaxError: invalid syntax

Πως (δεν) δουλεύει:
Αυτό το πρόγραμμα δεν δουλεύει! Η Python λέει ότι έχετε ένα συντακτικό λάθος αυτό σημαίνει ότι το σενάριο δεν ικανοποιεί την δομή της Python που περιμένεις να δεις. Όταν παρατηρούμε το λάθος που μας δίνει η Python, μας λέει επίσης το μέρος που ανιχνεύτηκε το λάθος. Οπότε ξεκινάμε να κάνουμε debugging από εκείνη την γραμμή. Με μια προσεκτική παρατήρηση, βλέπουμε ότι η μόνες γραμμές έχουν χωριστεί σε δυο φυσικές γραμμές άλλα δεν έχουμε προσδιορίσει ότι αυτές οι δυο φυσικές γραμμές ανήκουν μαζί. Βασικά, η Python έχει βρει τον τελεστή της πρόσθεσης (+) χωρίς κανένα τελεστή στην λογική γραμμή και ως εκ τούτου, δεν ξέρει πως να συνεχίσει. Να θυμάστε ότι μπορούμε να προσδιορίσουμε ότι η λογική γραμμή συνεχίζει στην επομένη φυσική γραμμή με το να χρησιμοποιούμε την κάθετη στο τέλος της φυσική γραμμή. Όποτε, κάνουμε αυτή την διόρθωση στο πρόγραμμα μας. **Αυτή η διόρθωση του προγράμματος γίνετε όταν βρίσκουμε λάθη και λέγεται bug fixing.**

Η τέταρτη έκδοση

```
#!/usr/bin/python
# Filename: backup_ver4.py

import os
import time
#1. τα αρχεία και τους καταλόγους που είναι αντίγραφα ασφαλείας καθορίζονται σε μια λίστα
source = ['"C:\\My Documents"', 'C:\\Code']

# Σημείωση πρέπει να χρησιμοποιούμε διπλά εισαγωγικά μέσα σε ένα string για ονόματα με κενά ενδιάμεσα τους.

#2. Τα αντίγραφα ασφαλείας πρέπει να αποθηκευτούν σε ένα κεντρικό κατάλογο
target_dir = 'E:\\Backup' # να θυμάστε να αλλάζετε αυτό με αυτό που θα χρησιμοποιήσουμε

#3. Τα αρχεία αντιγράφων ασφαλείας είναι σε ένα αρχείο zip

#4. Η σημερινή ημέρα είναι το όνομα του υποφακέλου στον κεντρικό κατάλογο
today = target_dir + os.sep + time.strdtime('%Y%m%d')

# Η τωρινή ώρα είναι το όνομα του zip αρχείου.
Now = time.strftime('%H%M%S')

#  Κάντε ένα σχόλιο για το χρήστη ώστε να δημιουργήσει το όνομα του zip αρχείου.
Comment = input('enter a comment-->')
if len (comment) == 0 : # ελέγξτε αν το σχόλιο έχει συμπεριφερθεί
        target = today + os.sep + now + '.zip'
else:
        target = today + os.sep + now + '_' +
        comment.replace(' ','_') + ' .zip'
# δημιουργήστε υποκατάλογο αν δεν υπάρχει ήδη.
If not os.path.exists(today):
        os.mkdir(today) # κάντε ένα κατάλογο
        print('successfully created directory', today)

#5. Χρησιμοποιούμε την εντολή zip για να βάλει τα αρχεία σε ένα αρχείο zip
zip_command = "zip -qr {0} {1} ".format(target,' ' .join(source))
```

```
#τρέξτε τα αντίγραφα ασφάλειας
if os.system(zip_command) ==0:
        print('successful backup to',target)
else:
        print('backup FAILED')
```

Έξοδος:
```
$ python backup_ver4.py
enter a commend → added new examplew
successful backup to
E:\backup\20080702\202836_added_new_examples.zip

$ python backup_ver4.py
enter a commend →
successful backup to E:\backup\20080702\202839.zip
```

Πως δουλεύει:
Αυτό το πρόγραμμα δουλεύει! Ας περάσουμε να δούμε τις πραγματικές βελτιώσεις που έχουμε κάνει στην εκδοσή 3. Πάμε στα σχόλια του χρήστη χρησιμοποιώντας την input function και ελέγχουμε αν όντως ο χρήστης εισήγαγε κάτι με το να βρούμε το μήκος του χρησιμοποιώντας την len function. Αν ο χρήστης έχει πατήσει το enter χωρίς να εισάγουμε οτιδήποτε (ίσως ήταν απλά ένα αντίγραφο ρουτίνας ή δεν υπήρξαν κάποιες ιδιαίτερες αλλαγές), μετά συνεχίζουμε όπως έχουμε κάνει και πριν. Ωστόσο, αν παρέχεται ένα σχόλιο, τότε αυτό επισυνάπτεται στο όνομα του zip αρχείου πριν την επέκταση .zip. Σημειώστε ανταλλάσσουμε τα κενά με σχόλια μαζί με να το υπογραμμίζουμε – αυτό γίνετε γιατί είναι πιο εύκολο να ελέγξουμε τα ονόματα των αρχείων χωρίς κενά.

Περισσότερες βελτιώσεις

Η τέταρτη έκδοση δουλεύει ικανοποιητικά το σενάριο για τους πιο πολλούς χρήστες, άλλα πάντα υπάρχει χορός για βελτίωση. Για παράδειγμα, μπορείτε να εισάγετε ένα επίπεδο πολυλογίας για το πρόγραμμα όπου μπορείτε να προσδιορίσετε ένα -v επιλογή ώστε το πρόγραμμα να γίνει πιο ομιλητικό. Άλλες θετικές ενισχύσεις θα μπορούσαν να αφήσουν επιπλέον αρχεία και κατάλογους για να περαστούν στο σενάριο στην εντολή γραμμών. Μπορούμε να έχουμε αυτά τα ονόματα από το sys.argv λίστες και μπορούμε να προσθέσουμε στην δικιά μας λίστα scource χρησιμοποιώντας την extend μέθοδο που παρέχετε από την λίστα της κλάσης.
Οι πιο σημαντικές βελτιώσεις δεν θα χρησιμοποιούν τον τρόπο os.system για την δημιουργία αρχείων και σε αντίθεση χρησιμοποιούν το zipfile ή tarfile ενσωματωμένο πρότυπο για την δημιουργίας αυτών των αρχείων. Είναι κομμάτι από την πρότυπη βιβλιοθήκη και είναι διαθέσιμη ήδη για να χρησιμοποιείται χωρίς εξωτερικές εξαρτήσεις στο zip πρόγραμμα για να είναι διαθέσιμο στο πρόγραμμα σας. Ωστόσο, έχω χρησιμοποιήσει τον τρόπο os.system για την δημιουργία αντιγράφων ασφάλειας στο παραπάνω παράδειγμα καθαρά για παιδαγωγικούς σκοπούς, όποτε το παράδειγμα είναι αρκετό για να καταλάβουμε από οποιονδήποτε άλλα είναι πραγματικά αρκετό για να είναι χρήσιμο. Μπορείτε να προσπαθήσετε να γράψετε την πέμπτη εκδοχή οπου χρησιμοποιούμε το zipfile (http://docs.python.org/dev/3.0/library/zipfile.html) πρότυπο αντί για να καλέσουμε os.system ?

Η διαδικασία ανάπτυξης λογισμικού

Τώρα έχουμε πάει για να περάσουμε τις σημαντικές φράσεις για να γράψουμε την προοπτική για να γράψουμε το λειτουργικό. Αυτές οι φράσεις μπορούν να περιληφθούν όπως παρακάτω:

1. Τι(ανάλυση)
2. Πως (σχεδιασμός)
3. Κάντε το(εκτέλεση)
4. δοκιμή (δοκιμές και debugging)
5. χρήση (λειτουργία ή ανάπτυξη)
6. διατήρηση (διύλιση)

Ένας προτιμόμενος τρόπος για να γράψουμε το πρόγραμμα είναι η διαδικασία που έχουμε ακολούθήσει για να δημιουργήσουμε το σενάριο για τα αντίγραφα ασφάλειας: κάντε την ανάλυση και τον σχεδιασμό. Ξεκινήστε αμέσως με μια απλή έκδοση. Δοκιμάστε και κάντε debug. Χρησιμοποιείστε το για να δείτε ότι δουλεύει όπως περιμέναμε. Τώρα, προσθέστε οποιοδήποτε χαρακτηριστικό που θέλετε και συνεχίσετε να επαναλαμβάνετε το αυτό-δοκιμή-χρηση κύκλος όσες πιο πολλές φόρες χρειάζεται να θυμάστε, το λειτουργικό μεγαλώνει, όχι να είναι ενσωματωμένο.

Περίληψη

Έχουμε δει πως να δημιουργούμε το δικό μας πρόγραμμα/σενάριο της python και το σημαντικό στάδιο περιλαμβάνει το πως γράφουν τέτοια προγράμματα. Μπορεί να σας φανεί χρήσιμο να δημιουργήσετε το δικό σας πρόγραμμα όπως έχουμε κάνει σε αυτό το κεφάλαιο ώστε να γίνετε ποιο οικείο με την python όπως στο πρόβλημα – λύση. Μέτα, θα το συζητήσουμε το προγραμματισμό του αντικείμενο – προσανατολισμό.

Python en: Αντικειμενοστραφής Προγραμματισμός

Εισαγωγή:

Σε όλα τα προγράμματα που έχουμε γράψει μέχρι τώρα, έχουμε σχεδιάσει τα προγράμματα μας γύρω από την function. π.χ. block εντολών που χειραγωγούν τα δεδομένα. Αυτό λέγεται αντικειμενοστραφής τρόπος προγραμματισμού. Υπάρχει και ένας άλλος τρόπος για να οργανώσουμε το πρόγραμμα όπου περιεχέι δεδομένα και λειτουργικότητα τυλιγμένα μέσα σε κάτι το οποίο λέγεται αντικείμενο. Αυτό λέγεται παράδειγμα αντικειμενοστραφή προγραμματισμού. Το περισσότερο χρόνο μπορείτε να χρησιμοποιήσετε διαδικαστικό προγραμματισμό, άλλα όταν γράφετε μεγάλα κομμάτια προγραμματισμού ή έχετε ένα πρόβλημα τότε αυτή είναι η μέθοδος κατάλληλη, μπορείτε να χρησιμοποιείτε τη τεχνική του αντικειμενοστραφή προγραμματισμού. Κλάσης και αντικείμενα είναι οι δυο κεντρικές πτυχές στον αντικειμενοστραφή προγραμματισμό. Μια κλάση δημιουργεί ένα νέο τύπο όπου τα αντικείμενα είναι περιπτώσεις των κλάσεων. Ανάλογο είναι ότι μπορείς να έχεις μεταβλητές του τύπου int όπου μεταφράζει ότι οι μεταβλητές που αποθηκεύουν ακεραίους είναι μεταβλητές όπου είναι περιπτώσεις (αντικειμένων) από την int κλάση.

Σημείωση για προγραμματιστές στατικών γλωσσών

Σημείωση ότι ακόμα και η ακέραιοι αντιμετωπίζονται σαν αντικείμενα (από την int κλάση). Αυτό είναι σε αντίθεση με την C++ και την Java (πριν την έκδοση την 1,5) όπου η ακέραιοι είναι αρχικοί ντόπιοι τύποι. δείτε στο help(int) για περισσότερες πληροφορίες για την κατηγορία. Οι προγραμματιστές τον C# και Java 1.5 θα το βρουν παρόμοιο με την boxing and unboxing έννοια. Αντικείμενα μπορούν να αποθηκεύουν δεδομένα χρησιμοποιώντας συνηθισμένες μεταβλητές όπου ανήκουν στο αντικείμενο. Μεταβλητές που ανήκουν σε ένα αντικείμενο ή κατηγόρια αναφέρονται σε εσάς σαν πεδία. Αντικείμενα μπορούν επίσης να έχουν λειτουργικότητα με το να χρησιμοποιούν function όπου ανήκουν στην κατηγορία. Τέτοιες functions ονομάζονται μέθοδοί των κατηγοριών. Τέτοιες ορολογίες είναι σημαντικές γιατί μας βοηθάνε να διακρίνουμε μεταξύ function και μεταβλητών όπου είναι ανεξάρτητες και σε αυτά που ανήκουν σε μια κατηγόριες ή ένα αντικείμενο. Συλλογικά τα πεδία και οι μέθοδοι μπορούν να αναφέρονται ως τα χαρακτηριστικά. Τα πεδία είναι δυο τύπων – μπορούν να ανήκουν σε κάθε παράδειγμα/αντικείμενο της κατηγορίας ή μπορεί να ανήκουν στις κατηγορίες από μόνες τους. Ονομάζονται παραδείγματα μεταβλητών και κατηγορίες μεταβλητών αντίστοιχα. Μια κλάση χρησιμοποιεί την λήξει-κλειδί class. Τα πεδία και οι μέθοδοί της κατηγορίας είναι στην εσοχή block της λίστα.

To self

Η κατηγορία μέθοδος έχει μόνο μια συγκεκριμένη διάφορα από συνηθισμένα functions – πρέπει να έχει επιπλέον το πρώτο όνομα που πρέπει να προσθέσει για να ξεκινήσουμε την λίστα της παραμέτρου, άλλα δεν δίνετε μια τιμή για αυτήν την παράμετρο όταν καλείς την μέθοδο, η python θα το παρέχει μόνο του. Αυτές οι πρακτικές μεταβλητές προτιμάνε το ίδιο το αντικείμενο, του έχει δοθεί το όνομα self. Παρόλα αυτά, μπορείτε να δώσετε ότι όνομα θέλετε για αυτή την παράμετρο, συνιστάτε να χρησιμοποιείτε το όνομα self – οποιοδήποτε άλλο όνομα είναι διαφορετικό αποδοκιμάζετε. Υπάρχουν πολλά πλεονεκτήματα για χρησιμοποιήσετε ένα πρότυπο όνομα –

οποιοσδήποτε διαβάσει το πρόγραμμα αμέσως θα το αναγνωρίσει και ακόμα πιο ειδικευμένα IDEs(integrated development environments) μπορεί να σας βοηθήσει αν χρησιμοποιείτε το self.

