ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Γλώσσες Προγραμματισμού

Εισαγωγικές έννοιες - Python

Χαρά Πρασσά, chprassa@gmail.com

Χαρακτηριστικά της γλώσσας Python

- Γλώσσα υψηλού επιπέδου
- Δωρεάν και ανοιχτού κώδικα
- Πληθώρα βιβλιοθηκών
- Εκδόσεις γλώσσας
  - Έκδοση 2
  - Έκδοση 3

Γλώσσα γενικού σκοπού με πολλές εφαρμογές, μπορεί να χρησιμοποιηθεί:

- Εφαρμογές Μηχανικής Μάθησης (Machine Learning) και Τεχνητής Νοημοσύνης (Artificial Intelligence) (Βιβλιοθήκες Scikit-Learn, Pandas, NumPy)
- Εφαρμογές οπτικοποίησης δεδομένων (Data Visualization) – Βιβλιοθήκες Seaborn – Matplotlib
- Ανάλυση δεδομένων
- Ανάπτυξη web εφαρμογών και πολλές ακόμα εφαρμογές

Γνωστές εταιρείες που χρησιμοποιούν Python: Facebook, YouTube, NETFLIX, Dropbox, Pinterest κλπ.

- ❑ Η Python δημιουργήθηκε από τον Ολλανδό Guido Van Rossum και κυκλοφόρησε πρώτη φορά το 1991.
- ❑ Η ονομασία της προέρχεται από τους Monty Python!

Ορισμός και χρήση της print() function

Η πιο απλή συνάρτηση της Python, είναι η print() με την οποία μπορούμε να εκτυπώσουμε ένα αποτέλεσμα/μήνυμα στο terminal παράθυρο.

- Για να καλέσουμε την **print( )** απλά γράφουμε το όνομα της και αμέσως μετά ανοίγουμε και κλείνουμε ένα σετ από παρενθέσεις. Μέσα στις παρενθέσεις γράφουμε το μήνυμα **(argument)** που θέλουμε να εκτυπώσουμε στο terminal παράθυρο του υπολογιστή μας.
- ***Argument***: πληροφορία που γράφουμε μέσα στην παρένθεση. Η αρχή και το τέλος του argument ορίζονται από ένα σετ single quotes (' ').
- ***Statement:*** Κάθε γραμμή κώδικα Python που εκτελεί μια ολοκληρωμένη πράξη

example1.py ×

example1.py

```
1   print('Όνομα: Μαρία')
2   print('Επώνυμο: Παπαδοπούλου')
3   print('Αριθμός μητρώου: 7123000')
```

η εκτέλεση του κώδικα γίνεται από πάνω προς τα κάτω και με την σειρά που έχουν γραφτεί τα functions

```
Όνομα: Μαρία
Επώνυμο: Παπαδοπούλου
Αριθμός μητρώου: 7123000
```

- **String:** μια ομάδα χαρακτήρων
- **String literal:** μια ομάδα χαρακτήρων που εμφανίζεται ή χρησιμοποιείται μέσα στον κώδικα
- Εξ ορισμού ένα string literal πρέπει να το περικλείουμε μέσα σε εισαγωγικά (quotes) για να καταλαβαίνει η Python από που ξεκινάει και που τελειώνει.

❑ Η Python όμως μας δίνει την ευελιξία να χρησιμοποιήσουμε είτε **single quotes(' '),** όπως ήδη έχουμε κάνει, **είτε double quotes (" ").** Αυτή η ευελιξία ίσως σας φανεί χρήσιμη σε εκείνες τις περιπτώσεις όπου το string literal περιέχει απόστροφο λόγω του αγγλικού συντακτικού.

```
1. print("UCLA's minimum admission requirements")
```

Output

```
UCLA's minimum admission requirements
```

```
2. print('University name: "University of California, Los Angeles"')
```

Output

```
University name: "University of California, Los Angeles"
```

- Η Python μας επιτρέπει να χρησιμοποιήσουμε επίσης και **triple quotes** είτε " " " είτε '''.

- Η χρήση των **triple quotes** συνίσταται στις περιπτώσεις όπου τα string literals περιέχουν single quotes ή και double quotes.
- Το κύριο χαρακτηριστικό τους όμως, είναι η δυνατότητα τους να περικλείουν πολλαπλές γραμμές από string literals κάτι που λείπει σαν ικανότητα από τα single quotes και τα double quotes.

```
3. print('''University name: "University of California, Los Angeles"
Location: Los Angeles, California, USA
Average GPA: 3.9''')
```

Output

University name: "University of California, Los Angeles"

Location: Los Angeles, California, USA

Average GPA: 3.9

- Εναλλακτικός τρόπος: αν τα εισαγωγικά πρέπει να είναι μέρος της ακολουθίας χαρακτήρων το δηλώνουμε χρησιμοποιώντας το πρόθεμα '\'. ο χαρακτήρας '\' ονομάζεται χαρακτήρας διαφυγής (escape character) και δηλώνει την ειδική μεταχείριση του χαρακτήρα που ακολουθεί.

```
4. print('UCLA\'s minimum admission requirements')
```

Output

UCLA's minimum admission requirements

- Ο χαρακτήρας αλλαγής γραμμής '\n', μεταφέρει την εκτύπωση στην επόμενη γραμμή.

```
5. print("University name: \"University of California, Los Angeles\"\n
   Location: Los Angeles, California, USA\n Average GPA: 3.9")
```

Output

University name: "University of California, Los Angeles"

Location: Los Angeles, California, USA

Average GPA: 3.9

- Τα **comments** είναι σύντομες επεξηγήσεις του κώδικα με την μορφή μικρού και συνοπτικού μηνύματος που τα αγνοεί η Python κατά την εκτέλεση της εφαρμογής αλλά είναι χρήσιμα στον προγραμματιστή γιατί μπορεί να γυρίσει πίσω στον κώδικα που έγραψε πριν καιρό και να θυμηθεί, διαβάζοντας τα comments (σχόλια), για ποιο σκοπό είχαν γραφτεί συγκεκριμένες λειτουργίες του κώδικα. Κάθε comment γραμμή ξεκινάει με το # σύμβολο.

example1.py

```
1   #UCLA's minimum admission requirements
2   print('''University name: "University of California, Los Angeles"
3   Location: Los Angeles, California, USA
4   Average GPA: 3.9''')
5   print('Acceptance rate: 14.3%')#UCLA average acceptance rate
```

end-line comment

Output

University name: "University of California, Los Angeles"
Location: Los Angeles, California, USA
Average GPA: 3.9
Acceptance rate: 14.3%

- **μεταβλητή (variable)** : όνομα που ορίζουμε μέσα στον κώδικα το οποίο αντιπροσωπεύει μια τιμή που υπάρχει στην μνήμη του υπολογιστή ή μια τιμή που επιθυμούμε να αποθηκεύσουμε προσωρινά στην μνήμη του υπολογιστή.

