6. ΣΥΝΑΡΤΗΣΕΙΣ – ΕΜΒΕΛΕΙΑ – ΠΑΡΑΜΕΤΡΟΙ - MODULES

6.1 Τι είναι οι συναρτήσεις

Οι συναρτήσεις είναι απλώς έτοιμα τμήματα κώδικα τα οποία εκτελούν μια συγκεκριμένη εργασία. Σαν παράδειγμα μπορούμε να θυμηθούμε τις μαθηματικές συναρτήσεις που είναι διαθέσιμες στο MS Excel. Για πρόσθεση αριθμών, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση sum() και να πληκτρολογήσουμε sum(A1:A5) αντί για A1+A2+A3+A4+A5.

Ανάλογα με το πώς γράφεται μια συνάρτηση, αν είναι μέρος μιας κλάσης (η κλάση είναι μια έννοια στον αντικειμενοστραφή προγραμματισμό την οποία θα καλύψουμε αργότερα) και πώς την εισάγουμε, μπορούμε να καλέσουμε μια συνάρτηση απλά πληκτρολογώντας το όνομα της συνάρτησης ή χρησιμοποιώντας τον συμβολισμό με την τελεία «.» (dot notation).

Επίσης, κάποιες συναρτήσεις απαιτούν από εμάς να τους δώσουμε δεδομένα σαν είσοδο, για να εκτελούν τις εργασίες τους. Αυτά τα δεδομένα είναι γνωστά ως ορίσματα και τα περνάμε στη συνάρτηση περικλείοντας τα σε παρενθέσεις ( ) διαχωρισμένα με κόμμα.

Για παράδειγμα, για να χρησιμοποιήσουμε τη συνάρτηση print() για την εμφάνιση κειμένου στην οθόνη, την καλούμε πληκτρολογώντας print("Γειά σου κόσμε") όπου print είναι το όνομα της συνάρτησης και "Γειά σου κόσμε " είναι το όρισμα.

Χρησιμοποιώντας τον συμβολισμό με τελεία (dot notation), για να καλέσουμε τη συνάρτηση replace() για τη διαχείριση συμβολοσειρών κειμένου, πρέπει να πληκτρολογήσουμε

newString = "Γειά σου κόσμε".replace("κόσμε", "σύμπαν")

όπου “replace” είναι το όνομα της συνάρτησης και τα "κόσμε" και "σύμπαν" τα ορίσματα. Η συμβολοσειρά πριν από την τελεία (δηλαδή η "Γειά σου κόσμε") είναι η συμβολοσειρά που θα επηρεαστεί.

Ως εκ τούτου, το «Γειά σου κόσμε» θα αλλάξει σε "Γειά σου σύμπαν".

Ορισμένες λειτουργίες ενδέχεται να εμφανίσουν ένα αποτέλεσμα μετά την εκτέλεση των εργασιών τους. Σε αυτήν την περίπτωση,

η συνάρτηση replace() επιστρέφει τη συμβολοσειρά "Γειά σου σύμπαν", την οποία εμείς στη συνέχεια αντιστοιχίζουμε στο newString. Εάν εκτυπώσουμε το newString πληκτρολογώντας:

print(newString)

θα πάρουμε

Γεια σου Σύμπαν

6.2 Δημιουργία δικών μας συναρτήσεων

Μπορούμε να ορίσουμε τις δικές μας συναρτήσεις στην Python και να τις χρησιμοποιήσουμε ξανά και ξανά μέσα σε ένα πρόγραμμα.

Η πρώτη γραμμή του ορισμού μιας συνάρτησης είναι η επικεφαλίδα της συνάρτησης. Η επικεφαλίδα ξεκινάει με τη δεσμευμένη λέξη def ακολουθούμενη από το όνομα της συνάρτησης, ένα ζευγάρι παρενθέσεων που προαιρετικά περικλείει μια διαχωριζόμενη με κόμματα λίστα παραμέτρων (parameters) και τελειώνει με μια άνω κάτω τελεία. Κάτω από την επικεφαλίδα ακολουθεί το σώμα της συνάρτησης (οι εντολές της συνάρτησης) σε εσοχή.

