Διαχείριση δεδομένων και αρχείων στο Raspberry

ΜΑΘΗΜΑΤΑ RASPBERRY

ΜΑΡΙΟΣ ΒΑΣΙΛΕΙΟΥ

ΔΟΥΜΑ ΑΝΑΣΤΑΣΙΑ

ΚΑΒΑΛΛΙΕΡΑΤΟΥ ΕΡΓΙΝΑ

Εισαγωγή & Στόχοι Μαθήματος

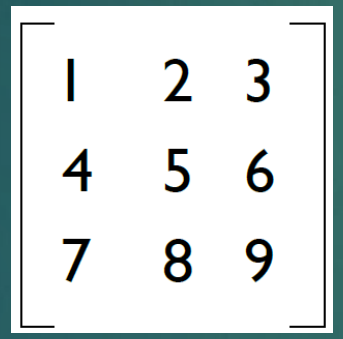
Σε αυτό το κεφάλαιο θα γίνει εκμάθηση βασικών εντολών και λειτουργιών της python με σκοπό την σωστή διαχείριση των δεδομένων, είτε από αισθητήρες του ρομπότ είτε από άλλες πηγές.

Περιεχόμενα

- Πίνακες
- Εξαιρέσεις
- Διαχείριση Αρχείων
- Ασκήσεις κατανόησης στα παραπάνω

Εισαγωγή στους πίνακες

- Η ανάγκη για εμφωλευμένες λίστες και του τρόπου πρόσβασης σε αυτές μας οδηγεί στην έννοια των (πολυδιάστατων) πινάκων
- Παρακάτω βρίσκεται ένα παράδειγμα δισδιάστατου πίνακα και η αποτύπωσή του σε python:




```
>>> mtx = [[1,2,3],[4,5,6],[7,8,9]]
>>> print(mtx)
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> print(mtx[0])
[1, 2, 3]
>>> print(mtx[1])
[4, 5, 6]
>>> print(mtx[2])
[7, 8, 9]
```

- Παρατήρηση: Ομοίως με τις λίστες γράφοντας το όνομα του πίνακα και 2 αγκύλες με τον αριθμό της γραμμής μπορεί να γίνει επιλογή της γραμμής (όπως φαίνεται παραπάνω)

Βιβλιοθήκη NumPy

- NumPy είναι μια βιβλιοθήκη της Python που χρησιμοποιείται για εργασία με πίνακες
- Έχει πολλές συναρτήσεις οι οποίες προορίζονται για εργασία πάνω στην γραμμική Άλγεβρα, στο μετασχηματισμό Fourier και σε πίνακες

Γιατί πίνακας τύπου numPy έναντι λιστών;

- Οι λίστες στην python εξυπηρετούν τον σκοπό των πινάκων αλλά είναι αργές στην επεξεργασία τους
- Οι πίνακες NumPy αποθηκεύονται σε μια συνεχή θέση στη μνήμη σε αντίθεση με τις λίστες, έτσι ώστε οι διαδικασίες να μπορούν να έχουν πρόσβαση και να τους χειρίζονται πολύ αποτελεσματικά

Χρήση NumPy

- Για να γίνει χρήση της βιβλιοθήκης θα πρέπει αυτή να είναι εγκατεστημένη στο Raspberry Pi:
  - Στο terminal δώστε την εντολή *pip3 install numpy*
- Για να είναι προσβάσιμη στο πρόγραμμά μας θα πρέπει να χρησιμοποιήσουμε την εντολή import όπως φαίνεται παρακάτω:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

- Το παραπάνω πρόγραμμα θα σας εμφανίσει τα στοιχεία του πίνακα

Πολυδιάστατος πίνακας NumPy

- Παράδειγμα δήλωσης δισδιάστατου πίνακα:

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
```

```
[[1 2 3]
 [4 5 6]]
```

- Παράδειγμα δήλωσης τρισδιάστατου πίνακα:

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
```

```
[[[1 2 3]
  [4 5 6]]

 [[1 2 3]
  [4 5 6]]]
```

Προσπέλαση στοιχείων πίνακα NumPy

- Ομοίως με τις λίστες, αρκεί το όνομα του πίνακα στο οποίο ακολουθεί ο αριθμός της θέσης μέσα σε αγκύλες (ξεκινώντας από την θέση 0):
  - Εμφάνιση 1ου στοιχείου μονοδιάστατου

```
arr = np.array([1, 2, 3, 4])
print(arr[0])
```

  - Εμφάνιση 3ου στοιχείου της 2ης διάστασης:

```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print(arr[1, 2])
```

