ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Ενότητα 1 – Εισαγωγικές έννοιες στη Python

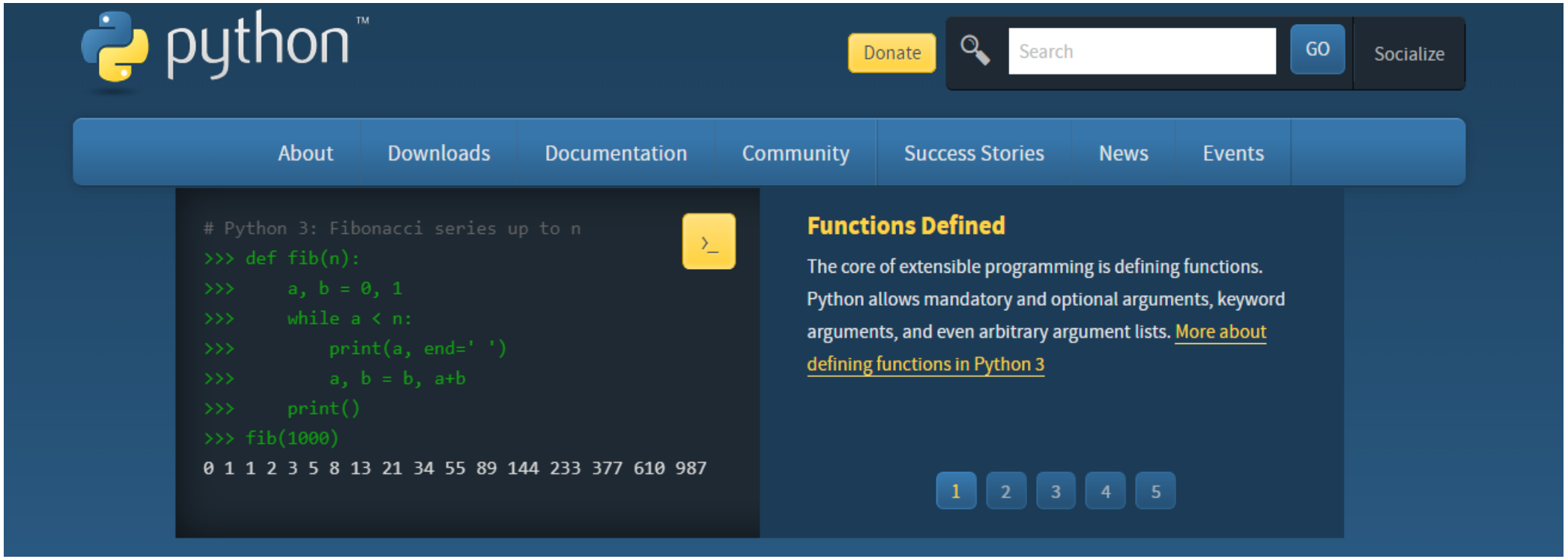
Μέθοδοι Μηχανικής Μάθησης στα Χρηματοοικονομικά

Αθανάσιος Σάκκας, Επ. Καθηγητής, ΟΠΑ

I. Εγκατάσταση της Python σε περιβάλλον Windows

1. Επισκεφτείτε την επίσημη ιστοσελίδα της Python στη διεύθυνση https://www.python.org/

2. Κατεβάστε την Python που αντιστοιχεί στο λειτουργικό σας σύστημα (πατήστε το κουμπί Downloads και θα σας εμφανίσει αυτόματα την Python που προτείνεται προς εγκατάσταση στον υπολογιστήσας).

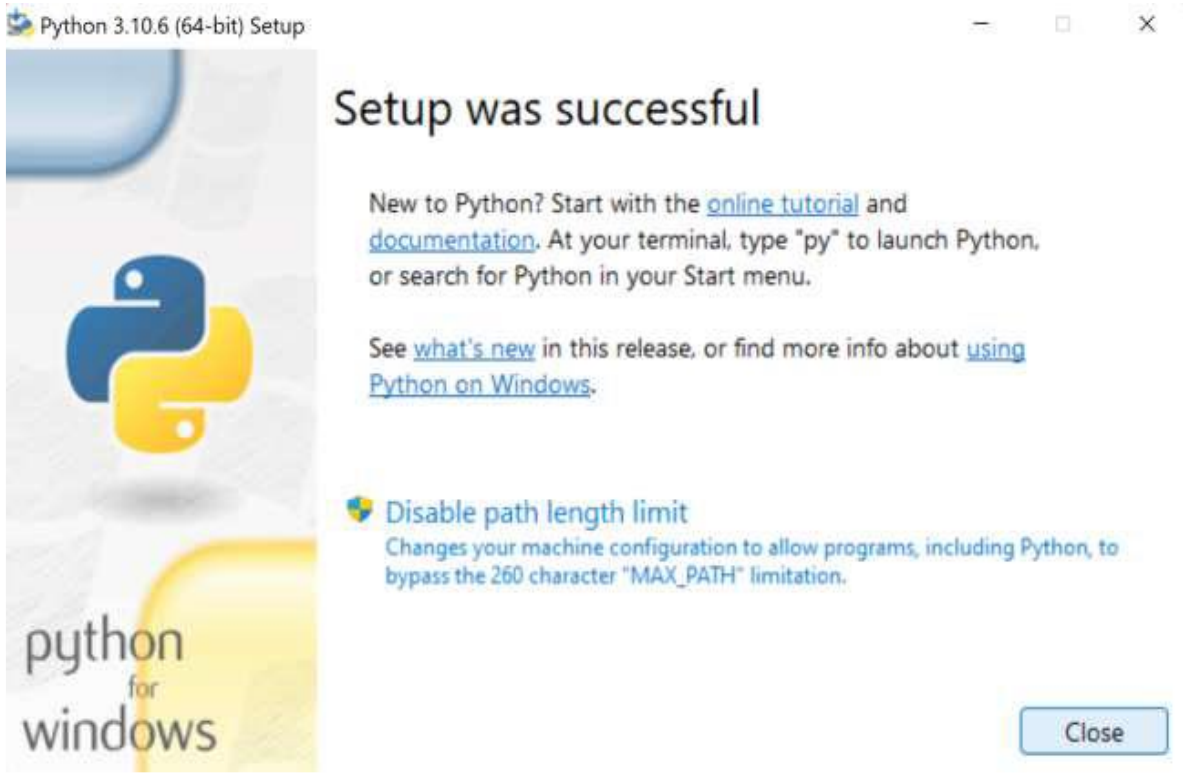


3. Μετά την επιλογή, ένα εκτελέσιμο αρχείο με προέκταση.exe μεταφορτώνεται στον υπολογιστή σας και τοποθετείται στον προκαθορισμένο φάκελο μεταφόρτωσης (συνήθως Downloads).

