Python για Όλους

Εξερευνώντας Δεδομένα με Χρήση της Python 3

Dr. Charles R. Severance

Συντελεστές

Συντακτική υποστήριξη: Elliott Hauser, Sue Blumenberg

Σχεδίαση εξωφύλλου: Aimee Andrion

Ιστορικό Εκτύπωσης

- 2016-Ιουλ-05 Πρώτη ολοκληρωμένη έκδοση σε Python 3.0
- 2015-Δεκ-20 Αρχική, κατά προσέγγιση μετατροπή, σε Python 3.0

Λεπτομέρειες πνευματικών δικαιωμάτων

Πρόλογος

Ανασκευή ενός "Ανοιχτού" Βιβλίου

Είναι πολύ φυσικό για τους ακαδημαϊκούς, στους οποίους λένε συνεχώς «δημοσιεύστε ή χάνετε» να θέλουν να δημιουργούν πάντα κάτι από την αρχή, που να είναι το δικό τους, φρέσκο δημιούργημα. Αυτό το βιβλίο είναι ένα πείραμα στο να μην ξεκινήσω από το μηδέν, αλλά αντίθετα να "αναμείξω" το βιβλίο με τίτλο *Think Python: How to Think Like a Computer Scientist*, που γράφτηκε από τους Allen B. Downey, Jeff Elkner και άλλους.

Τον Δεκέμβριο του 2009, ετοιμαζόμουν να διδάξω *SI502 - Networked Programming* στο Πανεπιστήμιο του Michigan, για πέμπτο συνεχόμενο εξάμηνο και αποφάσισα ότι ήρθε η ώρα να γράψω ένα εγχειρίδιο Python, που να επικεντρωνόταν στην διαχείριση δεδομένων αντί στην κατανόηση αλγορίθμων και στις αφαιρέσεις. Ο στόχος μου, στο SI502, είναι να διδάξω δεξιότητες δια βίου χειρισμού δεδομένων, χρησιμοποιώντας Python. Λίγοι από τους φοιτητές μου σχεδίαζαν να γίνουν επαγγελματίες προγραμματιστές υπολογιστών. Αντίθετα, σχεδίαζαν να γίνουν βιβλιοθηκονόμοι, διευθυντές, δικηγόροι, βιολόγοι, οικονομολόγοι κ.λπ., που έτυχε να θέλουν να χρησιμοποιήσουν επιδέξια την τεχνολογία, στον τομέα που επέλεξαν.

Δεν κατάφερα να βρω το τέλειο βιβλίο Python, με γνώμονα τα δεδομένα του μαθήματός μου, γι' αυτό ξεκίνησα να γράψω ένα τέτοιο βιβλίο. Ευτυχώς σε μια συνεδρίαση της σχολής, τρεις εβδομάδες πριν ξεκινήσω το νέο μου βιβλίο από την αρχή κατά τη διάρκεια των διακοπών, ο Δρ. Atul Prakash μου έδειξε το βιβλίο *Think Python* που είχε χρησιμοποιήσει για να διδάξει το μάθημά του για την Python εκείνο το εξάμηνο. Είναι ένα καλογραμμένο κείμενο Επιστήμης Υπολογιστών με έμφαση σε σύντομες, άμεσες επεξηγήσεις και στη διευκόλυνση της εκμάθησης.

Η συνολική δομή του βιβλίου έχει αλλάξει για να μπορεί κανείς να αντιμετωπίσει προβλήματα ανάλυσης δεδομένων, όσο το δυνατόν γρηγορότερα, και να έχει μια σειρά από παραδείγματα και ασκήσεις σχετικά με την ανάλυση δεδομένων από την αρχή.

Τα κεφάλαια 2–10 είναι παρόμοια με το βιβλίο *Think Python*, αλλά υπήρξαν σημαντικές αλλαγές. Παραδείγματα και ασκήσεις που προσανατολίζονται σε αριθμούς έχουν αντικατασταθεί με ασκήσεις προσανατολισμένες σε δεδομένα. Τα θέματα παρουσιάζονται με τη σειρά που απαιτείται για τη δημιουργία ολοένα και πιο εξελιγμένων λύσεων ανάλυσης δεδομένων. Ορισμένα θέματα όπως το `try` και `except` μεταφέρθηκαν και παρουσιάζονται ως μέρος του κεφαλαίου της δομής επιλογής. Οι συναρτήσεις αντιμετωπίζονται πολύ επιφανειακά, μέχρι να χρειαστούν για την αντιμετώπιση της πολυπλοκότητας του προγράμματος, αντί να εισαχθούν ως πρώιμο μάθημα αφαίρεσης. Σχεδόν όλες οι συναρτήσεις που καθορίζονται από τον χρήστη έχουν αφαιρεθεί από τον κώδικα των παραδειγμάτων και τις ασκήσεις, εκτός του Κεφαλαίου 4. Η λέξη "recursion (αναδρομή)"[1] δεν εμφανίζεται καθόλου στο βιβλίο.

Στα κεφάλαια 1 και 11–16, όλο το υλικό είναι ολοκαίνουργιο, εστιάζοντας σε πραγματικές χρήσεις και απλά παραδείγματα Python για ανάλυση δεδομένων, συμπεριλαμβανομένων των κανονικών εκφράσεων για αναζήτηση και ανάλυση, αυτοματοποίηση εργασιών στον υπολογιστή σας, ανάκτηση δεδομένων από όλο το δίκτυο. ιστοσυγκομιδή δεδομένων, αντικειμενοστραφή προγραμματισμό, χρήση διαδικτυακών υπηρεσιών, ανάλυση δεδομένων XML και JSON, δημιουργία και χρήση βάσεων δεδομένων με χρήση δομημένης γλώσσας ερωτημάτων και οπτικοποίηση δεδομένων.

Ο απώτερος στόχος όλων αυτών των αλλαγών είναι η στροφή από την Επιστήμη των Υπολογιστών στην Πληροφορική και η συμπερίληψη μόνο θεμάτων μιας πρώτης τάξεως τεχνολογίας, που μπορεί να είναι χρήσιμα ακόμα κι αν κάποιος επιλέξει να μην γίνει επαγγελματίας προγραμματιστής.

Οι μαθητές που βρίσκουν αυτό το βιβλίο ενδιαφέρον και θέλουν να το εξερευνήσουν περαιτέρω θα πρέπει να κοιτάξουν το βιβλίο *Think Python* του Allen B. Downey. Επειδή υπάρχει μεγάλη αλληλοεπικάλυψη μεταξύ των δύο βιβλίων, οι μαθητές θα αποκτήσουν γρήγορα δεξιότητες στους πρόσθετους τομείς του τεχνικού προγραμματισμού και της αλγοριθμικής σκέψης που καλύπτονται στο *Think Python*. Και δεδομένου ότι τα βιβλία έχουν παρόμοιο στυλ γραφής, θα

[1] Εκτός, φυσικά, από αυτήν τη γραμμή.

πρέπει να μπορούν να μεταβούν γρήγορα στο *Think Python*, με ελάχιστη προσπάθεια.

Ως κάτοχος πνευματικών δικαιωμάτων του *Think Python*, ο Allen μου έδωσε την άδεια να αλλάξω την άδεια χρήσης του βιβλίου, για το υλικό από το βιβλίο του που παραμένει σε αυτό το βιβλίο, από την άδεια GNU Free Documentation στην πιο πρόσφατη άδεια Creative Commons Attribution — Share Alike. Αυτό ακολουθεί μια γενική αλλαγή στις άδειες ανοιχτής τεκμηρίωσης που μετακινούνται από το GFDL στο CC-BY-SA (π.χ. Wikipedia). Η χρήση της άδειας CC-BY-SA διατηρεί την ισχυρή παράδοση copyleft του βιβλίου, ενώ καθιστά ακόμη πιο απλό για τους νέους συγγραφείς να επαναχρησιμοποιήσουν αυτό το υλικό, όπως τους βολεύει.

Πιστεύω ότι αυτό το βιβλίο χρησιμεύει ως παράδειγμα του γιατί το ανοιχτό υλικό είναι τόσο σημαντικό για το μέλλον της εκπαίδευσης και θέλω να ευχαριστήσω τους Allen B. Downey και Cambridge University Press για τη καινοτόμα απόφασή τους, να διαθέσουν το βιβλίο με ανοιχτά πνευματικά δικαιώματα . Ελπίζω να είναι ευχαριστημένοι με τα αποτελέσματα των προσπαθειών μου και ελπίζω ότι εσείς, οι αναγνώστες, να είστε ευχαριστημένοι με τις συλλογικές *μας* προσπάθειες.

Θα ήθελα να ευχαριστήσω τους Allen B. Downey και Lauren Cowles για τη βοήθειά τους, την υπομονή και την καθοδήγησή τους στην αντιμετώπιση και επίλυση των ζητημάτων πνευματικών δικαιωμάτων γύρω από αυτό το βιβλίο.

Charles Severance
www.dr-chuck.com
Ann Arbor, MI, USA
9 Σεπτεμβρίου 2013

Ο Charles Severance είναι Clinical Associate Professor στο University of Michigan School of Information.

Κεφάλαιο 1

Γιατί πρέπει να μάθετε να γράφετε προγράμματα;

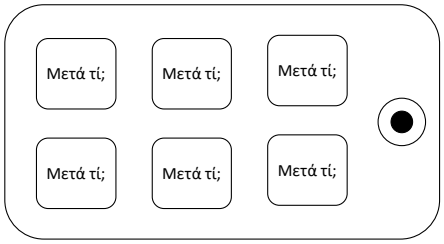
Η συγγραφή προγραμμάτων (ή προγραμματισμός) είναι μια πολύ δημιουργική και ανταποδοτική δραστηριότητα. Μπορείτε να γράψετε προγράμματα για πολλούς λόγους, που κειμένονται από το να αποκομίσετε τα προς το ζην έως το να επιλύσετε ένα δύσκολο πρόβλημα ανάλυσης δεδομένων ή να διασκεδάσετε ή να βοηθήσετε κάποιον άλλο να λύσει ένα πρόβλημα. Αυτό το βιβλίο το υποθέτει ότι *όλοι* πρέπει να γνωρίζουν πώς να προγραμματίζουν και ότι μόλις μάθετε πώς να πρόγραμματίζετε θα καταλάβετε τι θέλετε να κάνετε με τη νέα σας δεξιότητα.

Είμαστε περιτριγυρισμένοι στην καθημερινότητά μας με υπολογιστές που κυμαίνονται από φορητούς υπολογιστές έως κινητά τηλέφωνα. Μπορούμε να σκεφτούμε αυτούς τους υπολογιστές ως τους «προσωπικούς βοηθούς» μας που μπορούν να φροντίσουν πολλά πράγματα για λογαριασμό μας. Το υλικό στους σημερινούς υπολογιστές μας είναι ουσιαστικά κατασκευασμένο για να μας θέτει συνεχώς την ερώτηση "Τι θα θέλατε να κάνω στη συνέχεια;"

![Προσωπικός Ψηφιακός Βοηθός](c:/MAMP/htdocs/py4e/images/pda.png)

Οι προγραμματιστές προσθέτουν ένα λειτουργικό σύστημα και ένα σύνολο εφαρμογών στο υλικό και καταλήγουμε σε έναν Προσωπικό Ψηφιακό Βοηθό, που είναι αρκετά χρήσιμος και ικανός να μας βοηθήσει να κάνουμε πολλά διαφορετικά πράγματα.

Οι υπολογιστές μας είναι γρήγοροι και έχουν τεράστια ποσότητα μνήμης. Θα μπορούσαν να μας βοηθήσουν πολύ αν, μόνο, γνωρίζαμε τη γλώσσα στην οποία θα έπρεπε να μιλήσουμε, για να εξηγήσουμε στον υπολογιστή τι θα θέλαμε να «κάνει στη συνέχεια». Αν γνωρίζαμε αυτή τη γλώσσα, θα μπορούσαμε να πούμε στον υπολογιστή να κάνει διάφορες επαναλαμβανόμενες εργασίες για λογαριασμό μας. Είναι ενδιαφέρον ότι τα πράγματα που μπορούν να κάνουν οι υπολογιστές είναι συχνά τα πράγματα που εμείς οι άνθρωποι θεωρούμε βαρετά και μπερδεμένα.