Τα ονόματα των συναρτήσεων δίνονται με βάση τους ίδιους κανόνες που είδαμε και για τα ονόματα των μεταβλητών. Καλό είναι να δίνουμε ονόματα σχετικά με τη λειτουργία που επιτελούν οι συναρτήσεις. Οι συναρτήσεις ορίζονται συνήθως στην αρχή των προγραμμάτων. Είναι

προφανές όμως ότι μια συνάρτηση πρέπει να οριστεί πρώτα πριν χρησιμοποιηθεί (κληθεί). Κατά την κλήση μιας συνάρτησης εκτελούνται οι εντολές που περιέχονται στο σώμα της.

Μπορούμε να καλούμε μία συνάρτηση όσες φορές απαιτούνται για τη λύση του προβλήματος που αντιμετωπίζει το πρόγραμμά μας.

Η σύνταξη λοιπόν για τον ορισμό μιας συνάρτησης είναι η εξής:

def όνομαΣυνάρτησης(λίστα παραμέτρων):

κώδικας που περιγράφει τι πρέπει να κάνει η συνάρτηση

return [έκφραση]

Υπάρχουν δύο λέξεις-κλειδιά εδώ, def και return.

Το def λέει στο πρόγραμμα ότι ο κώδικας με εσοχή από την επόμενη γραμμή και μετά είναι μέρος της συνάρτησης. Return είναι μια λέξη-κλειδί που χρησιμοποιούμε για να επιστρέψουμε μια απάντηση (έξοδο)

από τη συνάρτηση. Μπορεί να υπάρχουν περισσότερες από μία δηλώσεις επιστροφής σε μια συνάρτηση.

Ωστόσο, μόλις η συνάρτηση εκτελέσει μια εντολή επιστροφής, η συνάρτηση θα ολοκληρωθεί και ο έλεγχος του κώδικα θα προχωρήσει. Εάν η συνάρτηση δεν χρειάζεται να επιστρέψει καμία τιμή, μπορούμε να παραλείψουμε τη δήλωση επιστροφής.

Εναλλακτικά, μπορούμε να γράψουμε “return” ή “return none”

Ας γράψουμε τώρα την πρώτη μας συνάρτηση.

Ας υποθέσουμε ότι θέλουμε να προσδιορίσουμε εάν ένας αριθμός είναι πρώτος.

Μπορούμε να ορίσουμε τη συνάρτηση χρησιμοποιώντας το

τελεστή modulus (%), τον βρόχο for και μια δήλωση if:

```
def def_protos_ar (ar_gia_elegxo):
  for x in range(2, ar_gia_elegxo):
    if (ar_gia_elegxo % x == 0):
      return False
    return True
```

Η παραπάνω συνάρτηση έχει μια παράμετρο που ονομάζεται ***ar_gia_elegxo***.

Οι παράμετροι είναι μεταβλητές που χρησιμοποιούνται για την αποθήκευση των ορισμάτων που μεταβιβάζουμε στη συνάρτηση.

Οι γραμμές 2 και 3 χρησιμοποιούν έναν βρόχο for για να διαιρέσουν την παράμετρο ***ar_gia_elegxo*** με όλους τους αριθμούς από το 2 έως το ***ar_gia_elegxo - 1*** για να προσδιορίσουμε αν το υπόλοιπο είναι ίσο με

μηδέν. Εάν το υπόλοιπο είναι μηδέν, το ***ar_gia_elegxo*** δεν είναι πρώτος αριθμός.

Η γραμμή 4 θα επιστρέψει ***False*** και η συνάρτηση θα βγει.