- Ο τρόπος με τον οποίο ορίζουμε μεταβλητές στην Python είναι ο εξής:

variable = expression

- Το σύμβολο της ισότητας (=) ονομάζεται **assignment operator** γιατί αναθέτει μια τιμή στην μεταβλητή
- **variable** είναι το όνομα της μεταβλητής
- **expression** είναι η τιμή ή ο κώδικας που μπορεί να μας επιστρέψει μια τιμή

- Για να μπορέσουμε να δούμε ποια είναι η τιμή ενός variable, καλούμε την print() function χωρίς όμως να χρησιμοποιήσουμε τα double quotes ή τα single quotes.

example1.py > ...

```
1   GPA=3.9
2   print("The average GPA at UCLA is")
3   print(GPA)
```

Output

```
The average GPA at UCLA is
3.9
```

- Η print () function μπορεί να επιστρέψει ένα αποτέλεσμα/μήνυμα σε βελτιωμένη μορφή αν δεχτεί πολλαπλά arguments αρκεί αυτά να χωρίζονται με κόμμα. Το κάθε argument θα μπορούσε να είναι ένα string μήνυμα, μια μεταβλητή ή κάποιο άλλο στοιχείο της Python.

example1.py > ...

```
1   GPA=3.9
2   print("The average GPA at UCLA is", GPA)
3
```

Output

The average GPA at UCLA is 3.9

```
GPA=3.9
Acceptance_rate=14.3
print("The average GPA at UCLA is", GPA, "and the average acceptance
rate is", Acceptance_rate, "%")
```

Output

The average GPA at UCLA is 3.9 and the average acceptance rate is 14.3 %

- Ένας εναλλακτικός τρόπος ανάθεσης τιμών σε μεταβλητές είναι να τις γράψουμε όλες σε μια γραμμή χωρισμένες από κόμμα (,).

```
GPA, Acceptance_rate =3.9, 14.3
print('The average GPA at UCLA is', GPA, 'and the average acceptance rate is', Acceptance_rate, "%")
```

```
Output
```

The average GPA at UCLA is 3.9 and the average acceptance rate is 14.3 %

- Αν ορίζαμε τις μεταβλητές στο terminal του Visual studio code ή στο Shell του IDLE, θα μπορούσαμε να δούμε την τιμή τους απλά γράφοντας το όνομα τους και δίχως τη χρήση της print(). Η print() είναι απαραίτητη εντολή για να δει ο χρήστης τα αποτελέσματα, όταν ο κώδικας συντάσσεται στον Editor.

```
>>> GPA, Acceptance_rate =3.9, 14.3
>>> GPA
3.9
>>> Acceptance_rate
14.3
```

- Το όνομα μεταβλητή (variable) σημαίνει ότι η τιμή της μπορεί να αλλάξει κατά την διάρκεια εκτέλεσης του κώδικα. Όταν αναθέσουμε μια τιμή σε μια μεταβλητή, η μεταβλητή θα αναφέρεται σε αυτήν μέχρι να της αναθέσουμε μια καινούργια τιμή.
- Το παρακάτω παράδειγμα μας δείχνει αυτήν ακριβώς την ιδιότητα των μεταβλητών.

example1.py > ...

```
1   GPA=3.9
2   print("The average GPA at UCLA is", GPA)
3   GPA=3.0
4   print('For students in California, the GPA requirement is', GPA)
```

Output

The average GPA at UCLA is 3.9

For students in California, the GPA requirement is 3.0

*Η τιμή 3.9, που είναι η παλιά τιμή, είναι ακόμα στην μνήμη του υπολογιστή, αλλά δεν μπορεί να χρησιμοποιηθεί γιατί δεν υπάρχει αναφορά επάνω της από καμία μεταβλητή. Όταν μια τιμή στην μνήμη του υπολογιστή δεν έχει κάποια μεταβλητή να αναφέρεται σε αυτήν, τότε ο interpreter της Python αυτόματα απομακρύνει αυτή την τιμή από την μνήμη (**garbage collection**).*

- Μπορούν να περιέχουν αλφαριθμητικούς χαρακτήρες, νούμερα και κάτω παύλα (_) αλλά πρέπει να ξεκινούν με γράμμα ή με κάτω παύλα.
- Τα πεζά διακρίνονται από τα κεφαλαία (case sensitive), value_1 ≠ Value_1
- Το όνομα της μεταβλητής δεν μπορεί να περιέχει κενά.
- Στην Python 3, είναι δυνατή η χρήση ελληνικών χαρακτήρων (δεν συνίσταται).
- Κεφαλαία γράμματα υποδηλώνουν σταθερές (σύμβαση η οποία δεν επιβάλλεται).

➢ *Παραδείγματα αποδεκτών ονομάτων μεταβλητών:*

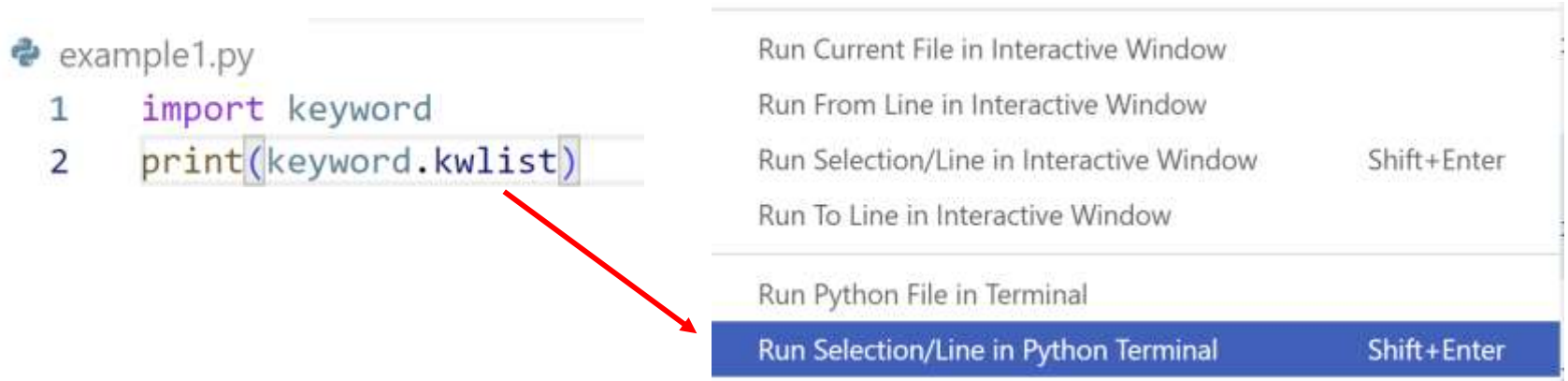
```
positive_number=3
Grade=10
```

➢ *Μη αποδεκτά ονόματα μεταβλητών:*

- 1number - αρχίζει με αριθμό
- number! - περιέχει το !
- total average – περιέχει κενά
- class - είναι λέξη κλειδί

- Οι λέξεις-κλειδιά είναι λέξεις τις οποίες έχει δεσμεύσει η ίδια η Python για δική της χρήση οπότε και δεν μπορούμε να τις χρησιμοποιήσουμε σαν ονόματα των δικών μας μεταβλητών.

