Προγραμματισμός στο Raspberry

ΜΑΘΗΜΑΤΑ RASPBERRY

ΜΑΡΙΟΣ ΒΑΣΙΛΕΙΟΥ

ΔΟΥΜΑ ΑΝΑΣΤΑΣΙΑ

ΚΑΒΑΛΛΙΕΡΑΤΟΥ ΕΡΓΙΝΑ

Εισαγωγή & Στόχοι Μαθήματος

Ρομποτική δεν είναι μόνο το υλικό (hardware) αλλά και το λογισμικό. Για να εκτελέσει το ρομπότ συγκεκριμένες λειτουργίες θα πρέπει να γίνει και ο σωστός προγραμματισμός.

Σε αυτό το κεφάλαιο θα γίνει εκμάθηση βασικών εντολών και λειτουργιών της python με σκοπό την σωστή διαχείριση των δεδομένων, είτε από αισθητήρες του ρομπότ μας είτε από άλλες πηγές.

Περιεχόμενα

Λογικές εκφράσεις και Τελεστές

- Ισότητα: '**==**' πχ. (x **==** y)
- Ανισότητα: '**!=**' πχ. (x **!=** y)
- Μεγαλύτερος: '**>**' πχ. (x **>** y)
- Μικρότερος: '**<**' πχ. (x **<** y)
- Μεγαλύτερος ή ίσος: '**>=**' πχ. (x **>=** y)
- Μικρότερος ή ίσος : '**<=**' πχ. (x **<=** y)
- Και: '**and**' πχ. (True **and** False) *έχει ως αποτέλεσμα False*
- Ή: '**or**' πχ. (True **or** False) *έχει ως αποτέλεσμα True*
- Όχι: '**no**t' πχ. (**not** True) *έχει ως αποτέλεσμα False*

Εκτέλεση υπό συνθήκη – If

- Κατά τη δημιουργία προγραμμάτων σχεδόν πάντα χρειαζόμαστε να κάνουμε κάποιον έλεγχο και κατά συνέπεια να αλλάζει η συμπεριφορά του προγράμματος ανάλογα με το αποτέλεσμα του ελέγχου
- Ο έλεγχος αυτός γίνεται με την εντολή '**if**' η οποία ακολουθείται από την συνθήκη. Π.χ.

```
>>> x = 8
>>> if x > 0:
        print('x is positive')


x is positive
```

- Ο παραπάνω κώδικας ελέγχει αν είναι θετικό το x. Τι θα πρέπει να γράψουμε όμως αν δεν είναι θετικό; Για αυτό τον λόγο θα γίνει η χρήση των εντολών '**elif**' και '**else**' όπως φαίνεται στο παρακάτω παράδειγμα:

```
>>> x = 0
>>> if x > 0:
        print('x is positive')
elif x < 0:
        print('x is negative')
else:
        print('x is neutral')


x is neutral
```

Άσκηση Κατανόησης

- Έστω μια μεταβλητή x στην οποία θα δίνει ο χρήστης μια οποιαδήποτε ακέραια τιμή.
- Να φτιάξετε ένα πρόγραμμα που θα ελέγχει και θα εμφανίζει αν η τιμή είναι:
  - μεγαλύτερη από 100,
  - μικρότερη ή ίση από 100 και μεγαλύτερη από 50,
  - μικρότερη ή ίση από 50 και μεγαλύτερη από 0,
  - Μηδέν (0)
  - Μικρότερη του μηδενός
- Προσοχή: Για να δώσετε τιμή τύπου integer θα πρέπει να κάνετε "casting" σε τύπου integer. Δηλ.: x = int(input())

Λύση


```
1   print("Enter value")
2   x = int(input())
3   if x > 100:
4       print("The value is greater than 100")
5   elif x <= 100 and x > 50:
6       print("The value is at range (50 - 100]")
7   elif x <= 50 and x > 0:
8       print("The value is at range (0 - 50]")
9   elif x == 0:
10      print("The value is zero")
11  else:
12      print("The value is less than zero")
```