Σημείωση για τους προγραμματιστές τον C++/Java/C#
Η self στην Python είναι ισοδύναμη σε αυτόν τον δείκτη στην C++ και αυτό αναφέρετε και στην Java και στην C#. θα πρέπει να αναρωτιέστε πως η python δίνει την τιμή για την self και γιατί δεν χρειάζεται να δίνετε τιμή για αυτή. Ένα παράδειγμα θα το κάνει αυτό ξεκάθαρο. Ας πούμε ότι έχετε μια κατηγορία που ονομάζεται MyClass και ένα παράδειγμα για αυτήν την κατηγορία ονομάζετε myobject. Όταν καλείται μια μέθοδο αυτού το αντικείμενο σαν myobject.method(arg1, arg2),τότε αυτό αυτόματα μετατρέπει από την Python σε MyClass.method(myobject, arg1, arg2) – αυτό είναι όλο για την self.
Αυτό επίσης σημαίνει ότι όταν έχεις μια μέθοδο όπου δεν παίρνει επιχειρήματα, τότε ακόμα χρειάζεται να έχουμε ένα επιχείρημα – η self.

Κατηγορίες
Η πιο απλή κατηγορία φαίνεται στο παρακάτω παράδειγμα.

```
#!/usr/bin/python
# Filename: simplestclass.py

class Person:
        pass # ένα άδειο block
p = Person()
print(p)
```

Έξοδος:

```
$ python simplestclass.py
<__main__.person object at 0x019F85F0>
```

Πως δουλεύει:
Δημιουργούμε μια κατηγορία χρησιμοποιώντας την class δήλωση και το όνομα της κατηγορίας. Αυτό ακολουθείτε από οδοντωτά block από δηλώσεις από το κορμί της κατηγορίας. Σε αυτή την περίπτωση, έχουμε ένα άδειο block όπου δηλώνεται με την χρησιμοποίησή της pass δήλωσης. Μέτα, δημιουργούμε ένα αντικείμενο/παράδειγμα από αυτήν την κατηγορία χρησιμοποιούμε το όνομα της κατηγορίας με ένα ζευγάρι από παρενθέσεις.(θα μάθουμε περισσότερα για την συγκεκριμενοποίηση στο επόμενο κεφάλαιο). Για την δικιά μας επαλήθευση, επιβεβαιώνουμε το τύπο της μεταβλητής με το να την εκτυπώσουμε απλά. Μας λέει ότι έχουμε ένα παράδειγμα από την Person κατηγορία στην __main__ πρότυπο. Σημειώστε ότι η διεύθυνση της μνήμης του ηλεκτρονικού υπολογιστή όπου τα αντικείμενα μας αποθηκεύονται επίσης εκτυπώνονται. Η διεύθυνση θα έχει μια διαφορετική τιμή στον ηλεκτρονικό υπολογιστή εφόσον η Python μπορεί να αποθηκεύσει τα αντικείμενα οπουδήποτε βρίσκει κενό.

Μέθοδο Αντικείμενων
Έχουμε ήδη συζητήσει την κατηγορία/αντικειμένων που μπορούμε να έχουμε μεθόδους όπως η function αυτό σημαίνει ότι έχουμε μια επιπλέον μεταβλητή self. Τώρα θα δούμε ένα παράδειγμα.

```
#!/usr/bin/python
# Filename: method.py

class Person:
def sayHi(self):
        print('Hello,how are you?')

p = person()
```

```
p.sayHi()

# το σύντομο παράδειγμα μπορεί να γραφτεί ως Person.()sayHi()
```

Έξοδος:

```
$ python method.py
Hello,how are you?
```

Πως δουλεύει:
Εδώ βλέπουμε την self σε δράση. Σημειώστε ότι η sayHi μέθοδο δεν παίρνει παραμέτρους αλλά ακόμα έχει τον ορισμό self function.

Η μέθοδο __int__

Υπάρχουν πολλά ονόματα μεθόδων οπού έχουνε ιδιαίτερη σημασία στην κατηγορία της Python. θα δούμε την σημασία της μεθόδου __int__ τώρα. Η __int__ τρέχει αμέσως μόλις ένα αντικείμενο αποστασιοποιείται. Η μέθοδος είναι χρήσιμη για να κάνετε οποιαδήποτε αρχικοποίηση που θέλετε να κάνετε στο αντικείμενο σας. Σημείωστε την διπλή κάτω παύλα και για τα δυο στην αρχή στο τέλος του ονόματος τους.

Παράδειγμα:

```
#!/usr/bin/python
# Filename: class_int.py

class Person:
        def __int__(self,name):
                self.name = name
        def sayHi(self):
                print('Hello,my name is',self.name)
p = Person('Swaroop')
p.sayHi()

# αυτό το σύντομο παράδειγμα μπορεί να γραφτεί ως person('swaroop').sayHi()
```

Έξοδος:

```
$ python class_int.py
Hello, my name is Swaroop
```

Πως δουλεύει:
Εδω, ορίζουμε την __int__ μέθοδο όπως το να λέμε το όνομα της παραμέτρου (μαζί με το συνηθισμένο self) Εδώ, μόλις δημιουργήσαμε ένα νέο πεδίο που επίσης ονομάζετε name. Σημείωση, υπάρχουν δυο διαφορετικές μεταβλητές παρόλο που ονομάζονται και οι δυο 'name'. Ο συμβολισμός με τελείες μας επιτρέπει να δούμε την διάφορα μετάξι τους. Πιο σημαντικά, σημειώστε ότι δεν καλούμε ρητά την __int__ μέθοδο άλλα περνάμε τα επιχειρήματα στις παρενθέσεις ακολουθώντας τις κατηγορίες των ονομάτων όταν δημιουργούνε ένα νέο παράδειγμα από κατηγορίες. Αυτό είναι σημαντική σημείωση για αυτή την μέθοδο. Τώρα, μπορούμε να χρησιμοποιήσουμε την self.name πεδίο στην μέθοδο σας όπου δείχνετε στην seyHi μέθοδο.

Μεταβλητές κατηγορίας και αντικείμενων

Έχουμε ήδη συζητήσει την λειτουργικότητα των κατηγοριών και των αντικειμένων(π.χ. Μεθόδων), τώρα, ας μάθουμε για το κομμάτι των δεδομένων. Το κομμάτι των δεδομένων, π.χ. Πεδία, δεν είναι τίποτα άλλο από συνηθισμένες μεταβλητές που το όριο του είναι **namespaces** των αντικειμένων και των κατηγοριών. Αυτό σημαίνει ότι αυτά τα ονόματα είναι έγκυρα μεταξύ των περιεχομένων των κατηγορητητηρίων και των αντικειμένων μόνο. Για αυτό λέγονται name spaces. Υπάρχουν δυο τύποι μεταβλητών πεδιών – κατηγοριών και αντικειμένων μεταβλητών όπου ανάλογα

διασαφηνίζονται σε ποιές κατηγορίες ή στο αντικείμενο ανήκει η μεταβλητή αντίστοιχα. Κατηγορίες μεταβλητών μοιράζονται – μπορούν να είναι προσβάσιμες με άλλες περιπτώσεις αυτής της κατηγορίας. Υπάρχει μόνο ένα αντίγραφο από τις κατηγορίες των μεταβλητών και όταν οποιαδήποτε αντικείμενο κάνει μια αλλαγή στην κατηγορία μεταβλητών, ότι η αλλαγή θα είναι εμφανής σε όλες τις άλλες περιπτώσεις. Οι μεταβλητές αντικειμένων ανήκουν σε κάθε μεμονωμένο αντικείμενο/παράδειγμα των κατηγοριών. Σε αυτή την περίπτωση, κάθε αντικείμενο έχει το δικό του αντίγραφο των πεδίων του π.χ. Δεν μοιράζονται και δεν σχετίζονται σε οποιοδήποτε τρόπο στο πεδίο με το ίδιο όνομα σε διαφορετικά παραδείγματα. Ένα παράδειγμα θα σας το κάνει εύκολο για να το καταλάβει:

```
#!/usr/bin/python
# Filename:objvar.py

class Rebot:
        ''' Represents a robot, with a name.'''
# Μια κατηγορία μεταβλητό, μετράνε τους αριθμούς τον ρομπότ
population = 0
def __int__(self,name):
''' initializes the data.'''
self.name = name
print('(initializing{0})'.format(self.name))

# όταν αυτός ο άνθρωπος δημιουργεί το ρομπότ
# προσθετή στο population
Robot.population += 1
def __def__(self)
''' I am dying '''
        print('{0} is being destroyed!'.format(self.name))

Robot.population -= 1
if Robot.population == 0:
        print('{0} was the last one.'.format(self.name))
else:
        print(there are still {0:d}robots working.'format(robot.population))

def sayHi(self):
''' Greeting by robot.
Yeah, they can do that.'''
print('greetings, my masters call me {0}.'.format(self.name))

def howMany():
''' Print the current population.'''
print('we have {0:d} robots.'.format(robot.population))
howMany = staticmethod(howMany)

droid1 = robot('R2-D2')
droid1.sayHi()
robot.howMany()

droid2 = robot('C-3PO')
droid2.sayHi()
robot.howMany()

print(“\n robots can do some work here.\n”)

print(“robots have finished their work. So let's destroy them.”)
```

```
del droid1
del droid2

robot.howMany()
```

Έξοδος:

```
(initializing R2-D2)
greetings,my masters call me R2-D2.
We have 1 robots.
(initializing C-3PO)
greetings, my master call me C-3PO.
We have 2 robots.
Robots can do some work here.
Robots have finished their work. So let's destroy them.
R2-D2 is being destroyed!
There are still 1robots working.
C-3PO is being destroyed!
C-3PO was the last one.we have 0 robots.
```

Πως δουλεύει:

Αυτό είναι ένα παράδειγμα που μας βοηθάει και μας δείχνει τη φύση των μεταβλητώς κατηγοριών και των αντικειμένων. Εδώ, το population ανήκει στο Robot κατηγόριων και ως εκ τούτου, είναι μια μεταβλητή κατηγοριών. Το name μεταβλητών ανήκει στο αντικείμενο (ανατεθεί χρησιμοποιώντας το self) και ως εκ τούτου είναι ένα αντικείμενο μεταβλητών. Έτσι αναφέρουμε το population μεταβλητών κατηγοριών σαν Robot.population και όχι σαν self.population. Αναφέρουμε τις μεταβλητές των αντικειμένων name χρησιμοποιώντας τη self.name σημειογραφία στη μέθοδο των αντικειμένων. Να θυμάστε την απλή διάφορα μετάξυ μεταβλητών κατηγοριών και αντικειμένων. Επίσης σημειώστε ότι ένα αντικείμενο μεταβλητών με το ίδιο όνομα σαν μεταβλητή κατηγοριών θα κρύψει τη μεταβλητή των κατηγοριών! Το howMany είναι μια μέθοδος όπου ανήκει στην κατηγορία και όχι στο αντικείμενο. Αυτό σημαίνει ότι μπορούμε να ορίσουμε είτε στην classmethod ή staticmethod ανάλογα στο που έχουμε την ανάγκη να ξέρουμε σε ποια κατηγορία του είμαστε μέρος της. Εφόσον δεν θέλουμε τέτοιες πληροφορίες, θα ξεκινήσουμε με την staticmethod. Θα μπορούσαμε να επιτύχουμε την ίδια χρήση διακοσμητών (http://www.ibm.com/developerworks/linux/library/i-cpdecor.html):

```
@staticmethod
def howMany():
        ''' Prints the current population.'''
        print('we have {0:d} robots.'.format(robot.population))
```

Η διακοσμητές μπορούν να φανταστείτε να γίνουν μια συντόμευση που ονομάζεται ρητή δήλωση, όπως έχουμε δει σε αυτό το παράδειγμα. Παρατηρώ ότι η μέθοδο __init__ χρησιμοποιεί για την αρχικοποίηση το robot παράδειγμα με ένα όνομα. Σε αυτή την μέθοδο, αυξάνουμε το population μετρείται με το 1 έχουμε ακόμα ένα robot να προστίθεται. Επίσης θα παρατηρήσετε τις τιμές του self.name ιδικά για κάθε γεγονός που δείχνει τη φύση των μεταβλητών αντικειμένου. Να θυμάστε, ότι πρέπει να αναφέρεστε στις μεταβλητές και στις μεθόδους με το ίδιο όνομα αντικειμένων χρησιμοποιώντας την self μόνο. Αυτό λέγεται αναφορά χαρακτηριστικού. Σε αυτό το πρόγραμμα, βλέπουμε την χρήση το **docstrings** για τις κατηγορίες όπως και στις μεθόδους. Μπορούμε να έχουμε πρόσβαση στην κατηγορία docstring χρόνο τρεξίματος χρησιμοποιώντας Robot. __doc__ και η μέθοδο docstring σαν obot.sayHi.__doc__. Όπως επίσης τη μέθοδο __init__, υπάρχει μια άλλη ειδική μέθοδο __del__ όπου λέγονται όταν ένα αντικείμενο πάει να πεθαίνει. π.χ. δεν είναι πλέον και επιστρέφουν στο σύστημα του υπολογιστή για την επαναχρησιμοποίηση σε εκείνο το κομμάτι της μνήμης. Σε αυτή τη μέθοδο, απλά μειώνουμε το Robot.population μετρώντας από το 1. Η μέθοδο __del__ τρέχει όταν το αντικείμενο δεν χρησιμοποιείται πια και δεν έχει εγγύηση για το πότε η μέθοδος θα ξεκινήσει να τρέχει. Αν θέλετε να δείτε ρητά σε δράση, πρέπει να χρησιμοποιήσουμε την del δήλωση όπου είναι ότι έχουμε κάνει εδώ τώρα.

Σημείωση για τους προγραμματιστές τον C++/Java/C#
Όλα τα μέλη της τάξης (συμπεριλαμβανόμενου τα μελή τον δεδομένων) είναι δημόσια και όλοι οι μέθοδοί είναι εικονικοί στην Python. Μια εξαίρεση: αν χρησιμοποιείτε μελή δεδομένων με ονόματα χρησιμοποιώντας την διπλή κάτω παύλα με πρόθεμα όπως το __privateval, η python χρησιμοποιεί το όνομα-ξέσχισμα όπου κάνει αποτελεσματικά προσωπική μεταβλητή. Έτσι, η συζήτηση που ακολουθεί είναι ότι η οποιαδήποτε μεταβλητή όπου χρησιμοποιείτε μόνο με την κατηγορία ή αντικείμενο θα πρέπει να ξεκινά με μια κάτω παύλα και όλα τα άλλα ονόματα είναι δημόσια και μπορεί να χρησιμοποιεί με την κατηγορία/αντικειμένων. Να θυμάστε ότι αυτή είναι μόνο μια σύμβαση και δεν επιβάλλονται από την Python (περιμένεις το πρόθεμα για την διπλή κάτω παύλα).

