LAB – 3
PYTHON BASIC PRACTICE – III

210905189
Kushala Sarada A V
Roll No. 33

22/01/2024

```
import numpy as np

# 1. Array creation
A = np.array([2, 5, 10])
print(A.dtype)  # int64

B = np.array([2.4, 10.6, 5.2])
print(B.dtype)  # float64

# Creating a 2-dimensional array
C = np.array([(3, 4, 5), (12, 6, 1)])

# Creating zero matrix of dimension 2x4
Z = np.zeros((2, 4))
print(Z)

# Creating ones matrix of dimension 3x3
O = np.ones((3, 3))
print(O)

# Creating a sequence of data using arange
S = np.arange(10, 30, 5)
print(S)  # [10 15 20 25]

# arange with float arguments
F = np.arange(0, 2, 0.3)
print(F)
# [0.  0.3 0.6 0.9 1.2 1.5 1.8]
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab3$ python3 sample1.py
int64
float64
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[10 15 20 25]
[0.  0.3 0.6 0.9 1.2 1.5 1.8]
```

```
import numpy as np
import random

# Using linspace to specify the total number of elements in the array
S1 = np.linspace(0, 2, 9)
print(S1)
```

```
# [0.   0.25 0.5  0.75 1.   1.25 1.5  1.75 2.  ]

# Random number generation using random module
print(random.choice([1, 2, 3, 4, 5]))  # Pick one number randomly from the list
print(random.choice('python'))  # Pick one character randomly from the string

# Pick one integer between 25 to 50
print(random.randrange(25, 50))

# Pick one integer between 25 to 50 with a step size of 2
print(random.randrange(25, 50, 2))

# Pick a random number between 0 to 1
print(random.random())

# Pick a floating-point number between 5 to 10
print(random.uniform(5, 10))

# Shuffle the elements of the list
my_list = [1, 2, 3, 4, 5]
random.shuffle(my_list)
print(my_list)

# Set the seed to get the same random value during every execution
random.seed(10)
print(random.random())
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab3$ python3 sample2.py
[0.   0.25 0.5  0.75 1.   1.25 1.5  1.75 2.  ]
2
o
38
37
0.2716046003387107
5.736477440962148
[1, 3, 2, 5, 4]
0.5714025946899135
```

```
import numpy as np
a = np.arange(15).reshape(3, 5)
print(a)
print(a.shape)  # (3, 5)
print(a.size)   # 15
print(a.T)      # Transpose to a 5x3 matrix

c = np.arange(24).reshape(2, 3, 4)
print(c)
print(c.shape)  # (2, 3, 4)
print(c[1, ...])  # Equivalent to c[1, :, :]
print(c[..., 2])  # Equivalent to c[:, :, 2]
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab3$ python3 sample3.py
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
(3, 5)
15
[[ 0  5 10]
 [ 1  6 11]
 [ 2  7 12]
 [ 3  8 13]
 [ 4  9 14]]
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
(2, 3, 4)
[[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]
[[ 2  6 10]
 [14 18 22]]
```

```
import numpy as np

a = np.array([20, 30, 40, 50])
b = np.arange(4)
c = a - b
print("a - b:", c)
print("b^2:", b**2)
print("10 * sin(a):", 10 * np.sin(a))
print("a < 35:", a < 35)

A = np.array([[1, 1], [0, 1]])
B = np.array([[2, 0], [3, 4]])
print("A * B (Elementwise product):", A * B)
print("np.dot(A, B) (Matrix product):", np.dot(A, B))

b = np.arange(12).reshape(3, 4)
print("Sum of each column:", b.sum(axis=0))
print("Sum of each row:", b.sum(axis=1))

print("Flattened array:", b.ravel())
B1 = b.reshape(2, 6)
print("Reshaped matrix (4x5):", B1)
```