4. Ανοίξτε (Open) το εκτελέσιμο αρχείο που μεταφορτώθηκε στο προηγούμενο βήμα,είτε κάνοντας διπλό κλικ στο αρχείο ή επιλέγοντας το Open στο διπλανό μενού.

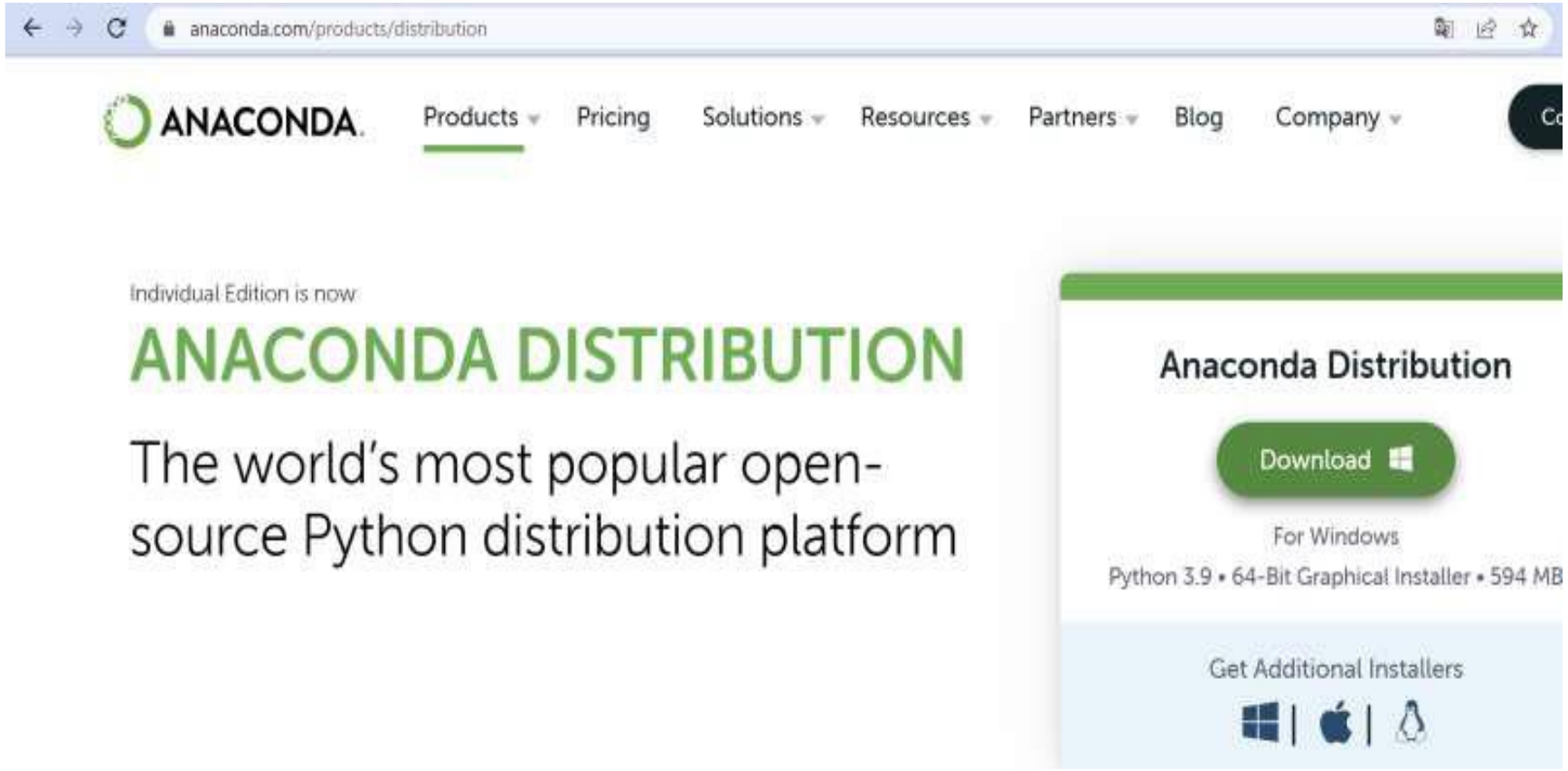
5. Στο παράθυρο εγκατάστασης που εμφανίζεται επιλέξτε Add Python…to Path και στη συνέχεια Install now.

6. Στο τέλος θα πρέπει να εμφανιστεί μήνυμα που σας πληροφορεί για την επιτυχή ολοκλήρωση της εγκατάστασης, όπως στην εικόνα:



7. Επιλέξτε “Close” για να κλείσει το παράθυρο. Η Python εγκαταστάθηκε επιτυχώς στον υπολογιστή σας!

