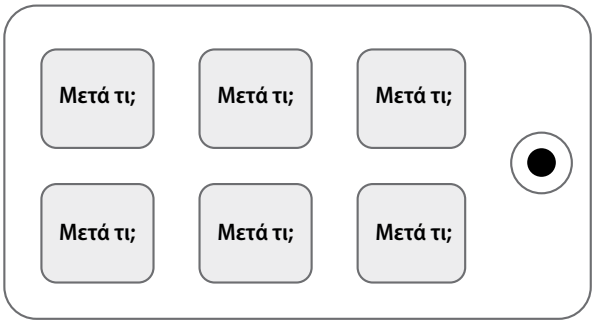
1

ΓΙΑΤΙ ΠΡΕΠΕΙ ΝΑ ΜΑΘΕΤΕ ΝΑ ΓΡΑΦΕΤΕ ΠΡΟΓΡΑΜΜΑΤΑ;

Η συγγραφή προγραμμάτων (ή προγραμματισμός) είναι μια πολύ δημιουργική και ανταποδοτική δραστηριότητα. Μπορεί να γράψετε ένα πρόγραμμα για πολλούς λόγους, που κυμαίνονται 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 κάποιο πρόβλημα. Αυτό το βιβλίο θεωρεί ότι όλοι πρέπει να γνωρίζουν πώς να προγραμματίζουν και ότι μόλις μάθουν πώς να προγραμματίζουν θα βρουν και τι θέλουν να κάνουν με τη νέα αυτή δεξιότητά τους.

Είμαστε περιτριγυρισμένοι στην καθημερινότητά μας από υπολογιστές, που κυμαίνονται από φορητούς υπολογιστές έως κινητά τηλέφωνα. Μπορούμε να φανταστούμε αυτούς τους υπολογιστές ως τους «προσωπικούς μας βοηθούς», που μπορούν να διεκπεραιώσουν πολλά πράγματα για λογαριασμό μας. Το υλικό των σημερινών υπολογιστών είναι ουσιαστικά κατασκευασμένο ώστε να μας θέτει συνεχώς το ερώτημα: «Τι θα θέλατε να κάνω στη συνέχεια;»



Εικόνα 1.1: Προσωπικός ψηφιακός βοηθός

Οι προγραμματιστές προσθέτουν ένα λειτουργικό σύστημα και ένα σύνολο εφαρμογών στο υλικό αυτό και καταλήγουμε σε έναν Προσωπικό Ψηφιακό Βοηθό (PDA), ιδιαίτερα χρήσιμο και ικανό στο να μας βοηθά να κάνουμε πολλά και διαφορετικά πράγματα.

Ο υπολογιστής, σήμερα, είναι γρήγορος και έχει τεράστια ποσότητα μνήμης. Θα μπορούσε να μας βοηθήσει πολύ αν, μόνο, γνωρίζαμε τη γλώσσα στην οποία πρέπει να του μιλήσουμε, για να του εξηγήσουμε 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.

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 δεν είναι αυτό το είδος των προβλημάτων για το οποίο έχει σχεδιαστεί, να λύνει, ο ανθρώπινος νους. Για έναν υπολογιστή, όμως, ισχύει το αντίθετο. Η ανάγνωση και η κατανόηση κειμένου από ένα κομμάτι χαρτί είναι δύσκολη για έναν υπολογιστή, αλλά το να μετρά λέξεις και να σας λέει πόσες φορές χρησιμοποιήθηκε η πιο συχνά επαναλαμβανόμενη λέξη, είναι πολύ εύκολο για αυτόν:

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

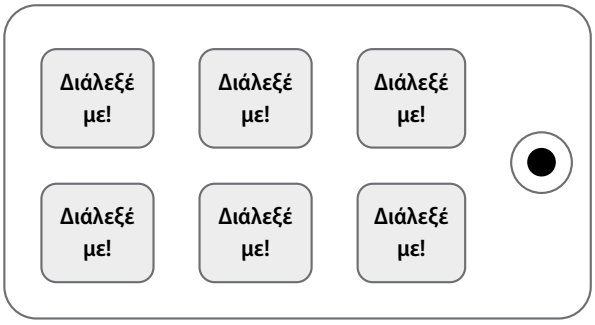
Ο «προσωπικός βοηθός ανάλυσης πληροφοριών» μας είπε γρήγορα ότι η λέξη «to» χρησιμοποιήθηκε δεκαέξι φορές στις τρεις πρώτες παραγράφους αυτού του κεφαλαίου. (Προφανώς στην αγγλική έκδοση του βιβλίου αυτού).