Εικόνα 1.1: Προσωπικός Ψηφιακός Βοηθός

Για παράδειγμα, κοιτάξτε τις τρεις πρώτες παραγράφους αυτού του κεφαλαίου και πείτε μου τη λέξη που χρησιμοποιείται περισσότερο και πόσες φορές χρησιμοποιείται η λέξη αυτή. Ενώ μπορούσατε να διαβάσετε και να καταλάβετε τις λέξεις σε λίγα δευτερόλεπτα, το να τις μετρήσετε είναι σχεδόν επώδυνο γιατί αυτό δεν είναι το είδος των προβλημάτων, που έχει σχεδιαστεί ο ανθρώπινος νους για να λύνει . Για έναν υπολογιστή, ισχύει το αντίθετο. Η ανάγνωση και η κατανόηση κειμένου από ένα κομμάτι χαρτί είναι δύσκολο για έναν υπολογιστή, αλλά το να μετράει τις λέξεις και να σας λέει πόσες φορές χρησιμοποιήθηκε η πιο συχνά επαναλαμβανόμενη λέξη, είναι πολύ εύκολο για τον υπολογιστή:

```
python words.py
Εισάγετε αρχείο: words.txt
to 16
```

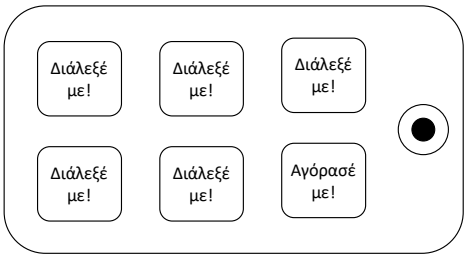
Ο "προσωπικός βοηθός ανάλυσης πληροφοριών" μας είπε γρήγορα ότι η λέξη "to" χρησιμοποιήθηκε δεκαέξι φορές στις τρεις πρώτες παραγράφους αυτού του κεφαλαίου. (Προφανώς στην αγγλική έκδοση του βιβλίου αυτού)

Αυτό ακριβώς το γεγονός, ότι δηλαδή οι υπολογιστές είναι καλοί σε πράγματα στα οποία δεν είναι οι άνθρωποι, είναι και ο λόγος που πρέπει να ειδικευτείτε στην ομιλία "γλώσσας υπολογιστών". Μόλις μάθετε αυτήν τη νέα γλώσσα, μπορείτε να αναθέσετε καθημερινές, τετριμμένες εργασίες στον σύντροφό σας (τον υπολογιστή), εξοικονομώντας χρόνο για τα πράγματα που σας ταιριάζουν και αγαπάτε. Εσείς συνεισφέρετε σε δημιουργικότητα, διαίσθηση και εφευρετικότητα σε αυτήν τη συνεργασία.

Δημιουργικότητα και κίνητρο

Παρόλο που αυτό το βιβλίο δεν προορίζεται για επαγγελματίες προγραμματιστές, ο επαγγελματικός προγραμματισμός μπορεί να είναι μια πολύ ανταποδοτική δουλειά τόσο οικονομικά όσο και προσωπικά. Η δημιουργία χρήσιμων, κομψών και έξυπνων προγραμμάτων για χρήση από άλλους είναι μια πολύ δημιουργική δραστηριότητα. Ο υπολογιστής σας ή ο προσωπικός ψηφιακός βοηθός (PDA) σας, συνήθως περιέχει πολλά διαφορετικά προγράμματα από πολλές διαφορετικές ομάδες προγραμματιστών,που το καθένα ανταγωνίζεται για την προσοχή και το ενδιαφέρον σας. Προσπαθούν με τον καλύτερο δυνατό τρόπο να καλύψουν τις ανάγκες σας και να σας προσφέρουν μια εξαιρετική εμπειρία χρήσης. Σε ορισμένες περιπτώσεις, όταν επιλέγετε ένα κομμάτι λογισμικού, οι προγραμματιστές αποζημιώνονται άμεσα λόγω της επιλογής σας.

Αν σκεφτούμε τα προγράμματα ως τη δημιουργική παραγωγή ομάδων προγραμματιστών, ίσως το παρακάτω σχήμα να είναι μια πιο λογική εκδοχή του PDA μας:

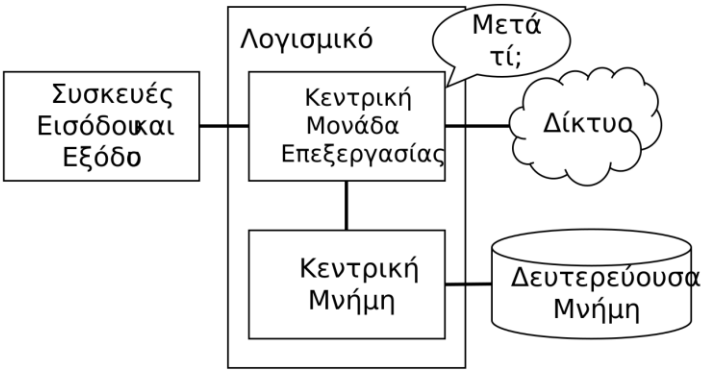


Εικόνα 1.2: Οι προγραμματιστές σας μιλάνε

Προς το παρόν, το κύριο κίνητρό μας δεν είναι να κερδίσουμε χρήματα ή να ευχαριστήσουμε τους τελικούς χρήστες, αλλά αντίθετα να είμαστε πιο παραγωγικοί στο χειρισμό των δεδομένων και των πληροφοριών που θα συναντήσουμε στην καθημερινή μας ζωή. Όταν ξεκινάτε για πρώτη φορά, θα είστε και ο προγραμματιστής και ο τελικός χρήστης των προγραμμάτων σας. Καθώς αποκτάτε δεξιότητες ως προγραμματιστής και ο προγραμματισμός σας φαίνεται πιο δημιουργικός, οι σκέψεις σας μπορεί να στραφούν στην ανάπτυξη προγραμμάτων για άλλους.

Αρχιτεκτονική υλικού υπολογιστών

Πριν ξεκινήσουμε να μαθαίνουμε τη γλώσσα που πρέπει να μιλάμε για να δίνουμε οδηγίες στους υπολογιστές για την ανάπτυξη λογισμικού, πρέπει να μάθουμε λίγα πράγματα για τον τρόπο κατασκευής των υπολογιστών. Αν αποσυναρμολογούσατε τον υπολογιστή ή το κινητό σας τηλέφωνο και κοιτούσατε βαθιά μέσα του, θα βρίσκατε τα ακόλουθα μέρη:



Εικόνα 1.3: Αρχιτεκτονική Υλικού

Οι ορισμοί υψηλού επιπέδου αυτών των τμημάτων είναι οι εξής:

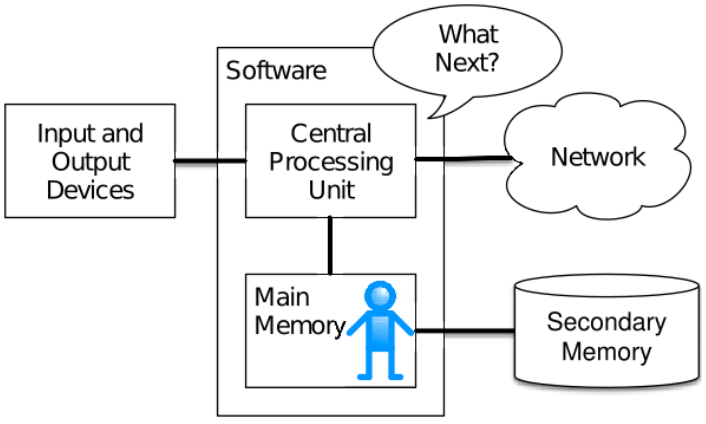
- Η *Κεντρική Μονάδα Επεξεργασίας* (ή CPU) είναι το τμήμα του υπολογιστή που έχει φτιαχτεί για να έχει εμμονή με το "και μετά τί;" Εάν ο υπολογιστής σας έχει χρονιστεί στα 3,0 Gigahertz, αυτό σημαίνει ότι η CPU θα ρωτήσει "μετά τί;" τρία δισεκατομμύρια φορές το δευτερόλεπτο. Θα πρέπει να μάθετε πώς να μιλάτε γρήγορα για να συμβαδίσετε με την CPU.
- Η *Κύρια Μνήμη* χρησιμοποιείται για την αποθήκευση πληροφοριών που χρειάζεται η CPU, γρήγορα. Η κύρια μνήμη είναι σχεδόν τόσο γρήγορη όσο η CPU. Αλλά οι πληροφορίες που είναι αποθηκευμένες στην κύρια μνήμη εξαφανίζονται όταν ο υπολογιστής είναι απενεργοποιημένος.
- Η *Δευτερεύουσα Μνήμη* χρησιμοποιείται επίσης για την αποθήκευση πληροφοριών, αλλά είναι πολύ πιο αργή από την κύρια μνήμη. Το πλεονέκτημα της δευτερεύουσας μνήμης είναι ότι μπορεί να κρατήσει αποθηκευμένες τις πληροφορίες ακόμη και όταν σταματά η τροφοδοσία ρεύματος στον υπολογιστή. Παραδείγματα δευτερεύουσας μνήμης είναι οι

μονάδες δίσκου ή μνήμη flash (συνήθως συναντώνται σε USB sticks και φορητές συσκευές αναπαραγωγής μουσικής).

- Οι συσκευές *Εισόδου και Εξόδου* είναι απλώς η οθόνη, το πληκτρολόγιο, το ποντίκι, το μικρόφωνο, το ηχείο, η επιφάνεια αφής κλπ. Είναι όλοι οι τρόποι αλληλεπίδρασης με τον υπολογιστή.
- Στις μέρες μας, οι περισσότεροι υπολογιστές διαθέτουν επίσης *Σύνδεση Δικτύου* για ανάκτηση πληροφοριών μέσω δικτύου. Μπορούμε να σκεφτόμαστε το δίκτυο ως ένα πολύ αργό μέρος για την αποθήκευση και την ανάκτηση δεδομένων, που μπορεί να μην είναι πάντα "up", δηλαδή διαθέσιμο. Κατά κάποιον τρόπο, το δίκτυο είναι μια πιο αργή και μερικές φορές αναξιόπιστη μορφή *Δευτερεύουσας Μνήμης*.

Ενώ οι περισσότερες λεπτομέρειες, για το πώς λειτουργούν αυτά τα εξαρτήματα είναι καλύτερα να αφεθούν στους κατασκευαστές υπολογιστών, βοηθά να γνωρίζουμε την ορολογία, ώστε να μπορούμε να μιλάμε για αυτά τα βασικά κομμάτια του υπολογιστή καθώς προχωράμε στη συγγραφή των προγράμματά μας.

Ως προγραμματιστές, η δουλειά σας είναι να χρησιμοποιήσετε και να ενορχηστρώσετε καθέναν από αυτούς τους πόρους, για να λύσετε το πρόβλημα που χρειάζεται να λύσετε, και για να αναλύσετε τα δεδομένα, που λαμβάνετε από τη λύση. Ως προγραμματιστές θα "μιλάτε" κυρίως με την CPU και θα της λέτε τι πρέπει να κάνει στη συνέχεια. Μερικές φορές θα πείτε στην CPU να χρησιμοποιήσει την κύρια μνήμη, τη δευτερεύουσα μνήμη, το δίκτυο ή τις συσκευές εισόδου/εξόδου.



Εικόνα 1.4: Πού Βρίσκεστε;

Είστε το άτομο που πρέπει να απαντά στη CPU, στην ερώτηση "Μετά τί;". Αλλά θα ήταν κάπως άβολο αν έπρεπε να σας συρρικνώσουμε, σε ύψος 5 χιλιοστών, και να σας εισάγουμε στον υπολογιστή, μόνο και μόνο για να μπορείτε να δίνετε μια εντολή, τρεις δισεκατομμύρια φορές το δευτερόλεπτο. Επομένως, πρέπει να γράψετε τις οδηγίες σας εκ των προτέρων. Αυτές τις αποθηκευμένες οδηγίες τις ονομάζουμε *πρόγραμμα* και την πράξη της καταγραφής αυτών των οδηγιών και την διασφάλιση της ορθότητας αυτών *προγραμματισμό*.