  - Εμφάνιση 3ου στοιχείου του 2ου πίνακα της 1ης διάστασης:

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
print(arr[0, 1, 2])
```

Πίνακας NumPy – Ένωση/ Διαχωρισμός

- Παράδειγμα ένωσης δύο πινάκων:

```
import numpy as np
arr1 = np.array([1, 2, 3])
print(arr1)
arr2 = np.array([4, 5, 6])
print(arr2)
arr = np.concatenate((arr1, arr2))
print(arr)
```

```
[1 2 3]
[4 5 6]
[1 2 3 4 5 6]
```

- Παράδειγμα διαχωρισμού σε τρεις πίνακες (χωρίζονται αυτόματα σε 3 πίνακες με τον ίδιο αριθμό κελιών):

```
arr = np.array([1, 2, 3, 4, 5, 6])
arr2 = np.array_split(arr, 3)
print(arr2)
```

```
[array([1, 2]), array([3, 4]), array([5, 6])]
```

Αναζήτηση σε πίνακα NumPy

- Εύρεση θέσης όπου η τιμή είναι 4:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 4, 4])
x = np.where(arr == 4)
print(x)
```

→ `(array([3, 5, 6]),)`

- Αν το στοιχείο υπάρχει στον πίνακα (x in arr)

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(5 in arr)
```

→ `True`

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(10 in arr)
```

→ `False`

Ταξινόμηση πίνακα NumPy

- Ταξινόμηση με την συνάρτηση sort()
- Ταξινόμηση μονοδιάστατου πίνακα:

```
import numpy as np
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))
```

```
[0 1 2 3]
```

- Ταξινόμηση δισδιάστατου πίνακα:

```
import numpy as np
arr = np.array([[3, 2, 4], [5, 0, 1]])
print(np.sort(arr))
```

```
[[2 3 4]
 [0 1 5]]
```

Άσκηση Κατανόησης

- Να δημιουργήσετε έναν πίνακα NumPy στον οποίο θα αποθηκεύετε τις μοίρες ενός κινητήρα (τιμές από το 0 μέχρι 180) και για κάθε τιμή θα αντιστοιχείτε την χρονική στιγμή (1, 2, 3 … κ.ο.κ.).
- Τις τιμές αυτές να τις δίνετε από το πληκτρολόγιο ενώ οι χρονικές στιγμές θα αντιστοιχίζονται αυτόματα ξεκινώντας από την χρονική στιγμή 0.
- Ο χρήστης θα δώσει συνολικά 10 τιμές και θα εμφανίζει τον πίνακα πριν το τέλος του προγράμματος.
- Παράδειγμα πίνακα:

0	1	2	3	4	5	6	7	8	9
0	45	90	120	180	180	135	60	0	0

Λύση 1

Στην πρώτη λύση θα δημιουργήσουμε έναν άδειο πίνακα 10 θέσεων και θα τον γεμίσουμε με δεδομένα από τον χρήστη:

```
1   import numpy as np
2
3   arr = np.empty([2, 10], dtype=int)
4
5   for i in range(0, 10):
6       arr[0][i] = i
7       print("Enter degrees:")
8       arr[1][i] = input()
```

Γραμμή 3: Δημιουργείται ένας άδειος πίνακας 2x10 τύπου integer. Στην παράμετρο dtype προσθέτουμε τον τύπο του πίνακα

Γραμμή 6: Αυτόματα προσθέτει την χρονική στιγμή

Γραμμή 8: δέχεται από τον χρήστη τις μοίρες

Λύση 2

- Στην δεύτερη λύση θα δημιουργήσουμε έναν άδειο δισδιάστατο πίνακα 0 θέσεων και θα τον γεμίσουμε με δεδομένα από τον χρήστη:

```
1   import numpy as np
2
3   arr = np.empty([2, 0], dtype=int)
4
5   for i in range(0, 10):
6       print("Enter degrees:")
7       arr = np.append(arr, [[i], [int(input())]], axis=1)
8
9   print(arr)
```

Γρ. 3: Δημιουργία άδειου πίνακα 0 θέσεων τύπου Integer
Γρ. 7: Με την εντολή append επεκτείνουμε τον πίνακα. Αν δώσουμε ως παράμετρο το *axis=0* προσθέτουμε έξτρα γραμμή ενώ με το *axis=1* προσθέτουμε στήλη.
Συνεπώς, θα επεκτείνουμε τον πίνακα *arr* προσθέτοντας νέα στήλη με τις παρακάτω θέσεις *[[i], [int(input())]]*.
Με το *i* δίνουμε την χρονική στιγμή, ενώ με το *int(input())]* τις μοίρες από τον χρήστη.