Εάν με την τελευταία επανάληψη του βρόχου ***for***, καμία από τη διαίρεση δεν δίνει υπόλοιπο ίσο με μηδέν, η συνάρτηση θα φτάσει στη γραμμή 5 και θα επιστρέψει ***True***. Κατόπιν θα βγει.

Για να χρησιμοποιήσουμε αυτή τη συνάρτηση, πληκτρολογούμε ***def_protos_ar(13)*** και την εκχωρούμε σε μια μεταβλητή όπως παρακάτω:

apantisi = def_protos_ar (13)

Εδώ περνάμε το 13 ως όρισμα, το οποίο θα αποθηκευτεί στην ***def_protos_ar***.

Στη συνέχεια, ο βρόχος for εκτελείται για να ελέγξει αν είναι η ***def_protos_ar*** πρώτος αριθμός και επιστρέφει είτε ***True*** είτε ***False***.

Μπορούμε να εκτυπώσουμε την απάντηση πληκτρολογώντας

print(apantisi).

Όλο το προγραμματάκι μας θα δείχνει κάπως έτσι:

```
def def_protos_ar (ar_gia_elegxo):
   for x in range(2, ar_gia_elegxo):
      if (ar_gia_elegxo % x == 0):
         return False
   return True
apantisi = def_protos_ar(137)
print(apantisi)
```

Θα πάρουμε το αποτέλεσμα: ***True***.

Μία συνάρτηση μπορεί να δεχθεί παραμέτρους (formal parameters ή απλά parameters), οι οποίες βοηθούν να δοθούν τιμές στη συνάρτηση ως είσοδος, έτσι ώστε αυτή να μπορεί να κάνει κάτι αξιοποιώντας αυτές τις τιμές. Οι παράμετροι μοιάζουν με τις μεταβλητές και οι τιμές τους ορίζονται, όταν καλούμε μία συνάρτηση. Οι παράμετροι υπάρχουν μόνο κατά τη διάρκεια εκτέλεσης της συνάρτησης. Κατά την κλήση μιας συνάρτησης οι τιμές που παίρνουν οι παράμετροι ονομάζονται ορίσματα (actual arguments ή απλά arguments).

Με απλά λόγια, για να μην μπερδευόμαστε, στον ορισμό μιας συνάρτησης έχουμε παραμέτρους (ονομασίες μεταβλητών) και στην κλήση μιας συνάρτησης έχουμε ορίσματα (τιμές ή μεταβλητές στις οποίες έχουν εκχωρηθεί τιμές).

Ας δούμε ένα παράδειγμα στο οποίο ορίζουμε και καλούμε μία συνάρτηση η οποία συγκρίνει δύο αριθμούς:

```
def compare(a,b): # Ορισμός της συνάρτησης compare με παραμέτρους a, b
   if a > b:
      print(a, '>', b)
   elif a< b:
      print(a, '<', b)
```

```
    else:
        print(a, '=', b)

compare(7,9) # κλήση συνάρτησης με ορίσματα απευθείας τιμές
x = 12
y = 5
compare(x,y) # κλήση συνάρτησης με ορίσματα μεταβλητές
compare(x,12)# κλήση συνάρτησης με ορίσματα μεταβλητή και τιμή
```

Άσκηση

Τι θα τυπωθεί και γιατί όταν τρέξει το παρακάτω προγραμματάκι;

```
def set10(x):
    x - 10

y = 0
set10(y)
print(y)
```

Μία συνάρτηση μπορεί να επιστρέφει και τιμή, η οποία μπορεί να

εκχωρηθεί σε μία μεταβλητή ή ακόμη και να χρησιμοποιηθεί ως μέρος μιας έκφρασης. Η επιστροφή τιμής γίνεται με την εντολή return η οποία σημαίνει:

«επίστρεψε αμέσως τον έλεγχο ροής από τη συνάρτηση στο σημείο του προγράμματος που αυτή κλήθηκε και χρησιμοποίησε την τιμή της έκφρασης που ακολουθεί ως επιστρεφόμενη τιμή».