Κατανόηση του προγραμματισμού

Στο υπόλοιπο αυτού του βιβλίου, θα προσπαθήσουμε να σας μετατρέψουμε σε ένα άτομο, εξειδικευμένο στην τέχνη του προγραμματισμού. Στο τέλος θα είστε *προγραμματιστής* - ίσως όχι επαγγελματίας προγραμματιστής, αλλά τουλάχιστον θα έχετε τις δεξιότητες να εξετάσετε ένα πρόβλημα ανάλυσης δεδομένων/πληροφοριών και να αναπτύξετε ένα πρόγραμμα για την επίλυση του προβλήματος.

Στην ουσία, χρειάζεστε δύο δεξιότητες για να είστε προγραμματιστής:

- Πρώτον, πρέπει να γνωρίζετε τη γλώσσα προγραμματισμού (Python) - πρέπει να γνωρίζετε το λεξιλόγιο και τη γραμματική της. Πρέπει να είστε σε θέση να γράψετε σωστά τις λέξεις, σε αυτήν τη νέα γλώσσα, και να ξέρετε πώς να δημιουργήσετε καλά σχηματισμένες "προτάσεις" σε αυτήν.
- Δεύτερον, πρέπει να είστε σε θέση να "πείτε μια ιστορία". Γράφοντας μια ιστορία, συνδυάζετε λέξεις και προτάσεις, για να μεταφέρετε μια ιδέα στον αναγνώστη. Απαιτείται μια ικανότητα και τέχνη στην κατασκευή της ιστορίας και η ικανότητα στη συγγραφή ιστοριών βελτιώνεται με το να γράφουμε και να λαμβάνουμε κάποια ανατροφοδότηση. Στον προγραμματισμό, το πρόγραμμά μας είναι η «ιστορία» και το πρόβλημα που προσπαθείτε να λύσετε είναι η «ιδέα».

Μόλις μάθετε μία γλώσσα προγραμματισμού, όπως η Python, θα είναι πολύ πιο εύκολο να μάθετε μια δεύτερη γλώσσα, όπως η JavaScript ή η C ++. Η νέα γλώσσα προγραμματισμού θα έχει πολύ διαφορετικό λεξιλόγιο και γραμματική, αλλά οι

δεξιότητες επίλυσης προβλημάτων που απαιτούνται είναι οι ίδιες, σε όλες τις γλώσσες προγραμματισμού.

Θα μάθετε το "λεξιλόγιο" και τις "προτάσεις" της Python αρκετά γρήγορα. Θα χρειαστεί περισσότερος χρόνος για να μπορέσετε να γράψετε ένα πρόγραμμα με συνοχή, για την επίλυση ενός ολοκαίνουργιου προβλήματος. Διδάσκουμε προγραμματισμό όπως και τη γραφή. Αρχίζουμε να διαβάζουμε και να εξηγούμε προγράμματα, μετά γράφουμε απλά προγράμματα και μετά γράφουμε όλο και πιο πολύπλοκα προγράμματα με την πάροδο του χρόνου. Κάποια στιγμή "βρίσκετε τη μούσα σας" και βλέπετε τα μοτίβα μόνοι σας και μπορείτε να δείτε πιο φυσικά πώς να αντιμετωπίσετε ένα πρόβλημα και να γράψετε ένα πρόγραμμα που να το λύνει. Και μόλις φτάσετε σε αυτό το σημείο, ο προγραμματισμός γίνεται μια πολύ ευχάριστη και δημιουργική διαδικασία.

Ξεκινάμε με το λεξιλόγιο και τη δομή των προγραμμάτων Python. Κάντε υπομονή καθώς τα απλά παραδείγματα θα σας θυμίζουν την εποχή που ξεκινήσατε να διαβάζετε για πρώτη φορά.

Λέξεις και προτάσεις

Σε αντίθεση με τις ανθρώπινες γλώσσες, το λεξιλόγιο της Python είναι πραγματικά πολύ μικρό. Αυτό το «λεξιλόγιο» το αποκαλούμε «δεσμευμένες λέξεις». Αυτές είναι λέξεις που έχουν πολύ ιδιαίτερη σημασία για την Python. Όταν η Python βλέπει αυτές τις λέξεις σε ένα πρόγραμμα Python, έχουν μία και μοναδική σημασία. Αργότερα καθώς γράφετε προγράμματα θα φτιάξετε τις δικές σας λέξεις, που έχουν νόημα για εσάς και ονομάζονται *μεταβλητές*. Θα έχετε ένα μεγάλο εύρος επιλογών για την ονοματολογία των μεταβλητών σας, αλλά δεν μπορείτε να χρησιμοποιήσετε καμία από τις δεσμευμένες λέξεις της Python, ως όνομα μεταβλητής.

Όταν εκπαιδεύουμε έναν σκύλο, χρησιμοποιούμε ειδικές λέξεις όπως "κάθισε", "μείνε" και "φέρε". Όταν μιλάτε σε ένα σκυλί και δεν χρησιμοποιείτε καμία από αυτές τις δεσμευμένες λέξεις, απλώς σας κοιτά με ένα ερωτηματικό βλέμμα στο πρόσωπό του μέχρι να πείτε μια δεσμευμένη λέξη. Για παράδειγμα, αν πείτε: "Μακάρι να περπατούσαν περισσότερο οι άνθρωποι, για να βελτιώσουν την υγεία

τους", αυτό που πιθανότατα ακούνε τα περισσότερα σκυλιά είναι "μπλα μπλα μπλα *περπάτα* μπλα μπλα μπλα μπλα." Αυτό συμβαίνει επειδή το "περπάτημα" είναι μια δεσμευμένη λέξη στη γλώσσα του σκύλου. Πολλοί μπορεί να αντιτείνουν ότι η γλώσσα μεταξύ ανθρώπων και γατών δεν έχει δεσμευμένες λέξεις[1] .

Κάποιες από τις δεσμευμένες λέξεις στη γλώσσα, που οι άνθρωποι μιλούν με την Python, είναι και οι ακόλουθες:

```
and       del       global      not       with
as        elif      if          or        yield
assert    else      import      pass
break     except    in          raise
class     finally   is          return
continue  for       lambda      try
def       from      nonlocal    while
```

Πράγματι, και σε αντίθεση με έναν σκύλο, η Python είναι ήδη πλήρως εκπαιδευμένη. Όταν λέτε "try", η Python θα δοκιμάζει, κάθε φορά που το λέτε, χωρίς αποτυχία.

Θα μάθουμε αυτές τις δεσμευμένες λέξεις και πώς χρησιμοποιούνται έγκυρα, αλλά προς το παρόν θα επικεντρωθούμε στο ισοδύναμο, της Python, του "μιλάω" (στη γλώσσα ανθρώπου-σε-σκύλο). Το ωραίο, όταν λέμε στην Python να μιλήσει, είναι ότι μπορούμε ακόμη να της πούμε τι να πει, δίνοντάς της ένα μήνυμα σε εισαγωγικά:

```
print('Γειά σου κόσμε!')
```

Και, μόλις, γράψαμε την πρώτη μας συντακτικά σωστή πρόταση, στην Python. Η πρόταση μας ξεκινά με τη συνάρτηση *print* ακολουθούμενη από μια σειρά κειμένου της επιλογής μας που περικλείεται σε απλά εισαγωγικά. Οι συμβολοσειρές στις εντολές εκτύπωσης περικλείονται σε εισαγωγικά. Τα απλά εισαγωγικά και τα διπλά εισαγωγικά είναι ισοδύναμα. Οι περισσότεροι χρησιμοποιούν απλά εισαγωγικά, εκτός από περιπτώσεις όπου ένα μόνο

[1] http://xkcd.com/231/

εισαγωγικό (το οποίο είναι μπορεί να δηλώνει και "απόστροφο") πρέπει να εμφανιστεί στη συμβολοσειρά.

Συνομιλία με την Python

ώρα που μάθαμε μια λέξη και μια απλή πρόταση, στην Python, πρέπει να γνωρίζουμε και πώς να ξεκινήσουμε μια συνομιλία με την Python, προκειμένου να δοκιμάσουμε τις νέες γλωσσικές μας δεξιότητες.

Για να καταφέρετε να συνομιλήσετε με την Python, πρέπει πρώτα να εγκαταστήσετε το λογισμικό Python στον υπολογιστή σας και να μάθετε πώς να εκκινείτε την Python στον υπολογιστή σας. Αυτό απαιτεί πάρα πολλές λεπτομερές για να συμπεριληφθεί σε αυτό το κεφάλαιο, οπότε προτείνω να συμβουλευτείτε το www.gr.py4e.com όπου σας παρέχω λεπτομερείς οδηγίες και βίντεο, για τη ρύθμιση και την εκκίνηση της Python σε συστήματα Macintosh και Windows. Έτσι, κάποια στιγμή, σε ένα τερματικό ή στο παράθυρο εντολών θα πληκτρολογήσετε *python* και ο διερμηνέας Python θα αρχίσει να εκτελείται σε διαδραστική λειτουργία, οπότε θα δείτε κάτι όπως το εξής:

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  6 2015, 01:54:25)
[MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Η προτροπή >>> είναι ο τρόπος του διερμηνέα Python να σας ρωτήσει: "Τι θέλετε να κάνω στη συνέχεια;" Η Python είναι έτοιμη να συζητήσει μαζί σας. Το μόνο που πρέπει να γνωρίζετε είναι πώς να της μιλήσετε, στη γλώσσα Python.

Ας πούμε για παράδειγμα, ότι δεν γνωρίζατε ούτε τις πιο απλές λέξεις ή προτάσεις της γλώσσας Python. Μπορεί να θέλετε να χρησιμοποιήσετε την κλασσική πρόταση, που χρησιμοποιούν οι αστροναύτες όταν προσγειώνονται σε έναν μακρινό πλανήτη και προσπαθούν να μιλήσουν με τους κατοίκους του πλανήτη:

```
>>> Ερχόμαστε ειρηνικά, παρακαλώ πηγαίνετέ μας στον αρχηγό σας
File "<stdin>", line 1
```

```
  Ερχόμαστε ειρηνικά, παρακαλώ πηγαίνετέ μας στον αρχηγό σας
       ^
SyntaxError: invalid syntax
>>>
```

Αυτό δεν πήγε και τόσο καλά. Αν δεν σκεφτείτε κάτι γρήγορα, οι κάτοικοι του πλανήτη είναι πιθανό να σας επιτεθούν με τα δόρατά τους, να σας βάλουν στη σούβλα, να σας ψήσουν στη φωτιά και να σας φάνε για δείπνο.

Ευτυχώς είχατε ένα αντίγραφο αυτού του βιβλίου, μαζί σας, στο ταξίδι σας και ανοίγοντάς το βρεθήκατε σε αυτήν τη σελίδα, οπότε προσπαθείτε ξανά:

```
>>> print('Γειά σου κόσμε!')
Γειά σου κόσμε!
```

Αυτό λειτούργησε πολύ καλύτερα, οπότε προσπαθείτε να επικοινωνήσετε περισσότερο:

```
>>> print('Πρέπει να είστε οι θρυλικοί θεοί που έρχεστε από τον
ουρανό')
Πρέπει να είστε οι θρυλικοί θεοί που έρχεστε από τον ουρανό
>>> print('Σας περιμέναμε πολύ καιρό')
Σας περιμέναμε πολύ καιρό
>>> print('Ο μύθος μας λέει ότι θα είστε πολύ νόστιμοι με μουστάρδα')
Ο μύθος μας λέει ότι θα είστε πολύ νόστιμοι με μουστάρδα
>>> print 'Θα έχουμε συμπόσιο απόψε, εκτός κι αν το πείτε
File "<stdin>", line 1
  print 'Θα έχουμε συμπόσιο απόψε, εκτός κι αν το πείτε
                                                      ^
SyntaxError: Missing parentheses in call to 'print'
>>>
```

Η συζήτηση πήγαινε τόσο καλά μέχρι που κάνατε ένα μικρό λαθάκι, χρησιμοποιώντας τη γλώσσα Python, και η Python ξανά έβγαλε τα δόρατα.