Διαχείριση εξαιρέσεων (1/2)

- Όταν συμβαίνει λάθος κατά τη διάρκεια της εκτέλεσης ενός προγράμματος, δημιουργείται μια εξαίρεση (exception), το πρόγραμμα σταματά και η Python τυπώνει μήνυμα λάθους
- Όταν συμβαίνει αυτό το λάθος το πρόγραμμα σταματάει. Πολλές φορές όμως δεν θέλουμε να συμβεί αυτό αλλά θέλουμε να διαχειριστούμε το λάθος
- Το πιο απλό παράδειγμα είναι αν διαιρέσουμε έναν αριθμό με το 0:

```
x = 5
y = 0
print(x/y)
```

```
    print(x/y)
ZeroDivisionError: division by zero
```

- Προφανώς ένας τρόπος είναι χρησιμοποιώντας έναν έλεγχο if πριν την εκτέλεση

Διαχείριση εξαιρέσεων (2/2)

- Για τη διαχείριση αυτού του λάθους, όμως, μπορούμε να χρησιμοποιήσουμε τις εντολές *try-except*:

```
x = 5
y = 0
try:
    print(x/y)
except ZeroDivisionError:
    print("Error: Zero Division. Y is 0")
```

```
Error: Zero Division. Y is 0


Process finished with exit code 0
```

- Η εντολή *try* εκτελεί τις εντολές στο πρώτο μπλοκ. Αν δεν υπάρξουν εξαιρέσεις αγνοεί την εντολή *except*. Αν συμβεί εξαίρεση τύπου *ZeroDivisionError* τότε εκτελεί τις εντολές στον κλάδο *except* και μετά συνεχίζει
- Με αυτό τον τρόπο δεν χρειάζεται το πρόγραμμά μας να τερματίσει αναπάντεχα

Διαχείριση Αρχείων (1/3)

- Για να διαβάσετε ένα αρχείο .txt αρκεί ο παρακάτω κώδικας:

```
f = open("file.txt", "r")
print(f.read())
```

- Υπάρχει όμως περίπτωση να μην ανοίξει το αρχείο (πχ αν δεν υπάρχει) και να τερματίσει το πρόγραμμα εμφανίζοντας μήνυμα λάθους
- Γι' αυτό το λόγο χρησιμοποιούμε τις εντολές που είδαμε προηγουμένως για τη διαχείριση εξαιρέσεων *try - except:*

```
try:
    f = open("filed.txt", "r")
    print(f.read())
except IOError:
    print("Error opening the file...")
print("something")
```



```
Error opening the file...
something

Process finished with exit code 0
```

Διαχείριση Αρχείων (2/3)

- Παρατηρούμε ότι ενώ δεν ανοίγει το αρχείο, εκτελεί τις εντολές μέσα στο block *except* και συνεχίζει να λειτουργεί το πρόγραμμα
- Για την εγγραφή ενός αρχείου χρησιμοποιούμε την εντολή *open("file.txt", "a"):*
- Έχοντας ως παράμετρο το "a" επεκτείνεται το αρχείο

```
f = open("file.txt", "a")
```

- Αν αντί για "a" χρησιμοποιήσουμε το "w" θα διαγράψει το περιεχόμενο που ήδη υπάρχει και θα γράψει τα νέα δεδομένα από την αρχή

```
f = open("file.txt", "w")
```

Διαχείριση Αρχείων (3/3)

- Παρακάτω ακολουθεί παράδειγμα εγγραφής αρχείου:

```
try:
    f = open("file.txt", "a")
    f.write("Now the file has more content!")
    f.close()

    #open and read the file after the appending:
    f = open("file.txt", "r")
    print(f.read())
except IOError:
    print("Error opening the file...")
print("do something")
```

```
Now the file has more content!
do something

Process finished with exit code 0
```

Εμφάνιση Ώρας συστήματος

- Η εμφάνιση της ώρας του συστήματος μπορεί να γίνει με την χρήση της βιβλιοθήκης *time* όπως φαίνεται παρακάτω:

```
import time

t = time.localtime()
current_time = time.strftime("%H:%M:%S", t)
print(current_time)
```

```
07:46:58
```

Άσκηση Κατανόησης

- Έστω μια λίστα με είδη αισθητήρων: [Temperature_sensor1, Temperature_sensor2, Humidity_sensor, ultrasonic_sensor].
- Να κατασκευάσετε πρόγραμμα που θα δέχεται από τον χρήστη τον αριθμό του αισθητήρα (Η αρίθμηση των αισθητήρων ξεκινά από 0 πχ. ο Humidity_ sensor στην παραπάνω λίστα θα έχει τον αριθμό 2) και την τιμή του.
- Μόλις γίνεται η εισαγωγή της τιμής, το πρόγραμμα θα καταγράφει την ώρα του συστήματος και θα εμφανίζει το όνομα του αισθητήρα, την τιμή του και την ώρα του συστήματος.