Παραδείγματα:

Φτιάξτε μια συνάρτηση που δέχεται ως όρισμα έναν αριθμό και επιστρέφει την απόλυτη τιμή του:

1.

```
def absValue (x):
   if x < 0:
      return -x
   else:
      return x


print(absValue(4))
print(absValue(-4))
```

Φτιάξτε μια συνάρτηση που δέχεται ως ορίσματα δύο αριθμούς ή δύο συμβολοσειρές και επιστρέφει το άθροισμα ή τη συνένωσή τους αντίστοιχα:

2.

```
def add(x,y):
   return x + y


print(add(9,7))
print(add("Γειά σου ", "κόσμε"))
```

Τι θα γίνει αν αλλάξουμε τον κώδικα αφαιρώντας την εντολή return, ώστε η συνάρτησή μας να μην επιστρέφει τιμή; Τι θα γίνει, αν προσπαθήσουμε να εκχωρήσουμε την κλήση της συνάρτησης σε μία μεταβλητή;

Για να δούμε:

3.

```
def add2(x,y):
    print(x + y)


add2(9,7)      # Θα τυπωθεί 16
sum = add2(9,7)# Θα τυπωθεί 16
```

6.3 Εμβέλεια μεταβλητών

Μια σημαντική έννοια που πρέπει να κατανοήσουμε κατά τον ορισμό μιας συνάρτησης είναι η έννοια της εμβέλειας των μεταβλητών.

Οι μεταβλητές που ορίζονται μέσα σε μια συνάρτηση αντιμετωπίζονται διαφορετικά από τις μεταβλητές που ορίζονται έξω.

Υπάρχουν δύο βασικές διαφορές.

Πρώτον, οποιαδήποτε μεταβλητή δηλώνεται μέσα σε μια συνάρτηση είναι προσβάσιμη μόνο εντός της συνάρτησης.

Αυτές είναι γνωστές και ως τοπικές μεταβλητές.

Οποιαδήποτε μεταβλητή δηλώνεται εκτός μιας συνάρτησης είναι γνωστή ως καθολική μεταβλητή και είναι προσβάσιμη απ’ οπουδήποτε στο πρόγραμμα.

Για να το καταλάβουμε, ας δούμε τον παρακάτω κώδικα:

```
gloVar = "Καθολική μεταβλητή"
def myFunction():
    print("\nΜΕΣΑ ΣΤΗ ΣΥΝΑΡΤΗΣΗ")
    #Οι καθολικές μεταβλητές ειναι ορατές μέσα σε μια συνάρτηση
    print (gloVar)
    #Δήλωση τοπικής μεταβλητής
    locVar = "Τοπική μεταβλητή"
    print(locVar)
'''
Καλούμε τη συνάρτηση παρακάτω
Παρατηρήστε ότι η myFunction() δεν έχει παραμέτρους.
Έτσι την καλούμε με κενές παρενθέσεις.
'''
myFunction()

print("\nΕΞΩ ΑΠΟ ΤΗ ΣΥΝΑΡΤΗΣΗ")

#Οι καθολικές μεταβλητές ειναι ορατές έξω από μια συνάρτηση
print(gloVar)

# Οι τοπικές μεταβλητές ΔΕΝ ειναι ορατές έξω από μια συνάρτηση
# Έτσι, στο τέλος του προγράμματος θα πάρουμε μήνυμα λάθους:
# NameError: name 'locVar' is not defined. Did you mean: 'gloVar'?
print(locVar)
```

Τι γίνεται στην περίπτωση που η τοπική μεταβλητή έχει το ίδιο όνομα με την καθολική;

Ας δούμε ένα παράδειγμα:

```
onoma = "Ελένη"
def pesGeia():
    print("Γειά " + onoma)

def alloOnoma(alloOnoma):
    onoma = alloOnoma

pesGeia()
alloOnoma("Μαρία")
pesGeia()
```

