ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Γλώσσες Προγραμματισμού

Συναρτήσεις (Functions) - Python

Χαρά Πρασσά, chprassa@gmail.com

- Η συνάρτηση είναι ένα μέρος ομαδοποιημένου και επαναχρησιμοποιήσιμου κώδικα που εκτελεί κάποια λειτουργία. Μπορεί να χρησιμοποιηθεί σε οποιοδήποτε σημείο του υπόλοιπου κώδικα και καλείται παραπάνω από μία φορά.

Μία συνάρτηση πρέπει να ακολουθεί τους παρακάτω κανόνες :

1. Η συνάρτηση πρέπει να ξεκινάει με τη λέξη-κλειδί def, ακολουθούμενη από το όνομά της, παρενθέσεις και το σύμβολο της άνω και κάτω τελείας. Ανάμεσα στις παρενθέσεις μπορούν να οριστούν τυχόν παράμετροι εισόδου.
2. Η δήλωση επιστροφής (return) είναι προαιρετική. Χρησιμοποιείται για να επιστρέψει επιχειρήματα (arguments) στον καλούντα. Μία δήλωση return χωρίς επιχειρήματα είναι ίδια με μία δήλωση return None.

Σύνταξη συνάρτησης:

Ως **όνομα συνάρτησης** είναι ένα οποιοδήποτε έγκυρο όνομα για την Python όπως αναφέρθηκε στον ορισμό των μεταβλητών

```
def <Όνομα συνάρτησης>([Ορίσματα]):
    '''
    docstring
    '''
    <Εντολές συνάρτησης>
```

Εισαγωγή σχολίων συνάρτησης (προαιρετικό)

- Σε αντίθεση με άλλες γλώσσες προγραμματισμού, η def δεν ορίζει απλά μια συνάρτηση, αλλά δημιουργεί ένα αντικείμενο συνάρτησης που σημαίνει ότι αυτή μπορεί να βρίσκεται σε οποιοδήποτε σημείο του κώδικα και όχι μόνο στην αρχή του.

Παράδειγμα:

```
def welcome():
    '''
    Just a welcome function that asks for the name and prints
    a welcome message
    '''
    name=input("Give me your name: ")
    print(f"Welcome {name}.")

welcome()
```

← *κλήση συνάρτησης*

Output:

```
Give me your name: George
Welcome George
```

❖ *Προσοχή στη στοίχιση των εντολών που περιλαμβάνονται μέσα στη συνάρτηση*

- Όρισμα είναι μια τιμή από την κανονική ροή του προγράμματος την οποία θα χρησιμοποιήσει η συνάρτηση μέσα στο σώμα της. Το όρισμα δηλώνεται μέσα στην παρένθεση στο όνομα της συνάρτησης.

Παράδειγμα με 1 όρισμα:

```
def greeting(name):
    print(f"Hello {name}! Welcome to Python")

user_name=input("Give me your name: ")
greeting(user_name)
```

3. name = George

4. Hello George! Welcome to Python

1. user_name = "George"

2. greeting("George")

Output:

```
Give me your name: George
Hello George! Welcome to Python
```

Παράδειγμα με 2 ορίσματα:

```
def expo(first_number, second_number):
    print(f"Raising number {first_number} to {second_number} is
{first_number**second_number}")
expo(3,2)
```

Output:

```
Raising number 3 to 2 is 9
```

- Το όρισμα μιας συνάρτησης μπορεί να είναι οποιοδήποτε προγραμματιστικό αντικείμενο όπως πχ. αριθμός, αλφαριθμητικό, λίστα, λεξικό κλπ.

Παράδειγμα με όρισμα λίστα:

```
def expolist(data):
    for value in range(len(data)):
        data[value]**=2
    print(f"New list: {data}")

demolist=[1,2,3,4,5]
expolist(demolist)
```

Υψώνει όλα τα στοιχεία της λίστας στο 2

Output:

New list: [1, 4, 9, 16, 25]

- `print(f"Previous list: {demolist}")`

Output:

Previous list: [1, 4, 9, 16, 25] *Εμφανίζει τα νέα δεδομένα και όχι τα παλιά. Γιατί;*

