Χρησιμοποιώντας την γλώσσα προγραμματισμού Python

Μιχαήλ Ανάργυρος Ζαμάγιας – ΤΠ5000

5 Ιουνίου 2020

Περιεχόμενα

Εισαγωγή

> How you look at it is pretty much how you'll see it.
>
> — Rasheed Ogunlaru

1.1 Πρόλογος

Άρχισα να προγραμματίζω, να γράφω Python συγκεκριμένα, στην πρώτη λυκείου ως χόμπι. Είχα χάσει το ενδιαφέρον μου για το hardware, ενώ παράλληλα μου άρεσε όλο και περισσότερο η διαδικασία του να κάνω το hardware χρήσιμο για εμένα. Κατά την διάρκεια του λυκείου είχα καταφέρει να γράψω αρκέτα ενδιαφέροντα προγράμματα, όπως μια μικρή μηχανή αναζήτησης, ένα πογραμμα με ρόλο μετερεολόγου και ένα ξυπνητήρι που έπαιζε βίντεο αντί ήχο όταν ερχόταν η ώρα. Τώρα, μερικά χρόνια μετά, θέλω να εξοικειώσω τον αναγνώστη με τα βασικά της Python μέσω αυτού του εγχειριδίου. Δεν χρειάζεται να επιμείνετε στα βασικά, ούτε χρειάζονται ανεπτυγμένες γνώσεις προγραμματισμού για να φτιάξετε κάποιο πρόγραμμα που θα σας διευκολύνει με κάποιες από τις εργασίες σας στον υπολογιστή, π.χ. την αυτοματοποίηση μιας διαδικασίας. Γιατί να επαναλαμβάνετε κάποια διαδικασία ξανά και ξανά ενώ μπορείτε να την αυτοματοποιήσετε; Η γιατί να εγκαθιστάτε λογισμικό από τρίτους στο μηχάνημά σας ενώ πιθανότατα μπορείτε να φτιάξετε μόνοι σας αυτό το εργαλείο;

1.2 Γιατί να ασχοληθεί κάποιος με τον προγραμματισμό;

Η γνώση του προγραμματισμού δίνει σε κάποιον την δεξιότητα να μετατρέπει ένα πρόβλημα σε κάτι που μπορεί να κατανοήσει και να λύσει, τις περισσότερες φορές, ένας υπολογιστής. Μάλιστα, θα υποστήριζα ότι είναι μια βασική δεξιότητα για κάποιον άνθρωπο στην εποχή μας, εάν εκείνος χρησιμοποιεί υπολογιστές.

> Codes are a puzzle. A game, just like any other game.
>
> — Alarn Turing

1.3 Γιατί να μάθει κάποιος Python;

Η Python είναι μια δερμηνευόμενη, γενικού σκοπού και υψηλού επιπέδου γλώσσα προγραμματισμού. Δηλαδή:

- μπορεί να εκτελεστεί κώδικας χωρίς να αποτελεί μέρος κάποιου προγράμματος μέσα από τον διερμηνέα,
- βρίσκεται πίσω από πολλές εφαρμογές και προσφέρει λύση σε πολλά προβλήματα και
- το συντακτικό της είναι απλούστερο, ευκολότερο στην κατανόηση σε σύγκριση με άλλες γλώσσες προγραμματισμού.

1.4 Ποιος είναι ο στόχος αυτού του εγχειριδίου;

Σε αυτό το εγχειρίδιο παρουσιάζονται συνοπτικά κι εν συντομία τα βασικά στοιχεία και χαρακτηριστικά της Python. Στόχος είναι ο αναγνώστης να δει πως «δουλεύει» η γλώσσα, να εξοικείωθει σε έναν ικανοποιητικό βαθμό με αυτήν και να καταφέρει να την αξιοποιήσει στην καθημερινότητά του, αλλά όχι να μάθει προγραμματισμό.

Βασικά Στοιχεία

Τα βασικά είναι βαρετά. Επιτρέψτε μου να εξηγήσω. Λόγω της φύσης της γλώσσας δεν χρειάζεστε να επιμείνετε στα βασικά. Είναι αρκετά εύκολο με μια καλή κατανόηση των βασικών «εργαλείων» της γλώσσας να χτίσετε αρκετά πράγματα. Τα βασικά «εργαλεία» σε αυτήν την περίπτωση είναι οι προτάσεις υπό συνθήκη, βρόχοι (αλλιώς επαναλήψεις), συναρτήσεις και τα δομοστοιχεία της γλώσσας. Ο πιο αποδοτικός και γρήγορος τρόπος να εξοικειωθεί ο αναγνώστης με την γλώσσα είναι να ασχοληθεί περισσότερο με το πρακτικό κομμάτι, με project, όπως π.χ. η δημιουργεία εργαλείων που θα του γλιτώσει αρκετό χρόνο από συγκεκριμένες ρουτίνες στον υπολογιστή.