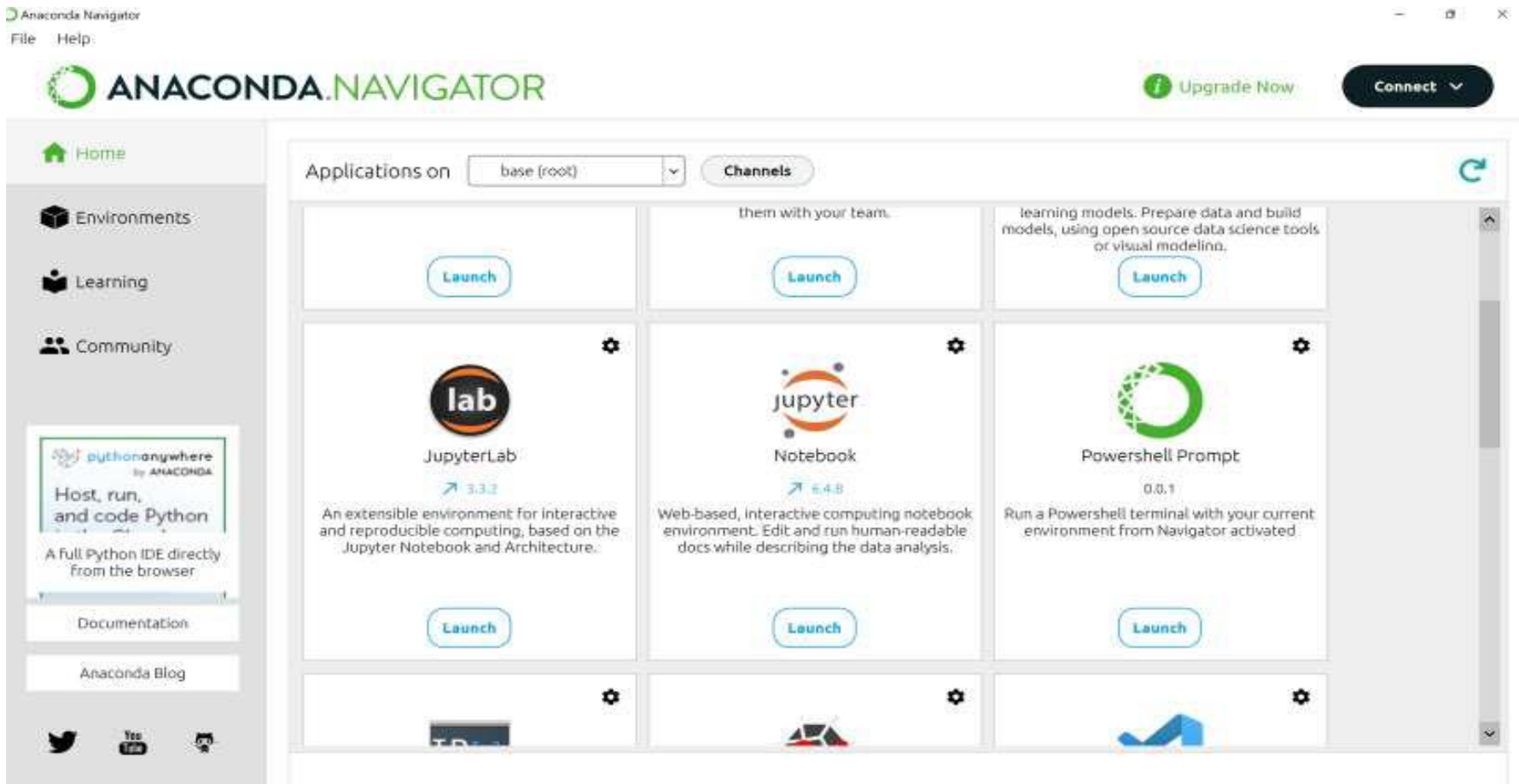
8 .Κατεβάστε την έκδοση Anaconda που αντιστοιχεί στο λειτουργικό σας σύστημα https://www.anaconda.com/



9. Προχωρήστε την εγκατάσταση με βάση τις προτεινόμενες (default) επιλογές.

10. Ανοίξτε το Anaconda Navigator.

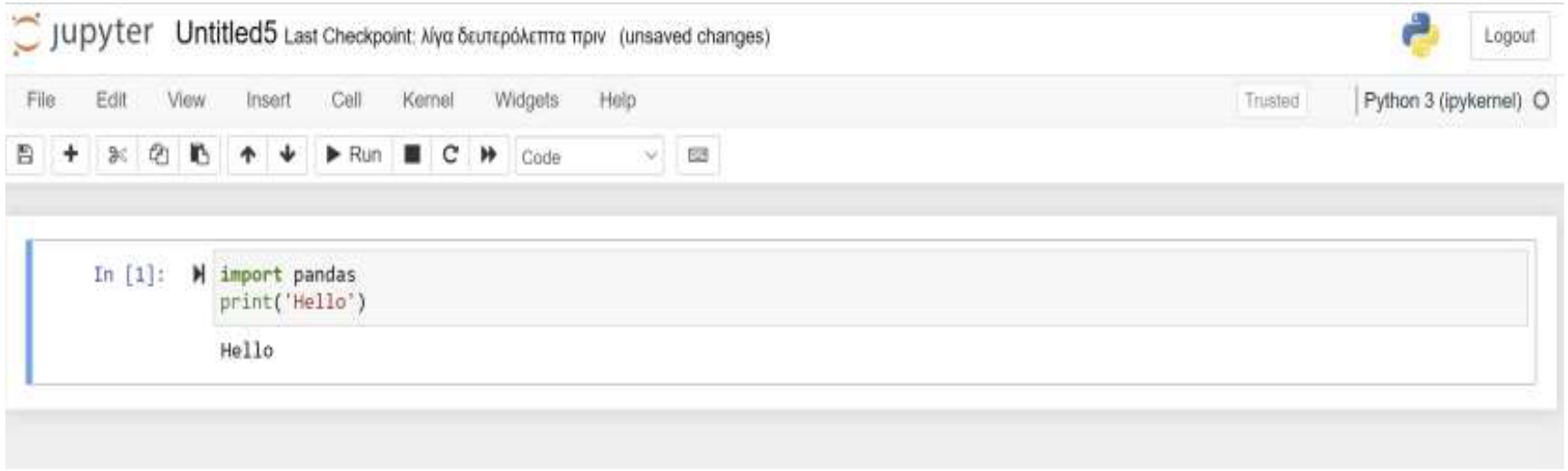
11. Από το Tab “Home” κάνετε launch το Jupyter Notebook.