Στο παραπάνω παράδειγμα, έχουμε τη μεταβλητή onoma σαν καθολική, αλλά και τοπική μεταβλητή. Όταν όμως καλείται η συνάρτηση alloOnoma με το όρισμα «Μαρία», τυπώνεται και πάλι «Ελένη», καθότι:

Όταν η τοπική μεταβλητή έχει το ίδιο όνομα με την καθολική, χρησιμοποιείται η καθολική μεταβλητή κάθε φορά που καλείται η συνάρτηση και όχι η τοπική της

για να πάρουμε την εκτύπωση που θέλουμε, δηλαδή να αλλάξουμε την τιμή της καθολικής μεταβλητής μέσα στη συνάρτηση, πρέπει να χρησιμοποιήσουμε τη λέξη κλειδί global, προσθέτοντας απλά στον κώδικά μας τη γραμμή «global onoma» όπως παρακάτω:

```
onoma = "Ελένη"
def pesGeia():
    print("Γειά " + onoma)
```

```
def alloOnoma(alloOnoma):
    global onoma
    onoma = alloOnoma

pesGeia()
alloOnoma("Μαρία")
pesGeia()
```

6.4 Συμβολοσειρές Τεκμηρίωσης (Doc strings)

Η τεκμηρίωση των συναρτήσεων (και κατ' επέκταση η τεκμηρίωση του κώδικά μας και με σχόλια) είναι μια καλή προγραμματιστική πρακτική, που εξασφαλίζει ο κώδικάς μας να είναι ευανάγνωστος και κατανοητός σε μας ή και σε άλλους οποτεδήποτε.

Για να τεκμηριώσουμε τον κώδικά μας, γράφουμε στην αρχή του κώδικα, μέσα σε τριπλά εισαγωγικά, τις βοηθητικές και περιγραφικές πληροφορίες για τη συνάρτησή μας.

Παράδειγμα:

```
"""
Αυτή η συνάρτηση δειχνει τη διαφορά
καθολικών και τοπικών μεταβλητών,
όταν τις καλούμε
"""
```

```
onoma = "Ελένη"
def pesGeia():
    print("Γειά " + onoma)

def alloOnoma(alloOnoma):
    global onoma
    onoma = alloOnoma

pesGeia()
alloOnoma("Μαρία")
pesGeia()
```

Τώρα, για να δούμε την τεκμηρίωση, πρώτα την εισάγουμε (όπως και τα modules που θα δούμε παρακάτω) και κατόπιν την καλούμε με την print(όνομα_αρχείου.__doc__), δηλαδή:

```
import glo_loc_var
print(glo_loc_var.__doc__)
```

και παίρνουμε:

```
Αυτή η συνάρτηση δείχνει τη διαφορά
καθολικών και τοπικών μεταβλητών,
όταν τις καλούμε
```

6.5 Προεπιλεγμένες παράμετροι – Default parameters

Η Python μας επιτρέπει να ορίσουμε προεπιλεγμένες τιμές για τις παραμέτρους μιας συνάρτησης. Αν μια παράμετρος έχει μια προεπιλεγμένη τιμή, δεν χρειάζεται να περάσουμε καμία τιμή για το

όρισμα κατά την κλήση της συνάρτησης. Για παράδειγμα, ας υποθέσουμε ότι μια συνάρτηση έχει 5 παραμέτρους a, b, c, d και e. Μπορούμε να ορίσουμε τη συνάρτηση ως

def someFunction(a, b, c=1, d=2, e=3):

print(a, b, c, d, e)

Εδώ, εκχωρούμε προεπιλεγμένες τιμές για τις τρεις τελευταίες παραμέτρους. Όλες οι παράμετροι με τις προεπιλεγμένες τιμές πρέπει να τοποθετηθούν στο τέλος της λίστας παραμέτρων. Δηλαδή, **ΔΕΝ** μπορούμε να ορίσουμε μια συνάρτηση ως εξής γιατί το r έρχεται μετά το q:

def someIncorrectFunction(p, q=1, r): # ΛΑΘΟΣ

print(p, q, r) ***# ΛΑΘΟΣ***

Για να καλέσουμε την someFunction(), μπορούμε να γράψουμε

someFunction(10, 20)

θα πάρουμε

10, 20, 1, 2, 3

Δεν χρειάζεται να περάσουμε καμία τιμή για τα c, d και e.