Κληρονομία

Ένα από τα ποιο σημαντικά πλεονεκτήματα των αντικειμένων του προσανατολισμένου προγραμματισμού είναι η επαναχρησιμοποίηση του κώδικα και μια από τους τρόπους που θα επιτευχθεί είναι μέσω του μηχανισμού κληρονομίας. Η κληρονομία μπορεί να είναι η καλύτερη αν φανταστείτε ως την υλοποίηση μιας σχέσης τύπου και προτύπου μεταξύ των τάξεων. Υποθέτω ότι θέλετε να γράψετε ένα πρόγραμμα όπου πρέπει να παρακολουθεί τους καθηγητές και τους μαθητές στο κολέγιο. Έχουν κάποια κοινά χαρακτηριστικά όπως το όνομα, ηλικίας και διεύθυνση. Επίσης έχουν συγκεκριμένους χαρακτηρισμούς όπως μισθός, μαθήματα και άδειες για τους καθηγητές, βαθμολογία και δίδακτρα για τους φοιτητές. Μπορείτε να δημιουργήσετε δυο αυτόνομες κατηγορίες για κάθε τύπο και επεξεργασία τους αλλά και την προσθήκη ενός νέου κοινού χαρακτηρισμού που θα σημαίνει να προσθέσετε για τους δυο από την ανεξάρτητη κατηγορία. Αυτό γίνεται γρήγορα δυσκίνητο. Ένας καλύτερος τρόπος θα είναι να δημιουργήσετε μια κοινή κατηγορία που ονομάζετε SchoolMember και μετά θα έχουμε τις κατηγορίες των καθηγητών και των μαθητών που κληρονομούν για αυτές τις κατηγορίες π.χ. Θα γίνουν υπό-είδη αυτών των τύπων (κατηγόριων) και μετά μπορούμε να προσθέσουμε συγκεκριμένα χαρακτηριστικά σε αυτούς τους υπό-τύπους. Υπάρχουν πολλά πλεονεκτήματα σε αυτή την προσέγγιση. Αν προσθέσουμε/αλλάξουμε οποιαδήποτε λειτουργικότητα στην SchoolMember, αυτό αυτόματα αντανακλάται σε υπό-τύπους επίσης. Για παράδειγμα, μπορείτε να προσθέσετε μια νέα ID κάρτα πεδίου για τους δυο καθηγητές και μαθητές με την απλή πρόσθεση στην κατηγορία SchoolMember. Ωστόσο, αλλάζοντας σε υπό-τύπους να μην επηρεάζουν άλλους υπό-τύπους. Ένα άλλο πλεονέκτημα είναι ότι αν αναφέρεται σε καθηγητή ή μαθητές αντικειμένων όπως SchoolMember αντικειμένων όπου μπορούν να είναι χρήσιμο σε κάποιες κατάστασης όπως να μετράτε τους αριθμούς των μαθητών των μελών. Αυτό λέγεται **polymorphism** όπου ένας υπό-τύπος μπορεί να είναι υποκαθίσταται σε οποιαδήποτε κατάσταση όπου ένα γονικό τύπο, αναμένεται π.χ. Το αντικείμενο μπορεί να αντιμετωπίζεται ως ένα παράδειγμα της γονικής κλάσης. Επίσης παρατηρώ ότι επαναχρησιμοποιηούμε τον κώδικα της γονικής κλάσης και δεν χρειάζεται να το επαναλάβω σε διαφορετικές κατηγορίες, όπως θα έπρεπε σε περίπτωση που είχε χρησιμοποιηθεί ανεξαρτήτως κατηγορίας. Η SchoolMember κατηγόριων σε αυτή την κατάσταση είναι γνωστή σαν την κατηγορία βάση ή υπέρ-κλάση. Οι κατηγορίες καθηγητές και μαθητές ονομάζονται παραγόμενες κλάσεις ή υπό-κλάσεις. Θα δούμε ένα παράδειγμα σε ένα πρόγραμμα:

```
#!/usr/bin/python
# Filename: inherit.py

class SchoolMember:
        ''' represents any school member'''
        def__init__(self,name,age):
        self.name = name
        self.age = age
        print('(initialized schollmamber:{0})'.format(self.name))

def tell(self):
        ""tell my details.'''
        print('name:"{0}" age:{1}"'.format(self.name,self.age),and="")

class Teacher(SchoolMember):
```

```
        ''' represents a teacher.'''
        def__init__(self,name,age)
        self.salary = salary
        print('(initialized teacher:{0})'.format(self.name))
def tell(self):
         SchoolMember.tell(self)
        print('salary: “{0:d}”'.format(self.salary))
class student(schoolmember);
'''represents a student.'''
def___init___ (self,name,age,marks):
schoolMember.__init__(self,name,age)
self.marks = maeks
print('(initialized student:{0}'.format(self.name))

def tell(self):
        schoolMember.tell(self)
        print('marks:”{0:d}”'.format(self.marks))

t = teacher('mrs. Shrividya',40,30000')
s = student('swaroop',25,75)

print() # εκτυπώνει μια κενή γραμμή

members = [t,s]
for member in members:
        member.tell() # δουλεύει και για τις δυο καθηγητές και μαθητές
```

Έξοδος:

```
$ python inherit.py
(initialized schoolmember: mrs. shrividya)
(initialized schoolmember: swaroop)
(initialized student: swaroop)

Name: “mrs. shrividya” age:”40” salary: “30000”
Name: “swaroop” age:”25” marks: “75”
```

Πως δουλεύει:
Όταν χρησιμοποιούμε την κληρονομία, ορίζουμε ονόματα τις βάσεις των κατηγοριών σε μια πλειάδα κατηγοριών ονομάτων ακολουθώντας τον ορισμό των κατηγοριών. Μέτα, παρατηρούμε την μέθοδο __init__ της βάσης των κατηγοριών που ονομάζονται ρητά χρησιμοποιώντας την self μεταβλητή όποτε αυτό που μπορούμε να κάνουμε αρχικοποίηση το κομμάτι της κλάσης βάσης τον αντικειμένων. Αυτό είναι πολύ σημαντικό να το θυμάστε - Η Python δεν καλεί αυτόματα τον κατασκευαστή της βάσης των κατηγοριών, θα πρέπει να ρητά να αποκαλούν τον εαυτό σας. Επίσης παρατηρούμε ότι μπορούμε να καλέσουμε όλες τις μεθόδους τις βάσης κατηγοριών με πρόθεμα τα ονομάτων των κατηγοριών, καλείτε την μέθοδο και μετά να περάσει η μεταβλητή self μαζί με τα τυχόν επιχειρήματα. Σημειώστε ότι μπορούμε να μεταχειριστούμε τις υποστάσεις των καθηγητών ή των μαθητών ως περιπτώσεις όπως την SchoolMember όταν χρησιμοποιούμε την tell μέθοδο τον κατηγοριών schollMember. Επίσης, παρατηρήστε την μέθοδο tell του υπό-τύπου και όχι τη μέθοδο tell της κατηγορίας schoolMember. Ένας τρόπος να καταλάβουμε αυτό είναι ότι η Python πάντα ξεκινά για μεθόδους στον πραγματικό τύπο, όπως σε αυτή την περίπτωση το κάνει. Αν δεν μπορείτε να βρείτε τη μέθοδο, ξεκινάτε να κοιτάτε την μέθοδο που ανήκει στην βάση των κατηγοριών μια μια στην με τη σειρά που αναφέρονται στην πλειάδα και στον ορισμό της τάξης. Μια σημείωση για την τερμινολογία - αν περισσότερες από μία κλάσεις παρατίθενται στην πλειάδα κληρονομικότητας, μετά ονομάζονται *πολλαπλή κληρονομικότητα.*

Περίληψη

Έχουμε μέχρι τώρα εξερευνήσει τις σημαντικές πτυχές τον κατηγοριών και τον αντικειμένων καθώς η σημαντικές ορολογίες συνδέονται μαζί. Έχουμε δει επίσης τα πλεονεκτήματα και τις παγίδες τον αντικειμένων – προσανατολισμό του προγραμματισμού. Η Python είναι υψηλών αντικειμένων – προσανατολισμού και καταλαβαίνει αυτή την έννοια θα σας βοηθήσει πολύ σε μεγάλες εκτέλεσης. Μέτα, θα μάθουμε πως να αντιμετωπίζουμε με εισόδους /εξόδους και πως να έχουμε πρόσβαση στους φακέλους τις Python.

Python en: Είσοδο – έξοδο

Εισαγωγή

Θα υπάρχουν καταστάσεις όπου το πρόγραμμα θα πρέπει να αλληλεπιδρά μαζί με τον χρήστη. Για παράδειγμα, θέλετε να παίρνετε εισαγωγή από τον χρήστη και μετά να εμφανίζει κάποια αποτελέσματα πίσω. Μπορούμε να επιτύχουμε αυτήν την χρήση με την input() και print() function αντίστοιχα. Για την έξοδο, μπορούμε επίσης να χρησιμοποιούμε τις σημαντικές μεθόδους της κατηγορίας str(string). Για παράδειγμα, μπορείτε να χρησιμοποιείτε την μέθοδο rjust για να έχουμε το string όπου είναι σωστό να δικαιολογείται σε ένα συγκεκριμένο πλάτος. Να δείτε το help(srt) για περισσότερες πληροφορίες. Ένας άλλος κοινός τύπος για εισόδους/εξόδους ασχολείται με τα αρχεία. Η δυνατότητα να δημιουργήσετε, να διαβάζετε και να γράφετε φακέλους είναι ουσιώδης σε πολλά προγράμματα και θα εξερευνήσουμε αυτή την άποψη σε αυτό το κεφάλαιο.

Εισαγωγή από τον χρήστη

```
#!/usr/bin/python
# user_input.py

def reverse(text):
        return text[ : :-1]
def is_palindrome(text):
        return text == reverse(text)

something = input('Enter text')
if (is_palindrome(something)):
        print(“yes, it is a palindrome”)
else:
        print(“no, it is not a palindrome”)
```

Έξοδος:

```
$ python user_input.py
Enter text: sir
no, it is not a palindrome

$ python user_input.py
Enter text: madam
yes, it is a palindrome

$ python user_input.py
Enter text: racecar
yes, it is a palindrome
```

Πως δουλεύει:

Χρησιμοποιούμε το χαρακτηριστικό τεμαχισμό για να αναστραφεί το κείμενο. Έχουμε ήδη δει πως μπορούμε να κάνουμε κομμάτια από την ακολουθία χρησιμοποιώντας τον κώδικα seq[a:b] ξεκινώντας από την θέση a στην θέση b. Μπορούμε επίσης να παρέχουμε ένα τρίτο επιχείρημα όπου προσδιορίζει το βήμα κατά το οποίο γίνεται το κομμάτιασμα. Η προεπιλογή έχει γίνει 1 επειδή όπου επιστρέφει ένα συνεχόμενο κομμάτι στο κείμενο. Δίνει ένα αρνητικό κομμάτι π.χ. -1

θα επιστρέφει το κείμενο ανάποδα. Η εισαγωγή input() function παίρνει ένα string σαν επιχείρημα και εμφανίζει στους χρήστες. Μέτα περιμένει από τον χρήστη να πληκτρολογείτε κάτι και να πατήσει το κουμπί της επιστροφής. Με το που ο χρήστης εισαγάγει, την input() function θα επιστρέψει το κείμενο. Περνούμε το κείμενο και το επιστρέφουμε. Αν το κείμενο είναι αυθεντικό και το επιστρέφουμε το κείμενο ίσα, μετά το κείμενο είναι παλιδρομο(**http://en.wiktionaly.org/wiki/palindrome**).

Άσκηση για το σπίτι:
Ελέγχουμε αν το κείμενο είναι παλίνδρομο. Επίσης πρέπει να αγνοήσετε σημεία στίξη, κενά και παρενθέσεις. Για παράδειγμα, “αυξηθεί για να ψηφίσετε, κύριε” είναι επίσης παλίνδρομο άλλα το τωρινό πρόγραμμα δεν λέει ότι είναι. Μπορείτε να βελτιώσετε το παραπάνω πρόγραμμα για να αναγνωρίσει αν είναι παλίνδρομο.?

Αρχεία

Μπορείτε να ανοίξετε και να χρησιμοποιήσετε αρχεία για το διάβασμα ή το γράψιμο με την δημιουργία ενός αντικειμένου των κατηγοριών τον αρχείων και χρησιμοποιώντας για το διάβασμα, readline ή να γράψετε μεθόδους κατάλληλα για να διαβάσετε ή γράψετε τα αρχεία. Η ικανότητα να διαβάσετε ή να γράψετε στα αρχεία εξαρτάται σχετικά με την κατάσταση λειτουργίας που έχεις ορίσει από το άνοιγμα τον αρχείον. Μέτα στο τέλος, όταν έχετε τελειώσει με το αρχείο, καλείτε την μέθοδο close για να πει στην python ότι έχουμε τελειώσει χρησιμοποιώντας το αρχείο.

Παράδειγμα:

```
#!/usr/bin/python
# Filename: using_file.py

poem = ' ' '\
programming is fun
when the work is done
if you wanna make work also fun:
        use python!
' ' '
f = open('poem.txt','w')#  ανοίγει για την 'w'riting
f.write(poem) # γράψτε το κείμενο του φακέλου
f.close() # κλείστε το αρχείο

f = open('poem.txt') # αν λειτουργία αν δεν έχει καθοριστεί, 'r' ead λειτουργία υποτίθεται
by default
while true:
        line = f.readline()
        if len(line) == 0: # μηδενικό μήκος υποδεικνύει EOF
                break
        print(line, end='')
f.close() # κλείστε το αρχείο
```

Έξοδος:

```
$ python using_file.py
programming is fun
when the work is done
if you wanna make toyr work also fun:
        use Python!
```

Πως δουλεύει:
Πρώτα, ανοίγετε ένα αρχείο με το να χρησιμοποιήσετε το ενσωματωμένο open function και προσδιορίστε το όνομα του αρχείου και ο τρόπος όπου θέλουμε να ανοίξουμε το αρχείο. Ο τρόπος μπορεί να είναι ο τρόπος διαβάσματος('r'), τρόπος γραψίματος ('w') ή ο τρόπος επισύναψης ('a').