Σε αυτό το σημείο, θα πρέπει επίσης να συνειδητοποιήσετε ότι, ενώ η Python είναι εκπληκτικά πολύπλοκη και ισχυρή και πολύ επιλεκτική σχετικά με τη σύνταξη που χρησιμοποιείτε για να επικοινωνήσετε μαζί της, η Python *δεν* είναι έξυπνη. Στην

πραγματικότητα συνομιλείτε με τον εαυτό σας, αλλά χρησιμοποιείτε σωστή σύνταξη.

Κατά μία έννοια, όταν χρησιμοποιείτε ένα πρόγραμμα γραμμένο από κάποιον άλλο, η συζήτηση γίνεται μεταξύ εσάς και εκείνου του άλλου προγραμματιστή, με την Python να ενεργεί ως ενδιάμεσος. Η Python είναι ένας τρόπος, για τους δημιουργούς προγραμμάτων, να εκφράσουν πώς υποτίθεται ότι θα προχωρήσει η συνομιλία. Και σε λίγα ακόμη κεφάλαια, θα είστε ένας από αυτούς τους προγραμματιστές, που χρησιμοποιούν την Python για να μιλήσουν στους χρήστες του προγράμματός τους.

Πριν αφήσουμε την πρώτη μας συνομιλία με τον διερμηνέα της Python, μάλλον θα πρέπει να μάθετε τον καθώς πρέπει τρόπο για να πείτε "αντίο" όταν αλληλεπιδράτε με τους κατοίκους του Πλανήτη Python:

```
>>> αντίο
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'αντίο' is not defined
>>> if you don't mind, I need to leave
File "<stdin>", line 1
  if you don't mind, I need to leave
           ^
SyntaxError: invalid syntax
>>> quit()
```

Θα παρατηρήσετε ότι το σφάλμα είναι διαφορετικό για τις δύο πρώτες εσφαλμένες προσπάθειες. Το δεύτερο σφάλμα είναι διαφορετικό γιατί το *if* είναι μια δεσμευμένη λέξη και η Python είδε την δεσμευμένη λέξη και σκέφτηκε ότι προσπαθούσαμε να πούμε κάτι, αλλά αντιλήφθηκε λάθος στη σύνταξη της πρότασης.

Ο σωστός τρόπος για να πείτε "αντίο" στην Python είναι να πληκτρολογήσετε *quit()* στη διαδραστική ερώτηση >>>. Πιθανότατα θα σας έπαιρνε αρκετή ώρα για να το μαντέψετε, οπότε το να έχετε ένα βιβλίο κοντά σας πιθανότατα θα σας φανεί χρήσιμο.

Ορολογία: Διερμηνέας και μεταγλωττιστής

Η Python είναι μια γλώσσα *υψηλού επιπέδου*, κατασκευασμένη ώστε να είναι σχετικά απλό για τους ανθρώπους να διαβάζουν και να γράφουν και για τους υπολογιστές να διαβάζουν και να επεξεργάζονται. Άλλες γλώσσες υψηλού επιπέδου είναι οι Java, C ++, PHP, Ruby, Basic, Perl, JavaScript και πολλές άλλες. Το πραγματικό υλικό μέσα στην κεντρική μονάδα επεξεργασίας (CPU) δεν καταλαβαίνει καμία από αυτές τις γλώσσες υψηλού επιπέδου.

Η CPU καταλαβαίνει μια γλώσσα που ονομάζουμε *γλώσσα μηχανής*. Η γλώσσα της μηχανής είναι πολύ απλή και, ειλικρινά, πολύ κουραστική για να γραφτεί, επειδή τα αναπαριστά όλα με μηδενικά και μονάδες:

```
00101000111010010010101010000001111
11100110000011101010010101101101
...
```

Η γλώσσα της μηχανής φαίνεται, επιφανειακά, αρκετά απλή, δεδομένου ότι υπάρχουν μόνο μηδενικά και μονάδες, αλλά η σύνταξή της είναι ακόμη πιο σύνθετη και πιο πολύπλοκη από της Python. Έτσι, πολύ λίγοι προγραμματιστές γράφουν, ενίοτε, σε γλώσσα μηχανής. Αντ 'αυτού, κατασκευάζουμε διάφορους μεταφραστές που επιτρέπουν στους προγραμματιστές να γράφουν σε γλώσσες υψηλού επιπέδου, όπως η Python ή η JavaScript και αυτοί οι μεταφραστές μετατρέπουν τα προγράμματα σε γλώσσα μηχανής για να εκτελεστεί πραγματικά από την CPU.

Δεδομένου ότι η γλώσσα μηχανής είναι συνδεδεμένη με το υλικό του υπολογιστή, η γλώσσα μηχανής δεν είναι *φορητή*, σε διαφορετικούς τύπους υλικού. Προγράμματα γραμμένα σε γλώσσες υψηλού επιπέδου μπορούν να μεταφερθούν μεταξύ διαφορετικών υπολογιστών χρησιμοποιώντας διαφορετικό διερμηνέα στο νέο μηχάνημα ή επαναμεταφράζοντας τον κώδικα για να δημιουργηθεί μια έκδοση του προγράμματος σε γλώσσας μηχανής για το νέο μηχάνημα.

Αυτοί οι μεταφραστές γλωσσών προγραμματισμού εμπίπτουν σε δύο γενικές κατηγορίες: (1) διερμηνείς και (2) μεταγλωττιστές.

Ένας *διερμηνέας* διαβάζει τον πηγαίο κώδικα του προγράμματος, όπως έχει γραφτεί από τον προγραμματιστή, αναλύει τον πηγαίο κώδικα και ερμηνεύει τις

οδηγίες εν κινήσει. Η Python είναι ένας διερμηνέας και όταν τρέχουμε την Python διαδραστικά, μπορούμε να πληκτρολογήσουμε μια γραμμή Python (μια πρόταση), η Python την επεξεργάζεται αμέσως και είναι έτοιμη για να πληκτρολογήσουμε την επόμενη γραμμή Python.

Μερικές από τις γραμμές του κώδικα λένε στην Python ότι θέλετε να θυμάται κάποια τιμή για αργότερα. Πρέπει να επιλέξουμε ένα όνομα για να απομνημονευθεί αυτή η τιμή και μπορούμε να χρησιμοποιήσουμε αυτό το συμβολικό όνομα για να ανακτήσουμε την τιμή αργότερα. Χρησιμοποιούμε τον όρο *μεταβλητή* για να αναφερθούμε στα ονόματα που χρησιμοποιούμε, για να χειριστούμε αυτά τα αποθηκευμένα δεδομένα.

```
>>> x = 6
>>> print(x)
6
>>> y = x * 7
>>> print(y)
42
>>>
```

Σε αυτό το παράδειγμα, ζητάμε από την Python να θυμάται την τιμή έξι και να χρησιμοποιήσει το όνομα *x*, ώστε να μπορούμε να ανακτήσουμε την τιμή αργότερα. Επαληθεύουμε ότι η Python έχει κρατήσει την τιμή, χρησιμοποιώντας το *print*. Στη συνέχεια, ζητάμε από την Python να ανακτήσει το *x* και να το πολλαπλασιάσει επί επτά και να θέσει την νέα τιμή που υπολογίστηκε στο *y*. Στη συνέχεια, ζητάμε από την Python να εκτυπώσει την τρέχουσα τιμή του *y*.

Παρόλο που πληκτρολογούμε αυτές τις εντολές σε Python μία γραμμή τη φορά, η Python τις αντιμετωπίζει ως μια διατεταγμένη σειρά δηλώσεων με μεταγενέστερες δηλώσεις που μπορούν να ανακτήσουν δεδομένα που δημιουργήθηκαν σε προγενέστερες δηλώσεις. Γράφουμε την πρώτη μας απλή παράγραφο με τέσσερις προτάσεις με λογική και ουσιαστική σειρά.

Είναι στη φύση ενός *διερμηνέα* να μπορεί να έχει μια διαδραστική συνομιλία όπως φαίνεται παραπάνω. Σε έναν *μεταγλωττιστής* πρέπει να παραδοθεί ολόκληρο το πρόγραμμα, σε ένα αρχείο και στη συνέχεια εκτελεί μια διαδικασία για τη

μετάφραση του πηγαίου κώδικα υψηλού επιπέδου σε γλώσσα μηχανής. Στη συνέχεια, ο μεταγλωττιστής αποθηκεύει την γλώσσα μηχανής, που προκύπτει σε ένα αρχείο για μετέπειτα εκτέλεση.

Εάν διαθέτετε σύστημα Windows, συχνά αυτά τα εκτελέσιμα προγράμματα γλώσσας μηχανής έχουν ένα επίθημα ".exe" ή ".dll", που σημαίνει "εκτελέσιμο" και "dynamic link library / βιβλιοθήκη δυναμικών συνδέσμων" αντίστοιχα. Σε Linux και Macintosh, δεν υπάρχει επίθημα που να χαρακτηρίζει μοναδικά ένα αρχείο ως εκτελέσιμο.

Εάν ανοίγατε ένα εκτελέσιμο αρχείο σε έναν επεξεργαστή κειμένου, θα φαινόταν εντελώς τρελό και δεν θα μπορούσε να διαβαστεί:

```
^?ELF^A^A^A^@^@^@^@^@^@^@^@^@^B^@^C^@^A^@^@^@\xa0\x82
^D^H4^@^@^@\x90^]^@^@^@^@^@^@4^@ ^@^G^@(^@$^@!^@^F^@
^@^@4^@^@^@4\x80^D^H4\x80^D^H\xe0^@^@^@\xe0^@^@^@^E
^@^@^@^D^@^@^@^C^@^@^@^T^A^@^@^T\x81^D^H^T\x81^D^H^S
^@^@^@^S^@^@^@^D^@^@^@^A^@^@^@^A\^D^HQVhT\x83^D^H\xe8
....
```

Δεν είναι εύκολο να διαβάσετε ή να γράψετε σε γλώσσα μηχανής, οπότε είναι ωραίο να έχουμε *διερμηνείς* και *μεταγλωττιστές*, που μας επιτρέπουν να γράφουμε σε γλώσσες υψηλού επιπέδου, όπως η Python ή η C.

Τώρα, σε αυτό το σημείο στη συζήτησή μας για μεταγλωττιστές και διερμηνείς, θα πρέπει να αναρωτηθείτε λίγο για τον ίδιο τον διερμηνέα της Python. Σε ποια γλώσσα γράφεται; Είναι γραμμένος σε μεταγλωττισμένη γλώσσα; Όταν πληκτρολογούμε "python", τι ακριβώς συμβαίνει;

Ο διερμηνέας της Python είναι γραμμένος σε μια γλώσσα υψηλού επιπέδου που ονομάζεται "C". Μπορείτε να δείτε τον πραγματικό πηγαίο κώδικα για τον διερμηνέα της Python πηγαίνοντας στο www.python.org και βρίσκοντας τη διαδρομή προς τον πηγαίο κώδικα. Έτσι η Python είναι ένα πρόγραμμα που είναι μεταγλωττισμένο σε γλώσσα μηχανής. Όταν εγκαταστήσατε την Python στον υπολογιστή σας (ή ο προμηθευτής την εγκατέστησε), αντιγράψατε ένα αντίγραφο κώδικα μηχανής του μεταφρασμένου προγράμματος Python στο σύστημά σας.

Στα Windows, ο εκτελέσιμος κώδικας μηχανής για την ίδια την Python είναι πιθανό σε ένα αρχείο με όνομα όπως:

```
C:\Python35\python.exe
```

Αυτά είναι περισσότερα από ό,τι πραγματικά πρέπει να γνωρίζετε για να είστε προγραμματιστής Python, αλλά μερικές φορές αξίζει να απαντήσετε σε αυτές τις μικρές ενοχλητικές ερωτήσεις στην αρχή.