Λύση


```python
import time

sensors = ["Temperature_sensor1", "Temperature_sensor2", "Humidity_sensor", "ultrasonic_sensor"]
n_sensors = 4

sensor_id = int(input("Enter number of the sensor: "))
sensor_value = int(input("Enter the value of the sensor: "))
t = time.localtime()
current_time = time.strftime("%H:%M:%S", t)
print(sensors[sensor_id], sensor_value, current_time)
```

```
Enter number of the sensor: 1
Enter the value of the sensor: 34
Temperature_sensor2 34 18:58:52

Process finished with exit code 0
```

Γρ. 6: Δέχεται ως είσοδο τον αριθμό του αισθητήρα εμφανίζοντας το κατάλληλο μήνυμα

Γρ. 7: Δέχεται ως είσοδο την τιμή του αισθητήρα εμφανίζοντας το κατάλληλο μήνυμα

Γρ. 8-9: καταγράφει την ώρα του συστήματος όπως είδαμε και σε προηγούμενο παράδειγμα

Άσκηση Κατανόησης

- Στην συνέχεια της προηγούμενης άσκησης να γράφει σε ένα αρχείο με όνομα "sensor_data.txt" τα δεδομένα από αισθητήρες με την παρακάτω μορφή:

```
Temperature_sensor1 25 11:40:52
Humidity_sensor 73 11:43:16
Temperature_sensor2 25 11:49:22
ultrasonic_sensor 354 12:40:57
```

- Το πρόγραμμα θα τερματίζεται όταν ο χρήστης δώσει τον αριθμό -1 ως είσοδο επιλογή αισθητήρα και θα εμφανίζει όλα τα εγγεγραμμένα στοιχεία του αρχείου.
- Ακόμα κατά την είσοδο του αριθμού του αισθητήρα να γίνεται έλεγχος εγκυρότητας ότι υπάρχει. Πχ αν έχετε 4 αισθητήρες ο αριθμός θα πρέπει να είναι στο εύρος [0 – 3].

Λύση (1/2)


```
1   import time
2   #                   0                   1                   2                   3
3   sensors = ["Temperature_sensor1", "Temperature_sensor2", "Humidity_sensor", "ultrasonic_sensor"]
4   n_sensors = 4
5
6   while True:
7       sensor_id = int(input("Enter number of the sensor: "))
8       if sensor_id == -1:
9           break
10      elif sensor_id < 0 or sensor_id >= n_sensors:
11          print("Not in range")
12          continue
13      sensor_value = int(input("Enter the value of the sensor: "))
14      t = time.localtime()
15      current_time = time.strftime("%H:%M:%S", t)
```

Γρ. 7: Δέχεται ως είσοδο τον αριθμό του αισθητήρα εμφανίζοντας το κατάλληλο μήνυμα
Γρ. 8: Ελέγχει τον αριθμό τερματισμού. Αν είναι -1 βγαίνει από την επανάληψη στην γρ. 9
Γρ. 10: Ελέγχει αν ο αριθμός είναι στα όρια του πίνακα με τους αισθητήρες, αν δεν είναι εμφανίζει κατάλληλο μήνυμα και με την εντολή continue στην γραμμή 12 προχωράει στην επόμενη επανάληψη (δλδ. Στην γραμμή 6)
Γρ. 13: Δέχεται ως είσοδο την τιμή του αισθητήρα εμφανίζοντας το κατάλληλο μήνυμα
Γρ. 14-15: καταγράφει την ώρα του συστήματος όπως είδαμε και σε προηγούμενο παράδειγμα

Λύση (2/2)


```
16      try:
17          f = open("sensor_data.txt", "a")
18          f.write(sensors[sensor_id])
19          f.write("  ")
20          f.write(str(sensor_value))
21          f.write("  ")
22          f.write(current_time)
23          f.write("\n")
24          f.close()
25      except IOError:
26          print("Error writing the file.. try again..")
27  print()
28  try:
29      f = open("sensor_data.txt", "r")
30      print(f.read())
31  except IOError:
32      print("Error reading the file.. ")
```

Όπως είδαμε και στα παραδείγματα, ανοίγουμε το αρχείο στην *γραμμή 17*.