- Όταν το προγραμματιστικό αντικείμενο είναι αμετάλλακτο όπως πχ. οι αριθμοί και τα αλφαριθμητικά, τότε δημιουργείται μέσα στη συνάρτηση ένα τοπικό αντίγραφο του προγραμματιστικού αντικειμένου. Όταν όμως το προγραμματιστικό αντικείμενο είναι μεταλλάξιμο, τότε οι αλλαγές γίνονται in place όπως έχουμε αναφέρει στις ιδιότητες των μεταλλάξιμων αντικειμένων. Στο παράδειγμα μας οι μεταβλητές demolist και data δείχνουν στην ίδια θέση μνήμης που βρίσκεται αποθηκευμένη η λίστα μας, οπότε όταν κάνουμε μια αλλαγή τότε και οι 2 μεταβλητές θα περιέχουν τις ίδιες τιμές.

```
def expolist(data):
    data_new=data.copy()
    for value in range(len(data_new)):
        data_new[value]**=2
    print(f"New list: {data_new}")

demolist=[1,2,3,4,5]
expolist(demolist)

print(f"Previous list: {demolist}")
```

Δημιουργείται αντίγραφο της λίστας

Output:

```
New list: [1, 4, 9, 16, 25]
Previous list: [1, 2, 3, 4, 5]
```

- Στα προηγούμενα παραδείγματα οι συναρτήσεις επέστρεφαν μέσω της print() ένα μήνυμα κειμένου. Αν θέλαμε η συνάρτηση να μην εκτυπώνει απλά ένα μήνυμα θα κάναμε χρήση της εντολής return. Η εντολή return επιστρέφει τις τιμές που ακολουθούν αυτή τη δήλωση πίσω στο πρόγραμμα.

Παράδειγμα με 1 επιστρεφόμενη τιμή:

```
def expo(first_number, second_number):
    return first_number**second_number

result=expo(2,3)
print(result)
print(expo(4,2))
```

Output:

```
8
16
```

Παράδειγμα με 2 επιστρεφόμενες τιμές:

```
def expo(first_number, second_number):
    exp=first_number**second_number
    sumnum=first_number+second_number
    return exp, sumnum

result1, result2=expo(2,3)
print(f"Expo: {result1} - Sum: {result2}")
```

Output:

```
Expo: 8 - Sum: 5
```

Παράδειγμα με εξαίρεση:

```
def division(number1, number2):
    return number1/number2

print(division(10,2))
print(division(6,0))
```

Output:

```
5.0
in division(number1, number2)
      1 def division(number1, number2):
----> 2     return number1/number2

ZeroDivisionError: division by zero
```

- Με χρήση της if θα δημιουργήσουμε μια εξαίρεση ώστε να μην σταματάει η εκτέλεση του προγράμματος

```
def division(number1, number2):
    if number2==0:
        return 'Division by zero error'
    else:
        return number1/number2

print(division(10,2))
print(division(6,0))
```

Output:

```
5.0
Division by zero error
```

- Όταν οι επιστρεφόμενες τιμές μιας συνάρτησης είναι πολλαπλές είναι προτιμότερο να τις επιστρέψουμε με άλλες προγραμματιστικές δομές όπως είναι οι λίστες ή τα λεξικά.

Παράδειγμα εξάσκησης: Δημιουργία συνάρτησης η οποία δέχεται ως όρισμα εισόδου ένα αλφαριθμητικό και επιστρέφει το πλήθος των χαρακτήρων, το πλήθος των λέξεων που περιέχει το αλφαριθμητικό, το πλήθος των φωνηέντων και το πλήθος των συμφώνων.

```
def strcount(data):
    chars=0 #πλήθος των χαρακτήρων
    words=1 #πλήθος των λέξεων
    vowels=0 #πλήθος των φωνηέντων
    consonants=0 #πλήθος των συμφώνων
    chars=len(data)
    for x in data:
        if x==" ":
            words+=1
        elif x in ('a','A','e','E','i','I','o','O','u','U'):
            vowels+=1
        else:
            consonants+=1
    return [chars, words, vowels, consonants]
values=strcount("We learn Matlab")
print(type(values))
print(f"Total length: {values[0]}
Total words: {values[1]}
Total vowels: {values[2]}
Total consonants: {values[3]}
")
```