- Γραμμή 2: Δέχεται τιμή από τον χρήστη, με το *int()* ορίζουμε ότι η τιμή είναι τύπου ακεραίου

Βρόχος – while loop

- Πολλές φορές στα προγράμματα μας χρειάζεται να εκτελέσουμε επαναλαμβανόμενες εργασίες.
- Ένας τρόπος για να επιτευχθεί αυτό είναι με την χρήση της εντολής '**while**' :

```
>>> x = 0
>>> while x<=3:
        print('x =',x)
        x = x+1


x = 0
x = 1
x = 2
x = 3
```

Όσο το x είναι μικρότερο του 3 -> θα εμφανίζει την τιμή του x και στην συνέχεια αυξάνει το x κατά μία μονάδα.

- Δηλαδή το πρόγραμμα θα εκτελείται όσο το x είναι μικρότερο ή ίσο του 3 'while x<=3'

Άσκηση Κατανόησης

- Έστω μια μεταβλητή x στην οποία θα δίνει ο χρήστης μια οποιαδήποτε ακέραια τιμή. Να εμφανίζεται κατάλληλο μήνυμα κατά την είσοδο της τιμής.
- Να φτιάξετε ένα πρόγραμμα που επαναληπτικά θα ελέγχει αν η τιμή που δίνει ο χρήστης είναι θετική ή αρνητική και θα εμφανίζει κατάλληλο μήνυμα. Η επανάληψη θα σταματάει όταν ο χρήστης δώσει την τιμή μηδέν (0).

Λύση


```
1    print("Enter value  (Press 0 to exit)")
2    x = int(input())
3    while x != 0:
4        if x > 0:
5            print("The value is a positive number")
6        elif x < 0:
7            print("The value is a negative number")
8        print("Enter new value  (Press 0 to exit)")
9        x = int(input())
```

- Γραμμή 3: Η επανάληψη θα εκτελείται όσο το x είναι διάφορο του μηδενός

- Προσθέτουμε είσοδο στην γραμμή 2 έτσι ώστε να μπορεί να κάνει τον έλεγχο στην γραμμή 3. Κατά συνέπεια, απαιτείται άλλη μία εντολή εισόδου στην γραμμή 9 έτσι ώστε όταν θα επιστρέψει στην γραμμή 3 (λόγω της επανάληψης) να γίνει ο έλεγχος με νέα τιμή

Λύση με την χρήση ‘break’


```
1    while True:
2        print("Enter new value  (Press 0 to exit)")
3        x = int(input())
4        if x > 0:
5            print("The value is a positive number")
6        elif x < 0:
7            print("The value is a negative number")
8        else:
9            break
```

- Αν έχει δοθεί η τιμή 0 θα μπει στην γραμμή 8-9 και με την εντολή break θα βγει από την επανάληψη

Εισαγωγή στις Συναρτήσεις

- Οι συναρτήσεις είναι «αυτόνομες» ενότητες κώδικα που ολοκληρώνουν μια συγκεκριμένη εργασία. Οι συναρτήσεις συνήθως «λαμβάνουν» δεδομένα, επεξεργάζονται και «επιστρέφουν» ένα αποτέλεσμα
- Μόλις γραφτεί μια συνάρτηση, μπορεί να χρησιμοποιηθεί ξανά και ξανά. Οι συναρτήσεις μπορούν να «κληθούν» από το εσωτερικό άλλων λειτουργιών
- Ένα παράδειγμα συνάρτησης είναι αυτό της print() η οποία εκτυπώνει την συμβολοσειρά ή τη μεταβλητή που λαμβάνει ως όρισμα

Συναρτήσεις

- Συνεπώς, οι συναρτήσεις αποτελούνται από τα εξής μέρη:
  1. Την εισαγωγή των δεδομένων ως παραμέτρους
  2. Την υλοποίηση των πράξεων-εντολών
  3. Την εξαγωγή των αποτελεσμάτων
- Χαρακτηριστικό και πολύ χρήσιμο παράδειγμα είναι αυτό της εύρεσης μέγιστου:

```
>>> def maximum(x,y):
        if x > y:
                return x
        else:
                return y


>>> maximum(10,35)
35
```

Ορίζουμε την συνάρτηση με την εντολή 'def' την οποία ακολουθεί το όνομα της συνάρτησης και μέσα στις παρενθέσεις οι παράμετροι της συνάρτησης.