> The most important property of a program is whether it accomplishes the intention of its user.
>
> — Tony Hoare

2.1 Ξεκινώντας

Η Python δεν έρχεται προεγκατεστημένη στα Windows και στις διανομές Linux συνήθως η προεγκατεστημένη έκδοση είναι η 2.7, η οποία πλέον έχει σταματήσει να λαμβάνει ενημερώσεις και υποστήριξη. Άρα, χρειάζεται μια νεότερη έκδοση της Python.

Η Python είναι μια cross-platform γλώσσα προγραμματισμού, με υποστήριξη και στα τρία κύρια λειτουργικά συστήματα (Windows, Linux και MacOS) με μερικές διαφοροποιήσεις ανά το λειτουργικό σύστημα.

Σε αυτό το εγχειρίδιο χρησιμοποιούνται οι `Python 3.7.6` και `conda 4.8.3` στην διανομή Linux `Pop!_OS`. Τέλος, τα προγράμματα στην Python έχουν κατάληξη `.py`.

2.1.1 Εγκατάσταση της Python

Προτείνω στον αναγνώστη να εγκαταστήσει την Anaconda, η οποία είναι μια δωρεάν και ανοιχτού κώδικα διανομή της Python προσφέροντας ένα μεγάλο σύνολο εργαλείων σε ένα σημείο. Η διαδικασία εγκατάστασης παραμένει εύκολη και καθιστά μελλοντικούς πειραματισμούς πιο εύκολους λόγω του πλήθους και ποικιλίας των εργαλείων που περιέχει.

Επισκεφθείτε την επίσημη σελίδα, κατεβάστε το αρχείο εγκατάστασης και ακολουθήστε τις οδηγίες εγκατάστης για το λειτουργικό σας σύστημα:

- Windows
- MacOS
- Linux

Επιβεβαίωση εγκατάστασης

Σε Windows, ανοίξτε το `Anaconda Prompt` σε Windows ή το terminal σε MacOS και Linux. Η εκτέλεση των εντολών `python --version` και `conda --version`, πρέπει να έχει ένα παρόμοιο αποτέλεσμα με `Python 3.7.6` και `conda 4.8.3`.

2.1.2 Εγκατάσταση επεξεργαστή κειμένου

Μπορούμε να γράψουμε Python σε οποιονδήποτε επεξεργαστή κειμένου, ακόμα και στο Σημειωματάριο που έρχεται με τον υπολογιστή μας. Υπάρχουν όμως άλλα προγράμματα τα οποία είναι ειδικά φτιαγμένα για προγραμματισμό, τα οποία φέρουν χαρακτηριστικά που θα κάνουν την ζωή μας πιο εύκολη. Μερικά τέτοια δωρεάν και ανοιχτού κώδικα προγράμματα είναι τα εξής: Sublime, Atom και VSCodium. Παρακάτω θα δείτε πως να εγκαταστήσετε το VSCodium.

Επισκεφθείτε την επίσημη σελίδα και ακολουθήστε τις οδηγίες εγκατάστασης για το λειτουργικό σας σύστημα.

Επιβεβαίωση εγκατάστασης

Σε οποιοδήποτε λειτουργικό σύστημα, αναζητήστε «VSCodium» στην λίστα προγραμμάτων σας. Εάν μπορείτε να το εκτελέσετε από εκεί, το έχετε εγκαταστήσει επιτυχώς. Διαφορετικά ξεκινήστε την διαδικασία εγκατάστασης από την αρχή.

2.2 Η γλώσσα

2.2.1 Διερμηνευτής

Ο διερμηνευτής της Python επιτρέπει στον χρήστη να εκτελέσει διαδραστικά εντολές Python, κάτι που μπορεί να φανεί χρήσιμο για πειραματισμό ή την δοκιμή κομματιών προγράμματος. Μπορείτε να χρησιμοποιήσετε τον διερμηνευτή,

- ανοίγοντας το Anaconda Prompt στα Windows και εκτελώντας την εντολή `python`.
- ανοίγοντας το terminal στα Linux και εκτελώντας την εντολή `python`.