μπορούμε επίσης να έχουμε να αντιμετωπίζουμε με το κείμενο του αρχείου ('t') ή ένα δυαδικό αρχείο('b'). υπάρχουν πραγματικά πολλοί τρόποι διαθέσιμοι και το help(open) θα σας δώσει πολλές πληροφορίες για αυτές. Από προεπιλογή, open() θεωρείστε τα αρχεία σαν να είναι ένα 't'ext αρχείο και να ανοίγει στον τρόπο 'r'ead. Στο δικό μας παράδειγμα, πρώτα ανοίγουμε το αρχείο στο τρόπο γραψίματος κείμενου και χρησιμοποιούμε την μέθοδο write των αρχείον του αντικειμένου για να γράψετε το αρχείο και μετά στο τέλος κλείστε το αρχείο. Μέτα, ανοίγουμε το ίδιο αρχείο ξανά για διάβασμα. Δεν χρειαζόμαστε να προσδιορίσουμε ένα τρόπο γιατί 'read text file' είναι η προεπιλεγμένη λειτουργία. Διαβάζουμε σε κάθε γραμμή των αρχείον χρησιμοποιώντας την μέθοδο readline σε μια επανάληψη. Αυτή η μέθοδος επιστρέφει μια ολοκληρωμένη γραμμή συμπεριλαμβανομένου την νέα γραμμή χαρακτήρων στο τέλος της γραμμής. Όταν ένα κενό string επιστρέφει, σημαίνει ότι έχουμε φτάσει στο τέλος των αρχείον και θα 'break' έξω από τον βρόχο. Από προεπιλογή, η print() function εμφανίζει το κείμενο όπως μια αυτόματη μια νέα γραμμή στην οθόνη. Καταστείλουμε μια νέα γραμμή με το να προσδιορίζετε με το end='' γιατί η γραμμή όπου διαβάζει από το αρχείο είναι ήδη τελειωμένη μαζί την νέα γραμμή χαρακτήρων. Μέτα, στο τέλος κλείνουμε με τον φάκελο. Τώρα, ελέγχουμε το περιεχόμενο του poem.txt φάκελο για να επιβεβαιώσουμε αυτό το πρόγραμμα έχει πραγματικά γραφτεί και θα διαβαστεί από αυτό το αρχείο.

Τουρσί (pickle)

Η Python μας παρέχει ένα σταθερό πρότυπο που ονομάζεται pickle(τουρσί) χρησιμοποιώντας την μπορείτε να αποθηκεύσετε οποιοδήποτε αντικείμενο στην python στο αρχείο και μετά θα το πάρει πίσω αργότερα. Αυτό ονομάζεται αποθήκευση των αντικειμένων επίμονα.
Παράδειγμα:

```
#!/usr/bin/python
# Filename: pickling.py

import pickle

# το όνομα του αρχείου όπου αποθηκεύουμε το αντικείμενο.
Shoplistfile = 'shoplist.data'
# λίστα από αντικείμενο για αγορά.
Shoplist = ['apple','mango','carrot']

# γράψτε στο φάκελο
f = open(shoplistfile,'wb')
pickle.dump(shoplist,f) # ρίξτε το αντικείμενο στο φάκελο
f.close()

del shoplist #καταστρέψατε τις μεταβλητές της λίστα αγορών.

#διαβάστε από την αποθήκευση.

F = open(shoplistfile, 'rb')
storedlist = pickle.load(f)
#φορτώστε το αντικείμενο από το φάκελο.
print(storedlist)
```

Έξοδος:

```
$ python pickling.py
['apple','mango','carrot']
```

Πως δουλεύει:
Για να αποθηκεύσουμε ένα αντικείμενο σε ένα αρχείο, πρώτα θα ανοίξουμε ένα αρχείο με τον τρόπο 'w'rite 'b'inary και μετά καλούμε την dump function της pickle πρότυπο. Αυτή η διαδικασία

ονομάζεται pickling. Μέτα, ανακτούμε το αντικείμενο χρησιμοποιώντας την load function του πρότυπου pickle όπου επιστρέφει το αντικείμενο. Αυτή η διαδικασία ονομάζεται unpickling.

Περίληψη
Έχουμε συζητήσει σημαντικούς τύπους από εισόδους/εξόδους και ο χειρισμό αρχείων και το πρότυπο pickle. Μετά, θα εξερευνήσουμε την έννοια των εξαιρέσεων.

Python en: Εξαιρέσεις
Εισαγωγή
Οι εξαιρέσεις συμβαίνουν όταν ορισμένες καταστάσεις εξαιρέσεων συμβαίνουν στο πρόγραμμα. Για παράδειγμα, αν πάτε να διαβάσετε ένα φάκελο και ο φάκελος δεν υπάρχει? Ή αν κατά λάθος διαγράψετε ένα φάκελο όταν το πρόγραμμα τρέχει? Τέτοιες καταστάσεις χειρίζονται χρησιμοποιώντας **εξαιρέσεις**. Παρόμοια, αν το πρόγραμμα έχει κάποιες άκυρες δηλώσεις? Αυτό χειρίζεται από την python όπου σηκώνει τα χέρια της και σας λέει οτι εκεί είναι λάθος (error).

Errors
Εξετάστε σε ένα απλό κάλεσμα print function. Αν γράψουμε ένα λάθος όπως print σαν Print? Παρατηρήστε το κεφάλαιο. Σε αυτή την περίπτωση, η Python εγείρει ένα συντακτικό λάθος.

```
>>>Print('hello world')
traceback (most recent call last):
File “<pyshell#0>”,line 1, in <module>
Print('hello world')
NameError: name 'Print' is not defined
>>> print ('hello world')
hello world
```

Παρατηρήστε ότι το NameError εγείρεται και επίσης και το που βρίσκεται, που εμφανίζεται ότι έχει γίνει το λάθος. Αυτός είναι ο χειρισμός σφαλμάτων για αυτό το σφάλμα.

Εξαιρέσεις
Θα δοκιμάσουμε να διαβάσουμε την εισαγωγή από τον χρήστη. Πιέστε ctrl – d και δείτε τι θα γίνει.

```
>>>s = input('enter something → ')
enter something →
traceback (most recent call last):
file “<pyshell#2>”, line 1, in <module>
s = input('enter something → ')
EOError: EOF when reading a line
```

Η Python που εγείρει το λάθος ονομάζετε EOFError όπου βασικά σημαίνει ότι θα βρει στο τέλος του αρχείου σύμβολου (όπου εκπροσωπούνται από το crtl – d) όταν δεν θα περιμένει να το δείτε.

Χειρισμό εξαιρέσεων
Μπορούμε να κάνουμε χειρισμό εξαιρέσεων χρησιμοποιώντας την try...except δήλωση. Βασικά βάζουμε την συνηθισμένη δήλωση μέσα στην try-block και βάλτε όλον το χειρισμό των λαθών στην except-block.

```
#!/usr/bin/python
# Filename: try_except.py

try:
        text = input('enter something –>')
except EOFError:
        print('why did you do an EOF an me?')
```

```
except KeyboardInterrupt:
        print('you cancelled the operation.')
else:
        print('you entered {0}'.format(text))
```

Έξοδος:

```
$ python try_except.py
enter something → # πιέστε ctrl-d
why did you do an EOF on me?

$ python try_except.py
enter something → # πιέστε ctrl-c
you cancelled the operation.

$ python try_except.py
enter something → no exception
you entered no exceptions
```

Πως δουλεύει:
Βάζουμε όλες τις δηλώσεις που θα μπορούσε να αυξήσει τις εξαιρέσεις/λάθη μέσα στο try block και μετά βάζουμε χειριστές για τα κατάλληλα λάθη/εξαιρέσεις, μέσα εκτός από την ρήτρα / block. Εκτός από ρήτρα μπορούμε να χειριστούμε ένα μόνο προσδιορισμένο λάθος ή εξαίρεση, ή μια λίστα σε παρένθεση του λάθος / εξαιρέσεις. Αν ούτε ονόματα ή εξαιρέσεις παρέχονται, θα χειριστεί όλα τα λάθη και τις εξαιρέσεις. Παρατηρήστε, ότι θα πρέπει να υπάρχει τουλάχιστον μια εξαίρεση από ρήτρα που συνδέονται με κάθε try ρήτρα. Αλλιώς, ποιος ο λόγος να έχουμε το try block?
Αν κάποιο λάθος ή εξαίρεση δεν χειρίζονται, μετά από προεπιλογή η Python χειρίζεται όπου σταμάτα την εκτέλεση του προγράμματος και εμφανίζει ένα μήνυμα του λάθους. Έχουμε ήδη δει αυτήν την δράση παραπάνω. Μπορείτε επίσης μια else ρήτρα να συνδέεται με το try...except block. Η else ρήτρα εκτελείται εάν δεν συμβαίνει καμία εξαίρεση. Στο επόμενο παράδειγμα, επίσης έχουμε δει πως να έχουμε την εξαίρεση αντικειμένων όποτε μπορούμε να ανακτήσουμε συμπληρωματικές πληροφορίες.

Αύξηση Εξαιρέσεις

Μπορείτε να αυξήσετε εξαιρέσεις χρησιμοποιώντας την αύξηση δήλωση με το να χορηγήσετε το όνομα του λάθους/εξαιρέσεις και η εξαίρεση του αντικείμενου που πρόκειται να ρίχνονται. Τα λήθη ή η εξαίρεση που μπορούν να προκύψουν με την κατηγορία όπου άμεσα ή έμμεσα, θα πρέπει να είναι η παράγωγη ή κατηγορία των εξαιρέσεων κατηγοριών.

```
#!/usr/bin/python
# Filename: raising.py

class shortinputexception(exception):
''' a user- defined exception clss.'''
def___init___ (self,length,atleast):
exception.__init__(self)
self.length = length
self.atleast = atleast

try:
text = inupt('enter something →  ')
if len(text)< 3:
        raise shortinputexception(len(text),3)
# η δουλειά μπορεί να συνεχίσει όπως συνηθίζεται
except EOFError:
        print('why did you do an EOF on me?')
```

```
except shortinputexception as ex:
        print('shortinputexception: the input was {0} long, excepted at least {1}'\.format(ex.length, exatleast))
else:
        print('no exception was raised')
```

Έξοδος:

```
$ python raising.py
enter something → a
shortinputexception: the input 1 long, expected at least 3

$ python raising.py
enter something → abc
no exception was raising.
```

Πως δουλεύει:
Εδώ, δημιουργούμε το δικό μας τύπο εξαίρεσης. Αυτή η νέα τύπου εξαίρεση ονομάζεται shortinputexception. Έχει δυο πεδία – μήκος από την εισαγωγή, και τουλάχιστον όπου είναι το μικρότερο μήκος το πρόγραμμα περιμένει. Στην ρήτρα εκτός από την ρήτρα, αναφέραμε τα λάθη της κατηγορίας όπου θα τα αποθηκεύουμε σαν τα ονόματα μεταβλητές για να κρατήσουμε αντίστοιχα αντικείμενα λάθη/εξαίρεση. Αυτή είναι η αναλογία στους παραμέτρους και στα επιχειρήματα στο κάλεσμα μιας function. Όπου αυτό ειδικότερα εκτός από την ρήτρα, χρησιμοποιούμε το μήκος και τουλάχιστον το πεδίο του αντικειμένου της εξαίρεσης για να εμφανίσει ένα κατάλληλο μήνυμα στον χρήστη.

Try...Finally

Ας υποθέσουμε ότι διαβάζουμε ένα αρχείο στο πρόγραμμα. Πως θα εξασφαλίσουμε ότι το αρχείο αντικειμένων θα το κλείσουμε κατάλληλα και αν η εξαίρεση αναδείχθηκε? Αυτό μπορεί να γίνει χρησιμοποιώντας το finally block. Παρατήστε ότι μπορείτε να χρησιμοποιήσετε μια ρήτρα εξαίρεσης μαζί με το finally block για το ίδιο αντίστοιχο του try block. Που θα πρέπει να ενσωματώσετε τη μια μέσα σε μια άλλη, εάν θέλετε να χρησιμοποιήσετε και τις δύο.

```
#!/usr.python
# Filename : finally.py

import time

try:
        f = open('poem.txt')
        while true: # συνηθισμένο αρχείο ανάγνωση ιδίωμα μας
                line = f.readline()
                if len(line) ==0:
                        break
        print(line, end='')
        time.sleep(2) # να είστε σίγουρη ότι τρέχει για λίγο
except keyboardinterrupt:
        print('!! you cancelled the reading from the file')
finally:
        f.close()
        print('(cleaning up:closed the file)')
```

Έξοδος:

```
$ python finally.py
programming is fun
```

when the work is done
if you wanna make your work also fun:
!!you cancelled the reading from the file
(cleaning up: closed the file)

Πως δουλεύει:
Κάνουμε τα συνηθισμένα αρχεία-διάβασμα. Άλλα έχουμε αυθαίρετα εισάγει τον ύπνο για 2 δευτερόλεπτα μετά από την εμφάνιση κάθε γραμμής χρησιμοποιώντας το time.sleep function όποτε το πρόγραμμα τρέχει αργά (η Python είναι πολύ γρήγορη από τη φύση της). Όταν το πρόγραμμα ακόμα τρέχει, πιέστε ctrl – c για να διακόψετε / ακυρώσετε το πρόγραμμα.
Παρατηρήστε ότι η keyboardinterrupt εξαίρεση ρίχνεται και το πρόγραμμα εγκαταλείπει. Ωστόσο, πριν το πρόγραμμα τερματίσει, η τελική ρήτρα εκτελείτε και το αρχείο των αντικειμένων είναι πάντα κλειστό.

Η with δήλωση

Την απόκτηση ενός πόρου στην try block και μεταγενέστερα ελευθερώνοντας τον πόρο στην finally block είναι κοινό πρότυπο. Ως εκ τούτου, για αυτό η with δήλωση που επιτρέπει αυτό να είναι ένας καθαρός τρόπος:

```
#!/usr/bin/python
# Filename: using_with.py

with open(“pem.txt”) as f:
        for line in f:
                print (line,end='')
```

Πως δουλεύει:
Η έξοδος πρέπει να είναι η ίδια με το προηγούμενο παράδειγμα. Η διάφορα εδώ είναι ότι χρησιμοποιούμε την open function μαζί με την with δήλωση – αφήνουμε να κλείσουν τα αρχεία μόνα τους αυτόματα με την with open. Αυτό που γίνεται πίσω από τις σκηνές είναι ότι υπάρχει ένα πρωτόκολλο που χρησιμοποιείτε με την with δήλωση. Προσκομίζει το αντικείμενο με την open δήλωση, ας το ονομάσουμε “thefile” σε αυτή την περίπτωση. Πάντα ονομάζεται η thefile.__enter__function πριν αρχίσει το κομμάτι του κώδικα και πάντα καλείται thefile.__exit__ τελειώνοντας το κομματικό του κώδικα. Όποτε ο κώδικας που θα έχουμε γράψει στο τελευταίο κομμάτι θα ασχοληθεί αυτόματα με την __exit__ μέθοδο. Αυτό είναι που μας βοηθάει να αποφύγουμε την σαφή χρήση της try...finally δήλωσης επαλειμμένα. Περισσότερες συζήτησης για αυτό το θέμα είναι πέρα από αυτό το βιβλίο, όποτε σας παρακαλώ κατευθυνθείτε στο PEP 343(http://www.python.org/dev/perps/pep-0343/) για ολοκληρωμένη εξήγηση.