Με την εντολή 'return' επιστρέφει/ εξαγάγει το αποτέλεσμα

Άσκηση Κατανόησης

- Να δημιουργήσετε 2 συναρτήσεις οι οποίες να βρίσκουν το μικρότερο μεταξύ:
  1. Δύο αριθμών
  2. Τριών αριθμών
- Στην συνέχεια να γράψετε πρόγραμμα που θα δέχεται ως είσοδο 3 αριθμούς από τον χρήστη και θα εμφανίζει με την χρήση των συναρτήσεων τον μικρότερο από τους 2 πρώτους και τον μικρότερο από τους τρεις.

Λύση


```
1   def minimum(a, b):
2       if a < b:
3           return a
4       else:
5           return b
6
7
8   def minimum3(a, b, c):
9       if a < b:
10          if a < c:
11              return a
12          else:
13              return c
14      else:
15          if b < c:
16              return b
17          else:
18              return c
```

Γρ.1-5:
Αν το a είναι μικρότερο του b το επιστρέφει αλλιώς επιστρέφει το b

Γρ. 8-18:

Για τον έλεγχο τριών αριθμών θα χρειαστούμε εμφωλευμένους ελέγχους όπως φαίνεται στο παράδειγμα

```
21  # Main program
22  print("Enter number 1")
23  x = int(input())
24  print("Enter number 2")
25  y = int(input())
26  print("Enter number 3")
27  z = int(input())
28
29  print("min 2: ", minimum(x, y))
30  print("min 3: ", minimum3(x, y, z))
```

Γρ. 22-27 δέχεται από τον χρήστη 3 ακέραιους αριθμούς
Γρ. 29-30: εμφανίζει τους ελάχιστους αριθμούς καλώντας τις συναρτήσεις

Λύση με χρήση εμφωλευμένων συναρτήσεων


```
1    def minimum(a, b):
2        if a < b:
3            return a
4        else:
5            return b
6
7
8    def minimum3(a, b, c):
9        if minimum(a, b) < minimum(b, c):
10           return minimum(a, b)
11       else:
12           return minimum(b, c)
13
14
```

```
15   # Main program
16   print("Enter number 1")
17   x = int(input())
18   print("Enter number 2")
19   y = int(input())
20   print("Enter number 3")
21   z = int(input())
22
23   print("min 2: ", minimum(x, y))
24   print("min 3: ", minimum3(x, y, z))
```

Η μόνη διαφορά με την προηγούμενη λύση είναι ότι χρησιμοποιούμε εμφωλευμένες συναρτήσεις για να βρούμε τον μικρότερο μεταξύ τριών αριθμών.

Για να βρούμε τον ελάχιστο ανάμεσα σε τρεις αριθμούς θα πρέπει να βρούμε τον ελάχιστο ανάμεσα από τους 2 και το αποτέλεσμα να το συγκρίνουμε με τον τρίτο αριθμό.

