

Προγραμματισμός, σε Python

Μιχαήλ Ανάργυρος Ζαμάγιας – ΤΠ5000

5 Ιουνίου 2020

Περιεχόμενα

1	Εισαγωγή	2
1.1	Πρόλογος	3
1.2	Γιατί να ασχοληθεί κάποιος με τον προγραμματισμό;	3
1.3	Γιατί να μάθει κάποιος Python;	3
1.4	Ποιος είναι ο σκοπός αυτού του εγχειριδίου;	3
2	Τα βασικά	4
2.1	Ξεκινώντας	5
2.1.1	Εγκατάσταση της Python	5
2.1.2	Εγκατάσταση επεξεργαστή κειμένου	5
2.2	Η γλώσσα	6
2.2.1	Διερμηνευτής	6
2.2.2	Εκτέλεση προγραμμάτων	6
2.2.3	Εύρεση κι επίλυση σφαλμάτων	7
2.2.4	Σχόλια	7
2.2.5	Μεταβλητές	8
2.2.6	Τύποι δεδομένων	8
2.2.7	Συμβολοσειρές	10
2.2.8	Λίστες	12
2.2.9	Πλειάδες	12
2.2.10	Σύνολα	12
2.2.11	Λεξικά	12
2.2.12	Προτάσεις υπό συνθήκη	12
2.2.13	Βρόχοι	12
2.2.14	Δομοστοιχεία	12
2.2.15	Δουλεύοντας με αρχεία	12
2.2.16	Δουλεύοντας με JSON αρχεία	12
3	Βιβλιογραφία	13
3.1	Βιβλία	14
3.2	Video	14
3.3	Σύνδεσμοι	14
3.4	Χρήσιμα αρχεία	14

Εισαγωγή

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 αρκετά ενδιαφέροντα προγράμματα, όπως μια μικρή μηχανή αναζήτησης, ένα πογραμμα με ρόλο μετεωρολόγου 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.

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. Γιατί να επαναλαμβάνετε κάποια ξανά και ξανά ενώ μπορείτε να την αυτοματοποιήσετε; Η γιατί να εγκαθιστάτε λογισμικό από τρίτους στο μηχανήμά σας ενώ πιθανότατα μπορείτε να φτιάξετε μόνοι σας αυτό το εργαλείο;

Codes are a puzzle. A game, just like any other game.

— Alan 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 σκοπός αυτού του εγχειριδίου;

Να εξοικειωθεί ο αναγνώστης με τα βασικά στοιχεία της γλώσσας και η εξοικείω του με τον τρόπο σκέψης ενός προγραμματιστή.

Τα βασικά

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 δημιουργία εργαλείων που θα του γλιτώσει αρκετό χρόνο από συγκεκριμένες ρουτίνες στον υπολογιστή.

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 εγκατάστασης 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 Διεργηνηνυτής

Ο διεργηνηνυτής 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 διεργηνηνυτή,

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

Αυτός είναι ο διεργηνηνυτής:

```
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 διεργηνηνυτή 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 διεργηνηνυτή, 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 διεργηνηνυτή. 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 εκτελείτε ή είτε να εκτελείτε 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 πολλαπλών γραμμών τα οποία χρησιμοποιούνται για να εξηγήσουν κάποια μεγαλύτερα κομμάτια κώδικα ή συναρτήσεις¹ 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 '''
```

¹Σε 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 με 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 περισσότερων 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 περισσότερων δεδομένων ενός ή διαφόρων τύπων, ανάμεσα σε αγκύλες ([]).

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

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

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

2.2.7 Συμβολοσειρές

Όπως είδατε πιο πριν, οι συμβολοσειρές επιτρέπουν να χρησιμοποιούνται λέξεις και κείμενο σαν δεδομένα, και είναι ένας από τους τρόπους επικοινωνίας με τον χρήστη. Στο παρακάτω πρόγραμμα, `string_format.py` θα δείτε πως γίνεται κάτι τέτοιο με περισσότερη λεπτομέρεια.

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.

```
python strigns.py
Hello, what's your name?
How old are you?
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 μεθόδους 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()}')
```

Η έξοδος του παραπάνω προγράμματος:

```
helen
HELEN
helen
Helen
Number of "l"s: 1
Hellen
Number of "l"s: 2
Index of "e": 1
Hellen contains just characters: True
Hellen contains characters or numbers: True
Hellen contains characters or numbers: False
```

2.2.8 Λίστες

2.2.9 Πλειάδες

2.2.10 Σύνολα

2.2.11 Λεξικά

2.2.12 Προτάσεις υπό συνθήκη

2.2.13 Βρόχοι

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 Individual 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](#)

3.4 Χρήσιμα αρχεία

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