Περίληψη

Έχουμε συζήτηση την χρήση του try..except και της try...finally δήλωσης. Έχουμε δει πως να δημιουργήσουμε την δίκια μας εξαίρεση στους τύπους και πως να αυξήσουμε τις εξαιρέσεις επίσης. Μέτα, θα εξερευνήσουμε την Python standerd library.

Python en: πρότυπη βιβλιοθήκη

Της Python η πρότυπη βιβλιοθήκη περιεχέι ένα μεγάλο αριθμό από χρήσιμα πρότυπα και είναι κομμάτι της εγκατάσταση της πρότυπης python. Είναι σημαντικό να γίνετε οικείοι με την Python την πρότυπη βιβλιοθήκη εφόσον πολλά προβλήματα μπορούν να λυθούν γρήγορα αν είστε οικείος με το εύρος των πραγμάτων που αυτές οι βιβλιοθήκες μπορούν να κάνουν. Θα εξερευνήσουμε κάποιες από τις συνήθεις χρήσεις των προτύπων στην βιβλιοθήκη. Μπορείτε να βρείτε ολόκληρες λεπτομερείς για όλα τα πρότυπα στην πρότυπη βιβλιοθήκη της Python στο κομμάτι της αναφοράς βιβλιοθήκης (**http://docs.python.org/dev/3.0/library/**) με τα δεδομένα που είναι μαζί με την Python που εγκαθιστάς. Ας εξερευνήσουμε μερικά χρήσιμα πρότυπα.

Παρατηρήστε
Αν βρίσκετε αυτά τα θέματα πολύ προχωρημένα, μπορείτε να παραλείψετε το κεφάλαιο. Ωστόσο, σας συνιστώ να επιστρέψετε στο κεφάλαιο όταν είστε πιο άνετος με το προγραμματισμό της Python.

Sys πρότυπο

Το πρότυπο της sys περιέχει τη λειτουργία του συγκεκριμένου συστήματος. Έχουμε ήδη δει ότι η λίστα sys.argv περιεχέι τις εντολές γραμμών επιχειρημάτων. Υποθέτουμε ότι θέλουμε να ελέγξουμε την έκδοση των εντολών της Python την ύπαρξη της χρήσης του, όποτε πείτε, ότι θέλουμε να εξασφαλίσουμε ότι θα χρησιμοποιούμε την τελευταία έκδοση 3. Το πρότυπο sys μας δίνει τέτοια λειτουργικότητα.

```
>>> import sys
>>> sys.version_info
(3, 0, 0, 'beta', 2)
>>> sys.version_info[0] >= 3
True
```

Πως δουλεύει:
Το sys πρότυπο έχει μια version_info πλειάδα όπου μας δίνει την πληροφορία της έκδοσης. Η πρώτη είσοδος είναι η μεγάλη έκδοση. Μπορούμε να ελέγξουμε αυτό, για παράδειγμα, να εξασφαλίζουν ότι το πρόγραμμα θα τρέχει μόνο με την μικρότερη έκδοση από την Python 3.0:

```
#!/usr/bin/python
# Filename: versioncheck.py
import sys, warnigs
if sys.version_info[0]<3:
        warning.warn(“need python 3.0 for this programma to run”, runtimewarning)
else:
        print('procced as normal')
```

Έξοδο:

```
$ python2.5 versioncheck.py
versioncheck.py:6: (runtimewarning: need python 3.0 for this program to run runtimewarning)

$ python3 versioncheck.py
proceed as normal
```

Πως δουλεύει:
Χρησιμοποιούμε ένα άλλο πρότυπο από την πρότυπη βιβλιοθήκη που ονομάζεται warning όπου χρησιμοποιείτε για να δείξουμε την προειδοποίηση το τέλος – χρήστης. Αν ο αριθμός έκδοση της python δεν είναι τουλάχιστον 3, εμφανίζουν αντίστοιχη προειδοποίηση.

Πρότυπο logging

Τι θα γίνει αν θέλεις να έχεις κάποια debugging μηνύματα ή σημαντικά μηνύματα για να αποθηκεύονται κάπου όποτε μπορείτε να ελέγχετε αν το πρόγραμμα σας τρέχει όπως το περιμένετε? Πως αποθηκεύετε κάπου αυτά τα μηνύματα? Αυτό μπορεί να γίνει χρησιμοποιώντας την logging πρότυπο.

```
#!/usr/bin/python
# Filename: usr_logging.py
import os, platform, logging

if platform.platform().startswith('windwos'):
logging_file = os.path.join(os.getenv('HOMEDRIVE')
```

```
os.getenv('HOMEPATH'), 'test.log')
else:
        logging_file = os.path.join(os.getenv('HOME'), 'test.log')
logging.basicconfig(
        level=logging.DEBUG,
        format='%(asctime)s: %(levelnam)s: %(massage)s',
        filename = logging_file,
        filemode = 'w',)

logging.debug(“start of the program”)
logging.info(“doing something”)
logging.warning(“dying now”)
```

Έξοδος:
$ python use_logging.py
logging to c:\users\swaroop\test.log

Αν ελέγξουμε το περιεχόμενο του test.log, θα δείχνει κάπως έτσι:
2008-09-03 13:18:16,233 : DEBUG : Άρχιζει το πρόγραμμα
2008-09-03 13:18:16,233 : INFO : να κάνει κάτι
2008-09-03 13:18:16,233 : WARNING : πεθαίνει τώρα

Πως δουλεύει:
Χρησιμοποιούμε τρία πρότυπα από την πρότυπη βιβλιοθήκη – το os πρότυπο αλληλεπιδρώντας με το λειτουργικό σύστημα, το πρότυπο platform για την platform π.χ. Το λειτουργικό σύστημα και το πρότυπο logging στην πληροφορία log. Πρώτα, ελέγχουμε το λειτουργικό σύστημα με το να χρησιμοποιήσουμε των ελέγχων της επιστροφής του string με το platform.platform() (για περισσότερες πληροφορίες, δείτε import platform; help(platform)). Αν είναι για τα windows, καταλαβαίνουμε από το σκληρό home, το home φάκελο και το όνομα του αρχείου όπου θέλουμε να αποθηκεύσουμε τις πληροφορίες. Βάζουμε αυτά τα τρία κομμάτια μαζί, περνούμε την ολοκληρωμένη τοποθεσία του αρχείου. Για άλλες platforms, θέλουμε να ξέρουμε απλά το φάκελο home του χρήστη και να πάρουμε την ολοκληρωμένη τοποθεσία του φακέλου. Χρησιμοποιούμε το os.path.join() function για να βάλουμε αυτά τα τρία κομμάτια τις τοποθεσίας μαζί. Ο λόγος να χρησιμοποιήσουμε την ειδική function από το να προσθέσουμε τα strings μαζί είναι γιατί αυτή η function εξασφαλίζει την ολοκληρωμένη τοποθεσία ταιριάζει με την μορφή που αναμένεται από το λειτουργικό σύστημα. Ρυθμίζουμε το πρότυπο logging για να γράψουμε όλα τα μηνύματα με συγκεκριμένη μορφή για τον φάκελο που έχουμε ορίσει. Στο τέλος, μπορούμε να βάλουμε μηνύματα ότι και τα δυο είναι για debugging, πληροφορίες, προσοχή ή τα κρίσιμα μηνύματα. Όταν το πρόγραμμα τρέχει, μπορούμε να ελέγχουμε αυτό το αρχείο και θα ξέρουμε τι θα γίνει στο πρόγραμμα, παρόλο που δεν έχουνε παρουσιάσει πληροφορίες στον χρήστη που τρέχει το πρόγραμμα.

Urllib και Json πρότυπα

Πόσο πολύ ωραία θα ήτανε αν μπορούσαμε να γράψουμε το δικό μας πρόγραμμα όπου θα μα δίνει αποτελέσματα από τον ηλεκτρονικό ιστό? Ας το εξερευνήσουμε τώρα. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας μερικά πρότυπα. Πρώτα είναι το urllib πρότυπο όπου μπορούμε να καλέσουμε οποιαδήποτε σελίδα από το ίντερνετ. Θα χρησιμοποιήσουμε το Yahoo! Εξερευνήστε για να αναζητήσουμε τα αποτελέσματα και ευτυχώς μπορεί να μας δώσει αποτελέσματα σε μορφή που καλείται JSON είναι εύκολο για εμάς να την αναλύσει γιατί το ενσωματωμένο json πρότυπο στην πρότυπη βιβλιοθήκη.

Να κάνουμε

Αυτό το πρόγραμμα δεν δουλεύει ακόμα όπου σημαίνει να είναι ένα bug στην Python 3.0 beta(http://bugs.python.org/issue3763).

```
#!/usr/bin/python
# Filename: yohoo_search.py

import sys
if sys.version_info[0] !=3:
        sys.exit('this program needs python 3.0')
import json
import urllib, urllib.parse, urllib.request, urllib.response

#αποκτήστε το δικό σας APP ID στην http://developer.yahoo.com/wsregapp/YAHOO_APP_ID =
'jl22psvV34HELWhdfUjbfDQzlj2B57KFS_qs4I8D0Wz5U5_yCI!Awv8.lBSfPhwr'
SEARCH_BASE = ' http://search.yahooapis.com/websearchservice/v1/websearch'
class YahooSearchError(Exception):
pass
# πάρτε από http://developer.yahoo.com/python/python-json.html
def search (query, results=20, start=1,**kwargs):
        kwargs.update({
                'appid': YAHOO_APP_ID,
                'query': query,
                'results': results,
                'start': start,
                'output': 'json'
                })
url = SEARCH_BASE + '?' + urllib.parse.urlencode(kwargs)
result = json.load(urllib.request.urlopen(url))
if 'Error' in result:
        raise YahooSearchError(result['Error'])
return result['resultset']

query = input('what do you to search for?')
for result in search(query)['result']:
        print('{0} : {1}'.format(result['title'],result['url']))
```

Έξοδος:

Να κάνω

Πως δουλεύει:

Μπορούμε να έχουμε αποτελέσματα αναζήτησης από μια συγκεκριμένη σελίδα στο ίντερνετ με το να δίνουμε το κείμενο που ψάχνουμε σε μια συγκεκριμένη μορφή. Πρέπει να προσδιορίσουμε πολλές επιλογές όπου τις συνδυάζουμε χρησιμοποιώντας key1=value1&key2=value2 μορφή όπου ο χειρισμός γίνετε από το urllib.parse.urlencode() function. Όποτε για παράδειγμα, ανοίξτε το ling στο πρόγραμμα περιήγησης στο Web (http://search.yahooapis.com/websearchservice/ v1/ websearch?query=byte+ of+ python&appid=jl22psvV34HELWhdfUjbfDQzlj2B57KFS_qs4I8D0Wz5U5_yCI1Awv8. IBSfPhwr&results=20&start=1& output=json)
και θα δείτε 20 αποτελέσματα, ξεκινώντας από τα πρώτα αποτελέσματα, για τις λέξεις “byte of python”, και θα ρωτάμε για την έξοδο σε μορφή JSON. Κάνουμε σύνδεση σε αυτή την διεύθυνση χρησιμοποιώντας την urllib.request.urlopen() function και περαστέ αυτό το αρχείο που χειρίζεται το json.laod() όπου θα διαβάσετε το περιεχόμενο και συγχρόνως θα μετατρέπει αυτόματα σε αντικείμενο στην Python. Μέτα τότε κάνουμε επανάληψη μέσα από αυτά τα αποτελέσματα και τα εμφανίζουμε στο τέλος στον χρήστη.

Πρότυπα της σειράς εβδομάδων

Υπάρχουν πολλά να εξερευνήσουμε στην πρότυπη βιβλιοθήκη όπως το debugging (http://docs.python.org/dev/3.0/library/pdb.html), χειρισμός εντολής γραμμής επιλογόν (http://diveintoython.org/regular_expressions/index.html) και ου το κάθε εξής. Για τον καλύτερο τρόπο να εξερευνήσουμε τα πρότυπα βιβλιοθήκης είναι να διαβάσουμε Doug Hellmann's εξαιρετικο πρότυπο python της βδομάδας της σειρας(http://www.doughellmann.com/projects/PyMOTW/).

Περίληψη

Έχουμε εξερευνήσει κάποιες από τις λειτουργίες από πολλά πρότυπα στην πρότυπη βιβλιοθήκη της Python. Συνιστάται να περιηγηθείτε μέσα από την πρότυπη βιβλιοθήκη της Python τεκμηρίωσης (http://docs.python.org/dev/3.0/library) για να έχουμε μια ιδέα για όλα τα πρότυπα που είναι διαθέσιμα. Μέτα, θα καλύψουμε σοβαρές πτυχές της Python όπου θα κάνουμε την περιοδεία της Python πιο ολοκληρωμένη.

Python en: Περισσότερα

Εισαγωγή

Μέχρι τώρα έχουμε καλύψει σημαντικές πτυχές της Python όπου θα χρησιμοποιήσουμε. Σε αυτό το κεφάλαιο θα καλύψουμε κάποιες πτυχές που θα μας μάθουν πολλά για την Python.

Περνώντας Γύρω Από Πλειάδες

Αν ποτέ επιθυμούσαμε να επιστρέψουμε δυο διαφορετικές τιμές από μια function? Μπορείς. Το μόνο που μπορείς να κάνεις είναι να χρησιμοποιήσεις μια πλειάδα.

```
>>> edf get_error_details():
… return(2, 'second error details')
…
>>> errnum, errstr = get_error_details()
>>> errnum
2
>>> errstr
'second error details'
```

Παρατηρήστε ότι η χρήση του a, b =< some expression> ερμηνεύει τα αποτελέσματα ων εκφράσεων σαν μια πλειάδα με δυο τιμές. Αν θέλετε να ερμηνεύσετε τα αποτελέσματα σαν (a, <everything else>), μετά θα θέλετε μόνο αν ξεκινήσετε όπως θα κάνατε στις παραμέτρους της function:

```
>>> a, *b = [1, 2, 3, 4]
>>> a
1
>>> b
[2, 3, 4]
```

Αυτό σημαίνει ότι ο γρηγορότερος τρόπος να ανταλλάξουμε δυο μεταβλητές στην Python είναι:

```
>>> a = 5; b = 8
>>> a, b = b, a
>>> a, b
(8, 5)
```

Ειδική μέθοδο

Υπάρχουν ορισμένοι μέθοδοι όπως το __init__and__del__ μέθοδος που έχουν ειδική σημασία στην κατηγορία. Ειδικές μέθοδοι χρησιμοποιούνται για να μιμούνται ορισμένες συμπεριφορές του ενσωματωμένου τύπου. Για παράδειγμα, αν θέλετε να χρησιμοποιήσετε το x[key] λειτουργία ευρετηρίασης της κατηγορίας (οπός χρησιμοποιείτε για την λίστα και τις πλειάδες), μετά το μόνο που έχετε να κάνετε είναι να εφαρμόσετε την __getitem__() μέθοδο και η δουλειά τέλος. Αν το σκέφτεστε, αυτό είναι που κάνει η Python για την list κατηγορία από μόνο του. Κάποιες χρήσιμες ειδικές μεθόδους είναι στο πίνακα που ακολουθεί. Αν θέλετε να ξέρετε για όλες τις ειδικές

μεθόδους, δείτε το εγχειρίδιο (http://docs.python.org/dev/3.0/reference/datamodel.html#special-method-names).