Γράφοντας ένα πρόγραμμα

Η πληκτρολόγηση εντολών στον διερμηνέα της Python είναι ένας πολύ καλός τρόπος για να πειραματιστείτε με τις δυνατότητες της Python, αλλά δεν συνιστάται για την επίλυση πιο πολύπλοκων προβλημάτων.

Όταν θέλουμε να γράψουμε ένα πρόγραμμα, χρησιμοποιούμε έναν συντάκτη κειμένου για να γράψουμε τις εντολές Python σε ένα αρχείο, το οποίο ονομάζεται *script / σενάριο*. Κατά συνθήκη, τα σενάρια Python έχουν ονόματα που τελειώνουν με `.py`.

\index{script} \index{σενάριο}

Για να εκτελέσετε το σενάριο, πρέπει να πείτε στον διερμηνέα της Python το όνομα του αρχείου. Στο τερματικό / γραμμή εντολών, πληκτρολογείτε `python hello.py` ως εξής:

```
$ cat hello.py
print('Γειά σου κόσμε!')
$ python hello.py
Γειά σου κόσμε!
```

Το "$" είναι η προτροπή του λειτουργικού συστήματος και το "cat hello.py" μας δείχνει ότι το αρχείο "hello.py" περιέχει ένα πρόγραμμα Python μιας γραμμής, για την εκτύπωση μιας συμβολοσειράς.

Καλούμε τον διερμηνέα Python και του λέμε να διαβάσει τον πηγαίο κώδικα από το αρχείο "hello.py" αντί να μας ζητήσει τις γραμμές του κώδικα Python διαδραστικά.

Θα παρατηρήσετε ότι δεν ήταν ανάγκη να γράψετε *quit()* στο τέλος του προγράμματος Python στο αρχείο. Όταν η Python διαβάζει τον πηγαίο κώδικα από ένα αρχείο, ξέρει να σταματά όταν φτάσει στο τέλος του αρχείου.

Τι είναι ένα πρόγραμμα;

Ο στοιχειώδης ορισμός ενός *προγράμματος* είναι μια ακολουθία δηλώσεων Python που έχουν σχεδιαστεί για να κάνουν κάτι. Ακόμα και το απλό σενάριο *hello.py* είναι ένα πρόγραμμα. Είναι ένα πρόγραμμα μιας γραμμής και δεν είναι ιδιαίτερα χρήσιμο, αλλά με τον αυστηρό ορισμό, είναι ένα πρόγραμμα Python.

Ίσως είναι πιο εύκολο να καταλάβετε τι είναι ένα πρόγραμμα σκεπτόμενοι ένα πρόβλημα, το οποίο θέλετε να επιλύσετε κατασκευάζοντας ένα πρόγραμμα και μετά να προσπαθήσετε να κατασκευάσετε το πρόγραμμα αυτό.

Ας υποθέσουμε ότι κάνετε μια έρευνα Κοινωνικής Δικτύωσης σε αναρτήσεις στο Facebook και σας ενδιαφέρει η πιο συχνά χρησιμοποιούμενη λέξη σε μια σειρά αναρτήσεων. Θα μπορούσατε να εκτυπώσετε τη ροή των αναρτήσεων στο Facebook και να μελετήσετε το αποτέλεσμα, αναζητώντας την πιο συνηθισμένη λέξη, αλλά αυτό θα πάρει πολύ χρόνο και θα είναι πολύ επιρρεπές σε λάθη. Θα ήταν έξυπνο να γράψετε ένα πρόγραμμα Python για να χειριστεί την εργασία γρήγορα και με ακρίβεια, ώστε να μπορείτε να περάσετε το Σαββατοκύριακο σας κάνοντας κάτι πιο διασκεδαστικό.

Για παράδειγμα, κοιτάξτε το παρακάτω κείμενο για έναν κλόουν και ένα αυτοκίνητο. Κοιτάξτε το κείμενο και εντοπίστε την πιο κοινή λέξη και πόσες φορές εμφανίζεται.

```
the clown ran after the car and the car ran into the tent
and the tent fell down on the clown and the car
```

Στη συνέχεια, φανταστείτε ότι κάνετε αυτήν την δουλειά κοιτάζοντας εκατομμύρια γραμμές κειμένου. Ειλικρινά θα ήταν πιο γρήγορο για εσάς να μάθετε Python και να γράψετε ένα πρόγραμμα Python, για να μετρήσετε τις λέξεις από ό,τι θα ήταν να σαρώσετε τις λέξεις με το χέρι.

Τα ακόμη καλύτερα νέα είναι ότι ετοίμασα ήδη ένα απλό πρόγραμμα, για να βρω την πιο κοινή λέξη σε ένα αρχείο κειμένου. Το έγραψα, το δοκίμασα και τώρα σας το δίνω για να το χρησιμοποιήσετε και να εξοικονομήσετε χρόνο.

```
όνομα = input('Εισάγετε αρχείο:')
handle = open(όνομα, 'r')
πλήθη = dict()

for γραμμή in handle:
    λέξεις = γραμμή.split()
    for λέξη in λέξεις:
        πλήθη[λέξη] = πλήθη.get(λέξη, 0) + 1

maxπλήθος = None
maxλέξη = None
for λέξη, πλήθος in list(πλήθη.items()):
    if maxπλήθος is None or πλήθος > maxπλήθος:
        maxλέξη = λέξη
        maxπλήθος = πλήθος

print(maxλέξη, maxπλήθος)
```

 # Κώδικας στο: http://www.gr-py4e.com/code3/word.py

Δεν χρειάζεται καν να γνωρίζετε την Python για να χρησιμοποιήσετε αυτό το πρόγραμμα. Θα χρειαστεί να ανατρέξετε στο Κεφάλαιο 10 αυτού του βιβλίου, για να κατανοήσετε πλήρως τις εκπληκτικές τεχνικές Python που χρησιμοποιήθηκαν για τη δημιουργία του προγράμματος. Είστε ο τελικός χρήστης, απλά χρησιμοποιείτε το πρόγραμμα και θαυμάζετε την εξυπνάδα του και πώς σας γλίτωσε από τόσο πολύ χειρωνακτική προσπάθεια. Απλώς πληκτρολογείτε τον κώδικα σε ένα αρχείο που ονομάζεται *words.py* και το εκτελείτε ή κατεβάζετε τον πηγαίο κώδικα από το http://www.gr.py4e.com/code3/ και το εκτελείτε.

Αυτό είναι ένα καλό παράδειγμα για το πώς η Python και η γλώσσα Python λειτουργούν ως ενδιάμεσος μεταξύ εσάς (του τελικού χρήστη) και εμένα (του προγραμματιστή). Η Python είναι ένας τρόπος για να ανταλλάξουμε χρήσιμες ακολουθίες οδηγιών (δηλ. προγράμματα) σε μια κοινή γλώσσα που μπορεί να χρησιμοποιηθεί από οποιονδήποτε εγκαταστήσει την Python στον υπολογιστή

του. Κανείς μας λοιπόν δεν μιλάει *με την Python*, αλλά επικοινωνούμε μεταξύ μας *μέσω* της Python.

Τα δομικά στοιχεία των προγραμμάτων

Στα επόμενα κεφάλαια, θα μάθουμε περισσότερα για το λεξιλόγιο, τη δομή των προτάσεων, τη δομή των παραγράφων και τη δομή της "ιστορίας" της Python. Θα μάθουμε για τις ισχυρές δυνατότητες της Python και πώς να συνδυάσουμε αυτές τις δυνατότητες για να δημιουργήσουμε χρήσιμα προγράμματα.

Υπάρχουν ορισμένα εννοιολογικά πρότυπα χαμηλού επιπέδου, που χρησιμοποιούμε για την κατασκευή προγραμμάτων. Αυτές οι κατασκευές δεν είναι μόνο για προγράμματα Python, είναι μέρος κάθε γλώσσας προγραμματισμού από τη γλώσσα μηχανής έως τις γλώσσες υψηλού επιπέδου.

είσοδος : Λήψη δεδομένα από τον "έξω κόσμο". Αυτό μπορεί να είναι η ανάγνωση δεδομένων από ένα αρχείο ή ακόμη και κάποιου είδους αισθητήρα όπως μικρόφωνο ή GPS. Στα αρχικά μας προγράμματα, η εισαγωγή μας θα προέρχεται από τον χρήστη που πληκτρολογεί δεδομένα στο πληκτρολόγιο.

έξοδος : Εμφάνιση των αποτελεσμάτων του προγράμματος σε μια οθόνη ή αποθήκευση τους σε ένα αρχείο ή ίσως εγγραφή τους σε μια συσκευή, όπως μια συσκευή αναπαραγωγής μουσικής ή εκφώνησης κειμένου.

σειριακή εκτέλεση : Εκτέλεση δηλώσεων της μίας μετά την άλλη, με τη σειρά που συναντώνται στο σενάριο.

υπό όρους εκτέλεση : Έλεγχος για ορισμένες συνθήκες και, στη συνέχεια, εκτέλεση ή παράληψη μιας ακολουθίας εντολών.

επαναλαμβανόμενη εκτέλεση : Εκτέλεση κάποιου συνόλου δηλώσεων κατ' επανάληψη, συνήθως με κάποια παραλλαγή.

επαναχρησιμοποίηση : Γραφή μιας σειράς εντολών μία φορά, απόδοση όνομα σε αυτές και, στη συνέχεια, επαναχρησιμοποίησή τους, όπως χρειάζεται κάθε φορά, σε όλο το πρόγραμμά σας.

Ακούγεται, σχεδόν, πάρα πολύ απλό για να είναι αληθινό, και φυσικά δεν είναι ποτέ τόσο απλό. Είναι σαν να λέμε ότι το περπάτημα είναι απλά «να βάζεις το ένα πόδι μπροστά από το άλλο». Η «τέχνη» της συγγραφής ενός προγράμματος συνθέτει και υφαίνει αυτά τα βασικά στοιχεία μαζί πολλές φορές, για να παράγει κάτι που είναι χρήσιμο για τους χρήστες του.

Το παραπάνω πρόγραμμα καταμέτρησης λέξεων χρησιμοποιεί άμεσα όλα τα παραπάνω μοτίβα, εκτός από ένα.

Τι θα μπορούσε να πάει στραβά;

Όπως είδαμε στις πρώτες μας συνομιλίες με την Python, πρέπει να επικοινωνούμε με μεγάλη ακρίβεια, όταν γράφουμε κώδικα Python. Η παραμικρή απόκλιση ή λάθος θα κάνει την Python να σταματήσει την εκτέλεση του προγράμματός σας.

Οι αρχάριοι προγραμματιστές συχνά θεωρούν το γεγονός ότι η Python δεν αφήνει περιθώρια για λάθη ως απόδειξη ότι η Python είναι κακιά, μισητή και σκληρή. Ενώ η Python φαίνεται να συμπαθεί όλους τους άλλους, αυτούς τους γνωρίζει προσωπικά και τους κρατά κακία. Λόγω αυτής της μνησικακίας, η Python παίρνει τα τέλεια γραμμένα προγράμματα μας και τα απορρίπτει ως "ακατάλληλα" μόνο και μόνο για να μας βασανίσει.

```
>>> primt 'Γειά σου κόσμε!'
File "<stdin>", line 1
  primt 'Γειά σου κόσμε!'
                    ^
SyntaxError: invalid syntax
>>> primt ('Γειά σου κόσμε')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'primt' is not defined
>>> Python σε μισώ!
File "<stdin>", line 1
  Python σε μισώ!
       ^
```

```
SyntaxError: invalid syntax
>>> αν βγεις από κει μέσα, θα σου δώσω ένα μαθηματάκι
File "<stdin>", line 1
   αν βγεις από κει μέσα, θα σου δώσω ένα μαθηματάκι
           ^
SyntaxError: invalid syntax
>>>
```

Δεν έχετε να κερδίσετε κάτι από τη διαμάχη με την Python. Είναι απλώς ένα εργαλείο. Δεν έχει συναισθήματα και είναι χαρούμενο και έτοιμο να σας εξυπηρετήσει όποτε το χρειαστείτε. Τα μηνύματα λάθους της ακούγονται σκληρά, αλλά είναι απλώς το κάλεσμα της Python για βοήθεια. Έχει εξετάσει τι πληκτρολογήσατε και απλά δεν μπορεί να καταλάβει τι έχετε εισαγάγει.