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

1.1 Δημιουργικότητα και κίνητρο

Παρόλο που αυτό το βιβλίο δεν προορίζεται για επαγγελματίες προγραμματιστές, ο προγραμματισμός επαγγελματικά μπορεί να είναι μια πολύ αποδοτική 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 καθένα αγωνίζεται 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:



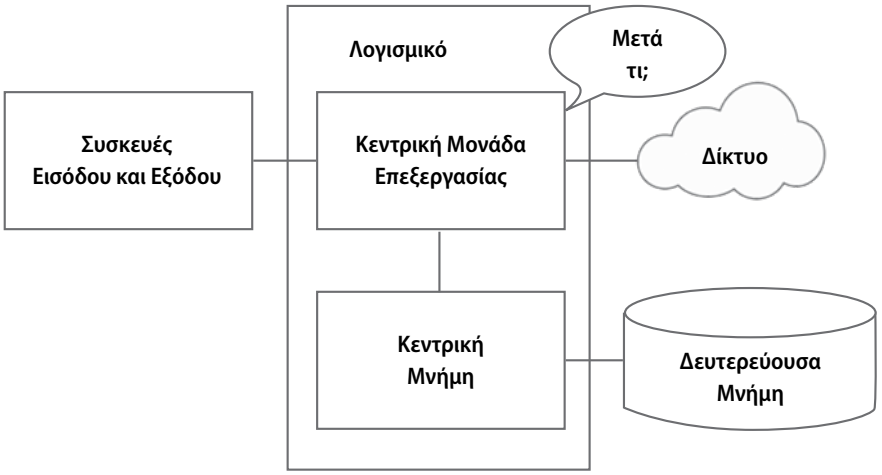
Εικόνα 1.2: Οι προγραμματιστές σάς μιλάνε

Προς το παρόν, το κύριο κίνητρό μας δεν είναι να κερδίσουμε χρήματα ή να ευχαριστήσουμε τους τελικούς χρήστες, αλλά αντίθετα να είμαστε πιο παραγωγικοί στον χειρισμό των δεδομένων και των πληροφοριών που θα συναντήσουμε στην καθημερινή μας ζωή. Όταν ξεκινάτε για πρώτη φορά, 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 και στην ανάπτυξη προγραμμάτων για άλλους.

1.2 Αρχιτεκτονική υλικού υπολογιστών

Πριν ξεκινήσουμε να μαθαίνουμε τη γλώσσα, την οποία πρέπει να μιλάμε, ώστε να δίνουμε οδηγίες στους υπολογιστές, για την ανάπτυξη λογισμικού, πρέπει πρώτα να μάθουμε λίγα πράγματα για τον τρόπο κατασκευής των υπολογιστών. Αν αποσυναρμο-

λογούσατε τον υπολογιστή ή το κινητό σας τηλέφωνο και κοιτούσατε βαθιά μέσα του, θα βρίσκατε τα ακόλουθα μέρη:



Εικόνα 1.3: Αρχιτεκτονική υλικού

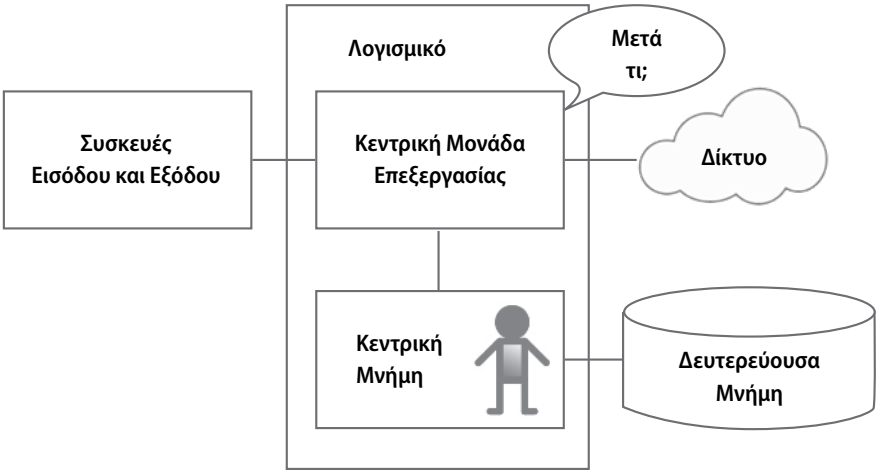
Οι ορισμοί, υψηλού επιπέδου, αυτών των τμημάτων είναι οι εξής:

- Η *Κεντρική Μονάδα Επεξεργασίας* (ή CPU) είναι το τμήμα του υπολογιστή που έχει κατασκευαστεί για να έχει εμμονή με το «και μετά τι;». Εάν ο υπολογιστής σας έχει χρονιστεί στα 3,0 Gigahertz, αυτό σημαίνει ότι η CPU θα ρωτάει «και μετά τι;» 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 γρήγορη αποθήκευση πληροφοριών, που χρειάζεται η CPU. Η κύρια μνήμη είναι σχεδόν τόσο γρήγορη όσο η CPU. Αλλά οι πληροφορίες που είναι αποθηκευμένες στην κύρια μνήμη εξαφανίζονται όταν ο υπολογιστής απενεργοποιηθεί.
- 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 η 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 μία πολύ αργή θέση, 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 θα 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 στη 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 στην ερώτηση τις CPU, «και μετά τι;». 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, τρία δισεκατομμύρια φορές το δευτερόλεπτο. Επομένως, πρέπει να γράψετε τις οδηγίες σας εκ των προτέρων. Αυτές τις αποθηκευμένες οδηγίες τις ονομάζουμε *πρό-*

γραμμα και την πράξη της καταγραφής αυτών των οδηγιών και την διασφάλιση της ορθότητας αυτών *προγραμματισμό*.

1.3 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, χρειάζονται 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 ικανότητα και τέχνη για την κατασκευή της ιστορίας και η ικανότητα αυτή, της συγγραφής ιστοριών, βελτιώνεται καθώς γράφουμε και λαμβάνουμε κάποια ανατροφοδότηση. Στον προγραμματισμό, η «ιστορία» είναι το πρόγραμμά μας και η «ιδέα» είναι το πρόβλημα που προσπαθούμε να λύσουμε.

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 όμως περισσότερος χρόνος για να μπορέσετε να γράψετε ένα πρόγραμμα με συνοχή, για την επίλυση ενός ολοκαίνουργιου προβλήματος. Διδασκόμαστε προγραμματισμό όπως και τη γραφή. Αρχίζουμε να διαβάζουμε και να εξηγούμε προγράμματα, μετά γράφουμε απλά προγράμματα και στη συνέχεια, με την πάροδο του χρόνου, γράφουμε όλο και πιο πολύπλοκα προγράμματα. Κάποια στιγμή «βρίσκετε τη μούσα σας» και βλέπετε τα μοτίβα μόνοι σας και μπορείτε να διακρίνετε, πιο εύκολα, πώς να αντιμετωπίσετε ένα πρόβλημα αλλά και πώς να γράψετε ένα πρόγραμμα που να το επιλύει. 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.

1.4 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 έχουν μία και μοναδική σημασία. Αργότερα, καθώς γράφετε κώδικα, θα δημιουργείτε τις δικές σας λέξεις, που θα έχουν νόημα για εσάς και αυτές 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 μιλήσετε σε ένα σκύλο και δεν χρησιμοποιήσετε καμία από αυτές τις δεσμευμένες λέξεις, αυτός απλά θα σας κοιτά με ένα ερωτηματικό βλέμμα στο πρόσωπό του μέχρι να πείτε κάποια δεσμευμένη λέξη. Για παράδειγμα, αν πείτε: «Μακάρι να περπατούσαν περισσότερο οι άνθρωποι, για να βελτιώσουν την υγεία τους», αυτό που πιθανότατα θα ακούσουν τα περισσότερα σκυλιά είναι «μπλα μπλα μπλα *περπάτα* μπλα μπλα μπλα μπλα.» Αυτό συμβαίνει επειδή το «περπάτα» είναι μια δεσμευμένη λέξη στη γλώσσα των σκύλων. (Φυσικά, 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 χρησιμοποιούν στην επικοινωνία τους με την Python, είναι και οι ακόλουθες:

```
and       continue     except       in         return
as        def          for          is         try
break     elif         if           not        while
class     else         import       or         with
```