Αυτός είναι ο διερμηνευτής:

```
Python 3.7.6 (default, Jan  8 2020, 19:59:22)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Για να εκτυπώσετε την φράση «hello world» στον διερμηνευτή εκτελείτε την εντολή `print("hello world")`:

```
Python 3.7.6 (default, Jan  8 2020, 19:59:22)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
```

Για να κλείσετε τον διερμηνευτή, αρκεί να εκτελέσετε `quit()`:

```
Python 3.7.6 (default, Jan  8 2020, 19:59:22)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> quit()
```

2.2.2 Εκτέλεση προγραμμάτων

Τα προγράμματα της Python εκτελούνται τις περισσότερες φορές μέσω του διερμηνευτή. Οπότε, αν έχετε το πρόγραμμα `my_first_program.py`, αρκεί να εκτελέσετε την εντολή `python my_first_program.py` στο terminal:

```
python hello_world.py
hello world
```

Το πρόγραμμα `my_first_program.py`:

```
1 print("hello world")
```

2.2.3 Εύρεση κι επίλυση σφαλμάτων

Καλώς ή κακώς, τα σφάλματα πάνε μαζί με τον προγραμματισμό. Δεν γίνεται να μην συναντήσετε κάποιο σφάλμα στο πρόγραμμά σας με αποτέλεσμα είτε το πρόγραμμά σας να μην εκτελείται ή είτε να εκτελείται αλλά να δίνει κάποια έξοδο που δεν περιμένατε. Για την εύρεση και επίλυση σφαλμάτων μπορείτε να δοκιμάσετε τα εξής:

- Όταν υπάρχει ένα σημαντικό σφάλμα σε ένα πρόγραμμα, η Python δημιουργεί ένα `traceback`, δηλαδή μια αναφορά σφαλμάτων. Η Python «ελέγχει» τον κώδικα του προγράμματος και προσπαθεί να εντοπίσει το σφάλμα. Ελέγξτε το `traceback`, μπορεί να σας δώσει μια ιδέα για το τι τυχόν πάει λάθος.
- Κάντε ένα διάλειμμα, απομακρυνθείτε για λίγο από τον υπολογιστή σας. Το συντακτικό είναι πολύ σημαντικό στον προγραμματισμό, και ένα λιγότερο ερωτηματικό, παρένθεση ή αυτάκια (“”, ‘’) θα δημιουργεί σφάλμα στο πρόγραμμά σας. Δείτε ξανά τον κώδικά σας και προσπαθήστε να εντοπίσετε το σφάλμα.
- Ξεκινήστε από την αρχή. Ενώ τις περισσότερες φορές δεν χρειάζεται να απεγκαταστήσετε κάποιο λογισμικό, μπορείτε να γράψετε από την αρχή το πρόγραμμά σας.
- Βρείτε κάποιον που γνωρίζει Python και ζητήστε του να σας βοηθήσει. Ρωτώντας τριγύρω, μπορεί και να βρείτε κάποιον που δεν περιμένατε να γνωρίζει Python.
- Ψάξτε για βοήθεια διαδικτυακά. Έχετε ήδη τον υπολογιστή μπροστά σας, πιθανότατα και σύνδεση στο διαδίκτυο, άρα δεν σας σταματάει τίποτα από το να αναζητήσετε το σφάλμα στο StackOverflow, στην Google ή στο YouTube.

2.2.4 Σχόλια

Ένα βασικό και χρήσιμο κομμάτι προγραμματισμού αποτελούν τα σχόλια. Τα σχόλια είναι μέρη του προγράμματος που δεν εκτελούνται και χρησιμοποιούνται από τους προγραμματιστές για να εξηγήσουν την υλοποίηση κάποιας συγκεκριμένης λογικής ή κάποιο κομμάτι του προγράμματος. Επιτρέπουν σε μια πιο αποτελεσματική και γρήγορη κατανόηση ενός προγράμματος ή κομματιού προγράμματος. Κάποιος που ενδιαφέρεται στο πρόγραμμά μπορεί να διαβάσει τα σχόλια που έχετε γράψει για τα κομμάτια του προγράμματός σας και δεν χρειάζεται να τα «εκτελεί» στο μυαλό του.

Υπάρχουν τα σχόλια μίας γραμμής με τα οποία ξεκινούν με `#` και τελειώνουν στο τέλος της γραμμής:

```
1 print("hello world")    # to show "hello world"
```

Ακόμα υπάρχουν τα σχόλια πολλάπλών γραμμών τα οποία χρησιμοποιούνται για να εξηγήσουν κάποια μεγαλύτερα κομμάτια κώδικα ή συναρτήσεις[1] και ξεκινάνε και τελειώνουν με `"""` ή `'''`:

```
1 """
2 This is a program that prints hello world to the terminal.
3 """
4 print("hello world")    # show hello world to the terminal
5 '''
6 This is multiline comment
7 too.
8 '''
```

[1] Σε αυτήν την περίπτωση τέτοια σχόλια ονομάζονται `docstring`

2.2.5 Μεταβλητές

Μία μεταβλητή, ή αλλιώς variable, είναι μια ετικέτα σε μία τιμή, η οποία τιμή έχει έναν από πολλούς τύπους δεδομένων, και αποθηκεύεται στην μνήμη του υπολογιστή.

Οι δηλώσεις μεταβλητών (αλλιώς οι ονομασίες τους):

- είναι case-sensitive, π.χ. `two` και `Two` είναι διαφορετικές μεταβλητές,
- πρέπει να ξεκινούν με γράμμα ή κάτω παύλα,
- μπορούν να εμπεριέχουν αριθμούς, αλλά δεν μπορούν να ξεκινάνε μα αριθμούς.

Για παράδειγμα:

```
1 two = 2
```

Παραπάνω, δίνεται στην μεταβλητή `two` ο ακέραιος αριθμός 2 κι έτσι αρκεί να καλέσετε την μεταβλητή `two` όποτε χρειάζεται να χρησιμοποιήσετε τον ακέραιο αριθμό 2 στην συνέχεια του προγράμματός σας.