12. Πατήστε “New” (πάνω δεξιά) και μετά Python3.

13. Σε ένα κελί του Jupyter Notebook κάντε copy-paste τον παρακάτω κώδικα και πατήστε Ctrl+Enter:

```
import pandas
print("Hello")
```



14. Αν από κάτω εμφανιστεί η λέξη Hello τότε όλα πήγαν καλά!

II. Εισαγωγή στη Python

- Η Python είναι μια ερμηνευμένη, υψηλού επιπέδου, γενικής χρήσης γλώσσα προγραμματισμού. Δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε για πρώτη φορά το 1991.
- Η Python έχει γίνει μια κοινή γλώσσα για την έρευνα μηχανικής μάθησης.
- Η Python 3.0, που κυκλοφόρησε το 2008, ήταν μια σημαντική αναθεώρηση της γλώσσας. Αυτό το μάθημα χρησιμοποιεί την Python 3.
- Θα ξεκινήσουμε με την εκτύπωση του Hello World

- Ο παραπάνω κώδικας μεταβιβάζει μια σταθερή συμβολοσειρά (string), που περιέχει το κείμενο "hello world" σε μια συνάρτηση που ονομάζεται print.
- Μπορείτε επίσης να αφήσετε σχόλια στον κώδικά σας για να εξηγήσετε τι κάνετε. Τα σχόλια μπορούν να ξεκινήσουν οπουδήποτε στη γραμμή με το σύμβολο #.
- Η σταθερή συμβολοσειρά, που περικλείεται σε εισαγωγικά, ορίζει τις κυριολεκτικές (literal) τιμές της συμβολοσειράς μέσα στο πρόγραμμά σας.

- Το triple quote επιτρέπει πολλές γραμμές κειμένου.

```
In [2]: # Το τριπλό απόσπασμα (triple quote) επιτρέπει πολλαπλές γραμμές κειμένου.
        print("""Print
        Multiple
        Lines
        """)

        Print
        Multiple
        Lines
```

- Η Python επιτρέπει τους αριθμούς ως κυριολεκτικές σταθερές (literal constants) σε προγράμματα. Η Python περιλαμβάνει υποστήριξη για αριθμούς κινητής υποδιαστολής (floating-point), ακέραιους (integer), μιγαδικούς (complex) και άλλους τύπους. Αυτό το μάθημα δεν θα κάνει χρήση μιγαδικών αριθμών. Σε αντίθεση με τις συμβολοσειρές, τα εισαγωγικά δεν περικλείουν αριθμούς.
- Η παρουσία υποδιαστολής διαφοροποιεί τους αριθμούς κινητής υποδιαστολής και ακέραιους αριθμούς. Για παράδειγμα, η τιμή 10 είναι ένας ακέραιος αριθμός, ενώ το 10.3 είναι ένας αριθμός κινητής υποδιαστολής. Για παράδειγμα, ο παρακάτω κώδικας εκτυπώνει δύο αριθμούς.

```
In [3]: print(10)
        print(10.3)

        10
        10.3
```

- Μέχρι στιγμής, έχουμε δει μόνο πώς να ορίσουμε κυριολεκτικές τιμές (literal values) αριθμών και συμβολοσειρών. Αυτές οι κυριολεκτικές τιμές είναι σταθερές και δεν αλλάζουν καθώς εκτελείται το πρόγραμμά σας.
- Οι μεταβλητές επιτρέπουν στο πρόγραμμά σας να διατηρεί τιμές που μπορούν να αλλάξουν καθώς εκτελείται το πρόγραμμα. Οι μεταβλητές έχουν ονόματα που σας επιτρέπουν να αναφέρετε τις τιμές τους. Ο παρακάτω κώδικας εκχωρεί μια ακέραια τιμή σε μια μεταβλητή με όνομα "a" και μια τιμή συμβολοσειράς σε μια μεταβλητή με όνομα "b".

```
In [4]: a = 50
        b = "πενήντα"
        print(a)
        print(b)

        50
        πενήντα
```

Το βασικό χαρακτηριστικό των μεταβλητών είναι ότι μπορούν να αλλάξουν. Ο παρακάτω κώδικας δείχνει πώς να αλλάξετε τις τιμές που διατηρούνται από τις μεταβλητές.

```
In [5]: a = 50
        print(a)
        a = a + 1
        print(a)

        50
        51
```

- Μπορείτε να αναμίξετε συμβολοσειρές και μεταβλητές για εκτύπωση. Αυτή η τεχνική ονομάζεται μορφοποιημένη ή παρεμβαλλόμενη συμβολοσειρά. Οι μεταβλητές πρέπει να βρίσκονται μέσα στα {}. Στην Python, αυτός ο τύπος συμβολοσειράς ονομάζεται γενικά f-string. Η συμβολοσειρά f συμβολίζεται με την τοποθέτηση ενός "f". Ο παρακάτω κώδικας δείχνει τη χρήση μιας συμβολοσειράς f για την ανάμειξη πολλών μεταβλητών με μια κυριολεκτική συμβολοσειρά.

```
In [6]: a = 50
        print(f'The value of a is {a}')

        The value of a is 50
```

- Μπορείτε επίσης να χρησιμοποιήσετε συμβολοσειρές f με μαθηματικά (που ονομάζεται έκφραση). Τα {} μπορούν να περικλείουν οποιαδήποτε έγκυρη έκφραση Python για εκτύπωση. Ο ακόλουθος κώδικας δείχνει τη χρήση μιας έκφρασης μέσα στα {} μιας συμβολοσειράς f.

```
In [7]: a = 50
        print(f'The value of a plus 5 is {a+5}')

        The value of a plus 5 is 55
```

```
In [8]: print(f'a is {a}')

        a is 50
```

