Εισαγωγή στην Python

Σημειώσεις μαθήματος

Ενότητα 5 – Βιβλιοθήκες python: Βιβλιοθήκη numPy

Δρ.Ν.Μανδέλλος, Δρ.Φ.Δογάνης

Εισαγωγή στην Python

Στόχοι ενότητας:

- Χρήση βιβλιοθηκών στην Python
- Η βιβλιοθήκη numPy

Εισαγωγή στη NumPy

Τι είναι η NumPy;

- Η **NumPy** είναι μια βιβλιοθήκη της Python για αριθμητικούς υπολογισμούς.
- Ειδικεύεται στη διαχείριση και επεξεργασία μεγάλων, πολυδιάστατων πινάκων (arrays).
- Είναι ιδανική για μαθηματικές πράξεις υψηλής απόδοσης, όπως πράξεις διανυσμάτων και πινάκων.

Γιατί να χρησιμοποιήσουμε την NumPy;

- Η NumPy είναι πιο γρήγορη από τις εγγενείς λίστες της Python για αριθμητικούς υπολογισμούς, καθώς η αποθήκευση δεδομένων είναι συνεχής στη μνήμη και οι πράξεις είναι βελτιστοποιημένες.

Εισαγωγή και Δημιουργία Arrays

Δήλωση της NumPy για να μπορούμε να την χρησιμοποιήσουμε.

Αρχικά θα πρέπει να δηλώσουμε τη βιβλιοθήκη.

Συνήθως γίνεται με το ψευδώνυμο (Alias) np:

Παράδειγμα:

```
import numpy as np
```

Δημιουργία ενός Array

Η NumPy παρέχει τη δυνατότητα να μετατρέπουμε λίστες της Python σε arrays.

Παράδειγμα Array:

Δημιουργία Array

```
import numpy as np

my_list = [1,2,3,4,5]
np_array = np.array(my_list)
print(np_array)
```

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

Πολυδιάστατα Arrays

Δημιουργία 2D ή μεγαλύτερων διαστάσεων πινάκων.

Παράδειγμα 2D Array:

Δημιουργία 2D Array

```
import numpy as np

my_list = [[11,12,13,14,15], [21,22,23,24,25]]
np_array = np.array(my_list)
print(np_array)
```

```
[[11 12 13 14 15]
 [21 22 23 24 25]]
```

Χρήσιμες Συναρτήσεις Δημιουργίας Arrays

Συνάρτηση	Συνάρτηση
np.zeros(): Δημιουργεί πίνακα με μηδενικά.	**np.ones():** Δημιουργεί πίνακα γεμάτο με άσσους.
import numpy as np *zeros_array = np.zeros((2, 3)) # 2 γραμμές, 3 στήλες* *print(zeros_array)*	*import numpy as np* *ones_array = np.ones((3, 2)) # 3 γραμμές, 2 στήλες* *print(ones_array)*
np.arange(): Δημιουργεί ακολουθία αριθμών με συγκεκριμένο βήμα.	**np.linspace():** Δημιουργεί σειρά αριθμών με συγκεκριμένο αριθμό βημάτων.
import numpy as np *range_array = np.arange(0, 10, 2) # από 0 έως 10 με βήμα 2* *print(range_array*	import numpy as np linspace_array = np.linspace(0, 1, 5) # 5 τιμές από 0 έως 1 print(linspace_array)

Βασικές Πράξεις και Συναρτήσεις Πινάκων

Πρόσβαση σε στοιχεία ενός array:

```
import numpy as np

array = np.array([10, 20, 30, 40])
print(array[1]) # 20
```

Βασικές Πράξεις και Συναρτήσεις Πινάκων

Πράξεις σε Arrays

Άθροιση, Αφαίρεση, Πολλαπλασιασμός, Διαίρεση (στοιχείο-προς-στοιχείο):

```
import numpy as np

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print(a + b) # [5, 7, 9]
print(a - b) # [-3, -3, -3]
print(a * b) # [4, 10, 18]
print(a / b) # [0.25, 0.4, 0.5]
```