Η Python μοιάζει πάρα πολύ με ένα σκύλο, σε αγαπάει άνευ όρων, έχει μερικές λέξεις κλειδιά που καταλαβαίνει, σε κοιτάζει με ένα γλυκό βλέμμα (>>>) και περιμένει να πεις κάτι που καταλαβαίνει. Όταν η Python λέει "SyntaxError: invalid syntax", απλά κουνάει την ουρά της και λέει: "Φαινόσαστε να λέτε κάτι, αλλά απλώς δεν καταλαβαίνω τι εννοούσατε, αλλά συνεχίστε να μου μιλάτε (>>>)."

Καθώς τα προγράμματά σας γίνονται όλο και πιο εξελιγμένα, θα συναντήσετε τρεις γενικούς τύπους σφαλμάτων:

Syntax errors - Συντακτικά λάθη : Αυτά είναι τα πρώτα λάθη που θα κάνετε και είναι τα πιο εύκολα στο να τα διορθώσετε. Ένα συντακτικό λάθος σημαίνει ότι έχετε παραβιάσει τους κανόνες "γραμματικής" της Python. Η Python κάνει ό,τι μπορεί για να δείξει ακριβώς σε ποια γραμμή και ποιον χαρακτήρα μπερδεύτηκε. Το μόνο δύσκολο κομμάτι συντακτικών σφαλμάτων είναι ότι μερικές φορές το λάθος που χρειάζεται διόρθωση είναι στην πραγματικότητα σε προηγούμενο σημείο στο πρόγραμμα και όχι εκεί όπου *παρατήρησε* η Python ότι μπερδεύτηκε. Έτσι, η γραμμή και ο χαρακτήρας που υποδεικνύει η Python σε ένα συντακτικό λάθος, μπορεί να είναι απλώς το σημείο εκκίνησης για την έρευνά σας.

Logic errors -Λογικά λάθη : Ένα λογικό λάθος προκύπτει όταν το πρόγραμμά σας έχει καλή σύνταξη αλλά υπάρχει κάποιο λάθος στη σειρά των δηλώσεων

ή ίσως λάθος στον τρόπο που οι προτάσεις σχετίζονται μεταξύ τους. Ένα καλό παράδειγμα λογικού σφάλματος μπορεί να είναι: "Πάρτε ένα ποτό από το μπουκάλι νερό σας, βάλτε το στο σακίδιο σας, περπατήστε στη βιβλιοθήκη και, στη συνέχεια, τοποθετήστε το καπάκι στο μπουκάλι".

Semantic errors - Σημασιολογικά λάθη : Ένα σημασιολογικό σφάλμα είναι όταν η περιγραφή των βημάτων, που πρέπει να ακολουθηθούν είναι συντακτικά τέλεια και με τη σωστή σειρά, αλλά απλώς υπάρχει ένα λάθος στο πρόγραμμα. Το πρόγραμμα είναι απόλυτα σωστό αλλά δεν κάνει αυτό που το *προορίζατε* για να κάνει. Ένα απλό παράδειγμα θα ήταν αν δίνατε σε κάποιον οδηγίες για το πώς θα φτάσει σε ένα εστιατόριο και λέγατε, "... όταν φτάσετε στη διασταύρωση με το βενζινάδικο, στρίψτε αριστερά, προχωρήστε ένα μίλι και το εστιατόριο είναι ένα κόκκινο κτίριο στα αριστερά σας". Ο φίλος σας έχει αργήσει πολύ και σας καλεί για να σας πει ότι βρίσκεται σε ένα αγρόκτημα και τριγυρνά πίσω από έναν αχυρώνα, χωρίς σημάδι εστιατορίου. Τότε του λέτε "έστριψες αριστερά ή δεξιά στο βενζινάδικο;" και σας λέει, "ακολούθησα τέλεια τις οδηγίες σου, τις έχω καταγράψει, λέει στρίψε αριστερά και προχώρησε ένα μίλι μετά το βενζινάδικο". Και τότε του λέτε: «Λυπάμαι πολύ, γιατί ενώ οι οδηγίες μου ήταν συντακτικά σωστές, δυστυχώς περιείχαν ένα μικρό, αλλά μη εντοπισμένο σημασιολογικό σφάλμα».

Και πάλι, και στους τρεις τύπους λαθών, η Python προσπαθεί σκληρά, απλώς να κάνει ό,τι ακριβώς ζητήσατε.

Εκσφαλμάτωση - Debugging

Όταν, στην Python, προκύπτει ένα λάθος ή ακόμα και όταν σας δίνει ένα αποτέλεσμα που είναι διαφορετικό από αυτό που θα έπρεπε, τότε ξεκινά η αναζήτηση της αιτίας του σφάλματος. Η εντοπισμός σφαλμάτων είναι η διαδικασία εύρεσης της αιτίας του σφάλματος στον κώδικά σας. Όταν κάνετε εντοπισμό σφαλμάτων σε ένα πρόγραμμα, και ειδικά εάν εργάζεστε σε ένα δύσκολο σφάλμα, υπάρχουν τέσσερα πράγματα που πρέπει να δοκιμάσετε:

ανάγνωση : Ελέγξτε τον κώδικά σας, διαβάστε τον ξανά στον εαυτό σας, και ελέγξτε ότι λέει αυτό που θέλατε να πείτε.

εκτέλεση : Πειραματιστείτε κάνοντας αλλαγές και εκτελώντας διαφορετικές εκδόσεις. Συχνά, εάν εμφανίζεται το σωστό πράγμα στο σωστό σημείο του προγράμματος, το πρόβλημα γίνεται εμφανές, αλλά μερικές φορές πρέπει να αφιερώσετε λίγο χρόνο για να φτιάξετε κρηπιδώματα.

μηρυκασμός : Αφιερώστε λίγο χρόνο στο να σκεφτείτε! Τι είδους σφάλμα είναι: σύνταξης, εκτέλεσης, σημασιολογικό; Τι πληροφορίες μπορείτε να πάρετε από τα μηνύματα σφάλματος ή από την έξοδο του προγράμματος; Τι είδους σφάλμα μπορεί να προκαλέσει το πρόβλημα που βλέπετε; Τι αλλάξατε τελευταία, πριν εμφανιστεί το πρόβλημα;

υποχώρηση : Κάποια στιγμή, το καλύτερο που μπορείτε να κάνετε είναι να κάνετε πίσω, αναιρώντας τις πρόσφατες αλλαγές, μέχρι να επιστρέψετε σε ένα πρόγραμμα που λειτουργεί και το καταλαβαίνετε. Στη συνέχεια, μπορείτε να ξανά ξεκινήσετε την κατασκευή του.

Οι αρχάριοι προγραμματιστές κολλάνε μερικές φορές σε ένα από τα παραπάνω και ξεχνούν τα υπόλοιπα. Ο εντοπισμός ενός δύσκολου σφάλματος απαιτεί ανάγνωση, εκτέλεση, μηρυκασμό και, μερικές φορές, υποχώρηση. Εάν κολλήσετε σε μία από αυτές τις δραστηριότητες, δοκιμάστε και τις υπόλοιπες. Κάθε δραστηριότητα συνοδεύεται από το δικό της τρόπο αποτυχίας.

Για παράδειγμα, η ανάγνωση του κώδικα μπορεί να βοηθήσει εάν το πρόβλημα είναι κάποιο τυπογραφικό λάθος, αλλά όχι εάν το πρόβλημα είναι κάποια εννοιολογική παρανόηση. Εάν δεν καταλαβαίνετε τι κάνει το πρόγραμμά σας, μπορεί να το διαβάσετε 100 φορές και να μην εντοπίσετε ποτέ το λάθος, επειδή το σφάλμα είναι στο μυαλό σας.

Η εκτέλεση πειραμάτων μπορεί να βοηθήσει, ειδικά αν εκτελείτε μικρές, απλές δοκιμές. Αλλά εάν εκτελείτε πειράματα χωρίς να σκεφτείτε ή να διαβάσετε τον κώδικά σας, μπορεί να πέσετε σε ένα μοτίβο που ονομάζω "προγραμματισμός τυχαίων περιπάτων", η οποία είναι η διαδικασία πραγματοποίησης τυχαίων αλλαγών έως ότου το πρόγραμμά σας λειτουργήσει σωστά. Περιττό να πούμε ότι ο προγραμματισμός τυχαίων περιπάτων μπορεί να διαρκέσει πολύ.

Πρέπει να αφιερώσετε χρόνο στο να σκεφτείτε. Το Debugging είναι σαν μια πειραματική επιστήμη. Θα πρέπει να έχετε τουλάχιστον μία υπόθεση για το ποιο είναι το πρόβλημα. Εάν υπάρχουν δύο ή περισσότερες υποθέσεις, προσπαθήστε να σκεφτείτε μια δοκιμή που θα εξαλείψει μία από αυτές.

Το διάλειμμα βοηθά στη σκέψη. Το ίδιο και η συζήτηση. Εάν εξηγήσετε το πρόβλημα σε κάποιον άλλο (ή ακόμα και στον εαυτό σας), μερικές φορές θα βρείτε την απάντηση πριν τελειώσετε την ερώτηση.

Αλλά ακόμη και οι καλύτερες τεχνικές εντοπισμού σφαλμάτων θα αποτύχουν εάν υπάρχουν πάρα πολλά λάθη ή εάν ο κώδικας που προσπαθείτε να διορθώσετε είναι πολύ μεγάλος και περίπλοκος. Μερικές φορές η καλύτερη επιλογή είναι να υποχωρήσετε, απλοποιώντας το πρόγραμμα μέχρι να φτάσετε σε κάτι που λειτουργεί και που καταλαβαίνετε.

Οι αρχάριοι προγραμματιστές είναι συχνά απρόθυμοι να υποχωρήσουν επειδή δεν αντέχουν να διαγράψουν ούτε μια γραμμή κώδικα (ακόμα και αν είναι λάθος). Εάν σας κάνει να νιώσετε καλύτερα, αντιγράψτε το πρόγραμμά σας σε άλλο αρχείο πριν αρχίσετε να το απογυμνώνετε. Στη συνέχεια, μπορείτε να αρχίσετε να επικολλάτε ξανά, μικρά κομμάτια κάθε φορά.

Το ταξίδι της μάθησης

Καθώς προχωράτε στο υπόλοιπο βιβλίο, μην φοβηθείτε εάν οι έννοιες δεν φαίνεται να ταιριάζουν και τόσο καλά, την πρώτη φορά. Όταν μαθαίνατε να μιλάτε, δεν ήταν πρόβλημα που τα πρώτα σας χρόνια κάνατε απλώς χαριτωμένους θορύβους και γρυλισμούς. Και δεν πείραζε, αν και χρειάστηκαν έξι μήνες για να μεταβείτε από το απλό λεξιλόγιο σε απλές προτάσεις και χρειάστηκαν 5-6 χρόνια ακόμη για να μεταβείτε από προτάσεις σε παραγράφους και μερικά χρόνια ακόμη για να μπορέσετε να γράψετε ένα ενδιαφέρον, μικρό και πλήρες διήγημα μόνοι σας.