Output:

```
<class 'list'>
Total length: 15
Total words: 3
Total vowels: 5
Total consonants: 8
```

- Μετατροπή του κώδικα ώστε η επιστρεφόμενη τιμή να είναι λεξικό αντί για λίστα

```
    return {"chars":chars, "words":words, "vowels":vowels,"consonants":consonants}

values=strcount("We learn Matlab")
print(type(values))
print(f'''Total length: {values['chars']}
Total words: {values['words']}
Total vowels: {values['vowels']}
Total consonants: {values['consonants']}
''')
```

Output:

```
<class 'dict'>
Total length: 15
Total words: 3
Total vowels: 5
Total consonants: 8
```

Παράμετροι με προκαθορισμένες τιμές

- Υπάρχουν ορισμένες παράμετροι στις συναρτήσεις οι οποίες είναι προαιρετικές και όταν δεν χρησιμοποιούνται κατά την κλήση της συνάρτησης αυτές λαμβάνουν κάποια προκαθορισμένη τιμή.

Παράδειγμα εξάσκησης: Δημιουργία συνάρτησης η οποία δέχεται ως ορίσματα εισόδου 2 αριθμούς και 1 τελεστή και επιστρέφει το αποτέλεσμα της αριθμητικής πράξης. Όταν ο τελεστής δεν ορίζεται τότε λαμβάνει την προκαθορισμένη τιμή + και εκτελείται η πρόσθεση των 2 αριθμών.

```
def calculations(first, second, operator='+'):
    if operator=='+':
        return first+second
    if operator=='-':
        return first-second
    if operator=='*':
        return first*second
    if operator=='/':
        return first/second
    if operator=='%':
        return first%second
    if operator=='**':
        return first**second

print(calculations(4,3))
print(calculations(4,3,'-'))
print(calculations(4,3,'*'))
```

Output:

```
7
1
12
```

- Στο προηγούμενο παράδειγμα θα μπορούσαμε να έχουμε ακόμα μια παράμετρο με προκαθορισμένη τιμή πχ. second=2.

```
def calculations(first, second=2, operator='+'):
    if operator=='+':
        return first+second
    if operator=='-':
        return first-second
    if operator=='*':
        return first*second
    if operator=='/':
        return first/second
    if operator=='%':
        return first%second
    if operator=='**':
        return first**second

print(calculations(4))
```

Output:
6

- Όταν έχουμε παραμέτρους με προκαθορισμένες τιμές τότε αυτές θα πρέπει να δηλώνονται μετά από τις παραμέτρους που δεν έχουν προκαθορισμένη τιμή.

```
def calculations(first=2, second, operator='+'):
```
← *Θα εμφάνιζε μήνυμα λάθους*

- Όταν δεν χρησιμοποιούμε λέξεις-κλειδιά η σειρά που δηλώνουμε τις παραμέτρους στις συναρτήσεις έχει σημασία.

Παράδειγμα εξάσκησης: Δημιουργία συνάρτησης που επιστρέφει τη διεύθυνση που εισάγει ο χρήστης σε ένα συγκεκριμένο format

```
def address(address,number,zipcode,city):
    print(f'''
Address: {address} {number}
Zip code: {zipcode}
City: {city}
''')
```

```
address('28is Oktovriou', 76, 10434,'Athina')
```

Output:

Address: 28is Oktovriou 76
Zip code: 10434
City: Athina

```
address(10434,'28is Oktovriou','Athina',76)
```

Το μήνυμα εκτυπώθηκε λάθος καθώς αλλάξαμε τη σειρά εισαγωγής των ορισμάτων της συνάρτησης

Output:

```
Address: 10434 28is Oktovriou
Zip code: Athina
City: 76
```

- Αν όταν καλέσουμε τη συνάρτηση δηλώσουμε τις παραμέτρους με τα ονόματα τους τότε το αποτέλεσμα δεν θα επηρεαστεί από τη σειρά εισαγωγής τους.

```
address(zipcode=10434,address='28is Oktovriou',city='Athina',number=76)
```

Output:

Address: 28is Oktovriou 76
Zip code: 10434
City: Athina

Απροσδιόριστος αριθμός παραμέτρων

- Αν ο αριθμός των παραμέτρων που θέλουμε να εισάγουμε σε μια συνάρτηση δεν είναι γνωστός είναι δηλαδή απροσδιόριστος μπορούμε να χρησιμοποιήσουμε το σύμβολο * πριν το όνομα της παραμέτρου. Συνηθίζεται, χωρίς να είναι υποχρεωτικό, το όνομα που δίνουμε σε αυτή την περίπτωση να είναι args (arguments).