Είναι δεσμευμένες λέξεις με ειδική σημασία. Μπορούμε να τις δούμε όλες:



['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

- Στην Python, τα αντικείμενα «ακολουθίες χαρακτήρων» ονομάζονται strings (συμβολοσειρές). Πέρα από τα strings, η Python χειρίζεται και άλλα αντικείμενα, τα λεγόμενα *βαθμωτά* αντικείμενα (scalar objects). Κάθε αντικείμενο έχει *τύπο* (type) ο οποίος ορίζει το τι μπορεί να κάνει ένα πρόγραμμα με ένα αντικείμενο του συγκεκριμένου τύπου. Οι τύποι των βαθμωτών αντικειμένων της Python είναι οι ακόλουθοι:

1. int (ακέραιος)
2. float (κινητής υποδιαστολής)
3. bool (λογική μεταβλητή: παίρνει τις τιμές True ή False)
4. NoneType (παίρνει την τιμή None, δηλώνει απουσία τιμής)

Οι αριθμοί που αποθηκεύουμε εκχωρούνται σε μεταβλητές.

```
>>> number=10
>>> pi=3.14
>>> d=True
>>> n=None
>>> b=1.23E-7 (επιστημονικός συμβολισμός)
```

ο πρώτος χαρακτήρας πρέπει να είναι κεφαλαίο

Η ακολουθία χαρακτήρων που αποθηκεύουμε εκχωρείται σε μεταβλητή.

```
>>> name='John'
```

- Με την σύνταξη type(όνομα μεταβλητής) μπορούμε να ελέγξουμε τον τύπο των μεταβλητών οι οποίες ήδη έχουν ορισθεί (έχουν τιμές).

```
>>> type(number)
<class 'int’>
>>> type(pi)
<class 'float’>
>>> type(d)
<class 'bool’>
>>> type(n)
<class 'NoneType’>
>>> type(name)
<class 'str’>
```

```
1    number=10
2    print('The type of number is',type(number))
```

Output

The type of number is <class 'int'>

- Μέχρι τώρα σε όλα μας τα παραδείγματα οι τιμές που λαμβάνουν οι μεταβλητές είναι γραμμένες μέσα στο κώδικα. Η συνάρτηση **input()** δίνει τη δυνατότητα στο χρήστη να αλληλοεπιδράσει με το πρόγραμμα εισάγοντας ένα μήνυμα/τιμή δια μέσω του terminal της Python.
- input( ): αφού διαβάσει την τιμή που θα εισάγουμε, θα πρέπει να την αναθέσει σε μια μεταβλητή, έτσι ώστε το πρόγραμμα μας να μπορεί να την χρησιμοποιήσει.

```
1   first_name=input("Please enter your first name: ")
2   last_name=input("Please enter your last name: ")
3   print('Your full name is', first_name, last_name)
```

Output

Please enter your first name: Maria
Please enter your last name: Papadopoulou
Your full name is Maria Papadopoulou

- *Το παραπάνω πρόγραμμα δείχνει ότι η όλη διαδικασία για να διαβάζουμε τιμές από τον χρήστη είναι πολύ εύκολη. Όμως υπάρχει ένα μικρό πρόβλημα – οι τιμές που περάσαμε στο πρόγραμμα είναι όλες string το οποίο είναι το default data type όταν η Python διαβάζει τιμές από το keyboard. Οπότε αν αντί για το όνομα Maria περνούσαμε στο πρόγραμμα τον αριθμό 1, για την Python δεν θα ήταν ο αριθμός 1 αλλά ο string χαρακτήρας 1 με τον οποίο φυσικά δεν μπορούμε να κάνουμε αριθμητικές πράξεις.*

```
1   number = input('Please enter an integer number: ')
2   print('The data type of number is', type(number))
```

Output

Please enter an integer number: 1
The data type of number is <class 'str'>

❑ *Το παραπάνω πρόβλημα μπορεί να διορθωθεί μέσω της μετατροπής του τύπου των μεταβλητών (data type conversion)*

- **int()** : μετατροπή του περιεχομένου της παρένθεσης σε ακέραιο
- **float()** : μετατροπή του περιεχομένου της παρένθεσης σε κινητής υποδιαστολής
- **str()** : μετατροπή του περιεχομένου της παρένθεσης σε συμβολοσειρά

```
1   first_name = input("Enter your first name: ")
2   last_name = input("Enter your last name: ")
3   years = int(input("Enter your years of study: "))
4   average_GPA = float(input("Enter your average GPA: "))
```

Output

Enter your first name: Maria
Enter your last name: Papadopoulou
Enter your years of study: 4
Enter your average GPA: 3.7

```
>>> type(years)
<class 'int'>
>>> type(average_GPA)
<class 'float'>
```

- Τα μαθηματικά σύμβολα στην Python δίνονται στον ακόλουθο πίνακα:

Σύμβολο	Πράξη	Περιγραφή
+	Πρόσθεση	Προσθέτει δύο αριθμούς
-	Αφαίρεση	Αφαιρεί έναν αριθμό από έναν άλλο
*	Πολλαπλασιασμός	Πολλαπλασιάζει δύο αριθμούς
/	Διαίρεση	Διαιρεί δύο αριθμούς και επιστρέφει το αποτέλεσμα σε δεκαδικό αριθμό
//	Διαίρεση	Επιστρέφει το ακέραιο μέρος της διαίρεσης
%	Υπόλοιπο	Επιστρέφει το υπόλοιπο της διαίρεσης
**	Δύναμη	Ύψωση σε δύναμη

```
>>> print("Πρόσθεση δύο αριθμών",2+3)
Πρόσθεση δύο αριθμών 5
>>> print("Αφαίρεση δύο αριθμών",5-2)
Αφαίρεση δύο αριθμών 3
>>> print("Διαίρεση 2 αριθμών",5/2)
Διαίρεση δύο αριθμών 2.5
>>> print("Ακέραια διαίρεση δύο αριθμών",5//2)
Ακέραια διαίρεση δύο αριθμών 2
>>> print("Υπόλοιπο διαίρεσης δύο αριθμών",5%2)
Υπόλοιπο διαίρεσης δύο αριθμών 1
>>> print("Ύψωση αριθμού σε δύναμη",2**3)
Ύψωση αριθμού σε δύναμη 8
```

έλεγχος αν αριθμός είναι άρτιος ή περιττός

- Σε περίπτωση αριθμητικών πράξεων με πολλούς τελεστές ισχύει η προσεταιριστική ιδιότητα. Πιο συγκεκριμένα,

1. Πρώτα θα εκτελεστούν οι πράξεις μέσα σε παρένθεση
2. Μετά θα εκτελεστούν οι τελεστές ύψωσης σε δύναμη (**)
3. Μετά ακολουθούν οι τελεστές πολλαπλασιασμού, διαίρεσης και υπολοίπου (*, /, //, %)
3. Και τέλος θα εκτελεστούν οι τελεστές της πρόσθεσης και αφαίρεσης (+, -)
4. Οι τελεστές σύγκρισης έχουν χαμηλότερη προτεραιότητα από όλους τους αριθμητικούς τελεστές.

- Αν υπάρχουν ταυτόχρονα στην ίδια αριθμητική πράξη τελεστές που έχουν την ίδια σειρά προτεραιότητας, π.χ πολλαπλασιασμός και διαίρεση, τότε η εκτέλεση θα γίνει από τα αριστερά προς τα δεξιά όπως είναι και ο κανόνας της άλγεβρας.

```
>>> 6-2*3+7-1
6
>>> 4 + 5 // 3
5
```

- Η χρήση των παρενθέσεων καθιστά πιο εύκολο τον έλεγχο της σειράς εκτέλεσης των πράξεων. Η πράξη που βρίσκεται μέσα στην παρένθεση θα εκτελεστεί πρώτη ανεξάρτητα από την σειρά προτεραιότητας των τελεστών.

```
>>> (6 - 3) * (2 + 7) - 1
26
>>> (4 + 5) // 3
3
```

❑ Όταν εκτελούμε πράξεις με την Python υπάρχουν κάποιες αυτόματες μετατροπές που γίνονται και αφορούν σε τρείς απλούς κανόνες.
Πιο συγκεκριμένα:

1. Όταν μια αριθμητική πράξη εκτελείται ανάμεσα σε δύο int τιμές, τότε το αποτέλεσμα θα είναι είδος int.
2. Όταν μια αριθμητική πράξη εκτελείται ανάμεσα σε δύο float, τότε το αποτέλεσμα θα είναι είδος float.
2. Όταν μια αριθμητική πράξη εκτελείται ανάμεσα σε έναν int και ένα float αριθμό, τότε ο int αριθμός μετατρέπεται αυτόματα σε float και το αποτέλεσμα θα είναι float.

```
>>> print(5*2.0)
10.0
>>> type(5*2.0)
<class 'float'>
```