Αν γράψουμε

someFunction(10, 20, 30, 40)

θα πάρουμε

10, 20, 30, 40, 3

Τα δύο πρόσθετα ορίσματα που περνάμε στο (30 και το 40) αντιστοιχίζονται στις παραμέτρους με προεπιλεγμένες τιμές κατά σειρά. Ως εκ τούτου, το 30 αντικαθιστά την προεπιλεγμένη τιμή για το c ενώ το 40 αντικαθιστά το d.

1.

Παράδειγμα:

```
def greet(name, greeting = 'Γεια'):
    print(greeting, name)


greet('Γιώργος')
greet('Γιώργος', 'Αντίο')
```

θα μας δώσει:

Γεια Γιώργος

Αντίο Γιώργος

Ερώτηση: Μπορώ να βάλω πρώτα το greeting = «Γιώργος»;

2.

Παράδειγμα:

```
def func(a, b=1, c=2):
    print("a = ",  a, "και b = ", b, "και c = ", c)
```

func(5, 10)

func(2, c=20)

func(c=30, a=10)

θα πάρουμε:

a = 5 και b = 10 και c = 2

a = 2 και b = 1 και c = 20

a = 10 και b = 1 και c = 30

6.6 Μεταβλητό μέγεθος λίστας παραμέτρων

Εκτός από τις προεπιλεγμένες τιμές για τις παραμέτρους, η Python μας επιτρέπει επίσης να έχουμε μεταβλητό αριθμό ορισμάτων σε μια συνάρτηση.

Αυτό είναι πολύ χρήσιμο εάν εμείς δεν γνωρίζουμε εκ των προτέρων τον αριθμό των ορισμάτων που έχει μια συνάρτηση.

Για παράδειγμα, μπορεί να έχουμε μια συνάρτηση που προσθέτει μια σειρά αριθμών, αλλά δεν ξέρουμε πόσοι αριθμοί υπάρχουν εκ των προτέρων. Σε περιπτώσεις όπως αυτή, μπορούμε να χρησιμοποιήσουμε το σύμβολο «*».

Παράδειγμα:

Def addNumbers(*num):

sum = 0

for i in num:

sum = sum + i

print(sum)

Όταν προσθέτουμε έναν μόνο αστερίσκο μπροστά από το num, λέμε στον μεταγλωττιστή ότι το num αποθηκεύει μια λίστα ορισμάτων μεταβλητού μήκους που περιέχει πολλά στοιχεία.

Στη συνέχεια, η συνάρτηση «βλέπει» τα ορίσματα για να βρει το άθροισμα όλων των αριθμών και επιστρέφει την απάντηση.

Για να καλέσουμε τη συνάρτηση, γράφουμε

addNumbers(1, 2, 3, 4, 5)

θα πάρουμε ***15*** ως έξοδο.

Μπορούμε επίσης να προσθέσουμε περισσότερους αριθμούς γράφοντας:

addNumbers(1, 2, 3, 4, 5, 6, 7, 8)

θα πάρουμε ***36*** ως έξοδο.

Όπως μπορούμε να δούμε στα παραπάνω παραδείγματα, όταν προσθέτουμε έναν αστερίσκο μπροστά από μία παράμετρο, μπορούμε να περάσουμε έναν μεταβλητό αριθμό ορισμάτων στη συνάρτηση.