- Μπορείτε να χρησιμοποιήσετε τις δηλώσεις if για να εκτελέσετε τη λογική. Προσέξτε τις εσοχές. Αυτές οι δηλώσεις if είναι ο τρόπος με τον οποίο η Python ορίζει μπλοκ κώδικα για να εκτελεστούν μαζί.
- Ένα μπλοκ αρχίζει συνήθως μετά από : και περιλαμβάνει οποιεσδήποτε γραμμές στο ίδιο επίπεδο εσοχής. Σε αντίθεση με πολλές άλλες γλώσσες προγραμματισμού, η Python χρησιμοποιεί κενό χώρο για να ορίσει μπλοκ κώδικα.

```
In [9]: a = 22
        if a>22:
            print('The variable a is greater than 22.')
        else:
            print('The variable a is not greater than 22')

The variable a is not greater than 22
```

- Είναι επίσης σημαντικό να σημειωθεί ότι ο τελεστής διπλό ίσο ("==") χρησιμοποιείται για τον έλεγχο της ισότητας δύο παραστάσεων. Το απλό ίσο ("=") χρησιμοποιείται μόνο για την εκχώρηση τιμών σε μεταβλητές στην Python. Το μεγαλύτερο από (">"), μικρότερο από ("<"), μεγαλύτερο από ή ίσο (">="), μικρότερο ή ίσο ("<=") όλα λειτουργούν όπως θα ήταν γενικά αποδεκτό. Ο έλεγχος για ανισότητα εκτελείται με τον τελεστή όχι ίσο ("!=").

```
In [10]: a = 22
         if a>=22:
             print('The variable a is greater or equal to 22.')
         else:
             print('The variable a is not greater or equal 22')

The variable a is greater or equal to 22.
```

- Είναι σύνηθες στις γλώσσες προγραμματισμού να κάνουν βρόχο σε μια σειρά αριθμών. Η Python το επιτυγχάνει αυτό με τη χρήση της λειτουργίας range. Εδώ μπορείτε να δείτε έναν βρόχο for και μια λειτουργία εύρους που κάνει το πρόγραμμα να κάνει βρόχο μεταξύ 1 και 9.

```
In [11]: for y in range(1, 10):
             print(y)

1
2
3
4
5
6
7
8
9
```

- Η εντολή range χρησιμοποιείται σε συνδυασμό με βρόχους για να περάσει πάνω από ένα συγκεκριμένο εύρος αριθμών.
- Το παρακάτω είναι ένα περαιτέρω παράδειγμα εκτύπωσης σε βρόχο συμβολοσειρών και αριθμών.

```
In [12]: ts = 0
         for x in range(1, 10):
             ts += x
             print(f"Adding {x}, sum so far is {ts}")

         print(f"Final sum: {ts}")

Adding 1, sum so far is 1
Adding 2, sum so far is 3
Adding 3, sum so far is 6
Adding 4, sum so far is 10
Adding 5, sum so far is 15
Adding 6, sum so far is 21
Adding 7, sum so far is 28
Adding 8, sum so far is 36
Adding 9, sum so far is 45
Final sum: 45
```

III. Λίστες, Λεξικά, Πλειάδες και Σύνολα

Στο μάθημα θα επικεντρωθούμε κυρίως σε Λίστες, Σύνολα και Λεξικά. Είναι σημαντικό να κατανοήσουμε τις διαφορές μεταξύ αυτών των τριών θεμελιωδών τύπων συλλογής.

- Λεξικό (**Dictionary)** - Ένα λεξικό είναι μια μεταβαλλόμενη μη διατεταγμένη συλλογή που ευρετηριάζει με ζεύγη ονομάτων και τιμών.
- Λίστα **(List)** - Μια λίστα είναι μια μεταβαλλόμενη διατεταγμένη συλλογή στοιχείων, που επιτρέπει να είναι διπλότυπα.
- Σύνολο (**Set)** - Ένα σύνολο είναι μια μεταβαλλόμενη μη διατεταγμένη συλλογή μοναδικών (χωρίς διπλότυπα) και αμετάβλητων (immutable) στοιχείων.
- Πλειάδα (**Tuple)** - Μια πλειάδα είναι μια αμετάβλητη διατεταγμένη συλλογή στοιχείων, που επιτρέπει να είναι διπλότυπα.

- Οι λίστες και οι πλειάδες είναι πολύ παρόμοιες στην Python και συχνά συγχέονται. Η σημαντική διαφορά είναι ότι μια λίστα είναι μεταβαλλόμενη, αλλά μια πλειάδα δεν είναι. Έτσι, περιλαμβάνουμε μια λίστα όταν θέλουμε να περιέχει παρόμοια αντικείμενα και περιλαμβάνουμε μια πλειάδα όταν γνωρίζουμε ποιες πληροφορίες περιλαμβάνονται σε αυτήν εκ των προτέρων.

- Πολλές γλώσσες προγραμματισμού περιέχουν μια συλλογή δεδομένων που ονομάζεται πίνακας (array). Ο τύπος πίνακα απουσιάζει αισθητά στην Python. Γενικά, ο προγραμματιστής θα χρησιμοποιήσει μια λίστα στη θέση ενός πίνακα στην Python. Η λίστα Python είναι πολύ πιο ευέλικτη καθώς το πρόγραμμα μπορεί να αλλάζει δυναμικά το μέγεθος μιας λίστας.