Παράδειγμα:

```
def max_number(*args):
    return max(args)

print(max_number(1,2,3))
print(max_number(7,8,-2,-9,0))
```

Επιστρέφει το μέγιστο μιας πλειάδας

Output:

```
3
8
```

- Η παράμετρος kwargs, εκτελεί τον ίδιο σκοπό απλά τις παραμέτρους τις εισάγουμε με λέξεις κλειδιά. Η διαφορά της args με την kwargs είναι ότι στην args οι παράμετροι είναι πλειάδα ενώ στην kwargs οι παράμετροι είναι λεξικό.

Παράδειγμα

```
def max_min_numbers (*args,**kwargs):
    maxnum= max(args) if args else '-'
    minnum=min(kwargs.values()) if kwargs else '-'
    return maxnum, minnum

print(max_min_numbers(1,2,4,val1=1,val2=2,val3=3))
print(max_min_numbers(1,2,4))
print(max_min_numbers(val1=1,val2=2,val3=4))
```

Επιστρέφει το μέγιστο μιας πλειάδας

Επιστρέφει το ελάχιστο ενός λεξικού

Output:

```
(4, 1)
(4, '-')
('-', 1)
```

- Όταν δηλώνουμε μια μεταβλητή στο μπλοκ των εντολών μιας συνάρτησης τότε αυτή η μεταβλητή αναγνωρίζεται μόνο μέσα σε αυτό το μπλοκ, είναι δηλαδή τοπική μεταβλητή.

Παράδειγμα: Δημιουργία συνάρτησης που υπολογίζει το παραγοντικό ενός αριθμού

```
def fact(number):
    result=1
    for i in range(1,number+1):
        result*=i
    return result

print(fact(5))
print(result)
```

Η μεταβλητή result έχει οριστεί μέσα στο σώμα της συνάρτησης για αυτό και δεν αναγνωρίζεται εκτός αυτής

Output:

```
120
NameError: name 'result' is not defined
```

```
def fact(number):
    result=1
    for i in range(1,number+1):
        result*=i
    return result

result=5
print(fact(5))
print(result)
```

Η μεταβλητή result ορίζεται εκτός της συνάρτησης, είναι καθολική, για αυτό και εκτυπώνεται. Μέσα στη συνάρτηση όμως η αρχική της τιμή είναι 1 όχι 5. Αυτό δεν θα άλλαζε ακόμα και αν η εντολή result=5 υπήρχε στην αρχή του κώδικα πριν τις γραμμές ορισμού της συνάρτησης.

Output:

```
120
5
```

Αναδρομικές και ανώνυμες συναρτήσεις

- Αναδρομική είναι μια συνάρτηση η οποία καλεί τον εαυτό της, δηλαδή η συνάρτηση αυτή ορίζεται κάνοντας χρήση την ίδια την συνάρτηση.

Παράδειγμα: Δημιουργία συνάρτησης που υπολογίζει το παραγοντικό ενός αριθμού μέσω αναδρομής

```
def fact(number):
    if number==0:
        par=1
    else:
        par=number*fact(number-1)
    return par

print(fact(5))
```

Output:

120

- Η ανώνυμη συνάρτηση χρησιμοποιείται για να οριστεί και να περαστεί σαν όρισμα κάπου αλλού. Αντί της λέξης def χρησιμοποιείται η λέξη lambda, το όνομα της συνάρτησης και οι παρενθέσεις από τα ορίσματα παραλείπονται και επιστρέφει την τιμή χωρίς τη χρήση του return.