Βασικές Πράξεις και Συναρτήσεις Πινάκων

Πράξεις σε Arrays

Πράξεις με αριθμούς: Μπορούμε να πολλαπλασιάσουμε ή να προσθέσουμε όλους τους αριθμούς του πίνακα με έναν αριθμό.

```
import numpy as np

print(a * 2) # [2, 4, 6]
```

Συναρτήσεις του NumPy για Στατιστική

- **np.mean()**: Μέση τιμή
- **np.sum()**: Άθροισμα
- **np.min(), np.max()**: Ελάχιστη και μέγιστη τιμή
- **np.std()**: Τυπική απόκλιση

```
import numpy as np

data = np.array([1, 2, 3, 4, 5])
print(np.mean(data)) # 3.0
print(np.sum(data)) # 15
print(np.min(data)) # 1
print(np.max(data)) # 5
print(np.std(data)) # 1.4142135623730951
```

Πράξεις μεταξύ Πινάκων και Διανυσμάτων

Πίνακας (Matrix) Πολλαπλασιασμός:

Ο πολλαπλασιασμός πινάκων μπορεί να γίνει με τη χρήση της συνάρτησης ***np.dot()*** *ή του operator* ***@****.*

```
import numpy as np

matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
result = np.dot(matrix1, matrix2) # ή matrix1 @ matrix2 print(result)
```

Πράξεις μεταξύ Πινάκων και Διανυσμάτων

Πίνακας (Matrix) Αντιστροφή:

Το αντίστροφο ενός πίνακα (A^-1), ***numPy.linalg.inv()***

```
import numpy as np

matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
result = np.linalg.inv(A)
```

Πράξεις μεταξύ Πινάκων και Διανυσμάτων

Διανυσματική Πρόσθεση και Αφαίρεση:

Οι πράξεις σε διανύσματα γίνονται στοιχείο-προς-στοιχείο.

```
import numpy as np

vector1 = np.array([1, 2, 3])
vector2 = np.array([4, 5, 6])
print(vector1 + vector2) # [5, 7, 9]
```

Πράξεις μεταξύ Πινάκων και Διανυσμάτων

Μετασχηματισμοί Πινάκων:

Ανάστροφο Πίνακα: np.transpose() ή T.

```
import numpy as np

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

Άσκηση

- **Άσκηση 1**: Δημιουργήστε έναν πίνακα 3x3, δώστε τιμές και βρείτε τη μέση τιμή.
- **Άσκηση 2**: Φτιάξτε δύο διανύσματα μήκους 5 και υπολογίστε το άθροισμα, τη διαφορά και το εσωτερικό γινόμενο.
- **Άσκηση 3**: Δημιουργήστε έναν πίνακα 2x3 και βρείτε τον ανάστροφο πίνακα.
- **Άσκηση 4**: Δημιουργήστε ένα διάστημα αριθμών από 0 έως 10 με βήμα 0.5 και υπολογίστε το άθροισμα και το γινόμενό τους.
- **Άσκηση 5**: Να λυθεί το σύστημα Ax = b, όταν A = [1 2 5, 3 6 8, 4 11 7] και b = [8 1 3]^T

Ασκήσεις

1

```python
import numpy as np

# Δημιουργία πίνακα 3x3 με τιμές
A = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])

# Υπολογισμός μέσης τιμής
mean_value = np.mean(A)

print("Πίνακας 3x3:")
print(A)
print("Μέση τιμή:", mean_value)
```

2

```python
v1 = np.array([1, 2, 3, 4, 5])
v2 = np.array([5, 4, 3, 2, 1])

sum_v  = v1 + v2
diff_v = v1 - v2
dot_v  = np.dot(v1, v2)

print("Άθροισμα:", sum_v)
print("Διαφορά:", diff_v)
print("Εσωτερικό γινόμενο:", dot_v)
```

3

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

B_T = B.T
B_pinv = np.linalg.pinv(B)
```

4

```python
x = np.arange(0, 10.5, 0.5)

S = x.sum()
P = x.prod()   # περιέχει το 0 → γινόμενο = 0

print("Άθροισμα:", S)
print("Γινόμενο:", P)
```

Ασκήσεις

5

```
A = np.array([[1, 2, 5],
              [3, 6, 8],
              [4, 11, 7]])

b = np.array([8, 1, 3])

x = np.linalg.solve(A, b)
print("Λύση x:", x)
```