Λίστες και πλειάδες

- Οι λίστες και οι πλειάδες διαθέτουν μια διατεταγμένη συλλογή αντικειμένων.
- Η κύρια διαφορά που θα δείτε συντακτικά είναι ότι μια λίστα περικλείεται με τετράγωνα άγκιστρα [] και μια πλειάδα περικλείεται από παρένθεση (). Ο παρακάτω κώδικας ορίζει τόσο τη λίστα όσο και την πλειάδα.

```
In [1]: l = ['α', 'β', 'γ', 'δ']
        t = ('α', 'β', 'γ', 'δ')

        print(l)
        print(t)

['α', 'β', 'γ', 'δ']
('α', 'β', 'γ', 'δ')
```

- Η Python έχει μια δήλωση *for*. Αυτή η δήλωση σας επιτρέπει να κάνετε βρόχο σε κάθε στοιχείο μιας συλλογής, όπως μια λίστα ή μια πλειάδα.

```
In [2]: for s in l:
            print(s)
```

```
α
β
γ
δ
```

- Η συνάρτηση *enumerate* είναι χρήσιμη για την απαρίθμηση σε μια συλλογή και για την πρόσβαση στο ευρετήριο του στοιχείου στο οποίο βρισκόμαστε αυτήν τη στιγμή.

```
In [3]: for i,l in enumerate(l):
            print(f"{i}:{l}")
```

```
0:α
1:β
2:γ
3:δ
```

- Σε μια λίστα μπορεί να προστεθούν πολλά αντικείμενα, όπως συμβολοσειρές. Επιτρέπονται διπλότυπες τιμές. Οι πλειάδες δεν επιτρέπουν στο πρόγραμμα να προσθέσει επιπλέον αντικείμενα μετά τον ορισμό.

```
In [4]: A = []
        A.append('α')
        A.append('β')
        A.append('γ')
        A.append('δ')
        print(A)
```

```
['α', 'β', 'γ', 'δ']
```

- Οι ταξινομημένες συλλογές, όπως λίστες και πλειάδες, σας επιτρέπουν να έχετε πρόσβαση σε ένα στοιχείο με βάση τον αριθμό ευρετηρίου του, όπως γίνεται στον παρακάτω κώδικα. Οι μη ταξινομημένες συλλογές, όπως λεξικά και σύνολα, δεν επιτρέπουν στο πρόγραμμα να έχει πρόσβαση σε αυτά με αυτόν τον τρόπο.

```
In [5]: print(A[1])
β
```

- Σε μια λίστα μπορεί να προστεθούν πολλά αντικείμενα, όπως συμβολοσειρές. Επιτρέπονται διπλότυπες τιμές. Οι πλειάδες δεν επιτρέπουν στο πρόγραμμα να προσθέσει επιπλέον αντικείμενα μετά τον ορισμό. Για τη συνάρτηση εισαγωγής, ένα ευρετήριο, ο προγραμματιστής πρέπει να καθορίσει ένα ευρετήριο. Αυτές οι λειτουργίες δεν επιτρέπονται για πλειάδες επειδή θα οδηγούσαν σε αλλαγή.

```
In [6]: # Insert
        A = ['α', 'β', 'γ']
        A.insert(0, 'α0')
        print(A)
        # Remove
        A.remove('β')
        print(A)
        # Remove at index
        del A[0]
        print(A)
['α0', 'α', 'β', 'γ']
['α0', 'α', 'γ']
['α', 'γ']
```

Σύνολα

- Ένα σύνολο περιέχει μια μη ταξινομημένη συλλογή αντικειμένων, αλλά τα σύνολα δεν επιτρέπουν διπλότυπα. Εάν ένα πρόγραμμα προσθέσει ένα διπλότυπο στοιχείο σε ένα σύνολο, μόνο ένα αντίγραφο κάθε στοιχείου παραμένει στη συλλογή. Η προσθήκη ενός διπλότυπου στοιχείου σε ένα σύνολο δεν οδηγεί σε σφάλμα. Οποιαδήποτε από τις παρακάτω τεχνικές θα ορίσει ένα σύνολο.

```
In [7]: s = set()
        s = { 'α', 'β', 'γ'}
        s = set(['α', 'β', 'γ'])
        print(s)

        {'γ', 'β', 'α'}
```

- Μια λίστα περικλείεται πάντα σε τετράγωνα άγκιστρα [], μια πλειάδα σε παρένθεση (), και ένα σύνολο σε {}. Τα προγράμματα μπορούν να προσθέτουν στοιχεία σε ένα σύνολο καθώς εκτελούνται.

```
In [8]: # Μη αυτόματη προσθήκη στοιχείων, τα σύνολα δεν επιτρέπουν διπλότυπα
        # Τα σύνολα προσθέτουν (add), ενώ οι λίστες όπως είδαμε επεκτείνουν (append).
        B = set()
        B.add('α')
        B.add('β')
        B.add('γ')
        B.add('δ')
        print(B)

        {'γ', 'δ', 'β', 'α'}
```

Λεξικά

- Η Python παρέχει ένα λεξικό, που είναι ουσιαστικά μια συλλογή ζευγών ονόματος-τιμής το οποίο ορίζετα χρησιμοποιώντας {}.

```
In [9]: d = {'name': "Thanos", 'age':"20"}
        print(d)
        print(d['name'])

        if 'name' in d:
            print("Name is defined")

        if 'address' in d:
            print("address defined")
        else:
            print("address undefined")
```

```
{'name': 'Thanos', 'age': '20'}
Thanos
Name is defined
address undefined
```

- Προσέξτε να μην επιχειρήσετε να αποκτήσετε πρόσβαση σε ένα απροσδιόριστο κλειδί, καθώς αυτό θα οδηγήσει σε σφάλμα. Μπορείτε να ελέγξετε εάν έχει οριστεί ένα κλειδί, όπως φαίνεται παραπάνω. Μπορείτε επίσης να αποκτήσετε πρόσβαση στον κατάλογο και να δώσετε μια προεπιλεγμένη τιμή, όπως δείχνει ο ακόλουθος κώδικας.

```
In [10]: d.get('unknown_key', 'default')
Out[10]: 'default'
```