Εισαγωγή Βιβλιοθηκών

- Προγραμματίζοντας στην python υπάρχει η δυνατότητα να γίνει χρήση συναρτήσεων από άλλα αρχεία τα οποία μπορεί να μην τα έχουμε φτιάξει εμείς
- Αυτό μπορεί να επιτευχθεί με την χρήση της εντολής 'import'
- Χαρακτηριστικό παράδειγμα είναι η χρήση της βιβλιοθήκης 'math' η οποία δίνει την δυνατότητα για κάποιες μαθηματικές πράξεις όπως:
- Την χρήση του αριθμού π (~3,14..):
- Τον υπολογισμό της τετραγωνικής ρίζας:

```
>>> import math
>>> print(math.pi)
3.141592653589793
>>> print(math.sqrt(9))
3.0
```

Εισαγωγή Βιβλιοθηκών

- Παρατηρούμε ότι για την χρήση της βιβλιοθήκης ‘math’ χρησιμοποιούμε την εντολή ‘import math’ ενώ για να καλέσουμε μια συνάρτηση της βιβλιοθήκης χρησιμοποιούμε το όνομα της βιβλιοθήκης, «τελεία» και το όνομα της συνάρτησης. Πχ. ‘math.sqrt()’
- Επιπλέον, υπάρχει η δυνατότητα να μετονομαστεί η βιβλιοθήκη για πιο εύκολη χρήση. Π.χ. αντί για ‘math’ να καλείται με το ‘m’:

```
>>> import math as m
>>> print(m.sqrt(4))
```

- Αυτό μπορεί να γίνει με την χρήση της εντολής ‘as’ όπως φαίνεται παραπάνω

Δημιουργία και κλήση της δικής μας Ενότητας (module)

- Για να δημιουργηθεί μια ενότητα, απλώς αποθηκεύστε τον κώδικα που θέλετε σε ένα αρχείο με την επέκταση αρχείου '.py'
- Για παράδειγμα:

1. Δημιουργήστε ένα αρχείο 'mymodule.py' με κώδικα:

```
def hello(name):
    print("hello, " + name)
```

2. Τώρα μπορείτε να χρησιμοποιήσετε την συνάρτηση που φτιάξατε στο αρχείο mymodule.py από άλλο αρχείο απλά με την χρήση της εντολής 'import' όπως φαίνεται παρακάτω:

```
import mymodule

mymodule.hello("Marios")
```

Άσκηση Κατανόησης

- Χρησιμοποιώντας τον κώδικα που γράψατε στην προηγούμενη άσκηση για την εύρεση του ελάχιστου αριθμού, να φτιάξετε ένα αρχείο με όνομα ‘find_min.py’ που θα περιέχει αυτό τον κώδικα
- Να δημιουργήσετε κατάλληλο πρόγραμμα σε διαφορετικό αρχείο που θα καλεί αυτές τις συναρτήσεις
- Κατά την κλήση, για δική σας διευκόλυνση, να κάνετε την χρήση της εντολής ‘as’ με όνομα ‘fm’

Λύση

find_min.py

```
1   def minimum(a, b):
2       if a < b:
3           return a
4       else:
5           return b
6
7
8   def minimum3(a, b, c):
9       if minimum(a, b) < minimum(b, c):
10          return minimum(a, b)
11      else:
12          return minimum(b, c)
```

- Στο αρχείο *find_min.py* χρησιμοποιούμε τον κώδικα της προηγούμενης άσκησης

ex4.py

```
1   import find_min as fm
2
3   print("Enter number 1")
4   x = int(input())
5   print("Enter number 2")
6   y = int(input())
7   print("Enter number 3")
8   z = int(input())
9
10  print("min 2: ", fm.minimum(x, y))
11  print("min 3: ", fm.minimum3(x, y, z))
```

- Στο αρχείο *ex4.py* χρησιμοποιούμε κώδικα από την προηγούμενη άσκηση για την είσοδο των αριθμών από τον χρήστη και την εμφάνιση των μικρότερων.
- Γραμμή 1: Για να μπορέσουμε να χρησιμοποιήσουμε τις συναρτήσεις που είναι σε άλλο αρχείο
- Στις γραμμές 10-11 για να καλέσουμε τις συναρτήσεις πρέπει να ορίσουμε σε ποιο αρχείο βρίσκονται γι'αυτό το λόγο πριν το όνομα της συνάρτησης βάζουμε το "*fm.*"