❑ Όταν η συγκεκριμένη πράξη εκτελείται, η τιμή του 5 θα μετατραπεί σε float, δηλαδή 5.0, και το αποτέλεσμα του πολλαπλασιασμού με το 2.0 θα μας επιστρέψει την τελική τιμή 10.0 που είναι float.

```
>>> print(int(5*2.0))
10
>>> type(int(5*2.0))
<class 'int'>
```

Η μετατροπή μπορεί να γίνει και από το χρήστη μέσω των γνωστών συναρτήσεων int() και float()

- Αν η αριθμητική πράξη που θέλετε να εκτελέσετε έχει πολλούς αριθμούς και θέλετε να την χωρίσετε σε 2 ή και περισσότερες γραμμές για να χωράει στην οθόνη σας, τότε μπορείτε να χρησιμοποιήσετε το backslash (\ ) όπως δείχνει το παρακάτω παράδειγμα.

```
>>> var1=2
>>> var2=3
>>> var3=4
>>> var4=7
>>> result=var1*5+var2*3+\
... var3*5+var4*6
>>> print("The result of the calculation is: ", result)
The result of the calculation is:  81
```

- Επίσης θα μπορούσατε να αποφύγετε την χρήση του backslash, αν η αριθμητική πράξη βρίσκεται εξ ολοκλήρου μέσα στην print( ) function.

```
>>> var1=2
>>> var2=3
>>> var3=4
>>> var4=7
>>> print("The result of the calculation is: ", var1*5+var2*3+
... var3*5+var4*6)
The result of the calculation is:  81
```

Στρογγυλοποίηση αριθμών

- Υπάρχουν πολλές χρήσιμες συναρτήσεις που μπορείτε να χρησιμοποιήσετε σε αριθμητικά δεδομένα, οι οποίες παρέχονται από την Python. Κάποιες από αυτές μπορείτε να τις χρησιμοποιήσετε απευθείας, κάποιες όμως χρειάζεται να εισάγετε πρώτα τη βιβλιοθήκη που τις περιέχει για να μπορέσετε να τις χρησιμοποιήσετε.
- Μια συνάρτηση είναι μια ομαδοποίηση εντολών οι οποίες εκτελούν μια λειτουργικότητα, πχ. η print() που έχουμε ήδη χρησιμοποιήσει.
- Η συνάρτηση round() κάνει στρογγυλοποίηση του αριθμού που δέχεται η συνάρτηση σαν όρισμα (μέσα στην παρένθεση).

```
>>> round(5.342)
5
```
← στρογγυλοποίηση αριθμού 5.342, επιστρέφει int

```
>>> round(5.342,2)
5.34
```
← στρογγυλοποίηση του αριθμού που δήλωσα σαν 1ο όρισμα σε τόσα δεκαδικά ψηφία όσα αναφέρει το 2ο όρισμα

```
>>> round(5.342,0)
5.0
```
← στρογγυλοποίηση αριθμού 5.342, επιστρέφει float

- Στην Python χρειάζεται προσοχή όταν δουλεύετε με αριθμούς τύπου float. Για παράδειγμα:

```
>>> number_1=1.1
>>> number_2=1.3
>>> number_1+number_2
2.4000000000000004
```

Η Python για να εκτελέσει πράξεις με πραγματικούς αριθμούς μετατρέπει τον float σε κλάσμα μετά εκτελεί τις πράξεις και το αποτέλεσμα αυτό ξανά πίσω σε float.

Στρογγυλοποίηση αριθμών

- Η χρήση της συνάρτησης round() θέλει ιδιαίτερη προσοχή καθώς με τον τρόπο που λειτουργούν οι πραγματικοί αριθμοί η Python θα μας δώσει το αντίστοιχο αποτέλεσμα.

```
>>> round(1.5)          στρογγυλοποίηση προς τα πάνω
2
>>> round(2.5)          στρογγυλοποίηση προς τα κάτω
2
```

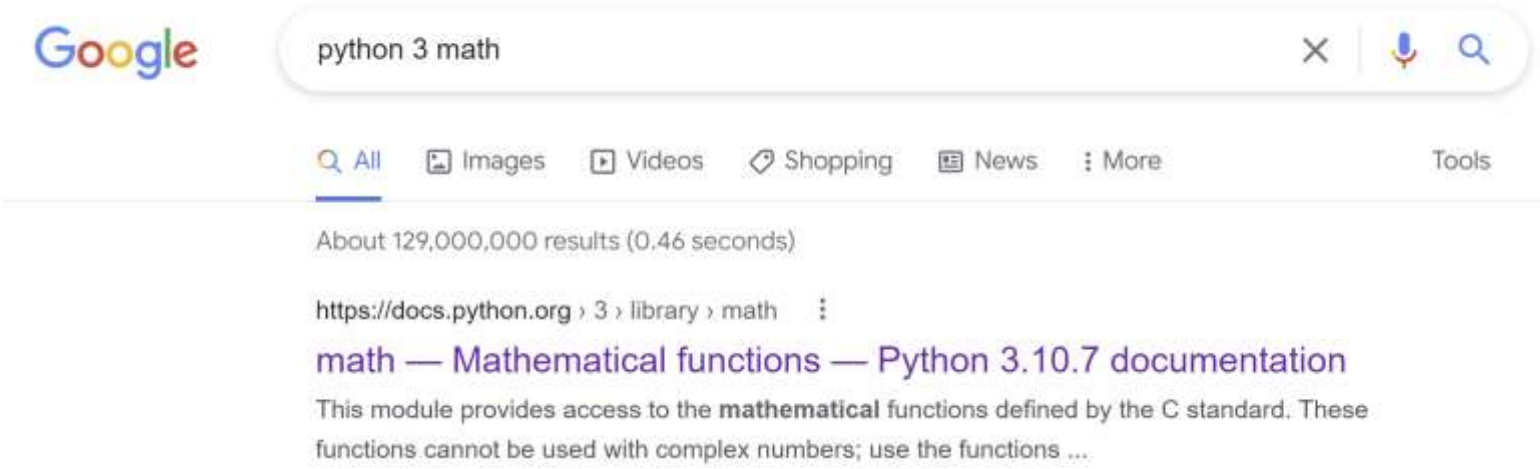
- Για να αποφύγουμε την εμφάνιση τέτοιων 'παράδοξων' αποτελεσμάτων μπορούμε να χρησιμοποιήσουμε συναρτήσεις της βιβλιοθήκης math.
  - Η συνάρτηση floor() στρογγυλοποιεί πάντα τον αριθμό προς τα κάτω στον αμέσως προηγούμενο ακέραιο αριθμό στην κλίμακα των ακεραίων
  - Η συνάρτηση ceil() στρογγυλοποιεί πάντα τον αριθμό προς τα πάνω στον αμέσως επόμενο ακέραιο αριθμό στην κλίμακα των ακεραίων
  - Για να χρησιμοποιήσετε τις συναρτήσεις μιας βιβλιοθήκης πρέπει πρώτα να εισάγετε τη βιβλιοθήκη στον κώδικα σας μέσω της συνάρτησης import.

```
>>> import math
>>> math.floor(1.5)
1
>>> math.floor(2.5)
2
>>> math.floor(-4.5)
-5
>>> math.ceil(3.1)
4
```

Για να χρησιμοποιήσετε τη συνάρτηση μιας βιβλιοθήκης πρέπει να ακολουθήσετε τη σύνταξη:

Όνομα βιβλιοθήκης.Όνομα συνάρτησης

- Για να δείτε όλες τις συναρτήσεις που περιέχει μια βιβλιοθήκη πχ. η math μπορείτε να μεταβείτε στο Google και να πληκτρολογήσετε python 3 math. *Προσοχή να εισάγετε την έκδοση της Python που χρησιμοποιείτε 2 ή 3.* Από εκεί ο πρώτος σύνδεσμος σας οδηγεί στο documentation της Python. Εκεί μπορείτε να δείτε και άλλες συναρτήσεις της math όπως για παράδειγμα την factorial() η οποία υπολογίζει το παραγοντικό ενός ακεραίου αριθμού. Μπορείτε ακόμα να δείτε πως συντάσσεται μια συνάρτηση, τα ορίσματα τα οποία δέχεται, όπως και μια μικρή περιγραφή της λειτουργίας της.