```
1 one = 1
2 two = 2
3 three = one + two
4 print(three)
```

Εδώ, η μεταβλητή `one` έχει τον ακέραιο αριθμό 1, η μεταβλητή `two` έχει τον ακέραιο αριθμό 2 και η μεταβλητή `three` έχει τον ακέραιο αριθμό 3 τελικά, από την πράξη `one + two`, δηλαδή `1 + 2`.

2.2.6 Τύποι δεδομένων

Οι τύποι δεδομένων επιτρέπουν την κατηγοριοποίηση των στοιχείων δεδομένων και καθορίζουν ποιες λειτουργίες μπορούν να εκτελεστούν σε αυτά τα στοιχεία δεδομένων. Παρακάτω θα δείτε συνοπτικά τους τύπους δεδομένων και πως τους καταλαβαίνει η γλώσσα, χρησιμοποιώντας την συνάρτηση `type()`.

Αριθμητικά δεδομένα

Τα αριθμητικά δεδομένα αντιπροσωπεύουν οποιαδήποτε αναπαράσταση δεδομένων που περιέχει αριθμούς. Η Python αναγνωρίζει τους εξής τύπους αριθμητικών δεδομένων:

- Integer numbers: Ακέραιοι αριθμοί, χωρίς κλασματικό μέρος.

```
>>> type(5)     # decimal system
<class 'int'>
>>> type(0b010) # binary system
<class 'int'>
>>> type(0o642) # octal system
<class 'int'>
>>> type(0xF3)  # hexadecimal system
<class 'int'>
```

- Float numbers: Οποιοσδήποτε πραγματικός αριθμός με κλασματικό μέρος, το οποίο αναπαριστάτε είτε με δεκαδική είτε με επιστημονική σημειογραφία.

```
>>> type(0.0)       #   decimal notation
<class 'float'>
>>> type(-1.7e-6)   #   scientific notation
<class 'float'>
```

- Complex numbers: Αριθμοί με πραγματικό και μιγαδικό μέρος, που αναπαριστώνται ως `x+yj`, όπου το `x` και το `y` αποτελούν το πραγματικό μέρος του αριθμού και το `j` την φανταστική μονάδα `-1`.

```
>>> type(4+6j)
<class 'complex'>
```

Boolean δεδομένα

Δεδομένα με μία από δύο built-in τιμές, τις `True` και `False`. Παρατηρήστε ότι τα `T` και `F` είναι με κεφαλαία. Αντιπροσωπεύουν τις λογικές τιμές `1` και `0` αντίστοιχα, ως αποτέλεσμα των προτάσεων υπό συνθήκη.

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

Τύποι ακολουθίας

Μια ακολουθία είναι μία ταξινομημένη συλλογή ίδιου ή διαφορετικών τύπων δεδομένων. Η Python αναγνωρίζει τους εξής τύπους ακολουθίας:

- String: Συμβολοσειρά, είναι ένα σύνολο ενός ή περισσοτέρων χαρακτήρων, μια σειρά χαρακτήρων, ανάμεσα σε μονά ή διπλά «αυτάκια» (`' '`,`" "`).

```
>>> greetings = 'Hello there!'
>>> type(greetings)
<class 'str'>
>>> type('1 + 2 = 3')
<class 'str'>
```

- List: Λίστα, είναι ένα σύνολο ενός ή περισσοτέρων δεδομένων ενός ή διαφόρων τύπων, ανάμεσα σε αγγύλες (`[ ]`).

```
>>> some_list = [0+1j, 2, 3.13, 4, 'numbers']
>>> type(some_list)
<class 'list'>
```

- Tuple: Πλειάδα, είναι ένα σύνολο ενός ή περισσοτέρων δεδομένων ενός ή διαφόρων τύπων, ανάμεσα σε παρενθέσεις (`( )`).