Όνομα	Εξήγηση
__init__(self,...)	Αυτή η μέθοδος καλείτε πριν το νεοσύστατο αντικείμενο επιστρέφεται για χρήση
__del__(self)	Καλείτε πριν το αντικείμενο καταστραφεί
__str__(self)	Καλείτε όταν χρησιμοποιούμε την print function ή όταν χρησιμοποιείτε η str().
__it__(self,other)	Καλείτε όταν χρησιμοποιείτε το λιγότερο χειριστή (<). παρόμοια, υπάρχουν μέθοδοί για όλους τους χειριστες(+,>,π.χ.)
__getitem__(self,key)	Καλείτε όταν το x[key] λειτουργία ευρετηρίασης χρησιμοποιείτε
__len__(self)	Καλείτε όταν το ενσωματωμένο len() function χρησιμοποιείτε για την ακολουθία αντικειμένων.

Απλή δήλωση block

Έχουμε δει ότι το κάθε block των δηλώσεων είναι μια συλλογή από την υπόλοιπη και το δικό της επίπεδο οδόντωσης. Οπότε, υπάρχει μια ανακοπή. Αν το block δήλωση περιεχέι μόνο μια απλή δήλωση ή βρόχο δήλωσης το επόμενο παράδειγμα θα σας το ξεκαθαρίσει.

```
>>> flag = true
>>> if flag: print 'yes'
…
yes
```

Παρατηρήστε ότι η απλή δήλωση χρησιμοποιείτε σε θέση και όχι σαν ξεχωριστώ block. Παρόλα αυτά, μπορείτε να χρησιμοποιήσετε αυτή για να κάνετε το πρόγραμμα σας μικρότερο, σας συνιστώ να αποφεύγετε αυτήν την μέθοδο συντόμευσης, να περιμένετε στον ελέγχο λαθών, κυρίως γιατί θα είναι πιο απλό να προσθέτετε μια παραπάνω δήλωση αν χρησιμοποιείτε την σωστή οδόντωση.

Μορφές Lambda

Μια Lambda δήλωση χρησιμοποιείτε για να δημιουργήσετε μια νέα function αντικειμένων και τα επιστρέφει στο χρόνο εκτέλεσης.

```
#!/usr/bin/python
# Filename: lambda.py

def make_repeater(n):
        return lambda s: s*n
twice = make_repeater(2)
print(twice('word'))
print(twice(5))
```

Έξοδος:

```
$ python lambda.py
wordword
10
```

Πως δουλεύει:
Εδώ, χρησιμοποιούμε την function make_repeater για να δημιουργήσετε μια νέα function αντικειμένων στο χρόνο εκτέλεσης και την επιστρέφει. Μια lamdba δήλωση χρησιμοποιείται για να δημιουργήσουν το function αντικείμενο. κατ 'ουσίαν, η lambda παίρνει μια παράμετρο που ακολουθείτε από την απλή έκφραση μονό που γίνετε το κορμί του function και επιστρέφει την τιμή

αυτή της έκφρασης με μια νέα function. Παρατηρήστε ότι ακόμα και μια print δήλωση δεν μπορεί να χρησιμοποιηθεί άμεσα σε μια μορφή lambda, μόνο εκφράσεις.

Να κάνω
Μπορούμε να κάνουμε μια list.sort() με το να παρέχουμε μια σύγκριση της function δημιουργώντας την χρίση στην lambda?

```
Points = [{ 'x' : 2,'y': 3}, {'x' : 4, 'y':1}]
# points.sort(lambda a, b : cmp(a['x'], b['x']))
```

Κατανόηση λίστας

Η κατανόηση της λίστας χρησιμοποιείτε για να αντλούν μια νέα λίστα από μια ήδη υπάρχουσα λίστα. Ας υποθέσουμε ότι έχουμε μια λίστα από αριθμούς και θέλουμε να έχουμε μια αντιστοιχία από λίστα με όλους τους αριθμούς πολλαπλασιασμένους με το 2 μόνο όταν οι αριθμοί από μόνο τους είναι η μεγαλύτερη από το 2. Η κατανόηση της λίστας είναι ιδανικές για τέτοιες καταστάσεις.

```
#!/usr/bin/python
# Filename: list_comprehension.py
listone = [2, 3, 4]
listtwo = [2*i for I  in listone if i>2]
print(listtwo)
```

Έξοδος:

```
$ python list_coprehension.py
[6, 8]
```

Πως δουλεύει:
Εδώ, αντλούμε μια νέα λίστα που προσδιορίζουμε τον χειρισμό για να γίνει (2*i) όταν κάποια συνθήκη είναι ικανοποιημένη (if i>2). Παρατηρήστε ότι η αυθεντική λίστα παραμένει αμετάβλητη. Το πλεονέκτημα της χρήσης της λίστας είναι η κατανόηση ότι μειώνει το πόσο του boilerplate και απαιτείται κωδικός όταν χρησιμοποιούμε βρόγχο για μια διαδικασία του κάθε στοιχείου από μια λίστα και αποθηκεύει σε μια νέα λίστα.

Πλειάδες που Λαμβάνουν και τους Καταλόγους των Λειτουργιών

Υπάρχει ένας ειδικός τρόπος για να λαμβάνουμε παραμέτρους σε μια function σαν μια πλειάδα ή λεξικό χρησιμοποιώντας το * or ** πρόθεμα. Αυτό είναι χρήσιμο όταν περνούμε μια μεταβλητή αριθμών των επιχειρημάτων στην function.

```
>>>  def powersum(power, *args):
… '" επιστρέφει την προσθέσει του κάθε επιχειρήματος αυξάνοντας την δύναμη που του προσδιορίζω."'
… total = 0
… for I in args:
… total += pow(i, power)
… return total
…
>>> powersum(2, 3, 4)
25
>>> powersum(2, 10)
100
```

Επειδή έχουμε ένα * πρόθεμα στην μεταβλητή args, όλα τα επιχειρήματα τα πέρασε στην function που αποθηκεύονται στην args σε μια πλειάδα. Αν ένα ** πρόθεμα χρησιμοποιείται αντί, για τους παραπάνω παραμέτρους θα θεωρούνται να είναι κλειδί/τιμή ζευγάρια των λεξικών.

exec και eval

Η exec function χρησιμοποιείται για να εκτελέσει δηλώσεις της Python όπου αποθηκεύονται σε ένα string ή φάκελο, σαν αντίθεση για να γράψει το πρόγραμμα μόνο του. Για παράδειγμα, μπορούμε να αναπαράγουμε ένα string που περιεχέι κώδικα της python στο χρόνο εκτέλεσης και μετά εκτελεί αυτές τις δηλώσεις χρησιμοποιώντας την exec δήλωση:

```
>>> exec('print(“hello world”)')
hello world
```

Παρόμοια, η eval function χρησιμοποιείτε για να εκτιμήσουμε την εγκυρότητα της Python της εκφράσεις όπου αποθηκεύονται σε ένα string. Ένα απλό παράδειγμα δείχνετε παρακάτω.

```
>>> eval('2*3')
6
```

Η εντολή assert

Η assert δήλωση χρησιμοποιείτε για να assert ότι κάτι είναι σωστό. Για παράδειγμα, αν είστε πολύ σίγουρος ότι έχετε τουλάχιστον ένα στοιχείο στην λίστα χρησιμοποιείστε και ελεξεται αυτό, και να αυξήσετε το λάθος αν δεν είναι σωστό, μετά assert δήλωση είναι ιδανική σε αυτή την περίπτωση. Όταν η assert δήλωση αποτυχαίνει, το assertionError αυξάνεται.

```
>>> mylist = ['item']
>>> assert len(mylist) >=1
>>> mylist.pop()
'item'
>>> mylist
[]
>>> assert len(mylist)>=1
tracedack (most recent call last):
file “<stdin>”, line 1, in <module>
assertionError
```

Η assert δήλωση θα πρέπει να χρησιμοποιείτε φρόνιμα. Της περισσότερες φόρες, είναι καλύτερα να πιάσετε εκφράσεις, είτε να χειριστεί το πρόβλημα ή να δείξετε ένα μήνυμα του λάθους στον χρήστη και μετά να τελειώσετε.

Η repr function

Η repr function χρησιμοποιείται για να αποκτήσετε κανονικά sting για την αντιπροσώπευση του αντικείμενου. Το σημαντικό κομμάτι είναι ότι θα έχετε eval(repr(object)) == odject της περισσότερες φόρες.

```
>>>i = []
>>> I.append('item')
>>> repr(i)
“['item']”
>>> eval(repr(i))
['item']
>>> eval(repr(i)) == i
true
```

Βασικά, η repr funtion χρησιμοποιείται για να αποκτήσετε μια εκτ'υπωση αναπαράστασης του αντικείμενου. Μπορείτε να ελέγχετε ποια κατηγορία για την repr function ορίζει την μέθοδο __repr__ στην κατηγορία.

Περίληψη

Πρέπει να καλύπτονται κάποια πιο πολλά χαρακτηριστικά της Python σε αυτό το κεφάλαιο και ακόμα δεν έχουμε να καλύψει όλα τα χαρακτηριστικά της Python. Ωστόσο, σε αυτή την κατάσταση, έχουμε καλύψει πολλές από αυτές που θα πάτε να χρησιμοποιήσετε σε πρακτική. Αυτό είναι επαρκές για να ξεκινήσουμε σε οποιοδήποτε πρόγραμμα θα πάμε να δημιουργήσουμε. Μέτα, θα το συζητήσουμε πως να εξερευνήσουμε την Python περισσότερο.

Python en: Μέτα τι

Αν έχετε διαβάσει διεξοδικά μέχρι τώρα και εξασκηθείτε στο να γράψετε πολλά προγράμματα, μετά θα πρέπει να έχετε γίνει άνετοι και οικείοι με την python. Θα έχετε πιθανόν δημιουργήσει κάποια προγράμματα της Python να δοκιμάσετε κάποια πράγματα και να εξασκήσετε την ικανότητα της python. Αν δεν το έχετε κάνει ακόμα, θα έπρεπε. Αυτή η ερώτηση είναι η 'τι μετά?'.
Θα σας προτείνω να αντιμετωπίσετε αυτό το πρόβλημα:

Δημιουργήστε τις δίκες σας γραμμές - εντολές προγράμματος address-book και χρησιμοποιήστε τες για να περιηγηθείτε, προσθέσετε, τροποποιήσετε, διαγράψτε ή να ερευνήσετε για τις επαφές σας όπως τους φίλους, οικογένεια και συναδέλφους και της πληροφορίες όπως τις διευθύνσεις email και/ή αριθμούς τηλεφώνων. Πληροφορίες πρέπει να αποθηκεύονται στην μεταγενέστερη ανάκτηση. Αυτό είναι εύκολο αν το σκεφτείτε στους όρους όλων των σημαντικών πραγμάτων που έχουμε συναντήσει μέχρι τώρα. Αν ακόμα θέλετε οδηγίες ως προς να το συνεχίσετε, εδώ είναι κρυμμένη.

Κρυμμένα (μην διαβάσετε)

Δημιουργήστε μια κατηγορία να παρουσιάσετε τις πληροφορίες. Χρησιμοποιείστε ένα λεξικό για να αποθηκεύσετε το πρόσωπο των αντικειμένων με το όνομα τους σαν κλειδί. Χρησιμοποιείστε ένα πρότυπο pickle για να αποθηκεύσετε αντικείμενα επίμονα στον σκληρό δίσκο. Χρησιμοποιείστε τη μέθοδο ενσωματωμένο λεξικό για να προσθέσετε, διαγράψετε και να τροποποιήσουμε το πρόσωπο. Όταν είστε ικανός να το κάνετε αυτό, μπορείτε να διεκδικήσετε να είστε ένας προγραμματιστής της Python. Τώρα, αμέσως στείλτε σε μένα ένα mail (http://www.swaroopch.com/contact/) για να με ευχαριστήσετε για αυτό το καλό βιβλίο. Αυτό το βήμα είναι προαιρετικό άλλα συνίστάται. Επίσης, παρακαλώ να σκεφτείτε να κάνετε μια δωρεά, συμβάλλοντας στην βελτίωση ή να γίνετε εθελοντής για να στηρίξετε την συνεχεία της ανάπτυξης του βιβλίου. Αν βρείτε αυτό το πρόγραμμα εύκολο, ορίστε και άλλο ένα:
Εφαρμόστε την αλλαγή των εντολών (http://unixhelp. ed. ac. Uk/ CGI/ man-cgi?replace). Αυτή η εντολή θα αντικαταστήσει ένα string μαζί με ένα άλλο από την λίστα των φακέλων που παρέχονται. Η αντικατάσταση των εντολών μπορούν να είναι απλά ή εξελιγμένα όπως επιθυμείς, για απλά string υποκατάστασης για να δούμε τα πρότυπα(κανονικές εκφράσεις). Μέτα από αυτά, εδώ είναι μερικοί τρόποι για να συνεχίσουμε το ταξίδι στην Python:
Η wxGlade(http://wxglade.sourceforge.net/) GUI χρήστη. Μπορείτε να δημιουργήσετε ελεύθερα σαν ιδιόκτητο λειτουργικό χρησιμοποιώντας wxPython. Για να ξεκινούμε, διαβάστε την wxPython φροντιστήριο (http://zetcode.com/wxpython/).

Παράδειγμα κώδικα

Ο καλύτερος τρόπος για να μάθουμε την γλώσσα είναι να γράψουμε πολλούς κώδικες και να διαβάσουμε πολλούς κώδικες:

- Το PLEAC πρόγραμμα (http://pleac.sourceforge.net/pleac_python/index.html)
- Python παραδείγματα στην java2s (http://www.java2s.om/Code/Python/CatalogPython.htm)
- Python Cookbook (http://code.activestate.com/recipes/langs/python/) αποτελεί
- εξαιρετικά πολύτιμη συλλογή από συνταγές και συμβουλές για το πώς να λύσει ορισμένα είδη προβλημάτων χρησιμοποιώντας Python. Αυτό πρέπει να διαβάσει για κάθε χρήστη της Python.