Αυτό είναι γνωστό ως λίστα ορισμάτων μεταβλητού μήκους **χωρίς λέξεις-κλειδιά**.

Αν θέλουμε να περάσουμε σε μια λίστα ορισμάτων μεταβλητού μήκους **με λέξεις-κλειδιά** στη συνάρτηση, μπορούμε να χρησιμοποιήσουμε διπλό αστερίσκο.

Παράδειγμα:

def printMemberAge(**age):

for i, j age.items():

print("Name = %s, Age = %s" %(i, j))

Αυτή η συνάρτηση έχει μια παράμετρο που ονομάζεται age (ηλικία). Οι διπλοί αστερίσκοι δηλώνουν ότι αυτή η παράμετρος αποθηκεύει μια λίστα ορισμάτων μεταβλητού μήκους με λέξεις-κλειδιά, η οποία είναι

ουσιαστικά ένα λεξικό.

Στη συνέχεια, ο βρόχος for, περνά μέσα από το όρισμα και εκτυπώνει τις τιμές.

Για να καλέσουμε τη συνάρτηση, γράφουμε:

printMemberAge(Peter = 5, John = 7)

θα πάρουμε:

Name = Peter, Age = 5

Name = John, Age = 7

Αν γράψουμε:

printMemberAge(Peter = 5, John = 7, Yvonne = 10)

θα πάρουμε:

Name = Peter, Age = 5

Name = John, Age = 7

Name = Yvonne, Age = 10

Εάν η συνάρτησή μας χρησιμοποιεί ένα κανονικό όρισμα (formal argument), μια λίστα ορισμάτων μεταβλητού μήκους χωρίς λέξεις-κλειδιά και μια λίστα ορισμάτων μεταβλητού μήκους με λέξεις-κλειδιά, πρέπει να ορίσουμε τη συνάρτηση χρησιμοποιώντας την ακόλουθη σειρά:

Def someFunction2 (farg, *args, **kwargs):

Δηλαδή, πρέπει πρώτα να έρθει το κανονικό όρισμα και μετά το όρισμα και το όρισμα χωρίς λέξεις-κλειδιά και μετά αυτό με λέξεις-κλειδιά,

με αυτή τη σειρά.

6.7 Εισαγωγή modules

Η Python συνοδεύεται από μεγάλο αριθμό ενσωματωμένων συναρτήσεων. Αυτές οι συναρτήσεις αποθηκεύονται σε αρχεία γνωστά ως modules. Για να χρησιμοποιήσουμε τον κώδικα που βρίσκεται στα modules, πρέπει πρώτα να τα εισάγουμε στα προγράμματά μας.

Αυτό το κάνουμε χρησιμοποιώντας τη λέξη-κλειδί import.

Υπάρχουν τρεις τρόποι:

Ο **πρώτος τρόπος** είναι να εισάγουμε ολόκληρο το module γράφοντας import onoma_module

Για παράδειγμα, για να εισαγάγουμε το ενσωματωμένο module random της Python, γράφουμε:

import random.

Για να χρησιμοποιήσουμε τη συνάρτηση randrange() που υπάρχει μέσα στο module της random γράφουμε:

random.randrange(1, 10).

Αν το θεωρούμε ενοχλητικό να γράφουμε "random" κάθε φορά που χρησιμοποιούμε τη συνάρτηση (**δεύτερος τρόπος**), μπορούμε να εισάγουμε το module γράφοντας import random as r (όπου r είναι οτιδήποτε θέλουμε).

Τώρα για να χρησιμοποιήσουμε τη συνάρτηση randrange(), απλά

γράφουμε ***r.randrange(1, 10)***.

Ο **τρίτος τρόπος** εισαγωγής modules είναι η εισαγωγή συγκεκριμένων συναρτήσεων από το module γράφοντας:

from moduleName import name1[, name2[, ... nameN]].