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 του «μιλάω», της Python (στη γλώσσα

1 http://xkcd.com/231/

ανθρώπου-σε-σκύλο). Το ωραίο, όταν λέμε στην Python να μιλήσει, είναι ότι μπορούμε ακόμη και να της υπαγορεύσουμε τι να πει, δίνοντάς της ένα μήνυμα σε εισαγωγικά:

```
print('Γεια σου κόσμε!')
```

Και, μόλις, γράψαμε την πρώτη μας, συντακτικά σωστή, πρόταση σε Python. Η πρότασή μας ξεκινά με τη συνάρτηση *print*, ακολουθούμενη από ένα κείμενο της επιλογής μας, που περικλείεται σε απλά εισαγωγικά. Το κείμενο (συμβολοσειρά) στις εντολές εκτύπωσης περικλείεται σε εισαγωγικά. Τα απλά εισαγωγικά και τα διπλά εισαγωγικά είναι ισοδύναμα. Οι περισσότεροι χρησιμοποιούν απλά εισαγωγικά, εκτός από την περίπτωση όπου ένα μονό εισαγωγικό (το οποίο μπορεί να δηλώνει και «απόστροφο») πρέπει να περιέχεται στη συμβολοσειρά.

1.5 Συνομιλία με την Python

Τώρα που κατέχουμε μια λέξη και μια απλή πρόταση, στην Python, πρέπει να μάθουμε και πώς να ξεκινήσουμε μια συνομιλία μαζί της, προκειμένου να δοκιμάσουμε τις νέες γλωσσικές μας δεξιότητες.

Για να καταφέρετε να συνομιλήσετε με την Python, πρέπει πρώτα να εγκαταστήσετε το λογισμικό της Python στον υπολογιστή σας καθώς 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 διερμηνευτής της Python θα αρχίσει να εκτελείται σε διαδραστική λειτουργία, οπότε θα δείτε κάτι σαν το παρακάτω:

```
Python 3.11.6 (main, Nov  2 2023, 04:39:43)
   [Clang 14.0.3 (clang-1403.0.22.14.1)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Η προτροπή >>> είναι ο τρόπος του διερμηνευτή της 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, σε γλώσσα Python.

Ας πούμε για παράδειγμα, ότι δεν γνωρίζατε ούτε τις πιο απλές λέξεις ή προτάσεις της γλώσσας Python. Μπορεί να θέλατε 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, 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 έχετε ένα αντίγραφο αυτού του βιβλίου, μαζί σας, στο ταξίδι σας και ανοίγοντάς το βρίσκεστε σε αυτήν ακριβώς τη σελίδα, οπότε προσπαθείτε ξανά:

```
>>> print('Γεια σου κόσμε!')
Γεια 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'
SyntaxError: unterminated string literal (detected at line 1)
>>>
```

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

Σε αυτό το σημείο, θα πρέπει 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 κεφάλαια παρακάτω, θα είστε ένας από αυτούς τους προγραμματιστές, που χρησιμοποιούν την Python για να μιλήσουν με τους χρήστες του προγράμματός τους.

Πριν αφήσουμε την πρώτη μας συνομιλία με τον διερμηνευτή 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
SyntaxError: unterminated string literal (detected at line 1)
>>> quit()
```

Θα παρατηρήσετε ότι το σφάλμα είναι διαφορετικό στις 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 προτροπή >>>. Πιθανότατα θα σας έπαιρνε αρκετή ώρα για να το μαντέψετε, οπότε το να έχετε ένα βιβλίο κοντά σας πιθανότατα θα σας φανεί χρήσιμο.

1.6 Ορολογία: Διερμηνευτής και Μεταγλωττιστής

Η Python είναι μια γλώσσα *υψηλού επιπέδου*, κατασκευασμένη ώστε να είναι σχετικά απλό, για τους ανθρώπους, να διαβάζουν και να γράφουν σε αυτή και για τους υπολογιστές να διαβάζουν και να επεξεργάζονται. Άλλες γλώσσες υψηλού επιπέδου είναι