```
>>> some_tuple = (0+1j, 2, 3.13, 4, 'numbers')
>>> type(some_tuple)
<class 'tuple'>
```

Για να δείτε το μέγεθος μιας ακολουθίας καλείτε την συνάρτηση `len( )` και της δίνετε όρισμα την ακολουθία που θέλετε. Θα δείτε παραδείγματα για την κάθε ακολουθία στην συνέχεια του εγχειριδίου.

2.2.7 Συμβολοσειρές

Οι συμβολοσειρές είναι ένα ταξινομημένο σύνολο χαρακτήρων και είναι ο τύπως των λέξεων και προτάσεων σε ένα πρόγραμμα.

Μορφοποίηση

```
1 name = 'Mike'
2 age = 20
3
4 #   print(name)
5 #   print(age)
6
7 print('Hello, what\'s your name?')
8 print('How old are you?')
9
10 greetings_reply = f'Hi, my name is {name}!'
11 age_reply = f'I am {age} years old.'
12 reply = f'{greetings_reply} {age_reply}'
13 print(reply)
```

Στις γραμμές 1 και 2 δηλώνονται οι μεταβλητές `name` και `age` που έχουν τιμές `'Mike'` και `20`, αντίστοιχα. Μπορείτε να εμφανίσετε στο terminal αυτές τις τιμές εκτελώντας τις εντολές που είναι σχολιασμένες στις γραμμές 4 και 5. Με τις γραμμές 7 και 8 εμφανίζονται τα μηνύματα `'Hello, what\'s your name?'` και `'How old are you?'`. Παρατηρήστε την κάθετο πριν το δεύτερο `'`. Αυτό είναι για αποτραπεί το τέλος της συμβολοσειράς σε εκείνο το σημείο, για να μην δει η γλώσσα εκείνο το σημείο ως το τέλος του string. Έπειτα, παρατηρήστε στις γραμμές 10, 11, 12. Όπως βλέπετε αυτά τα strings έχουν ένα `f` αμέσως πριν ξεκινήσουν και εμπεριέχουν μέσα σε άγκιστρα μεταβλητές που έχουν ήδη δηλωθεί. Τα συγκεκριμένα strings ονομάζονται f-strings και επιτρέπουν στις κανονικές συμβολοσειρές να γίνουν «διαδραστικές».

Παρακάτω βλέπετε την έξοδο του προγράμματος, εκτελώντας την εντολή `python strigns.py` στο terminal.

```
1 python strigns.py
2 Hello, what's your name?
3 How old are you?
4 Hi, my name is Mike! I am 20 years old.
```

Μέθοδοι

Όλοι οι τύποι δεδομένων που είδατε παραπάνω είναι κλάσεις, και όλες οι κλάσεις έχουν μεθόδους. Μέθοδοι είναι συναρτήσεις για κλάσεις και χρησιμοποιώντας τις κατάλληλες μεθόδους μπορείτε να «επεξεργαστείτε» συμβολοσειρές. Παρακάτω θα δείτε μερικές από τις μεθόδους της κλάσης συμβολοσειρών και τι κάνουν.

```
1 name = 'helen'
2 print(name)
3
4 #   make all characters uppercase
5 name = name.upper()
6 print(name)
7
8 #   make all characters lowercase
9 name = name.lower()
```

```
10 print(name)
11
12 #   make first character uppercase and the rest lowercase
13 name = name.capitalize()
14 print(name)
15
16 #   count the times there's the character l in name
17 l_count = name.count('l')
18 print(f'Number of "l"s: {l_count}')
19
20 #   replace l with ll
21 name = name.replace('l', 'll')
22 print(name)
23
24 #   count the times there's the character l in name
25 l_count = name.count('l')
26 print(f'Number of "l"s: {l_count}')
27
28 #   find index of the first e in name
29 e_index = name.find('e')
30 print(f'Index of "e": {e_index}')
31
32 #   check if name contains just characters
33 print(f'"{name}" contains just characters: {name.isalpha()}')
34
35 #   check if name contains characters or numbers
36 print(f'"{name}" contains characters or numbers: {name.isalnum()}')
37
38 #   check if name contains just numbers
39 print(f'"{name}" contains characters or numbers: {name.isnumeric()}')
40
41 #   print length of name
42 print(f'len({name}): {len(name)}')
```

Η έξοδος του παραπάνω προγράμματος:

```
1 helen
2 HELEN
3 helen
4 Helen
5 Number of "l"s: 1
6 Hellen
7 Number of "l"s: 2
8 Index of "e": 1
9 "Hellen" contains just characters: True
10 "Hellen" contains characters or numbers: True
11 "Hellen" contains characters or numbers: False
12 len(Hellen): 6
```

2.2.8 Λίστες

Οι λίστες είναι μια ταξινομημένη συλλογή δεδομένων ίδιου ή διαφορετικών τύπων και δηλώνονται με τα δεδομένα που περιέχουν γύρω από αγγύλες. Παρακάτω μπορείτε να δείτε μερικές μεθόδους τους από το πρόγραμμα `lists.py`.

Για παράδειγμα:

```
veggies = ['carrot', 'onion', 'lettuce']