Εισαγωγή στις Λίστες

- Η python έχει έναν τύπο μεταβλητών ο οποίος επιτρέπει την ομαδοποίηση άλλων δεδομένων. Ένας τέτοιος τύπος είναι οι lists, οι οποίες δηλώνουν διατεταγμένα σύνολα τιμών
- Οι τιμές (μέλη μιας λίστας) λέγονται στοιχεία (elements)
- Ο όρος «διατεταγμένα» σημαίνει ότι μπορούμε να ορίσουμε σειρά μεταξύ των στοιχείων δηλαδή ποιο είναι το πρώτο, ποιο είναι το δεύτερο, το τρίτο κ.λπ.
- Τα στοιχεία μιας λίστας μπορεί να είναι οποιουδήποτε τύπου. Μπορεί να είναι νούμερα, ακέραιοι, πραγματικοί, μπορεί ακόμα να είναι και άλλες λίστες

Δημιουργία λίστας

- Ο πιο εύκολος τρόπος δημιουργίας μιας λίστας είναι η τοποθέτηση των στοιχείων μέσα σε '[ ]' διαχωρισμένα με κόμμα ','
- Παρακάτω βλέπουμε δύο ομοιογενείς λίστες, δηλαδή έχουν στοιχεία ίδιου είδους, η πρώτη ακεραίους ενώ η δεύτερη συμβολοσειρές:

```
list1 = [2, 14, 23, 35]

list2 = ['sensor', 'motor', 'microcontroller', 'battery']
```

- Τα στοιχεία μιας λίστας δε χρειάζεται να είναι ίδιου τύπου
- Παρακάτω βλέπουμε μια ανομοιογενή λίστα η οποία περιέχει έναν ακέραιο, έναν πραγματικό αριθμό, μια συμβολοσειρά, αλλά και μια άλλη λίστα:

```
list3 = [2, 6.2, 'sensor', [2,5,10]]
```

Χρήσιμες συναρτήσεις για Λίστες

- Μια πολύ χρήσιμη λειτουργία είναι η αυτόματη παραγωγή από λίστες με ακεραίους βάσει της συνάρτησης range
- Η συνάρτηση range, φτιάχνει ένα εύρος (range) τιμών, μια ακολουθία τιμών από το πρώτο όρισμα της συνάρτησης range έως το δεύτερο όρισμα «μείον ένα»:

```
>>> nums = list(range(1,11))
>>> print(nums)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

- Παρατηρούμε ότι τα διαστήματα της range είναι κλειστά στο κάτω άκρο και ανοιχτά στο πάνω άκρο
- Προσοχή: το range είναι μια συνάρτηση, το αποτέλεσμα του range δεν είναι λίστα, είναι τύπου range, το οποίο το παίρνει η «γεννήτρια»-συνάρτηση της λίστας και το μετατρέπει σε λίστα

Χρήσιμες συναρτήσεις για Λίστες

- Υπάρχουν και δύο άλλες μορφές της range:
  - Με ένα μόνο όρισμα, επιστρέφει μια ακολουθία τιμών που ξεκινάει από το 0 με βηματισμό 1:

```
>>> nums = list(range(11))
>>> print(nums)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

  - Αν υπάρχει και τρίτο όρισμα, τότε αυτό το όρισμα προσδιορίζει το βηματισμό της ακολουθίας τιμών

```
>>> nums = list(range(1,11,2))
>>> print(nums)
[1, 3, 5, 7, 9]
```

Προσπέλαση στοιχείων λίστας

- Για την εμφάνιση στοιχείου σε συγκεκριμένη θέση της λίστας αρκεί να γράψουμε μέσα σε μια print() το όνομα της μεταβλητής στο οποίο ακολουθεί ο αριθμός της θέσης μέσα σε αγκύλες (ξεκινώντας από την θέση 0):

```
>>> nums = list(range(1,11))
>>> print(nums[5])
6
```