- Μπορείτε επίσης να αποκτήσετε πρόσβαση στα μεμονωμένα κλειδιά και τις τιμές ενός λεξικού.

```
In [11]: d = {'name': "Thanos", 'age':"20"}
         # All of the keys
         print(f"Key: {d.keys()}")

         # All of the values
         print(f"Values: {d.values()}")
```

```
Key: dict_keys(['name', 'age'])
Values: dict_values(['Thanos', '20'])
```

- Τα λεξικά και οι λίστες μπορούν να συνδυαστούν. Τα λεξικά και οι λίστες μαζί είναι ένας καλός τρόπος για τη δημιουργία πολύ περίπλοκων δομών δεδομένων. Ο παρακάτω κώδικας δείχνει μια υβριδική χρήση λεξικών και λιστών.

```
In [12]: # Python list & map structures
         customers = [
             {"name": "Thanos S.", "pets": ["A", "B"]},
             {"name": "Helen K. ", "pets": ["A"]},
             {"name": "Maria P."}
         ]

         print(customers)

         for customer in customers:
             print(f"{customer['name']}:{customer.get('pets', 'no pets')}")
```

```
[{'name': 'Thanos S.', 'pets': ['A', 'B']}, {'name': 'Helen K. ', 'pets': ['A']}, {'name': 'Maria P.'}]
Thanos S.:['A', 'B']
Helen K. :['A']
Maria P.:no pets
```

Σύνθετες λίστες

- Πολλές προηγμένες λειτουργίες είναι διαθέσιμες για λίστες. Μια τέτοια λειτουργία είναι το zip. Δύο λίστες μπορούν να συνδυαστούν σε μια ενιαία λίστα με την εντολή zip. Ο παρακάτω κώδικας δείχνει την εντολή zip.

```
In [1]: a = [1,2,3,4,5]
        b = [5,4,3,2,1]

        print(zip(a,b))
```

```
<zip object at 0x0000011169F58EC0>
```

- Για να δούμε τα αποτελέσματα της συνάρτησης zip, μετατρέπουμε το επιστρεφόμενο αντικείμενο zip στη λίστα. Μπορείτε να δείτε, η συνάρτηση zip επιστρέφει μια λίστα πλειάδων. Κάθε πλειάδα αντιπροσωπεύει ένα ζεύγος στοιχείων. Η σειρά στις δύο λίστες διατηρήθηκε.

```
In [2]: a = [1,2,3,4,5]
        b = [5,4,3,2,1]

        print(list(zip(a,b)))
```

```
[(1, 5), (2, 4), (3, 3), (4, 2), (5, 1)]
```

- Η συνήθης μέθοδος για τη χρήση της εντολής zip είναι μέσα σε έναν βρόχο for. Ο παρακάτω κώδικας δείχνει πώς ένας βρόχος for μπορεί να εκχωρήσει μια μεταβλητή σε κάθε συλλογή που επαναλαμβάνει το πρόγραμμα.

```
In [3]: a = [1,2,3,4,5]
        b = [5,4,3,2,1]

        for x,y in zip(a,b):
            print(f'{x} - {y}')
```

```
1 - 5
2 - 4
3 - 3
4 - 2
5 - 1
```

- Συνήθως, και οι δύο συλλογές θα έχουν το ίδιο μήκος όταν περάσουν στην εντολή zip. Δεν είναι λάθος να έχουμε συλλογές διαφορετικού μήκους. Όπως δείχνει ο ακόλουθος κώδικας, η εντολή zip θα επεξεργάζεται στοιχεία μόνο μέχρι το μήκος της μικρότερης συλλογής.

```
In [4]: a = [1,2,3,4,5]
        b = [5,4,3]

        print(list(zip(a,b)))
```

```
[(1, 5), (2, 4), (3, 3)]
```

- Μερικές φορές μπορεί να θέλετε να γνωρίζετε το τρέχον αριθμητικό ευρετήριο όταν ένας βρόχος for επαναλαμβάνεται μέσω μιας διατεταγμένης συλλογής. Χρησιμοποιήστε την εντολή enumerate για να παρακολουθήσετε τη θέση ευρετηρίου για ένα στοιχείο συλλογής. Επειδή η εντολή enumerate ασχολείται με αριθμητικά ευρετήρια της συλλογής, η εντολή zip θα εκχωρήσει αυθαίρετα ευρετήρια σε στοιχεία από μη ταξινομημένες συλλογές.

- Σκεφτείτε πώς μπορείτε να κατασκευάσετε ένα πρόγραμμα Python για να αλλάξετε κάθε στοιχείο μεγαλύτερο από 5 στην τιμή του 5. Το παρακάτω πρόγραμμα εκτελεί αυτόν τον μετασχηματισμό. Η εντολή enumerate επιτρέπει στον βρόχο να γνωρίζει σε ποιο ευρετήριο στοιχείων βρίσκεται αυτήν τη στιγμή, επιτρέποντας έτσι στο πρόγραμμα να μπορεί να αλλάξει την τιμή του τρέχοντος στοιχείου της συλλογής.

```
In [5]: a = [2, 10, 3, 11, 10, 3, 2, 1]
        for i, x in enumerate(a):
            if x>5:
                a[i] = 5
        print(a)

        [2, 5, 3, 5, 5, 3, 2, 1]
```

- Η εντολή κατανόησης (comprehension command) μπορεί να δημιουργήσει δυναμικά μια λίστα. Η παρακάτω κατανόηση μετράει από το 0 έως το 9 και προσθέτει κάθε τιμή (πολλαπλασιασμένη επί 10) σε μια λίστα.

```
In [6]: lst = [x*10 for x in range(10)]
        print(lst)

        [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
```

- Ένα λεξικό μπορεί επίσης να είναι μια κατανόηση. Η γενική μορφή για αυτό είναι:

dict_variable = {key:value for (key,value) in dictonary.items()}

- Μια κοινή χρήση για αυτό είναι η δημιουργία ενός ευρετηρίου για συμβολικά ονόματα στηλών.

```
In [7]: text = ['col-zero','col-one', 'col-two', 'col-three']
        lookup = {key:value for (value,key) in enumerate(text)}
        print(lookup)

        {'col-zero': 0, 'col-one': 1, 'col-two': 2, 'col-three': 3}
```