Βιβλιοθήκη math

math — Mathematical functions

This module provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the cmath module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

The following functions are provided by this module. Except when explicitly noted otherwise, all return values are floats.

Number-theoretic and representation functions

`math.ceil(x)`

Return the ceiling of *x*, the smallest integer greater than or equal to *x*. If *x* is not a float, delegates to `x.__ceil__`, which should return an `Integral` value.

`math.comb(n, k)`

Return the number of ways to choose *k* items from *n* items without repetition and without order.

Evaluates to `n! / (k! * (n - k)!)` when `k <= n` and evaluates to zero when `k > n`.

Also called the binomial coefficient because it is equivalent to the coefficient of k-th term in polynomial expansion of the expression `(1 + x) ** n`.

- Η Python παρέχει αρκετές χρήσιμες συναρτήσεις παραγωγής (ψεύδο)-τυχαίων αριθμών στη βιβλιοθήκη random.
- Όπως και με τη math έτσι και με τη random πρέπει πρώτα να εισάγετε τη βιβλιοθήκη στον κώδικα σας μέσω της συνάρτησης import.

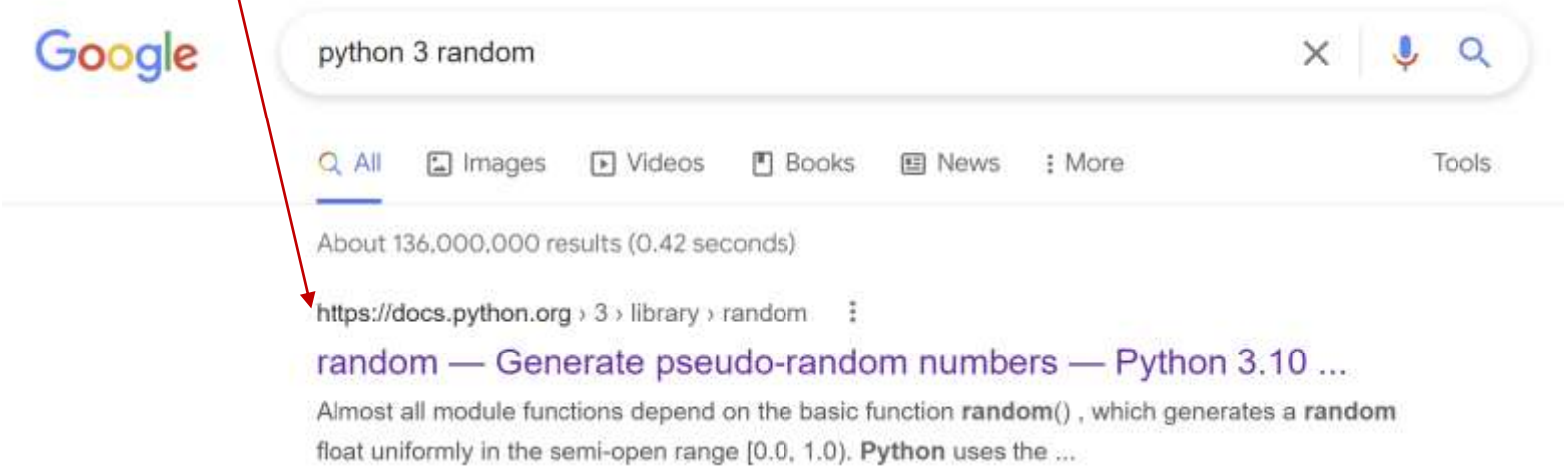
```
>>> import random
```

- Μπορείτε να αλλάξετε τον τρόπο αναφοράς στη βιβλιοθήκη που εισάγετε αν δεν πατήσετε Enter και συνεχίσετε την εντολή γράφοντας για παράδειγμα as rnd.

```
>>> import random as rnd
```

δίνετε ένα πιο μικρό όνομα ή ένα όνομα που εσείς επιθυμείτε για να αναφερθείτε στη βιβλιοθήκη

Ο πρώτος σύνδεσμος σας οδηγεί στο documentation της random



- Στα επόμενα παραδείγματα παρουσιάζουμε μερικές χρήσιμες συναρτήσεις της βιβλιοθήκης random

1. Η συνάρτηση randint() δέχεται σαν ορίσματα 2 ακέραιες τιμές και επιστρέφει ένα ψευδοτυχαίο ακέραιο αριθμό ανάμεσα σε αυτό το κλειστό διάστημα.

```
>>> rnd.randint(0,10)
3
>>> rnd.randint(0,10)
9
```

Κάθε φορά που τρέχετε τη συνάρτηση το αποτέλεσμα είναι ένας νέος αριθμός

✓ *Αν επιθυμούμε να παράγουμε ψευδοτυχαίους αριθμούς αλλά με την ίδια σειρά κάθε φορά, τότε μπορούμε να τρέξουμε τη συνάρτηση* **rnd.seed()** *και σαν όρισμα να περάσουμε μια οποιαδήποτε τιμή. Καλώντας τη συνάρτηση seed() με όρισμα κάποιο ακέραιο αριθμό (αν δεν δοθεί νοείται ένας ακέραιος που προκύπτει από την ώρα της ημέρας), αρχικοποιούμε την ακολουθία τυχαίων αριθμών.*

```
>>> rnd.seed(123)
>>> rnd. randint(0,10)
0

>>> rnd.seed(123)
>>> rnd. randint(0,10)
0
```

Επιστρέφει ένα τυχαίο αριθμό βάσει του εσωτερικού αλγόριθμου υλοποίησης της seed()

Επιστρέφει την ίδια τιμή ξανά, παράγει μεν ψευδοτυχαίους αριθμούς αλλά με την ίδια σειρά όσο εισάγουμε τον ίδιο ακέραιο αριθμό σαν όρισμα στη seed()

2. Η συνάρτηση randrange() δέχεται σαν ορίσματα 2 ακέραιες τιμές και επιστρέφει έναν ψευδοτυχαίο ακέραιο αριθμό ανάμεσα σε αυτό το διάστημα χωρίς να περιλαμβάνεται η άνω τιμή. Υπάρχει επίσης η δυνατότητα να εισάγουμε και τρίτο όρισμα το οποίο υποδηλώνει το βήμα.

```
>>> rnd.randrange(1,10,2)
7
```

τυχαίος περιττός ακέραιος αριθμός μεταξύ 1 και 9

3. Η συνάρτηση uniform() δέχεται σαν ορίσματα 2 τιμές και επιστρέφει ένα ψευδοτυχαίο αριθμό τύπου float ανάμεσα σε αυτό το κλειστό διάστημα.

```
>>> rnd.uniform(1,3)
2.6770070329487154
```