Θέλουμε να μάθετε την Python πολύ πιο γρήγορα, οπότε τα διδάσκουμε όλα ταυτόχρονα στα επόμενα κεφάλαια. Αλλά είναι σαν να μαθαίνετε μια νέα γλώσσα, που χρειάζεται χρόνο για να την απορροφήσετε και να την κατανοήσετε, πριν κάνετε κτήμα σας. Αυτό οδηγεί σε κάποια σύγχυση, καθώς εξετάζουμε και

επανεξετάζουμε θέματα για να προσπαθήσουμε να σας κάνουμε να δείτε τη μεγάλη εικόνα ενώ ορίζουμε τα μικροσκοπικά κομμάτια που συνθέτουν αυτή τη μεγάλη εικόνα. Ενώ το βιβλίο είναι γραμμένο γραμμικά, και αν παρακολουθείτε ένα μάθημα θα προχωρήσει με γραμμικό τρόπο, μην διστάσετε να είστε μη γραμμικοί στον τρόπο με τον οποίο προσεγγίζετε το υλικό. Κοιτάξτε μπροστά και πίσω και διαβάστε χαλαρά. Μια γρήγορη ανάγνωση του πιο προηγμένου υλικού, χωρίς απαραίτητα να κατανοείτε πλήρως τις λεπτομέρειες, μπορεί να οδηγήσει σε καλύτερη κατανόηση του "γιατί;" του προγραμματισμού. Επανεξετάζοντας το προηγούμενο υλικό και ακόμη επαναλαμβάνοντας προηγούμενες ασκήσεις, θα συνειδητοποιήσετε ότι στην πραγματικότητα μάθατε πολύ υλικό ακόμα κι αν το υλικό που κοιτάτε επί του παρόντος φαίνεται λίγο ακατανόητο.

Συνήθως όταν μαθαίνετε την πρώτη σας γλώσσα προγραμματισμού, υπάρχουν μερικές υπέροχες "Αχά!" στιγμές, που μπορείτε να κοιτάξετε από ψηλά το πελέκημα της πέτρας με σφυρί και σμίλη και, κάνοντας ένα βήμα πίσω, να δείτε ότι πράγματι δημιουργείτε ένα όμορφο γλυπτό.

Αν κάτι φαίνεται ιδιαίτερα δύσκολο, συνήθως δεν αξίζει να ξενυχτάτε και να το κοιτάτε επίμονα. Κάντε ένα διάλειμμα, πάρτε έναν υπνάκο, φάτε ένα σνακ, εξηγήστε σε κάποιον τι αντιμετωπίζετε (ή ίσως το σκυλί σας) και, στη συνέχεια, επιστρέψτε σε αυτό με φρέσκια ματιά. Σας διαβεβαιώνω ότι μόλις μάθετε τις έννοιες προγραμματισμού του βιβλίου, θα κοιτάξετε πίσω και θα δείτε ότι όλα ήταν πραγματικά εύκολα και κομψά και απλώς σας πήρε λίγο χρόνο για να το απορροφήσετε.

Γλωσσάριο

bug : Ένα λάθος του προγράμματος.

ανάλυση - parse : Η εξέταση ενός προγράμματος και η ανάλυση της συντακτικής δομής του.

γλώσσα μηχανής : Η γλώσσα χαμηλότερου επιπέδου για λογισμικό, η οποία είναι η γλώσσα που εκτελείται απευθείας από την κεντρική μονάδα επεξεργασίας (CPU). \index{γλώσσα μηχανής}

γλώσσα υψηλού επιπέδου : Μια γλώσσα προγραμματισμού, όπως η Python, που έχει σχεδιαστεί για να είναι εύκολη η ανάγνωσή της και η γραφή της από ανθρώπους.

γλώσσα χαμηλού επιπέδου : Μια γλώσσα προγραμματισμού που έχει σχεδιαστεί για να είναι εύκολη στην κατανόησή της από τον υπολογιστή. Ονομάζεται επίσης "γλώσσα μηχανής" ή "γλώσσα assembly".

δευτερεύουσα μνήμη : Αποθηκεύει προγράμματα και δεδομένα και διατηρεί τις πληροφορίες ακόμη και όταν σταματήσει η τροφοδοσία ρεύματος. Γενικά πιο αργή από την κύρια μνήμη. Παραδείγματα δευτερεύουσας μνήμης αποτελούν οι μονάδες δίσκου και η μνήμη flash σε USB sticks.

διαδραστική λειτουργία - interactive mode : Ένας τρόπος χρήσης του διερμηνέα της Python, πληκτρολογώντας εντολές και εκφράσεις στη γραμμή προτροπής.

διερμηνεία : Για να εκτελέσετε ένα πρόγραμμα σε γλώσσα υψηλού επιπέδου, μεταφράζοντάς το μία προς μία γραμμή.

επίλυση προβλήματος : Η διαδικασία διατύπωσης ενός προβλήματος, εύρεσης μιας λύσης και έκφρασης της λύσης αυτής.

κεντρική μονάδα επεξεργασίας : Η καρδιά κάθε υπολογιστή. Είναι αυτό που τρέχει το λογισμικό που γράφουμε. Ονομάζεται επίσης "CPU" ή "ο επεξεργαστής".

κύρια μνήμη : Αποθηκεύει προγράμματα και δεδομένα. Η κύρια μνήμη χάνει τις πληροφορίες της όταν σταματήσει η τροφοδοσία ρεύματος. \index{κύρια μνήμη}

μεταγλώττιση - compile : Η μετάφραση ενός ολόκληρου προγράμματος, γραμμένου σε γλώσσα υψηλού επιπέδου, ταυτόχρονα, σε γλώσσα χαμηλού επιπέδου, προετοιμάζοντάς το για μετέπειτα εκτέλεση.

μεταφερσιμότητα : Η ιδιότητα ενός προγράμματος να μπορεί να εκτελεστεί σε περισσότερα από ένα είδη υπολογιστών.

πηγαίος κώδικας : Ένα πρόγραμμα σε γλώσσα υψηλού επιπέδου.

πρόγραμμα : Ένα σύνολο οδηγιών, που καθορίζει έναν υπολογισμό.

προτροπή - prompt : When a program displays a message and pauses for the user to type some input to the program.

σημασιολογία : Το νόημα ενός προγράμματος.

σημασιολογικό λάθος : Ένα σφάλμα σε ένα πρόγραμμα που ως αποτέλεσμα έχει το να κάνει κάτι διαφορετικό από αυτό που ήθελε ο προγραμματιστής.

συνάρτηση print : Μια οδηγία προς τον διερμηνέα της Python, που προκαλεί την εμφάνιση τιμών στην οθόνη.

Ασκήσεις

Άσκηση 1: Ποια είναι η λειτουργία της δευτερεύουσας μνήμης σε έναν υπολογιστή;

a) Εκτελέστε όλο τον υπολογισμό και τη λογική του προγράμματος
b) Ανακτά ιστοσελίδες μέσω Διαδικτύου
c) Αποθηκεύστε πληροφορίες μακροπρόθεσμα, ακόμη και πέρα από έναν κύκλο ισχύος
d) Δέχεται είσοδο από τον χρήστη

Άσκηση 2: Τι είναι ένα πρόγραμμα;

Άσκηση 3: Ποια είναι η διαφορά μεταξύ μεταγλωττιστή και διερμηνέα;

Άσκηση 4: Ποιο από τα παρακάτω περιέχει "κώδικα μηχανής";

a) Ο διερμηνέας της Python
b) Το πληκτρολόγιο
c) Το πηγαίο αρχείου Python
d) Ένα έγγραφο επεξεργαστή κειμένου

Άσκηση 5: Τί λάθος υπάρχει στον ακόλουθο κώδικα;

```
>>> primt 'Hello world!'
File "<stdin>", line 1
  primt 'Hello world!'
```

```
                    ^
SyntaxError: invalid syntax
>>>
```

Άσκηση 6: Πού αποθηκεύεται στον υπολογιστή μια μεταβλητή όπως το "x", μετά την ολοκλήρωση της ακόλουθης γραμμής Python;

```
x = 123
```

a) Κεντρική μονάδα επεξεργασίας
b) Κύρια μνήμη
c) Δευτερεύουσα μνήμη
d) Συσκευές εισόδου
e) Συσκευές εξόδου

Άσκηση 7: Τί θα εμφανίσει το ακόλουθο πρόγραμμα;

```
x = 43
x = x + 1
print(x)
```

a) 43
b) 44
c) x + 1
d) Σφάλμα επειδή το x = x + 1 δεν είναι μαθηματικά σωστό

Άσκηση 8: Εξηγήστε καθένα από τα παρακάτω χρησιμοποιώντας ως παράδειγμα μία ανθρώπινη ικανότητα: (1) Κεντρική μονάδα επεξεργασίας, (2) Κύρια μνήμη, (3) Δευτερεύουσα μνήμη, (4) Συσκευή εισόδου και (5) Συσκευή εξόδου. Για παράδειγμα, "Τι είναι το ανθρώπινο ισοδύναμο με μια κεντρική μονάδα επεξεργασίας";

Άσκηση 9: Πώς διορθώνετε ένα "Συντακτικό Λάθος";

Παράρτημα A

Συνεισφορές - Contributors

A.1 Λίστα συνεισφερόντων για το Python for Everybody

Andrzej Wójtowicz, Elliott Hauser, Stephen Catto, Sue Blumenberg, Tamara Brunnock, Mihaela Mack, Chris Kolosiwsky, Dustin Farley, Jens Leerssen, Naveen KT, Mirza Ibrahimovic, Naveen (@togarnk), Zhou Fangyi, Alistair Walsh, Erica Brody, Jih-Sheng Huang, Louis Luangkesorn, and Michael Fudge

Μπορείτε να δείτε λεπτομέρειες συνεισφερόντων στη διεύθυνση:

https://github.com/csev/py4e/graphs/contributors

A.2 Λίστα συνεισφερόντων για το Python for Informatics

Bruce Shields for copy editing early drafts, Sarah Hegge, Steven Cherry, Sarah Kathleen Barbarow, Andrea Parker, Radaphat Chongthammakun, Megan Hixon, Kirby Urner, Sarah Kathleen Barbrow, Katie Kujala, Noah Botimer, Emily Alinder, Mark Thompson-Kular, James Perry, Eric Hofer, Eytan Adar, Peter Robinson, Deborah J. Nelson, Jonathan C. Anthony, Eden Rassette, Jeannette Schroeder, Justin Feezell, Chuanqi Li, Gerald Gordinier, Gavin Thomas Strassel, Ryan Clement, Alissa Talley, Caitlin Holman, Yong-Mi Kim, Karen Stover, Cherie Edmonds, Maria Seiferle, Romer Kristi D. Aranas (RK), Grant Boyer, Hedemarrie Dussan,

A.3 Πρόλογος για το "Think Python"

A.3.1 Η περίεργη ιστορία του "Think Python"

(Allen B. Downey)

Τον Ιανουάριο του 1999 ετοιμαζόμουν να διδάξω ένα εισαγωγικό μάθημα προγραμματισμού στην Java. Το είχα ήδη διδάξει τρεις φορές και είχα απογοητευτεί. Το ποσοστό αποτυχίας στην τάξη ήταν πολύ υψηλό αλλά, ακόμη και για τους φοιτητές που πέτυχαν, το συνολικό επίπεδο επίδοσης ήταν πολύ χαμηλό.

Ένα από τα προβλήματα που εντόπισα ήταν τα βιβλία. Ήταν πολύ μεγάλα, με πάρα πολλές, περιττές λεπτομέρειες σχετικά με την Java και όχι αρκετή καθοδήγηση υψηλού επιπέδου σχετικά με τον τρόπο προγραμματισμού. Και όλοι υπέφεραν από το φαινόμενο της παγίδας: ξεκινούσαν εύκολα, προχωρούσαν σταδιακά και μετά κάπου γύρω στο Κεφάλαιο 5 έχανες τη γη κάτω από τα πόδια σου. Οι φοιτητές θα δεχόταν πολύ νέο υλικό, πολύ γρήγορα, και έπρεπε να περάσουν το υπόλοιπο του εξαμήνου μαζεύοντας τα κομμάτια.