Γρ. 18: γράφουμε το όνομα του αισθητήρα.

Γρ. 19, 21: γράφουμε κενά στο αρχείο

Γρ. 20: Επειδή η τιμή του αισθητήρα είναι τύπου integer τον μετατρέπουμε σε τύπο string και τότε τον γράφουμε

Γρ. 22: γράφουμε την ώρα στο αρχείο

Γρ. 23: Αλλάζουμε γραμμή

Γρ. 25-26, 31-32: Σε περίπτωση που δεν ανοίξει το αρχείο εμφανίζουμε κατάλληλο μήνυμα

Πίνακας NumPy – Άντληση δεδομένων από αρχείο .txt (1/2)

- Ο πιο απλός τρόπος για να αποθηκευτούν δεδομένα, όπως είδαμε, είναι η αποθήκευση τους σε ένα αρχείο .txt
- Αν θέλουμε, για παράδειγμα, να αποθηκεύσουμε συντεταγμένες [x, y, z] σε ένα τέτοιο αρχείο θα πρέπει να βρεθεί ένας τρόπος να αντληθούν από αυτό το αρχείο για να αξιοποιηθούν
- Πχ. Έχουμε αποθηκεύσει στο παρακάτω αρχείο συντεταγμένες διαχωρισμένες με «|»

x y z

```
100|40|210
123|342|323
421|543|125
254|432|564
```

Πίνακας NumPy – Άντληση δεδομένων από αρχείο .txt (2/2)

- Η Βιβλιοθήκη NumPy δίνει τη δυνατότητα να αντληθούν αυτά τα δεδομένα και να αποθηκευτούν απευθείας σε μορφή πίνακα σε μία μόνο γραμμή όπως φαίνεται παρακάτω:

```
import numpy as np
coordinates = np.genfromtxt('coordinates.txt', delimiter = '|', dtype = 'int')
print(coordinates)
```

- Η συνάρτηση genfromtxt() δέχεται ως παραμέτρους: (i) το όνομα του αρχείου, (ii) τον διαχωριστή (με την μορφή: delimiter = ‘’), και (iii) τον τύπο (με την μορφή: dtype = ‘’).
- Περισσότερες πληροφορίες σχετικά με την συνάρτηση: https://numpy.org/doc/stable/reference/generated/numpy.genfromtxt.html

Random

- Πολλές φορές στα προγράμματα που υλοποιούνται χρειάζεται η παραγωγή τυχαίων αριθμών
- Παρακάτω είναι ένα εύκολο παράδειγμα παραγωγής τυχαίων αριθμών με την χρήση της βιβλιοθήκης random:

```
import random


print(random.randint(0, 500))
```

- Η συνάρτηση randint() δέχεται ως ορίσματα τα κλειστά όρια των αριθμών που θα παραχθούν. Δηλαδή στο παραπάνω παράδειγμα θα εμφανιστεί τυχαία ένας αριθμός στο διάστημα [0-500]

Άσκηση Κατανόησης

- Να δημιουργήσετε μια συνάρτηση που θα παράγει τυχαία 100 συντεταγμένες [x, y, z] και θα τις γράφει σε αρχείο, διαχωρισμένες με ‘ | ’
- Οι τιμές θα είναι στο διάστημα [0 – 500]
- Να δημιουργήσετε συνάρτηση που θα αντλεί τις συντεταγμένες από το αρχείο, θα τις τοποθετεί σε έναν πίνακα και θα τις εμφανίζει από εκεί

Λύση


```
1   import numpy as np
2   import random
3
4   for i in range (0, 100):
5       try:
6           f = open("coordinates.txt", "a")
7           f.write(str(random.randint(0, 500)))
8           f.write("|")
9           f.write(str(random.randint(0, 500)))
10          f.write("|")
11          f.write(str(random.randint(0, 500)))
12          f.write("\n")
13          f.close()
14      except IOError:
15          print("Error writing the file..")
16
17
18  coordinates = np.genfromtxt('coordinates.txt', delimiter='|', dtype='int')
19
20  print(coordinates)
```

Γρ. 6: ανοίγουμε το αρχείο

Γρ. 7, 9, 11: γράφουμε τον τυχαίο αριθμό στο αρχείο ως string

Γρ. 8, 10: Γράφουμε το «διαχωριστικό» για να μπορούμε να αναγνωρίζουμε τους αριθμούς

Γρ. 18: Ομοίως με το προηγούμενο παράδειγμα (διαφάνεια 25)

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"