- Παρατήρηση: Οι θέσεις της λίστας ξεκινάνε από το 0 και όχι από το 1. Αν δώσετε αρνητική τιμή μετράει από το τέλος της λίστας προς την αρχή, άρα το -1 πάει στο τελευταίο στοιχείο, το -2 στο προτελευταίο κ.ο.κ.:

```
>>> print(nums[-2])
9
```

- Προσοχή: Είναι λάθος να προσπαθήσετε να γράψετε/διαβάσετε στοιχείο το οποίο είναι εκτός των ορίων της λίστας. Σε αυτήν την περίπτωση θα εμφανίσει το λάθος *IndexError: list assignment index out of range*

Μήκος Λίστας

- Η συνάρτηση len() επιστρέφει το μήκος μιας λίστας

```
>>> nums = list(range(1,11))
>>> print(nums)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> print(len(nums))
10
```

- Παρατήρηση: Σε βρόχους οι οποίοι διατρέχουν λίστες, είναι καλύτερα να χρησιμοποιούμε την τιμή αυτής της συνάρτησης αντί μιας σταθεράς, γιατί προσαρμόζεται αυτόματα σε αυξομειώσεις του μήκους της λίστας. Έτσι, αν το μήκος της λίστας αλλάξει, δε θα χρειάζεται να κάνουμε αλλαγές στο πρόγραμμα.
- Παρατήρηση: Στην περίπτωση εμφωλευμένης λίστας, αυτή μετράει σαν απλό στοιχείο.

```
>>> list=[1,2,[1,2]]
>>> print(list)
[1, 2, [1, 2]]
>>> print(len(list))
3
```

Λίστες και λογικός τελεστής « in »

- Ο « in » είναι λογικός τελεστής ο οποίος ελέγχει αν μια τιμή ανήκει σε μια λίστα (επιστρέφοντας true ή false):

```
>>> list1 = ['sensor', 'motor', 'microcontroller', 'battery']
>>> 'battery' in list1
True
```

- Ακόμα μπορεί να χρησιμοποιηθεί και στον βρόχο for:

for *VARIABLE* in LIST:

…..

"BODY"

…..

Βρόχος – For loop με χρήση λίστας

- Ένας άλλος τρόπος για να γίνει επανάληψη κάποιων εργασιών είναι με την χρήση της εντολής '**for**' :

```
>>> sensors = ['s1', 's2', 's3']
>>> for x in sensors:
        print(x)


s1
s2
s3
```

- Όπως φαίνεται στο παράδειγμα, αυτό που κάνει είναι: Για κάθε στοιχείο στην λίστα 'sensors' εμφάνισε το περιεχόμενο

Βρόχος – For loop με χρήση range()

- Ακόμα, ο βρόχος ‘for’ συνηθίζεται να χρησιμοποιείται με την συνάρτηση range() που είδαμε νωρίτερα όπως φαίνεται παρακάτω:

```
>>> for i in range(0,15,2):
        print(i)


0
2
4
6
8
10
12
14
```

- Παρατήρηση: η ‘range()’ δημιουργεί μια λίστα με αριθμούς και μέσα στον βρόχο ‘for’ εμφανίζει τους αριθμούς

Λίστες – Άσκηση κατανόησης

Έστω μια λίστα με θερμοκρασίες από 10 διαφορετικούς αισθητήρες: [15.6, 17.3, 20.8, 23.5, 25.2, 25.8, 26.6, 24.6, 22.7, 20.0]

1. Να γίνει καταμέτρηση με την χρήση βρόχου των αισθητήρων που η θερμοκρασία βρέθηκε να είναι πάνω από 25 βαθμούς και να εμφανίσετε το αποτέλεσμα.
2. Να δημιουργηθεί καινούρια λίστα με τον αριθμό θέσης των αισθητήρων που η θερμοκρασία είναι πάνω από 25 βαθμούς και να εμφανίζεται η λίστα. (π.χ. η θερμοκρασία 15.6 βρίσκεται στην θέση 0).