Τι θα κάναμε αν θέλαμε μόνο 2 δεκαδικά ψηφία;

```
>>> round(rnd.uniform(1,3),2)
2.75
```

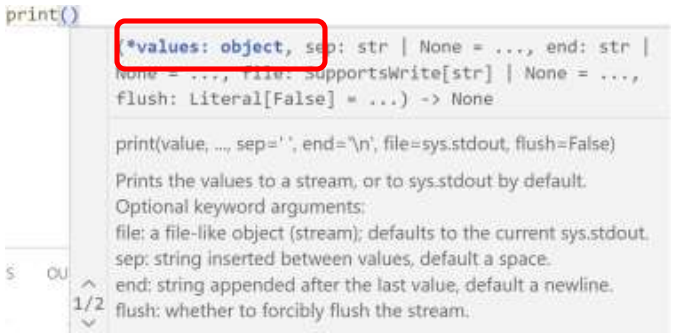
- Η βασική λειτουργία της print( ) είναι να τυπώνει το αποτέλεσμα σε μια καινούργια γραμμή σαν αποτέλεσμα.

```
print("Δευτέρα")
print("Τρίτη")
print("Τετάρτη")
print("Πέμπτη")
print("Παρασκευή")
print("Σάββατο")
print("Κυριακή")
```

terminal output

```
Δευτέρα
Τρίτη
Τετάρτη
Πέμπτη
Παρασκευή
Σάββατο
Κυριακή
```

- Από το documentation που αφορά στη σύνταξη της συνάρτησης, παρατηρούμε ότι μπορούμε να έχουμε πολλαπλά μηνύματα μέσα στην ίδια print() που μπορούμε να τυπώσουμε. Το κάθε μήνυμα χωρίζεται από το προηγούμενο με κόμμα.




```
print("Δευτέρα","Τρίτη","Τετάρτη","Πέμπτη","Παρασκευή","Σάββατο","Κυριακή")
```

terminal output

```
Δευτέρα Τρίτη Τετάρτη Πέμπτη Παρασκευή Σάββατο Κυριακή
```

- Όταν έχουμε πολλαπλά μηνύματα μέσα στην ίδια print( ) τότε μπορούμε να χρησιμοποιήσουμε το (+) σύμβολο για να ενώσουμε τα μηνύματα και επηρεάσουμε το τρόπο που εμφανίζεται το τελικό αποτέλεσμα. Για παράδειγμα, να μεγαλώσουμε το κενό που υπάρχει ανάμεσα στα μηνύματα ή να προσθέσουμε ένα καινούργιο σύμβολο. Όταν έχουμε String μηνύματα και ανάμεσα τους υπάρχει το (+) σύμβολο, η Python δεν εκτελεί την πράξη της πρόσθεσης (άλλωστε δεν είναι δυνατόν αυτό) αλλά εκτελεί συνένωση των μηνυμάτων.

```
a="Δευτέρα"
b="Τρίτη"
c="Τετάρτη"
d="Πέμπτη"
e="Παρασκευή"
f="Σάββατο"
g="Κυριακή"
result=a+b+c+d+e+f+g
print(result)
```

terminal output → ΔευτέραΤρίτηΤετάρτηΠέμπτηΠαρασκευήΣάββατοΚυριακή

- Για να υπάρχει μια οριζόντια παύλα ανάμεσα στα μηνύματα, προσθέτουμε ένα ="-"

```
result=a+"-"+b+"-"+c+"-"+d+"-"+e+"-"+f+"-"+g
print(result)
```

terminal output ↓

```
Δευτέρα-Τρίτη-Τετάρτη-Πέμπτη-Παρασκευή-Σάββατο-Κυριακή
```

- Εκτός από το σύμβολο της συνένωσης (+), μπορούμε να χρησιμοποιήσουμε το (*) σύμβολο για να εμφανίσουμε το μήνυμα τόσες φορές όσες δηλώσαμε μετά τον τελεστή (*).

```
a="Δευτέρα"
print(a*3)
```

terminal output

ΔευτέραΔευτέραΔευτέρα

- Αν θέλαμε να αφήσουμε και ένα κενό μεταξύ των λέξεων αφήνουμε ένα κενό όταν ορίζουμε το a.

```
a="Δευτέρα "
print(a*3)
```

terminal output

Δευτέρα Δευτέρα Δευτέρα

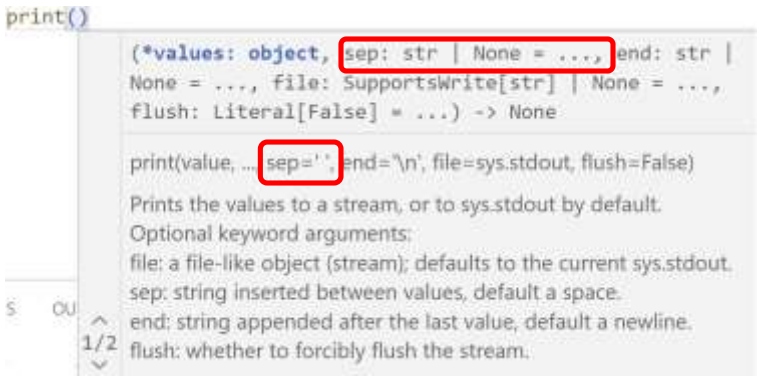
- Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε το σύμβολο της συνένωσης (+) με το κενό " "

```
a="Δευτέρα"
print((a+" ")*3)
```

terminal output

Δευτέρα Δευτέρα Δευτέρα

- Επόμενο argument στη συνάρτηση print() είναι η sep, που καθορίζει πως χωρίζονται τα μηνύματα μεταξύ τους.



Η προκαθορισμένη συμπεριφορά, όταν έχουμε πολλαπλά strings, είναι να έχουν ένα κενό ανάμεσα τους. Αυτή είναι και η default τιμή του sep. Αλλάζοντας την τιμή του sep, αλλάζουμε και τον τρόπο που χωρίζονται τα strings ανάμεσα τους.

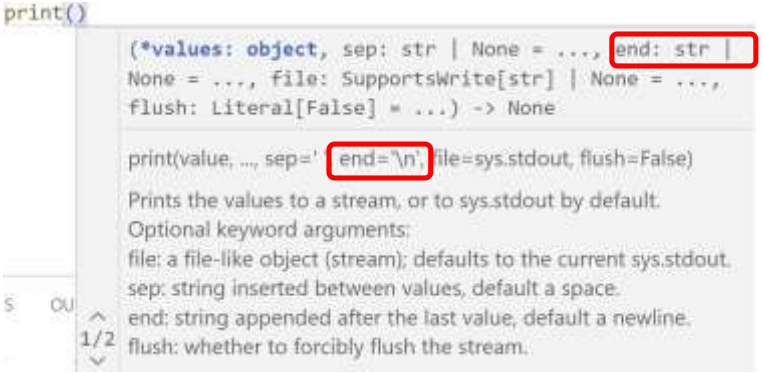
➢ Παράδειγμα αντικατάστασης του κενού με το σύμβολο *

```
print("Δευτέρα","Τρίτη","Τετάρτη","Πέμπτη","Παρασκευή","Σάββατο","Κυριακή",sep="*")
```

terminal output

```
Δευτέρα*Τρίτη*Τετάρτη*Πέμπτη*Παρασκευή*Σάββατο*Κυριακή
```

- Επόμενο argument στη συνάρτηση print() είναι η end, που καθορίζει τον τρόπο που ολοκληρώνεται η εντολή. Η προκαθορισμένη (default) τιμή του end είναι ο χαρακτήρας νέας γραμμής (\n) δηλαδή μετά το τέλος εκτέλεσης του print( ) ο κέρσορας μεταφέρεται στην αμέσως επόμενη γραμμή.