#   print list
print(f'veggies: {veggies}')

#   reverse list and print list
veggies.reverse()
print(f'veggies reversed: {veggies}')

#   sort list alphabetically and print list
veggies.sort()
print(f'veggies sorted: {veggies}')

#   reverse sort list alphabetically and print list
veggies.sort(reverse=True)
print(f'veggies reverese sorted: {veggies}')

#   print the position of the first item
first_item_index = veggies.index('carrot')
print(f'first_item_index: {first_item_index}')

#   print first item
print(f'veggies[first_item_index]: {veggies[first_item_index]}')

#   append to list and print list
veggies.append('broccoli')
print(f'added broccoli to the end of veggies: {veggies}')

#   remove last item and print list
veggies.pop(-1)
print(f'removed last veggie: {veggies}')

#   insert item into position and print list
veggies.insert(first_item_index, 'potato')
print(f'inserted potato to the start of veggies: {veggies}')

#   remove onion and print list
veggies.remove('onion')
print(f'removed onion from veggies: {veggies}')

#   print length of veggies
print(f'len(veggies): {len(veggies)}')
```

Η έξοδος του παραπάνω προγράμματος:

```
veggies: ['carrot', 'onion', 'lettuce']
veggies reversed: ['lettuce', 'onion', 'carrot']
veggies sorted: ['carrot', 'lettuce', 'onion']
veggies reverese sorted: ['onion', 'lettuce', 'carrot']
first_item_index: 2
veggies[first_item_index]: carrot
added broccoli to the end of veggies: ['onion', 'lettuce', 'carrot', 'broccoli']
removed last veggie: ['onion', 'lettuce', 'carrot']
inserted potato to the start of veggies: ['onion', 'lettuce', 'potato', 'carrot
```

```
    ']
removed onion from veggies: ['lettuce', 'potato', 'carrot']
len(veggies): 3
```

2.2.9 Πλειάδες

Μία πλειάδα είναι ένα ταξινομημένο σύνολο δεδομένων ίδιου ή διαφορετικών τύπων και δηλώνεται με τα στοιχεία που περιέχει ανάμεσα σε παρενθέσεις (( )). Μοιάζει με την λίστα, αλλά η σημαντικότερη και βασικότερη διαφορά μεταξύ αυτών των δύο τύπων είναι ότι η πλειάδα δεν επιτρέπει να γίνουν αλλαγές εφόσον δηλωθεί. Έαν κάπου στο πρόγραμμα πάει να γίνει κάποια αλλαγή σε κάποια πλειάδα, τότε η εκτέλεση του προγράμματος θα σταματήσει και θα εμφανιστεί σφάλμα.

Για παράδειγμα:

```
veggies = ('carrot', 'onion', 'lettuce')

#   print tuple
print(f'veggies: {veggies}')

#   print the position of the first item
first_item_index = veggies.index('carrot')
print(f'first_item_index: {first_item_index}')

#   print first item
print(f'veggies[first_item_index]: {veggies[first_item_index]}')

#   print length of veggies
print(f'len(veggies): {len(veggies)}')
```

Η έξοδος του παραπάνω προγράμματος:

```
veggies: ('carrot', 'onion', 'lettuce')
first_item_index: 0
veggies[first_item_index]: carrot
len(veggies): 3
```

2.2.10 Λεξικά

Ένα λεξικό είναι ένα μη ταξινομημένο σύνολο δεδομένων ίδιου ή διαφορετικών τύπων σε ζευγάρια μορφής "key": "item".

Για παράδειγμα:

```
person = {
    'first_name': 'Mike',
    'last_name': 'Zamayias',
    'age': 20
}

#   print person
print(f'person: {person}')

#   print person type
print(f'type(person): {type(person)}')

#   print person's first name
print(f"person['first_name']: {person['first_name']}")

#   print person's last name
print(f"person['last_name']: {person['last_name']}")

#   print person's age
print(f"person['age']: {person['age']}")

#   add key:item pair to person
person['id'] = 'TP5000'

#   print person
print(f'person: {person}')

#   remove key:item pair from person
del person['age']
person.pop('id')

#   print person
print(f'person: {person}')

#   print person's keys
print(f'person.keys(): {person.keys()}')

#   print person's items
print(f'person.items(): {person.items()}')

#   copy person to another_person and print another_person
another_person = person.copy()
print(f"another_person: {another_person}")

#   clear person and print person
person.clear()
print(f"person: {person}")

#   print length of person and another_person
```

```
50 print(f"len(person): {len(person)}")
51 print(f"len(another_person): {len(another_person)}")
```

Η έξοδος του παραπάνω προγράμματος:

```
1 type(person): <class 'dict'>
2 person['first_name']: Mike
3 person['last_name']: Zamayias
4 person['age']: 20
5 person: {'first_name': 'Mike', 'last_name': 'Zamayias', 'age': 20, 'id': 'TP5000
    '}
6 person: {'first_name': 'Mike', 'last_name': 'Zamayias'}
7 person.keys(): dict_keys(['first_name', 'last_name'])
8 person.items(): dict_items([('first_name', 'Mike'), ('last_name', 'Zamayias')])
9 another_person: {'first_name': 'Mike', 'last_name': 'Zamayias'}
10 person: {}
11 len(person): 0
12 len(another_person): 2
```

2.2.11 Συναρτήσεις

Οι συναρτήσεις είναι κομμάτια κώδικα που εκτελούνται όταν καλεστούν. Κάποιες συναρτήσεις χρειάζονται ορίσματα για να εκτελεστούν ορθά, ενώ άλλες δεν χρειάζονται. Για παράδειγμα η συνάρτηση `len()` χρειάζεται ένα όρισμα ενώ οι μέθοδοι `keys()` και `items()` των λεξικών δεν χρειάζονται ορίσματα. Οι μέθοδοι `pop` και `sort` των λιστών μπορούν να εκτελεστούν χωρίς κάποιο όρισμα καθώς έχουν προκαθορισμένες τιμές για τα ορίσματα. Παρακάτω θα δείτε πως δηλώνονται κι εκτελούνται οι συναρτήσεις.

```
1  def function_zero():
2      """
3      A function that prints 'Hello World!' to the terminal.
4      """
5      print('Hello World!')
6
7
8  def function_one(name='user'):
9      """
10     A function that greets the user. It has a default value for the name set to
       'user', in the case that the suer forgets to enter his'/hers' name.
11     """
12     print(f'Hello {name} from function_one!')
13
14
15 def function_two(name='user'):
16     """
17     A function that returns a greeting with the name of the user.
18     """
19     return f'Hello {name} from function_two!'
20
21
22 #   call function_zero
23 function_zero()
24
25 #   call function_one without an arguement
26 function_one()
27
28 #   call function_one with an arguement
29 function_one('Mike')
30
31 #   call function_two that returns a string with an arguement
32 function_two('Helen')
33
34 #   call function_two that returns a string without an arguement
35 string = function_two()
36 print(string)
```

Η έξοδος του παραπάνω προγράμματος:

```
1 Hello World!
2 Hello user from function_one!
3 Hello Mike from function_one!
4 Hello user from function_two!
```

Παρατηρήστε ότι δεν εμφανίζετε κάποια έξοδος με την εκτέλεση της `function_two('Helen')`. Αυτό συμβαίνει επειδή η συνάρτηση επιστρέφει μια τιμή, εδώ το string `'Hello Helen from function_two!'`,

αλλά δεν εκτυπώνει κάτι.

2.2.12 Προτάσεις υπό συνθήκη

Ένα πρόγραμμα, σε οποιαδήποτε γλώσσα προγραμματισμού εκτελείται σειριακά, δηλαδή γραμμή γραμμή. Όμως κάτι τέτοιο δεν είναι χρήσιμο εκείνες τις φορές που πρέπει να ικανοποιείται κάποια ή κάποιες συνθήκες πρώτα, όπως για παράδειγμα η αυτόματη εκτέλεση ενός προγράμματος σε μία συγκεκριμένη ώρα. Αυτό επιτυγχάνεται χρησιμοποιώντας τις εντολές `if`, `elif`, `else`:

```
1  person = {
2      'first_name': 'Mike',
3      'last_name': 'Zamayias',
4      'age': 20,
5      'id': 'TP5000'
6  }
7
8  if person['age'] >= 18:
9      print(f"{person['first_name']} is an adult.")
10 else:
11     print(f"{person['first_name']} is a child.")
12
13
14 if 'TP' in person['id']:
15     print(f"{person['first_name']} studies informatics science.")
16 else:
17     print(f"What does {person['first_name']} study?")
18
19 x = y = 10
20
21 if x > y:
22     print(f'{x} is greater than {y}')
23 elif x == y:
24     print(f'{x} is equal to {y}')
25 else:
26     print(f'{y} is greater than {x}')
```

Σε αυτό το πρόγραμμα φτιάχνεται το αντικείμενο `person` κλάσης `dict` στις γραμμές 1 ως 6. Στις γραμμές 8 ως 11 ελέγχεται αν το κλειδί `age` του αντικειμένου lstinline[language=]person είναι μεγαλύτερο από 18 και εκτυπώνει το αντίστοιχο μήνυμα, δηλαδή αν ισχύει `20 > 18` εκτυπώνεται το μήνυμα `'Mike is an adult.'`, αλλίως εκτυπώνεται `'Mike is a child.'`. Στις γραμμές 14 ως 17 ελέγχεται αν υπάρχει η συμβολοσειρά `'TP'` στην τιμή του κλειδιού `'id'` του αντικειμένου `person`. Αν ισχύει αυτή πρόταση τότε εκτυπώνεται το μήνυμα `'Mike studies informatics science.'`, αλλιώς εκτυπώνεται `'What does Mike study?'`. Έπειτα, στην γραμμή 19 δηλώνονται οι μεταβλητές `x` καιy με τιμή `10` και γίνεται ο εξής έλεγχος:

- αν ικανοποιείται η συνθήκη `x > y` εκτελείται η εντολή στην γραμμή 22
- αλλίως αν ικανοποιείται η συνθήκη `x == y` εκτελείται η εντολή στην γραμμή 24
- αλλιώς εκτελείται η εντολή στην γραμμή 26

Η έξοδος του παραπάνω προγράμματος:

```
1 Mike is an adult.
2 Mike studies informatics science.
3 10 is equal to 10
```

2.2.13 Βρόχοι

Οι βρόχοι χρησιμοποιούνται για την επανάληψη κάποιου κομμάτι προγράμματος, όπως κάποια πράξης σε μία λίστα ακεραίων ή την τροποίηση μιας λίστας συμβολοσειρών.

```
1  numbers = list(range(0, 10))
2
3  #   print numbers
4  print(f'numbers: {numbers}')
5
6  evens, odds = [], []
7
8  #   iterate numbers list using the for loop
9  for number in numbers:
10     #   if number is even
11     if number % 2 == 0:
12         #   append number to evens
13         evens.append(number)
14     #   if number is not even
15     else:
16         #   append number in odds
17         odds.append(number)
18
19 #   print evens
20 print(f'evens: {evens}')
21
22 #   print odds
23 print(f'odds: {odds}')
24
25 #   print numbers in numbers using using the while loop
26 index = 0
27 length = len(numbers)
28 while True:
29     check = index < length
30     print(f'--- index: {index}')
31     print(f'--- length: {length}')
32     print(f'--- index < length: {check}')
33     if check == False:
34         break
35     else:
36         number = numbers[index]
37         print(f'number: {number}')
38     index += 1
```

Στην γραμμή 1 δημιουργείται μια λίστα με ακέραιους αριθμούς από το 0 έως το 9 και στην γραμμή 4 εκτυπώνεται η λίστα numbers. Μπορείτε να διαβάσετε την γραμμή 9 ως «για κάθε αριθμό στους αριθμούς κάνε:», με «αριθμό» να αποτελεί το αντικείμενο "number" και «αριθμούς» το αντικείμενο "numbers".

Η έξοδος του παραπάνω προγράμματος:

```
1 numbers: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
2 evens: [0, 2, 4, 6, 8]
3 odds: [1, 3, 5, 7, 9]
4 --- index: 0
5 --- length: 10
6 --- index < length: True
```

```
number: 0
--- index: 1
--- length: 10
--- index < length: True
number: 1
--- index: 2
--- length: 10
--- index < length: True
number: 2
--- index: 3
--- length: 10
--- index < length: True
number: 3
--- index: 4
--- length: 10
--- index < length: True
number: 4
--- index: 5
--- length: 10
--- index < length: True
number: 5
--- index: 6
--- length: 10
--- index < length: True
number: 6
--- index: 7
--- length: 10
--- index < length: True
number: 7
--- index: 8
--- length: 10
--- index < length: True
number: 8
--- index: 9
--- length: 10
--- index < length: True
number: 9
--- index: 10
--- length: 10
--- index < length: False
```

2.2.14 Δομοστοιχεία

2.2.15 Δουλεύοντας με αρχεία

2.2.16 Δουλεύοντας με JSON αρχεία

Βιβλιογραφία

> Total 0 knowledge, entirely parallel with programming of any kind. Heard Python is simple, why would I want to learn it ?
>
> u/Azsras_Zuralix on r/learnpython

3.1 Βιβλία

- Python Crash Course, 2nd Edition — by Eric Matthes

3.2 Video

- Python Crash Course

3.3 Σύνδεσμοι

- WordReference Dictionary
- Λεξικό της κοινής νεοελληνικής
- Βιβλιογραφία, Wikipedia
- Python, Wikipedia
- Anaconda (Python distribution), Wikipedia
- Anaconda Individiual Edition, Anaconda | The World's Most Popular Data Science Platform
- Conditional statements, Wikipedia
- Python Cheatsheet
- VSCodium is a community-driven, freely-licensed binary distribution of Microsoft's editor VSCode
- Python data types
- Quotes on programming

3.4 Χρήσιμα αρχεία

- Βιβλιογραφική ανασκόπηση, Δημοκρίτειο Πανεπιστήμιο Θράκης
- Εισαγωγή στη LaTeX για φοιτητές. (An Introduction to Latex in Greek)
- Python Cheat Sheet