Σύνταξη:

lambda x1, x2, ..: έκφραση, όπου x1, x2 ορίσματα εισόδου

- Τη συνάρτηση map() μπορείτε να τη χρησιμοποιήσετε σε περιπτώσεις όπου έχετε μια επαναληπτική δομή, για παράδειγμα μια λίστα, και θέλετε με βάση αυτές τις τιμές να δημιουργήσετε ένα ίδιο αντικείμενο, μεταλλάσσοντας τις τιμές του αρχικού αντικειμένου αλλά κρατώντας τις τιμές που είχε το αρχικό αντικείμενο πάνω στο οποίο βασιστήκατε, χωρίς δηλαδή να τις αλλάξετε.

Παράδειγμα: Δημιουργία συνάρτησης που δέχεται μια λίστα αλφαριθμητικών και επιστρέφει τα ίδια δεδομένα αλλά σε μορφοποίηση ώστε να είναι όλα μικροί χαρακτήρες. *(χωρίς χρήση της map())*

```
def lowercase(data):
    result=[]
    for value in data:
        result.append(value.lower())
    return result

print(lowercase(['Python','Matlab','R']))
```

Output:

```
['python', 'matlab', 'r']
```

Παράδειγμα: Δημιουργία συνάρτησης που δέχεται μια λίστα αλφαριθμητικών και επιστρέφει τα ίδια δεδομένα αλλά σε μορφοποίηση ώστε να είναι όλα μικροί χαρακτήρες. *(με χρήση της map())*

```
def lowercase(data):
    return data.lower()
```

Η ενέργεια που θα εφαρμοστεί στο αντικείμενο

Το επαναληπτικό αντικείμενο πάνω στο οποίο η map() θα εφαρμόσει την ενέργεια

```
print(map(lowercase,['Python','Matlab','R']))

print(list(map(lowercase,['Python','Matlab','R'])))
```

Μετατροπή του αντικειμένου map σε λίστα μέσω της list()

Output:

```
<map object at 0x00000227E247A350>

['python', 'matlab', 'r']
```

- *Η map φροντίζει να περνάει κάθε φορά ένα αντικείμενο από τη λίστα και να τρέχει τη συνάρτηση lowercase()*

Η συνάρτηση filter

- Τη συνάρτηση filter() μπορείτε να τη χρησιμοποιήσετε για να επιλέξετε κάποια δεδομένα από το επαναληπτικό αντικείμενο που της περνάτε σαν όρισμα τα οποία να ικανοποιούν μια συνθήκη.
- Η σύνταξη της filter() είναι η ίδια με της map()

Παράδειγμα: Έστω ότι έχετε ένα πρόγραμμα στο οποίο οι χρήστες έχουν δημιουργήσει συνθηματικά usernames και passwords για να συνδεθούν. Εσείς αποφασίζετε να αλλάξετε την πολιτική ασφάλειας της εφαρμογής σας και δεν θέλετε να υπάρχουν usernames μικρότερα από 4 χαρακτήρες.

```
def checklen(data):
    return len(data)<=3

values =['Anna','George','Maria','Bob']
print(list(filter(checklen,values)))
print(values)
```

Output:

```
['Bob']
['Anna', 'George', 'Maria', 'Bob']
```

Όπως η map() έτσι και η filter() δεν έχει επηρεάσει το αρχικό αντικείμενο

Μία συνάρτηση:

- Καλείται με το όνομά της.
- Μπορεί να μην έχει ορίσματα.
- Μπορεί να δεχθεί ένα ή περισσότερα ορίσματα ως παραμέτρους εισόδου αρκεί να ακολουθείται η εξής σειρά:
  1. Κανονικές παράμετροι
  2. Παράμετροι args
  3. Παράμετροι με προκαθορισμένη τιμή
  4. Παράμετροι kwargs
- Μπορεί να μην επιστρέφει καμία τιμή στο κύριο πρόγραμμα.
- Μπορεί να επιστρέφει στο κύριο πρόγραμμα μια ή περισσότερες τιμές. Για την επιστροφή τιμών στο κύριο πρόγραμμα χρησιμοποιούμε εντός της συνάρτησης την εντολή **return** η οποία ακολουθείται από μια ή περισσότερες μεταβλητές. Εάν επιστρέφει περισσότερες από μια τιμές τότε στην κλήση της συνάρτησης στο αριστερό μέλος της εντολής αναγράφονται οι μεταβλητές εξόδου διαχωριζόμενες με κόμμα. πχ.

par1, par2, par3, par4=onoma_sunartisis()