- Αυτό μπορεί να χρησιμοποιηθεί για την εύκολη εύρεση του ευρετηρίου μιας στήλης με βάση το όνομα.

```
In [8]: print(f'The index of "col-two" is {lookup["col-two"]}')

        The index of "col-two" is 2
```

IV. Χειρισμός αρχείων

Υπάρχουν πολλοί διαφορετικοί τύποι αρχείων που πρέπει να επεξεργαστείτε. Μερικοί από αυτούς τους τύπους αρχείων παρατίθενται εδώ:

- **Τα αρχεία CSV** (γενικά έχουν την επέκταση .csv) περιέχουν δεδομένα πίνακα που μοιάζουν με δεδομένα υπολογιστικού φύλλου (excel).
- **Τα αρχεία εικόνων** (γενικά με την επέκταση .png ή .jpg) περιέχουν εικόνες για όραση υπολογιστή.
- **Τα αρχεία κειμένου** (συχνά έχουν την επέκταση .txt) περιέχουν μη δομημένο κείμενο και είναι απαραίτητα για την επεξεργασία φυσικής γλώσσας.
- **Το JSON** (συχνά έχει την επέκταση .json) περιέχει ημι-δομημένα δεδομένα κειμένου σε μορφή κειμένου με δυνατότητα ανάγνωσης από τον άνθρωπο.
- **Τα αρχεία ήχου** (συχνά έχουν επέκταση όπως .au ή .wav) περιέχουν ηχογραφημένο ήχο.

Read a CSV File

- Τα προγράμματα Python μπορούν να διαβάσουν αρχεία CSV με Pandas (λεπτομέρειες για το Pandas στην επόμενη ενότητα). Η γενική τους μορφή είναι:

```
]: import pandas as pd

   df = pd.read_csv("https://raw.githubusercontent.com/thanossakkas/data/main/GDP_EU.csv")
   df
```

]:

	TIME	AUT	BEL	CZE
0	1970	3829.731721	3876.367950	NaN
1	1971	4210.441192	4210.433244	NaN
2	1972	4638.534351	4606.148069	NaN
3	1973	5103.430018	5140.661848	NaN
4	1974	5772.158418	5820.638690	NaN
5	1975	6300.543674	6258.089288	NaN
6	1976	6963.477714	6964.342545	NaN
7	1977	7768.883976	7435.042458	NaN
8	1978	8304.538089	8177.509638	NaN
9	1979	9491.450941	9056.835512	NaN
10	1980	10527.998100	10306.817050	NaN
11	1981	11478.234850	11242.562010	NaN
12	1982	12423.597360	11998.441080	NaN
13	1983	13315.535430	12500.938120	NaN
14	1984	13803.863450	13270.032670	NaN
15	1985	14589.375570	13912.992310	NaN
16	1986	15216.037180	14448.962580	NaN
17	1987	15793.916420	15130.905660	NaN
18	1988	16866.050590	16348.194070	NaN
19	1989	18126.710730	17514.815960	NaN
20	1990	19473.504510	18687.891450	12689.40152
21	1991	20618.036940	19599.302470	11655.62003
22	1992	21293.741180	20269.731530	11850.36547
23	1993	21733.353720	20471.138380	12123.72874
24	1994	22643.344320	21518.981950	12736.02198
25	1995	23698.629270	22447.550310	13855.63267
26	1996	24561.160500	22744.399560	14689.90856

Read a Text File

In [1]:
```
import urllib.request
import codecs
url = "https://raw.githubusercontent.com/thanossakkas/data/main/ithaki.txt"
with urllib.request.urlopen(url) as urlstream:
    for line in codecs.iterdecode(urlstream, 'utf-8'):
        print(line.rstrip())
```

```
ITHAKI
(I have used the greek written form of Ulysse's island:
Ithaki (I'thaki) -note that the accent is on the second syllable.)
When you set out on the way to Ithaki
wish for a long, eventful journey,
full of adventure, full of understanding.
Of Laistrygonians and Cyclops,
the raging Poseidon, never be afraid.
You'll never find such things on your way
if all your thoughts remain noble, and if your spirit
and body are touched by worthy emotion.
The Laistrygonians and Cyclops,
the fierce Poseidon, will never be encountered
if they're not carried in your soul, and if your soul
does not erect them on your path, before you.
```

Read an Image

Μπορείτε να χρησιμοποιήσετε το πακέτο Python PIL για επεξεργασία εικόνας. Ο παρακάτω κώδικας δείχνει πώς να φορτώσετε μια εικόνα από μια διεύθυνση URL και να την εμφανίσετε.

In [1]:
```
%matplotlib inline
from PIL import Image
import requests
from io import BytesIO

url = "https://upload.wikimedia.org/wikipedia/en/a/a9/Example.jpg"

response = requests.get(url)
img = Image.open(BytesIO(response.content))

img
```

Out[1]:

V. Συναρτήσεις

- Η συνάρτηση ορίζεται με το def.
- Μπορείτε να ορίσεται μια συνάρτηση με τις παραμέτρους της. Οι παράμετροι συνάρτησης μπορούν να ονομαστούν

```
In [1]: def say_hello(teacher, person_to_greet, greeting = "Hello"):
            print(f'{greeting} {person_to_greet}, this is {teacher}.')

        say_hello('Panos', "George")
        say_hello('Panos', "George", "Goodnight")
        say_hello(teacher="Panos", person_to_greet="George", greeting = "Goodnight")

        Hello George, this is Panos.
        Goodnight George, this is Panos.
        Goodnight George, this is Panos.
```

- ή να μην ονομαστούν

```
In [2]: def fun():
          print("Welcome to GFG")
```

- Επίσης μπορείτε να την καλείτε αρκετά εύκολα.

```
In [3]: fun()

        Welcome to GFG
```