Αν δεν επιθυμούμε αυτή την προκαθορισμένη συμπεριφορά, μπορούμε να ορίσουμε μια καινούργια τιμή για την end argument η οποία θα αντικαταστήσει την προκαθορισμένη τιμή του end.

- Παράδειγμα ορισμού σαν τιμή του end ενός κενού διαστήματος

```
print("Δευτέρα", end=' ')
print("Τρίτη", end=' ')
print("Τετάρτη", end=' ')
print("Πέμπτη", end=' ')
print("Παρασκευή", end=' ')
print("Σάββατο", end=' ')
print("Κυριακή", end=' ')
```

terminal output

```
Δευτέρα Τρίτη Τετάρτη Πέμπτη Παρασκευή Σάββατο Κυριακή
```

- Παράδειγμα χρήσης των arguments sep και end

default values

```
print('01','10','2022', sep='/', end='\n')
print('Προσωπικό','email:','chprassa',sep=' ', end='@')
print('gmail.com')
```

terminal output

```
01/10/2022
Προσωπικό email: chprassa@gmail.com
```

- Βέβαια, ο ορισμός των arguments βάσει των προκαθορισμένων (default) τιμών μπορεί να παραληφθεί χωρίς να επηρεαστεί το αποτέλεσμα στο τερματικό της Python

```
print('01','10','2022', sep='/')
print('Προσωπικό','email:','chprassa', end='@')
print('gmail.com')
```

terminal output

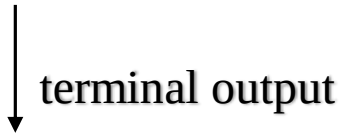
```
01/10/2022
Προσωπικό email: chprassa@gmail.com
```

- Μπορούμε επίσης να καθορίσουμε το πως εμφανίζονται τα μηνύματα στο terminal, χρησιμοποιώντας χαρακτήρες διαφυγής. Οι χαρακτήρες διαφυγής είναι ειδικοί χαρακτήρες που δεν εμφανίζονται στο output, αρχίζουν με το backslash σύμβολο (\ ) και μετά ένα συγκεκριμένο χαρακτήρα από την λίστα που δείχνει ο παρακάτω πίνακας:

\n	Αλλάζει τη γραμμή τύπωσης του μηνύματος
\t	Αφήνει ένα στηλοθέτη (tab) διάστημα
\'	Εκτυπώνει μια μονή απόστροφο
\"	Εκτυπώνει μια διπλή απόστροφο
\\	Εκτυπώνει ένα backslash σύμβολο
\s	Εκτυπώνει ένα κενό χαρακτήρα

- Παράδειγμα χρήσης των χαρακτήρων διαφυγής για να αλλάξουμε τον τρόπο τύπωσης του μηνύματος στο terminal.

```
print("Η διαδρομή του αρχείου είναι η ακόλουθη:\n\"C:\\Users\\Documents\\Python\"")
```




```
Η διαδρομή του αρχείου είναι η ακόλουθη:
"C:\Users\Documents\Python"
```

```
Annual_income = 25000
Monthly_income = Annual_income / 12
print("Το μηνιαίο εισόδημα είναι", Monthly_income)
```

χωρίς μορφοποίηση

terminal output

```
Το μηνιαίο εισόδημα είναι 2083.3333333333335
```

➢ Η function format( ) μας δίνει τη δυνατότητα να μορφοποιήσουμε ένα αριθμητικό αποτέλεσμα με βάση τον ορισμό που θα της δώσουμε αφού περάσουμε σε αυτήν δύο παραμέτρους:

1. Τον αριθμό που θέλουμε να μορφοποιήσουμε
2. Τα επιθυμητά ψηφία που θέλουμε να εμφανίσουμε

```
print("Το μηνιαίο εισόδημα είναι", format(Monthly_income, '.2f'))
```

terminal output

αριθμός των ψηφίων μετά την υποδιαστολή

μορφοποίηση float αριθμού

```
Το μηνιαίο εισόδημα είναι 2083.33
```

```
print("Το μηνιαίο εισόδημα είναι", format(Monthly_income, ',.2f'))
```

terminal output

κόμμα στις χιλιάδες

```
Το μηνιαίο εισόδημα είναι 2,083.33
```

```
print("Το μηνιαίο εισόδημα είναι \u20AC", format(Monthly_income, ',.2f'), sep='')
```

terminal output

€

```
Το μηνιαίο εισόδημα είναι €2,083.33
```

➢ Μπορούμε επίσης αντί για f να χρησιμοποιήσουμε το σύμβολο % έτσι ώστε να εμφανίσουμε το τελικό αποτέλεσμα σαν ποσοστό.

```
print(format(0.5, '.0%'))
```

terminal output

μορφοποίηση ποσοστού

```
50%
```

➢ Τέλος, υπάρχει και η d επιλογή με την οποία ενημερώνουμε την format( ) function ότι αντί για float θα χρησιμοποιήσουμε integer αριθμούς. Στους ακέραιους αυτή είναι η μόνη επιλογή χωρίς να ορίζουμε αριθμό ψηφίων.

```
print(format(1000, ',d'))
```

terminal output

μορφοποίηση ακεραίου

```
1,000
```

- Η πιο συνήθης μορφοποίηση των strings αφορά στην **ενοποίηση/συνένωση (concatenation)** δύο ή περισσότερων strings με σκοπό τη δημιουργία ενός μεγαλύτερου string. Το σύμβολο που χρησιμοποιείται για αυτό το σκοπό είναι το (+).

```
University_name="UC"
Location="Berkeley"
Full_name=University_name+" "+Location
print(Full_name)
```

terminal output

εμφάνιση κενού ανάμεσα στα δύο strings

```
UC Berkeley
```

- Ο πιο απλός τρόπος να εισάγουμε μια μεταβλητή μέσα σε ένα string είναι να χρησιμοποιήσουμε f-strings. Ο τρόπος αυτός είναι πιο εύκολος από ότι η τεχνική του concatenation που πρέπει να προσθέτουμε το + σύμβολο μετά από κάθε όρο.

```
University_name="UC"
Location="Berkeley"
print(f"The complete university name is {University_name} {Location}")
```

terminal output

```
The complete university name is UC Berkeley
```

- *Τα f-strings έχουν εφαρμογή και στην εισαγωγή αριθμών μέσα σε strings.*

Παράδειγμα:

```
username='Chara'
days_ago=2
print('Welcome back '+username+' ('+days_ago+' days ago since your last visit)')
```

terminal output

TypeError: can only concatenate str (not "int") to str

✓ Το λάθος δημιουργείται από την παρουσία του ακεραίου, καθώς η πράξη της συνένωσης αφορά σε ένωση αλφαριθμητικών. Τρόποι επίλυσης του προβλήματος:

1. Μετατροπή του integer σε string και επανεκτέλεση της print()

```
print('Welcome back '+username+' ('+str(days_ago)+' days ago since your last visit)')
```

terminal output

Welcome back Chara (2 days ago since your last visit)

2. Ισοδύναμα μπορούμε να χρησιμοποιήσουμε τη μέθοδο .format(). Τα ορίσματα της format θα αντικαταστήσουν μία-προς-μία τις αγκύλες μέσα στην έκφραση στα ' '.

```
print('Welcome back  {} ({} days ago since your last visit)'.format(username,days_ago))
```

terminal output

Welcome back Chara (2 days ago since your last visit)