Ερωτήσεις και απαντήσεις

- Αυθεντικά της Python Dos και Don'ts (http://docs.python.org/dev/howto/doanddont.tml)
- Αυθεντικά της Python FAQ (http://www.python.org/doc/faq/general/)

- Norvig λίστα των Σπάνιων Ερωτήσεων (http://norvig.com/python-iaq.html)
- Η Python συνέντευξη Q & A (http://dev.fyicenter.com/Interview-Questions/Python/index.html)
- Ερωτήσεις StackOverflow με ετικέτα python (http://beta.stackoverflow.com/questions/agged/python)

Συμβουλές και Κόλπα

- Python Tips & Tricks (http://www.siafoo.net/article/52)
- Σύνθετη Ξυλουργική λογισμικού χρησιμοποιώντας την Python (http://ivory.idyll.org/articles/advanced-swc/)
- Καλαίσθητη Python (http://gnosis.cx/publish/tech_index_cp.html) είναι μια εξαιρετική σειρά
  της Python με τα άρθρα από τον David Mertz.

Βιβλία, Χαρτιά, Σεμινάρια, Βίντεο

Η λογική συνέχεια μετά από αυτό το βιβλίο είναι να διαβάσεις από τον Mark Pilgrins το τρομερό βιβλίο για την εισαγωγή στην Python (http://www.diveintopython.org) όπου μπορείτε να το διαβάσετε online επίσης. Η βουτιά στο βιβλίο της Python εξερευνάει της συνήθεις εκφράσεις, XML επεξεργασία, υπηρεσίες ιστού, μονάδα ελέγχου, κ.τ.λ. Λεπτομέρειες.
Άλλοι χρήσιμοι πόροι:

- Δείξε μου τα βίντεο της Python (http://showmedo.com/videos/python)
- GoogleTechTalks βίντεο στην Python (http://youtube.com/results?earch_query=googletechtalks+python)
- Πλήρη κατάλογο Awaretek της Python στο προγράμματα εκμάθησης (http://www.awaretek.com/tutorials.html)
- Η Effbot's Python ζώνη (http://effbot.rg/zone/)
- Links στο τέλος του κάθε Python-URL! email (http://groups.google.com/group/comp.lang.python.announce/t/37de95ef0326293d)
- Python σελίδες (http://pythonpapers.org)

Συζήτηση

Αν έχετε κολλήσει σε ένα πρόβλημα στην Python και δεν ξέρετε που να ρωτήσετε για αυτό, τότε μπορείτε στην comp.langpython ομάδα συζήτησης (http://groups.google.com/group/comp.land.python/topics) είναι το καλύτερο μέρος για να κάνετε την ερώτηση σας. Κάντε πρώτα τις εργασίες ας και να προσπαθήσετε να λύσετε το πρόβλημα από μόνη σας και μετά κάντε την ερώτηση.

Νέα

Αν θα θέλατε να δείτε τα τελευταία νέα στην Python, τότε ακολουθήστε την Official Python Planet (http://planet.python.org) και ή στην Unofficial Python Planet (http://www.planetpython.org).

Εγκατάσταση βιβλιοθηκών

Υπάρχει μεγάλος αριθμός από ανοιχτές πήγες βιβλιοθήκες στο πακέτο της Python (http://pypi.python.org/pypi) όπου μπορείτε να χρησιμοποιήσετε στο δικό σας πρόγραμμα. Αν θέλετε να εγκαταστήσετε τις βιβλιοθήκες, μπορείτε να χρησιμοποιήσετε του Philip J. Eby's είναι πολύ εύκολη σαν εργαλείο.(http://peak.telecommunity.com/DevCenter/EasyInstall#using-easy-install).

Γραφικά του Λογισμικού

Ας υποθέσουμε ότι θέλετε να δημιουργήσετε τα δικά σας γραφικά στο πρόγραμμα της Python. Αυτό μπορεί να γίνει χρησιμοποιώντας μια GUI (Graphical User Interface) βιβλιοθήκη στης δέστρες της Python. Οι δέστρες είναι αυτές που σου επιτρέπουν να γράψεις το πρόγραμμα στην Python και να χρησιμοποιήσεις τις βιβλιοθήκες όπου από μόνες τους είναι γραμμένες στις γλώσσες C ή C++.

PyQt
Αυτό είναι το δέσιμο της Python για το Qt toolkit όπου είναι η βάση για να κτίσουμε την KDE. Η Qt είναι πολύ εύκολη να γραφτή και να χρησιμοποιηθεί και πολύ δυνατή και ειδικότερα στο Qt σχεδιασμό και στην Qt τεκμηρίωση. Η PyQt είναι ελεύθερη αν θελήσετε να δημιουργήσετε μια ανοιχτή πυγή (GPL'ed) λειτουργικού και αν χρειάζεται να αγοράσετε για να δημιουργήσετε μια κλειστή πυγή λειτουργικού. Ξεκινώντας με την Qt 4.5 μπορείτε να την χρησιμοποιήσετε για να δημιουργήσετε μια non-GPL λειτουργικού επίσης. Για να ξεκινήσουμε, διαβάστε την PyQt (**http://zetcode.com/tutorials/pyqt4/**) ή το PyQt βιβλίο (http://www.qtrac.eu/pyqtbook.html)

PyGTK
Αυτό είναι της Python για την GKT+ toolkit όπου είναι η βάση όπου κτίζετε η GNOME. Η GTK+ έχει πολλές ιδιορρυθμίες άλλα όταν σας γίνει οικεία μπορείτε να δημιουργήσετε GUI εφαρμογές γρήγορα. Ο σχεδιαστής γραφικής διεπαφής Glade είναι απαραίτητος. Η τεκμηρίωση θέλει ακόμα κάποιες βελτιώσεις. Η GTK+ δουλεύει καλά στην Linux άλλα μπορεί να δουλέψει και στα Windows. Μπορείτε να δημιουργήσετε και τα δυο επίσης σε ιδιόκτητο λογισμικό χρησιμοποιώντας το GTK+. Για να ξεκινήσουμε, διαβάστε την PyGTK στο πρόγραμμα εκμάθησης (**http://www.pygtk.org/tutorial.html**).

WXPython
Αυτό είναι το δέσιμο της Python για την wxWidgets toolkit. Η wxPython μας άμαθε την καμπύλη εκμάθησης που συνεργάζεται μαζί της. Ωστόσο, είναι φορητό και τρέχει και στην Linux, Windows, Mac και σε ενσωματωμένες πλατφόρμες. Υπάρχουν πολλές IDEs ελεύθερες για την wxPython μαζί και την GUI σχεδιαστές επίσης όπως και το SPE(Stani's Python Editor) (http://spe.pycs.net/) και την wxGlade (http://wxglade.sourceforge.net/) GUI κτίστης. Μπορείτε να φτιάξετε ελεύθερα λειτουργικό χρησιμοποιώντας wxPython. Για να ξεκινήσουμε, διαβάστε την wxPython πρόγραμμα εκμάθησης (http://zetcode.com/wxpython/).

TkInter
Αυτό είναι ένα από τα ποιο παλαιά GUI εργαλείων στην ύπαρξη. Αν χρησιμοποιείτε IDLE, έχετε δει ένα TkInter πρόγραμμα να δουλεύει. Δεν έχει μια από της καλύτερες εμφανίσεις & αισθάνεται γιατί έχει μια παλαιά εμφάνιση. Η TkIner είναι φορητή και δουλεύει και στην Linux/Unix επίσης στα windows. Σημαντικό, Η TkInter είναι κομμάτι της βασικής διανομής της Python. Για να ξεκινήσουμε, διαβάστε της οδηγίες του TkInter
(**http://www.pythonware.com/library/tkinter/introduction/**)
Για περισσότερες επιλογές, δείτε το GuiProgramming wiki σελίδα στον επίσημο δικτυακό τόπο (**http://www.python.org/cgi-bin/moinmoin/GuiProgramming**).

Περίληψη για τα εργαλεία GUI

Δύστυχος, δεν υπάρχουν βασικά εργαλεία του GUI για την Python. Σας προτείνω να διαλέξετε ένα από τα παραπάνω εργαλεία που εξαρτιούνται στην δικιά σας κατάσταση. Ο πρώτος παράγοντας είναι όπου είναι το πρόγραμμα για να τρέχει μόνο με Windows ή με Mac και Linux ή όλα. Ο τρίτος παράγοντας, αν η Linux επιλεγεί την πλατφόρμα, είναι αν είστε χρηστής του KDE ή GNOME στη Linux. Για περισσότερες πληροφορίες και περιεκτική ανάλυση, δείτε την σελίδα 26 της Python papers,Volume 3, Issue 1(http://archive.pythonpapers.org/thePythonPapersVolume3Issue1.pdf).

Σημαντικές υλοποιήσεις

Υπάρχουν συνήθως 2 κομμάτια από γλώσσα προγραμματισμού – η γλωσσά και η λειτουργιά. Μια γλωσσά είναι πως γράφεις κάτι. Η λειτουργιά είναι αυτή που τρέχει το πρόγραμμα.
Χρησιμοποιούμε το λειτουργικό Cpython για να τρέχει το πρόγραμμα. Μας αναφέρεται σαν Cpython γιατί γράφετε στην C γλώσσα και είναι το κλασικό διερμηνέα της Python. Υπάρχουν άλλα λειτουργικά για να τρέξουμε τα προγράμματα της Python:
jython(http://www.jython.org)
Μια εκτέλεση της Python τρέχει σε πλατφόρμα της java. Αυτό σημαίνει ότι μπορείς να χρησιμοποιείς java βιβλιοθήκη και κατηγορίες από την γλώσσα της Python και αντίστροφα.
IronPython(http://www.codeplex.com/wiki/view/aspx?ProjectName=IronPython)

Η εκτέλεση της Python που τρέχει στην πλατφόρμα .NET. Αυτό σημαίνει ότι μπορείς να χρησιμοποιήσεις τις βιβλιοθήκες και τις κατηγορίες της .NET μέσα στην γλώσσα και αντίστροφα της Python.

PyPy(http://codespeak.net/pypy.disk/pypy/doc/home.html)
Η εκτέλεση της Python γράφετε στην Python! Αυτό είναι το σχέδιο έρευνας για να το κάνεις γρήγορο και εύκολο για να βελτιώσει τον διερμηνέα εφόσον ο διερμηνέας από μόνος του γράφεται με γλώσσα δυναμική (σε αντίθεσή με την στατική γλώσσα όπως η C, Java ή C# στις παραπάνω τρεις εκτελέσεις).

Stackless Python(http://www.stackless.com)
Η εκτέλεση της Python είναι ειδικευμένη για την επίδοση του νήματος που βασίζονται. Υπάρχουν επίσης άλλα όπως το CLPython (http://common-lisp.net/project/clipython/)- Η εκτέλεση της Python γράφεται στην common λίστα και το IronMonkey(http://wiki.mozilla.org/Tamarin:ironMonkey) όπου είναι μια πύλη του IronPython για δουλέψει πάνω στο JavaScript διερμηνέα όπου θα μπορούσε να σημαίνει ότι μπορείς να χρησιμοποιήσεις Python(αντί για JavaScript) για να γράψετε το δικό σας ιστό - περιήγηση (“Ajax”) προγράμματος. Κάθε ένα από αυτές τις εκτελέσεις έχουν ειδικευμένες περιοχές όπου είναι χρήσιμες.

Περίληψη
Έχουμε φτάσει στο τέλος αυτού το βιβλίου άλλα, όπως λένε, αυτή είναι η αρχή του τέλους. Είστε τώρα ένας έμπειρος χρήστης της Python και είστε σε θέση να λύσετε πολλά προβλήματα χρησιμοποιώντας την Python. Μπορείτε να ξεκινήσετε αυτόματα τον υπολογιστή σας για να κάνετε όλων των ειδών που αναφέραμε προηγουμένως και αδιανόητα πράγματα για εσάς ή να γράψετε το δικό σας παιχνίδι και πολλά πολλά περισσότερα. Όποτε, ξεκινήστε!

Python en: παράρτημα FLOSS
Ελευθερα/libre και ανοικτού κώδικα λειτουργικό. (FLOSS)
Η FLOSS(http://en.wikipedia.org/wiki/FLOSS) βασίζεται την ιδέα της κοινωνίας, όπου από μόνο της αυτή έχει στηριχτεί στην ιδέα να μοιράζεσαι, και συγκεκριμένα να μοιράζεσαι την γνώση. Η FLOSS είναι ελεύθερη για χρίση, επεξεργασία και διαμοιρασμού. Αν έχετε διαβάσει αυτό το βιβλίο, τότε είστε οικείος με την FLOSS εφόσον έχετε χρησιμοποιήσει την Python και επίσης η Python είναι ανοιχτό λειτουργικό.
Εδώ έχουμε κάποια παραδείγματα του FLOSS για να σας δώσουν μια ιδέα τι μπορεί να δημιουργήσει η κοινωνία του να μοιράζεσαι και της κατασκευής.