Για παράδειγμα, για να εισάγουμε τη συνάρτηση randrange() από το module της random γράφουμε:

from random import randrange.

Αν θέλουμε να εισάγουμε περισσότερες από μια συναρτήσεις, τις χωρίζουμε με κόμμα. Για να εισάγουμε την randrange()

και την randint(), γράφουμε:

from random import randrange, randint.

Για να χρησιμοποιήσουμε τη συνάρτηση τώρα, δεν χρειάζεται να χρησιμοποιήσουμε τον συμβολισμό με την τελεία (dot notation).

Απλώς γράφουμε:

randrange(1, 10)

Μπορούμε επίσης, να εισάγουμε και τις δικές μας συναρτήσεις σε ένα πρόγραμμα.

Ας ανοίξουμε ένα νέο αρχείο κι ας το ονομάσουμε ***usecheckifprime.py***. Το αποθηκεύμε και γράφουμε

import def_protos_ar

answer = def_protos_ar.checkIfPrime(13)

print(answer)

Τώρα αν εκτελέσουμε ***το usecheckifprime.py*** θα λάβουμε την έξοδο ***True***. Τόσο απλό.

Ωστόσο, ας υποθέσουμε ότι θέλουμε να αποθηκεύσουμε τα ***def_protos_ar.py*** και ***usecheckifprime.py*** σε διαφορετικούς φακέλους. Θα πρέπει να προσθέσουμε κάποιο κώδικα στο αρχείο usecheckifprime.py για να πει στον διερμηνέα της Python πού να βρει τη συνάρτηση.

Ας υποθέσουμε ότι δημιουργήσαμε έναν φάκελο με το όνομα «MyPythonModules» στο σκληρό μας δίσκο C για αποθήκευση του

def_protos_ar.py.

Πρέπει να προσθέσουμε τον παρακάτω κώδικα στο επάνω μέρος του

αρχείου ***usecheckifprime.py*** (πριν από την αρχική εισαγωγή γραμμής).

import sys

if 'C:\\MyPythonModules' not in sys.path:

sys.path.append('C:\\MyPythonModules')

Το sys.path αναφέρεται στη διαδρομή συστήματος της Python σας. Αυτή είναι η λίστα των φακέλων στην οποία ψάχνει η Python για να αναζητήσει modules και αρχεία. Ο παραπάνω κώδικας προσθέτει το φάκελο «***C:\MyPythonModules***» στη διαδρομή του συστήματός σας.

Τώρα μπορούμε να βάλουμε την ***def_protos_ar***.py στο ***C:\MyPythonModules*** και το ***usecheckifprime.py*** σε οποιοδήποτε φάκελο θέλουμε.

Ασκήσεις

1. Γράψτε μια συνάρτηση με το όνομα powerTo, η οποία όταν καλείται να ζητάει από τον χρήστη πρώτα έναν αριθμό (σαν βάση) και κατόπιν έναν άλλο για να τον υψώνει σε δύναμη. Τέλος, να τυπώνει το αποτέλεσμα.
2. Γράψτε μια συνάρτηση countVowels η οποία να δέχεται μια λέξη ή πρόταση στα Ελληνικά και να μετράει πόσα φωνήεντα έχει. Στο τέλος να τυπώνει τον αριθμό φωνηέντων
3. Γράψτε μια συνάρτηση findLargest, η οποία θα δέχεται σαν παράμετρο μια λίστα αριθμών και όταν τρέχει, θα τυπώνει τον μεγαλύτερο απ' αυτούς
4. Γράψτε μια συνάρτηση printEvens η οποία θα δέχεται σαν παράμετρο έναν φυσικό αριθμό και θα επιστρέφει όλους τους άρτιους από το 0 μέχρι τον αριθμό.
5. Γράψτε μια συνάρτηση filterEvens η οποία δέχεται μια λίστα αριθμών και τυπώνει στο τέλος μόνο τους ζυγούς