- Εναλλακτικός τρόπος:

```
username='Chara'
days_ago=2
print(f'Welcome back {username} ({days_ago} days ago since your last visit)')
```

terminal output

Welcome back Chara (2 days ago since your last visit)

Παράδειγμα: Να δημιουργήσετε ένα πρόγραμμα το οποίο να εκτυπώνει τα ονόματα των φοιτητών και δίπλα την ηλικία τους και το μέσο όρο των μαθημάτων τους.

```
student1='Giorgos'
student2='Anna'
student1_age=19
student2_age=20
student1_average=8.6523
student2_average=7.1834
print("{} {} {}".format("Name","Age","Average_grade"))
print("{} {} {}".format(student1,student1_age,student1_average))
print("{} {} {}".format(student2,student2_age,student2_average))
```

terminal output

```
Name Age Average_grade
Giorgos 19 8.6523
Anna 20 7.1834
```

- Για να έχουμε καλύτερη στοίχιση μπορούμε να αλλάξουμε τις θέσεις στις οποίες εκτυπώνεται κάθε μεταβλητή ως εξής:

```
print("{:12} {:^12} {:12}".format("Name","Age","Average_grade"))
print("{:12} {:^12d} {:12f}".format(student1,student1_age,student1_average))
print("{:12} {:^12d} {:12f}".format(student2,student2_age,student2_average))
```

12 θέσεις – στοίχιση στη μέση ακέραιος αριθμός κινητής υποδιαστολής

Για στοίχιση στα αριστερά <
Για στοίχιση στα δεξιά >

- String Index: Για κάθε χαρακτήρα ενός string, η Python αναθέτει έναν αριθμό ξεκινώντας την αρίθμηση από το 0.

P y t h o n
[0] [1] [2] [3] [4] [5]

- Για να διαβάσουμε τον χαρακτήρα που αντιστοιχεί σε μια συγκεκριμένη index θέση, γράφουμε το όνομα της string μεταβλητής και αμέσως μετά μέσα σε square brackets ([ ]) τον αριθμό του index. Αν γράψουμε αρνητικό αριθμό τότε διαβάζουμε το string από το τέλος. Για παράδειγμα:

```
>>> name="Python"
>>> print(name[0])
P
>>> print(name[-1])
n
```

❖ String Slicing: Αν και η ιδιότητα του index είναι πολύ χρήσιμη, συνήθως δεν ζητάμε ένα μεμονωμένο χαρακτήρα αλλά ένα υποσύνολο του string.

```
>>> name="Python"
>>> print(name[0:3])
Pyt
```

- Ο πρώτος αριθμός δηλώνει το index από το οποίο θα αρχίζουμε να διαβάζουμε το string, ενώ ο δεύτερος αριθμός δηλώνει σε ποιο index θα σταματήσουμε. *Τον χαρακτήρα στο index στο οποίο σταματάμε δεν το λαμβάνουμε υπόψη στο τελικό αποτέλεσμα*. Άρα η εξήγηση στο παραπάνω παράδειγμα είναι: ξεκίνησε να διαβάζεις από το index 0 και σταμάτησε στο index 3 χωρίς όμως να διαβάσεις το στοιχείο στο index 3.

- Υπάρχει και μια τρίτη παράμετρος που είναι το *βήμα (step)* και δηλώνει ανά πόσους χαρακτήρες θα διαβάζουμε. Εάν δεν το δηλώσουμε, όπως κάναμε στο προηγούμενο παράδειγμα, τότε εξ ορισμού διαβάζονται όλοι οι χαρακτήρες από την αρχή μέχρι το index που έχει δηλωθεί γιατί η αύξηση του index αριθμού γίνεται κατά έναν αριθμό. Αν, για παράδειγμα, δηλώσουμε τον αριθμό 2 για step, τότε θα διαβάζονται οι χαρακτήρες που αντιστοιχούν ανά δύο index αριθμούς.

```
>>> print(name[0:3:2])
Pt
```

Έτοιμες συναρτήσεις για συμβολοσειρές (strings)

- len( ): επιστρέφει το πλήθος των χαρακτήρων

```
>>> name="python tutorial"
>>> print(len(name))
15
```

- lower( ): μετατρέπει όλους τους χαρακτήρες ενός string σε πεζούς

```
>>> name="Python Tutorial"
>>> print(name.lower())
puthon tutorial
```

- upper(): μετατρέπει όλους τους χαρακτήρες ενός string σε κεφαλαίους

```
>>> name="python tutorial"
>>> print(name.upper())
PYTHON TUTORIAL
```

- swapcase(): μετατρέπει τους κεφαλαίους χαρακτήρες ενός string σε πεζούς και το αντίστροφο

```
>>> name="Python Tutorial"
>>> print(name.swapcase())
pYTHON tUTORIAL
```

- count(): επιστρέφει το πλήθος των φορών που εμφανίζεται ένας χαρακτήρας σε ένα string. Κάνει διάκριση πεζών και κεφαλαίων.

```
>>> print(name.count("o"))
2
```

❖ title( ): μετατρέπει το πρώτο γράμμα από κάθε λέξη σε κεφαλαίο.

```
>>> name="python tutorial"
>>> print(name.title())
Python Tutorial
```

❖ replace( ): αντικαθιστά χαρακτήρες μέσα σε ένα string. Δέχεται δύο παραμέτρους – η πρώτη παράμετρος είναι ο χαρακτήρας που θα αντικαταστήσουμε και η δεύτερη παράμετρος είναι ο καινούργιος χαρακτήρας που επιθυμούμε να εισάγουμε στο string ως αντικατάσταση.

```
>>> print(name.replace("y","u"))
puthon tutorial
```

❖ find(): επιστρέφει έναν αριθμό ο οποίος αντιπροσωπεύει το index από τον οποίο αρχίζει ο χαρακτήρας ή η λέξη σε ένα string.

```
>>> print(name.find("tutorial"))
7
>>> print(name.find("t"))
2
```

❖ strip(): αφαιρεί τα κενά από την αρχή και το τέλος ενός string.

```
>>> name=" python tutorial "
>>> print(name.strip())
python tutorial
```

Παράδειγμα: Έστω ότι έχω ένα πρόγραμμα το οποίο ελέγχει τα συνθηματικά username και password ενός χρήστη και τον εισάγει ή όχι στο σύστημα.

```
username='safename'
password='safepass'
given_name='  safename '
given_password='safepass   '
print((given_name==username) and (given_password==password))
```

Output

False

- Το αποτέλεσμα είναι False εξαιτίας των κενών, χρήση της .strip()

```
username='safename'
password='safepass'
given_name='  safename '
given_password='safepass   '
print((given_name.strip()==username) and (given_password.strip()==password))
```

Output

True

❖ Για να βρείτε από μόνοι σας όλες αυτές τις μεθόδους και ακόμα περισσότερες που δεν έχουμε αναφέρει μπορείτε να γράψετε το όνομα της μεταβλητής που περιέχει μια string τιμή και να γράψετε την τελεία (.). Αμέσως θα εμφανιστεί μια λίστα με όλα τα functions που μπορείτε να εφαρμόσετε στο συγκεκριμένο string. Αν η λίστα δεν εμφανιστεί μπορείτε να πατήσετε Ctrl + Space Bar για να την εμφανίσετε.

❖ Επίσης για κάθε ένα από τα functions υπάρχει μια γρήγορη επεξήγηση της λειτουργίας τους όπως και για τις παραμέτρους που πρέπει να χρησιμοποιήσουμε για να εκτελεστούν σωστά.