Linux. Αυτή είναι ένα FLOSS λειτουργικό σύστημα όπου όλος ο κόσμος την αγκαλιάζει! Ξεκίνησε από την Linus Torvalds ως μαθητής. Τώρα, συγκρίνονται με τα Windows της Microsoft. [Linux Kernel(http://www.kernel.org)]
Ubuntu. Αυτή είναι μια κοινωνία διαμοιρασμός, που χορηγείτε από την Canonical και είναι η πιο διάσημη έκδοση της Linux σήμερα. Σου επιτρέπει να εγκαταστήσεις μια πληθώρα από FLOSS και όλα αυτά είναι εύκολο να χρησιμοποιείς και εύκολο να το εγκαθιστάς. Το καλύτερο από όλα είναι ότι άπλα κάνεις επανεκκίνηση τον υπολογιστή και να τρέξετε τα Linux από το cd! Αυτό σας επιτρέπει να χρησιμοποιήσετε τα νέα OS πριν τα εγκαταστήσετε στον υπολογιστή σας. [ubuntu linux(http://www.ubuntu.com)]
OpenOffice.org Αυτό είναι μια εξαιρετική έκδοση για να γράφεις, παρουσιάζεις, λογιστικό φύλλο, να ζωγραφίσεις και άλλα πολλά. Μπορεί επίσης να ανοίξει ένα αρχείο MS Word και MS PowerPoint αρχεία εύκολα. Τρέχει σε όλες της πλατφόρμες. [openoffice (http://www.openoffice.org)]
Mozilla Firefox Αυτό είναι επόμενης γενιά ιστός περιήγησης όπου μας δίνει ένα ανταγωνισμό με τον Enternet Explorer. Είναι γρήγορο blazingly και κερδίζει τις κριτικές γιατί έχει καλή αναγνώριση και λογική και εντυπωσιακά χαρακτηριστικά. Αυτή η επέκταση επιτρέπει οποιαδήποτε plugins να χρησιμοποιείτε.
Το **thunderbird** είναι ένα εκπληκτικό email όπου μπορείς να έχεις όλους τους λογαριασμούς από τα mail σου γρήγορα. [Mozila Firefox (http://www.mozila.org/products/firefox), Mozila Thunderbird (http://www.mozilla.org/products/thunderbird)]

Mono Αυτό είναι ανοιχτό λειτουργικό για την εκτέλεση της Microsoft.NET πλατφόρμα. Μας επιτρεπτή εφαρμογές .NET για να δημιουργήσουμε και να τρέξουμε το Linux, Windows,FreeBSD, Mac OS και άλλες πλατφόρμες επίσης. [Mono (http://www.mono-project.com), ECMA (http://www.ecma-international.org), Microsoft .NET (http://microsoft.com/net)]
Apache web server. Αυτό είναι από τους γνωστούς ελεύθερους server. Είναι γεγονός, είναι ο διασημότερος server στον πλανήτη. Τρέχει σχεδόν τα μισά από τις ιστοσελίδες. Ναι αυτό που ακούτε- το Apache ελέγχει τις περισσότερες ιστοσελίδες όπως όλοι οι ανταγωνισμοί (συμπεριλαμβανόμενου και της Microsoft IIS) σε συνδυασμό. [apache (http://httpd.apache.org)]
MySQL Αυτό είναι ένα διάσιμο ανοιχτό λειτουργικό δεδομένον server. Ειναι διάσιμο για την υψηλή ταχύτητα. Είναι το M στο διάσιμο LAMP τρέχει στις περισσότερες ιστοσελίδες στο ίντερνετ. [MySQL(http://www.mysql.com)]
VLC Player. Αυτό είναι το πρόγραμμα που παίζει όλα τα βίντεο τον DivX, Mp3, Ogg, VCDs και DVDs ποιος λέει ότι το ελεύθερο λειτουργικό δεν έχει πλακα?[VLC media player (http://www.videolan.org/vlc/)]
GeexBox είναι ένα από τους διαμοιρασμούς της Linux όπου σχεδιαστικέ να παίζει ταινίες με το που ξεκινήσει το CD[GeexBox (http://geexbox.org/en/stert.html)]
Αυτή η λίστα είναι άπλα μια ιδέα – υπάρχουν πολλά εξαιρετικά FLOSS, όπως η Perl γλωσσά, PHP γλωσσά, Drupal περιεχόμενο λειτουργικού για ιστοσελίδες, PostgreSQL serve δεδομένων, TORCS παιχνίδι αγώνων, Kdevelop IDE, Xine – αναπαραγωγής ταινιών, VIM συντάκτης, Quanta+ συντάκτης, Banshee αναπαραγωγής ήχου, GIMP πρόγραμμα επεξεργασίας εικόνας, και αυτή η λίστα μπορεί να συνεχίσει και άλλο.
Για να έχουμε το τελευταίο buzz στο κόσμο του FLOSS, δείτε τα ακόλουθα τις ιστοσελίδες:

- linux.com(thhp://www.linux.com)
- linuxtoday(http://linuxtoday.com)
- nwesforge(http://www.newsforge.com)
- disrtowatch(http://www.distrowatch.com)

Επισκεφτείτε τις παρακάτω ιστοσελίδες για περισσότερες πληροφορίες FLOSS:

- sourceforge (**http://www.**sourceforge.net)
- freshmeat(**http://freshmeat.net**)

Όποτε προχωρήστε και εξερευνήστε ελεύθερα στον κόσμο του FLOSS.

Python en: παράρτημα για

Σχεδόν όλα τα λειτουργικά που έχω χρησιμοποιήσει για την δημιουργία και τα βιβλία είναι ελεύθερα και ανοιχτά.

Η γέννηση του βιβλίου

Στο πρώτο προσχέδιο του βιβλίου, έχω χρησιμοποιήσει το Red Hat 9.0 Linux για το θεμέλιο του συστήματός μου και στο έκτο προσχέδιο, έχω χρησιμοποιήσει την Fedora Core 3 Linux σαν το βασικό μου θεμέλιο. Αρχικά, χρησιμοποιούσα το Kword για να γράψω το βιβλίο (όπως εξηγήθηκε στα ιστορικά μαθήματα στον πρόλογο).

Εφηβική Ηλικία

Μέτα, άλλαξα στο DocBook XML Kate άλλα το βρήκα κουραστικό. Οπότε, άλλαξα σε OpenOffice όπου ήταν άπλα υπέροχο γιατι εχει το επίπεδο να ελέγχεις και να το μετατρέπεις σε PDF, άλλα παράγει και ένα πρόχειρο HTML από το αρχείο. Τέλος, ανακάλυψα το Xemacs άλλα το ξαναέγραψα στο βιβλίο από τα αρχεία DocBook XML(ξανά) μετά αποφάσισα ότι αυτή η μορφή ήταν η μακροπρόθεσμη λύση. Στο έκτο προσχέδιο, αποφάσισα να χρησιμοποιήσω Quanta+ να κάνει όλη την επεξεργασία. Η βασική XSL stylesheets αυτά υπάρχουν μαζί με την Fedora Core 3 Linux όπου ξεκίνησα να χρησιμοποιώ. το πρότυπο της προεπιλεγμένης γραμματοσειράς που χρησιμοποιείται, επίσης. Τέλος πάντων, ξεκίνησα να γράφω CSS έγγραφο για να δίνει χρώμα και στυλ στην σελίδα HTML. Επίσης έχω γράψει ένα λεξικογραφικό αναλυτή, στην Python, όπου αυτόματα περεχεί συντακτικό αναδεικνύοντας σε όλα τα προγράμματα της λίστας.

Τώρα

Για το έβδομο προσχέδιο, χρησιμοποιώ το MediaWiki (http://www.mediawiki.org) ως βασικό για την εγκατάσταση (http://www.swaroopch.com/notes/). Τώρα έκανα όλα σε απευθείας σύνδεση και αυτή που το διαβάζουν μπορούν αμέσως να διαβάσουν/επεξεργαστούν/συζητήσεις μαζί με το Wiki ιστοσελίδες.
Ακόμα χρησιμοποιώ το Vim για επεξεργασία στο ViewSourceWith επέκταση για το Firefox (http://addons.mozilla.org/en-US/firefox/addon/394) ενσωματώνει mazi to Vim.

Για τον συγγραφέα

http://www.swaroopch.com/about/

Python en:Appendix Revision History

- 1.90
- 04/09/2008 είναι ακόμα σε εξέλιξη
- Αναβίωση μετά από ένα κενό 3,5 χρόνια!
- Αναβάθμιση σε Python 3.0
- Ξαναγράψτε χρησιμοποιώντας MediaWiki (πάλι)
- 1.20
- 13/01/2005
- Πλήρες ξαναγράψιμο με χρήση του Quanta + σε FC3 με πολλές διορθώσεις και ενημερώσεις. Πολλά νέα παραδείγματα. Ξανάγραψα το DocBook μου από το μηδέν.
- 1.15
- 28/03/2004
- Μικρές αναθεωρήσεις
- 1.12
- 16/03/2004
- Προσθήκες και διορθώσεις.
- 1.10
- 09/03/2004
- Περισσότερες ορθογραφικές διορθώσεις, χάρη σε πολλούς ενθουσιώδεις αναγνώστες.
- 1.00
- 08/03/2004
- Μετά από πολλή ανάδραση και εισηγήσεις από αναγνώστες, έχω κάνει σημαντικές αναθεωρήσεις στο περιεχόμενο, καθώς και ορθογραφικές διορθώσεις.
- 0.99
- 22/02/2004
- Προστέθηκε ένα νέο κεφάλαιο σχετικά με τις ενότητες. Προστέθηκε λεπτομέρειες σχετικά με δ μεταβλητό αριθμό ορισμάτων σε συναρτήσεις.
- 0.98
- 16/02/2004
- Έγραψε ένα σενάριο Python και ένα CSS stylesheet για τη βελτίωση έξοδο σε XHTML, συμπεριλαμβανομένου ενός
  αργού-ακόμη-λειτουργική λεξικογραφικό αναλυτή για αυτόματη VIM-όπως επισημάνετε τη σύνταξη του
  τηλεοπτικών προγραμμάτων.
- 0.97
- 13/02/2004
- Μια άλλη εντελώς ξαναγραφεί το σχέδιο, σε DocBook XML (ξανά). Κάντε κράτηση έχει δ βελτιωθεί πολύ
  είναι πιο συνεκτικό και ευανάγνωστο.
- 0.93
- 25/01/2004
- Προσθήκη αναφορών στο IDLE και πιο ειδικά για Windows πράγματα
- 0.92
- 05/01/2004

- Αλλαγές σε μερικά παραδείγματα.
- 0.91
- 30/12/2003
- Διόρθωση ορθογραφικών λαθών. Αυτοσχεδιασμός σε πολλά θέματα.
- 0.90
- 18/12/2003
- Προστέθηκε πριν 2 περισσότερα κεφάλαια. Μορφή OpenOffice με τις αναθεωρήσεις.
- 0.60
- 21/11/2003
- Πλήρης αναθεώρηση και επέκταση.
- 0.20
- 20/11/2003
- Διόρθωση μερικών ορθογραφικών και σφαλμάτων.
- 0.15
- 20/11/2003
- Μετατράπηκε σε DocBook XML.
- 0.10
- 14/11/2003
- Αρχικό σχέδιο χρησιμοποιώντας Kword.

Python en:Appendix Changes for Python 3000

- Vim και Emacs συνακτες
- http://henry.precheur.org/2008/4/8/Indenting%20Python%20with%20VIM.html
- http://www.enigmacurry.com/2008/05/09/emacs-as-a-powerful-python-ide/
- String - unicode μόνο
- http://.python.org/dev/3.0/tutorial/introduction.html#about-unicode
- Non-ASCII αναγνωριστικά επιτρέπεται
- http://www.python.org/dev/peps/pep-3131/
- *print()* function
- http://www.python.org/dev/peps/pep-3105/
  r*aw_input() b*ecomes *input()*
- http://www.python.org/dev/peps/pep-3111/
- Ακέραιος Literal Υποστήριξη και Σύνταξη
- http://www.python.org/dev/peps/pep-3127/
  nonlocal δηλωση
- http://www.python.org/dev/peps/pep-3104/
- Λειτουργίες μπορούν να λάβουν * επιχείρημα (varargs) για τις λίστες λέξεων-κλειδιών και μόνο επιχειρήματα
- http://www.python.org/dev/peps/pep-3102/
- Λειτουργίες μπορεί να έχει σημειώσεις (κάνει ένα πέρασμα σημείωμα;)
- http://www.python.org/dev/peps/pep-3107/
- Καλύτερη εξήγηση των ενοτήτων, πακέτα και την οργάνωσή τους (μαζι με την *__init__.py*etc.)
- http://ivory.idyll.org/articles/advanced-swc/#packages
- String.*format() αντί για το %* χειριστής
- http://www.python.org/dev/peps/pep-3101/
- http://docs.python.org/dev/library/string.html#formatstrings
- Αλλαγές μέθοδος Dict
- http://www.python.org/dev/peps/pep-3106/
- Ενσωματωμένο στην κατηγορία σετ, σε δομές δεδομένων κεφάλαιο
- Επίλυση Προβλημάτων
- Χρησιμοποιείστε http://gnuwin32.sourceforge.net/packages/zip.htm στα Windows
- Τάξεις
- http://docs.python.org/dev/3.0/reference/datamodel.html
- Μετακλάσεις
- http://www.python.org/dev/peps/pep-3115/
- Περίληψη κατηγορίες βάσεων

- http://www.python.org/dev/peps/pep-3119/
- Δεν είμαι σίγουρος αν τυχόν αλλαγές που απαιτούνται για τη Νέα I / O
- http://www.python.org/dev/peps/pep-3116/
- Εξαίρεση χειρισμου
- http://www.python.org/dev/peps/pep-0352/
- http://www.python.org/dev/peps/pep-3109/
- http://www.python.org/dev/peps/pep-3110/
- Βασική Βιβλιοθήκη - ενδιαφέρουσες προσθήκες
- Reorganization : http://www.python.org/dev/peps/pep-3108/
  http://docs.python.org/dev/library/warnings.html
- http://docs.python.org/dev/library/logging.html (important)
- http://docs.python.org/dev/library/urllib.html
- http://docs.python.org/dev/library/json.html
- Debugging
- http://docs.python.org/dev/library/pdb.html
- http://docs.python.org/dev/3.0/library/trace.html
- eval, repr/ascii λειτουργίες
- getopt/optparse - πώς να γράψετε ένα πρότυπο γραμμής εντολών του προγράμματος με python;
- κάτι σαν αντικατάσταση;
- http://hpux.connect.org.uk/hppd/hpux/Users/replace-2.24/man.html
- Περισσότερα
- Αποσυσκευασία μπορεί να πάρει το επιχείρημα *
- http://www.python.org/dev/peps/pep-3132/
- μαζί με την δήλωση
- http://www.python.org/dev/peps/pep-0343/
- Μέτα τι?
- Εφαρμογή «αντικαταστήσεις»
- http://unixhelp.ed.ac.uk/CGI/man-cgi?replace
- Αναφέρετε χρήση τουPyPI
- Q&A
- http://docs.python.org/dev/howto/doanddont.html
- http://www.python.org/doc/faq/general/
- http://norvig.com/python-iaq.html
- Βιβλία & Πόροι
- http://www.coderholic.com/free-python-programming-books/
- http://pythonpapers.org
- http://www.mobilepythonbook.org
- http://effbot.org/zone/
- Links στο τέλος του κάθε Python-URL! e-mail
- http://groups.google.com/group/comp.lang.python.announe/t/37de95ef0326293d
- Παράδειγματα
- http://www.rosettacode.org
- http://dev.fyicenter.com/Interview-Questions/Python/index.html
- http://www.java2s.com/Code/Python/CatalogPython.htm
- Συμβουλές & Κόλπα
- http://www.siafoo.net/article/52Source: http://www.swaroopch.com/mediawiki/index.php?oldid=242