Λύση ερ. 1


```
1    counter = 0
2    temp_list = [15.6, 17.3, 20.8, 23.5, 25.2,
3                 25.8, 26.6, 24.6, 22.7, 20.0]
4    for x in temp_list:
5        if x > 25:
6            counter = counter + 1
7    print("counter = ", counter)
```

Γρ.1: αρχικοποιούμε έναν μετρητή ο οποίος θα μετράει τις θερμοκρασίες πάνω από 25 βαθμούς

Γρ.4: Για κάθε στοιχείο της λίστας εκτελεί τον κώδικα στις γραμμές 5-6

Γρ.5-6: Αν η τιμή είναι μεγαλύτερη από 25 τότε αυξάνει τον μετρητή κατά 1

Λύση ερ. 2


```
2    temp_list = [15.6, 17.3, 20.8, 23.5, 25.2,
3                 25.8, 26.6, 24.6, 22.7, 20.0]
4    list2 = []
5    for i in range(0, len(temp_list)):
6        if temp_list[i] > 25:
7            list2.append(i)
8    print(list2)
```

Γρ. 4: Αρχικοποιούμε μία άδεια λίστα

Γρ. 5: Για τις θέσεις i της λίστας

Γρ. 6-7: Αν η θερμοκρασία είναι μεγαλύτερη του 25 τότε προσθέτει την τρέχουσα θέση i στην λίστα

Διαγραφή και προσθήκη στοιχείων σε μια Λίστα

- Για να γίνει προσθήκη στοιχείων σε μια λίστα αρκεί να «στριμώξουμε» αυτά τα στοιχεία στις θέσεις που θέλουμε:

```
>>> nums = list(range(1,11))
>>> print(nums)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> nums[5:5] = [0]
>>> print(nums)
[1, 2, 3, 4, 5, 0, 6, 7, 8, 9, 10]
```

- Παρατηρούμε ότι στην θέση 5 προσθέτει τον αριθμό 0 (nums[5:5] = [0])
- Στην ίδια λίστα με την εντολή "del" και τον αριθμό της θέσης της λίστας διαγράφουμε το στοιχείο της λίστας:

```
>>> print(nums)
[1, 2, 3, 4, 5, 0, 6, 7, 8, 9, 10]
>>> del nums[5]
>>> print(nums)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Λίστες – Άσκηση κατανόησης

Έστω μια λίστα με κινητήρες: ["dc_motor1", "dc_motor2", "dc_motor3", "stepper_motor1", "servo_motor1", "servo_motor2", "servo_motor3"]

1. Να προσθέσετε το ‘dc_motor4’ ανάμεσα από το ‘dc_motor3’ και το ‘stepper_motor1’
2. Να προσθέσετε το ‘stepper_motor2’ ανάμεσα από το ‘stepper_motor1’ και το ‘servo _motor1’
3. Να διαγράψετε το τελευταίο στοιχείο της λίστας

Λύση


```
motor_list = ["dc_motor1", "dc_motor2", "dc_motor3", "stepper_motor1",
              "servo_motor1", "servo_motor2", "servo_motor3"]

motor_list[3:3] = ["dc_motor4"]
print(motor_list)
motor_list[5:5] = ["stepper_motor2"]
print(motor_list)
del motor_list[len(motor_list)-1]
print(motor_list)
```

Γρ. 4 προσθέτει στην θέση 3 το dc_motor4

Γρ. 6 προσθέτει στην θέση 5 το stepper_motor2

Γρ. 8 διαγράφει το τελευταίο στοιχείο του πίνακα

Βιβλιογραφία

[1] https://www.raspberrypi.org/software/

[2] https://opensource.com/resources/raspberry-pi

[3] https://www.python.org/

[4] Κ. ΜΑΓΚΟΥΤΗΣ, Χ. ΝΙΚΟΛΑΟΥ, "ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΜΕ PYTHON"