Δύο εβδομάδες πριν από την πρώτη μέρα των μαθημάτων, αποφάσισα να γράψω το δικό μου βιβλίο. Οι στόχοι μου ήταν:

- Κράτα το σύντομο. Είναι καλύτερο για τους μαθητές να διαβάσουν 10 σελίδες παρά να μην διαβάσουν 50 σελίδες.
- Να είσαι προσεκτικός με το λεξιλόγιο. Προσπάθησα να ελαχιστοποιήσω την ορολογία και να ορίσω κάθε έννοια στην πρώτη χρήση της.
- Προχώρα βήμα βήμα. Για να αποφύγω τις παγίδες, πήρα τα πιο δύσκολα θέματα και τα χώρισα σε μια σειρά από μικρά βήματα.
- Εστίασε στον προγραμματισμό, όχι στη γλώσσα προγραμματισμού. Συμπεριέλαβα το ελάχιστο χρήσιμο υποσύνολο της Java και άφησα έξω τα υπόλοιπα.

Χρειαζόμουν έναν τίτλο, οπότε από μια ιδιοτροπία επέλεξα το *How to Think Like a Computer Scientist*.

Η πρώτη μου έκδοση ήταν άκομψη, αλλά λειτούργησε. Οι μαθητές διάβασαν και κατάλαβαν αρκετά ώστε μπορούσα να αφιερώσω χρόνο στην τάξη για τα δύσκολα θέματα, τα ενδιαφέροντα θέματα και (το πιο σημαντικό) μπορούσα να αφήσω τους φοιτητές να εξασκηθούν.

Κυκλοφόρησα το βιβλίο με την άδεια GNU Free Documentation License, η οποία επιτρέπει στους χρήστες να αντιγράφουν, να τροποποιούν και να διανέμουν το βιβλίο.

Το καλύτερο είναι αυτό που συνέβη στη συνέχεια. Ο Jeff Elkner, καθηγητής γυμνασίου στη Βιρτζίνια, υιοθέτησε το βιβλίο μου και το μετέγραψε σε Python. Μου έστειλε ένα αντίγραφό του και είχα την ασυνήθιστη εμπειρία να μάθω Python διαβάζοντας το δικό μου βιβλίο.

Ο Jeff και εγώ αναθεωρήσαμε το βιβλίο, ενσωματώσαμε μια μελέτη περίπτωσης από τον Chris Meyers και το 2001 κυκλοφορήσαμε το *How to Think Like a Computer Scientist Learning with Python*, επίσης υπό την άδεια GNU Free Documentation. Ως Green Tea Press, δημοσίευσα το βιβλίο και άρχισα να πουλάω έντυπα αντίτυπα μέσω του Amazon.com και των κολεγιακών βιβλιοπωλείων. Άλλα βιβλία από το Green Tea Press είναι διαθέσιμα στο greenteapress.com.

Το 2003 άρχισα να διδάσκω στο Olin College και διδάσκω για πρώτη φορά Python. Η σύγκριση με την Java ήταν εντυπωσιακή. Οι μαθητές δυσκολεύτηκαν λιγότερο, έμαθαν περισσότερα, δούλεψαν σε πιο ενδιαφέρουσες εργασίες και γενικά διασκέδασαν πολύ περισσότερο.

Τα επόμενα πέντε χρόνια συνέχισα να αναπτύσσω το βιβλίο, διορθώνοντας λάθη, βελτιώνοντας ορισμένα από τα παραδείγματα και προσθέτοντας υλικό, ειδικά ασκήσεις. Το 2008 άρχισα να εργάζομαι σε μια σημαντική αναθεώρηση — την ίδια στιγμή, επικοινώνησε μαζί μου ένας συντάκτης του Cambridge University Press που ενδιαφερόταν να δημοσιεύσει την επόμενη έκδοση. Καλός συγχρονισμός!

Ελπίζω να σας απολαύσετε τη δουλειά με αυτό το βιβλίο και να σας βοηθήσει να μάθετε να προγραμματίζετε και να σκέφτεστε, τουλάχιστον λίγο, σαν επιστήμονας υπολογιστών.

A.3.2 Ευχαριστίες για το "Think Python"

(Allen B. Downey)

Πρώτον και πιο σημαντικό, ευχαριστώ τον Jeff Elkner, ο οποίος μετέγραψε το Java βιβλίο μου σε Python, με το οποίο ξεκίνησε αυτό το έργο και με σύστησε σε αυτήν που αποδείχθηκε η αγαπημένη μου γλώσσα.

Ευχαριστώ επίσης τον Chris Meyers, ο οποίος συνέβαλε σε πολλές ενότητες του *How to Think Like a Computer Scientist*.

Και ευχαριστώ το Free Software Foundation για την ανάπτυξη της GNU Free Documentation License, η οποία βοήθησε να καταστεί δυνατή η συνεργασία μου με τον Jeff και τον Chris.

Ευχαριστώ επίσης τους συντάκτες του Lulu που εργάστηκαν στο *How to Think Like a Computer Scientist*.

Ευχαριστώ όλους τους φοιτητές που εργάστηκαν με προηγούμενες εκδόσεις αυτού του βιβλίου και όλους τους συντελεστές (που αναφέρονται σε ένα Παράρτημα), που έστειλαν διορθώσεις και προτάσεις.

Και ευχαριστώ τη σύζυγό μου, Lisa, για τη δουλειά της σε αυτό το βιβλίο, και το Green Tea Press, καθώς και οτιδήποτε άλλο.

Allen B. Downey
Needham MA

Ο Allen Downey είναι Associate Professor της Επιστήμης Υπολογιστών στο Franklin W. Olin College of Engineering.

A.4 Λίστα συνεισφερόντων για το "Think Python"

(Allen B. Downey)

Περισσότεροι από 100 οξυδερκείς και προσεκτικοί αναγνώστες έχουν στείλει προτάσεις και διορθώσεις τα τελευταία χρόνια. Η συνεισφορά τους και ο ενθουσιασμός τους για αυτό το έργο ήταν τεράστια βοήθεια.

Για λεπτομέρειες σχετικά με τη φύση καθεμιάς από τις συνεισφορές από αυτά τα άτομα, δείτε το κείμενο "Think Python".

Lloyd Hugh Allen, Yvon Boulianne, Fred Bremmer, Jonah Cohen, Michael Conlon, Benoit Girard, Courtney Gleason and Katherine Smith, Lee Harr, James Kaylin, David Kershaw, Eddie Lam, Man-Yong Lee, David Mayo, Chris McAloon, Matthew J.

Moelter, Simon Dicon Montford, John Ouzts, Kevin Parks, David Pool, Michael Schmitt, Robin Shaw, Paul Sleigh, Craig T. Snydal, Ian Thomas, Keith Verheyden, Peter Winstanley, Chris Wrobel, Moshe Zadka, Christoph Zwerschke, James Mayer, Hayden McAfee, Angel Arnal, Tauhidul Hoque and Lex Berezhny, Dr. Michele Alzetta, Andy Mitchell, Kalin Harvey, Christopher P. Smith, David Hutchins, Gregor Lingl, Julie Peters, Florin Oprina, D. J. Webre, Ken, Ivo Wever, Curtis Yanko, Ben Logan, Jason Armstrong, Louis Cordier, Brian Cain, Rob Black, Jean-Philippe Rey at Ecole Centrale Paris, Jason Mader at George Washington University made a number Jan Gundtofte-Bruun, Abel David and Alexis Dinno, Charles Thayer, Roger Sperberg, Sam Bull, Andrew Cheung, C. Corey Capel, Alessandra, Wim Champagne, Douglas Wright, Jared Spindor, Lin Peiheng, Ray Hagtvedt, Torsten Hübsch, Inga Petuhhov, Arne Babenhauserheide, Mark E. Casida, Scott Tyler, Gordon Shephard, Andrew Turner, Adam Hobart, Daryl Hammond and Sarah Zimmerman, George Sass, Brian Bingham, Leah Engelbert-Fenton, Joe Funke, Chao-chao Chen, Jeff Paine, Lubos Pintes, Gregg Lind and Abigail Heithoff, Max Hailperin, Chotipat Pornavalai, Stanislaw Antol, Eric Pashman, Miguel Azevedo, Jianhua Liu, Nick King, Martin Zuther, Adam Zimmerman, Ratnakar Tiwari, Anurag Goel, Kelli Kratzer, Mark Griffiths, Roydan Ongie, Patryk Wolowiec, Mark Chonofsky, Russell Coleman, Wei Huang, Karen Barber, Nam Nguyen, Stéphane Morin, Fernando Tardio, and Paul Stoop.

Παράρτημα Β

Λεπτομέρειες πνευματικών δικαιωμάτων

Θα προτιμούσα να αδειοδοτήσω το βιβλίο με τη λιγότερο περιοριστική άδεια CC-BY-SA. Όμως, δυστυχώς, υπάρχουν μερικοί αδίστακτοι οργανισμοί, που αναζητούν και βρίσκουν βιβλία με ελεύθερη άδεια και στη συνέχεια δημοσιεύουν και πωλούν σχεδόν αυτούσια αντίγραφα των βιβλίων σε κάποια υπηρεσία εκτύπωσης κατ' απαίτηση, όπως η LuLu ή η KDP. Το KDP έχει προσθέσει (ευτυχώς) μια πολιτική που κατοχυρώνει στις επιθυμίες του πραγματικού κατόχου πνευματικών δικαιωμάτων έναντι ενός μη κατόχου πνευματικών δικαιωμάτων, που προσπαθεί να δημοσιεύσει ένα έργο με ελεύθερη άδεια. Δυστυχώς, υπάρχουν πολλές υπηρεσίες εκτύπωσης κατ' απαίτηση και πολύ λίγες έχουν σκεφτεί τόσο καλά την πολιτική τους όσο το KDP.

Δυστυχώς, πρόσθεσα το στοιχείο NC στην άδεια χρήσης αυτού του βιβλίου για να έχω διέξοδο, σε περίπτωση που κάποιος προσπαθήσει να αντιγράψει αυτό το βιβλίο και να το πουλήσει εμπορικά. Δυστυχώς, η προσθήκη NC περιορίζει τις χρήσεις αυτού του υλικού που θα ήθελα να επιτρέψω. Έτσι, έχω προσθέσει αυτήν την ενότητα του εγγράφου για να περιγράψω συγκεκριμένες καταστάσεις όπου δίνω εκ των προτέρων την άδειά μου να χρησιμοποιηθεί το υλικό αυτού του βιβλίου σε καταστάσεις που κάποιοι μπορεί να θεωρήσουν εμπορικές.

- Εάν εκτυπώνετε περιορισμένο αριθμό αντιγράφων ολόκληρου ή μέρους αυτού του βιβλίου για χρήση σε ένα μάθημα (π.χ., όπως ένα πακέτο μαθημάτων), τότε σας εκχωρείται άδεια CC-BY για αυτό το υλικό για αυτόν τον σκοπό.

- Εάν είστε καθηγητής σε πανεπιστήμιο και μεταφράζετε αυτό το βιβλίο σε άλλη γλώσσα, εκτός από τα αγγλικά και διδάσκετε χρησιμοποιώντας το μεταφρασμένο βιβλίο, τότε μπορείτε να επικοινωνήσετε μαζί μου και θα σας χορηγήσω άδεια CC-BY-SA για αυτό το υλικό, σε σχέση με τη δημοσίευση της μετάφρασής σας. Συγκεκριμένα, θα σας επιτραπεί να πουλήσετε εμπορικά το μεταφρασμένο βιβλίο που προκύπτει.

Εάν σκοπεύετε να μεταφράσετε το βιβλίο, μπορεί να επικοινωνήσετε μαζί μου για να βεβαιωθούμε ότι έχετε όλο το σχετικό υλικό μαθημάτων ώστε να μπορέσετε να το μεταφράσετε και αυτό.

Φυσικά, μπορείτε να επικοινωνήσετε μαζί μου και να ζητήσετε άδεια εάν αυτές οι ρήτρες δεν επαρκούν. Σε όλες τις περιπτώσεις, η άδεια επαναχρησιμοποίησης και αναμίξεως αυτού του υλικού θα χορηγείται εφόσον υπάρχει σαφής προστιθέμενη αξία ή όφελος για μαθητές ή καθηγητές που θα προκύψουν ως αποτέλεσμα της νέας εργασίας.

Charles Severance
www.dr-chuck.com
Ann Arbor, MI, ΗΠΑ
9 Σεπτεμβρίου 2013