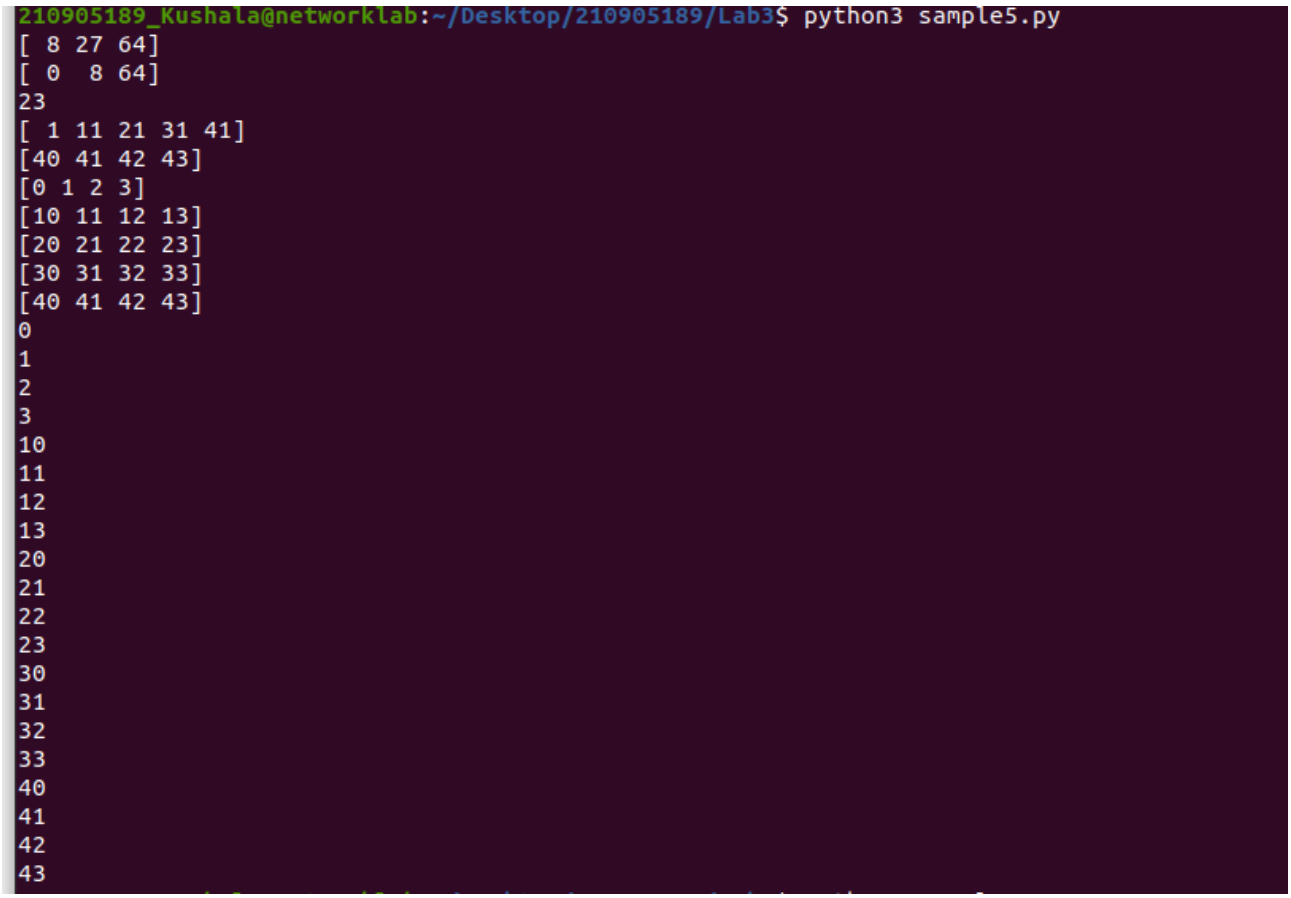
```
210905189_Kushala@networklab:~/Desktop/210905189/Lab3$ python3 sample4.py
a - b: [20 29 38 47]
b^2: [0 1 4 9]
10 * sin(a): [ 9.12945251 -9.88031624  7.4511316  -2.62374854]
a < 35: [ True  True False False]
A * B (Elementwise product): [[2 0]
 [0 4]]
np.dot(A, B) (Matrix product): [[5 4]
 [3 4]]
Sum of each column: [12 15 18 21]
Sum of each row: [ 6 22 38]
Flattened array: [ 0  1  2  3  4  5  6  7  8  9 10 11]
Reshaped matrix (4x5): [[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]
```

```
import numpy as np
a = np.arange(10)**3
print(a[2:5])
print(a[0:6:2])

b = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23], [30, 31, 32, 33], [40, 41, 42, 43]])
print(b[2, 3])
print(b[0:5, 1])
print(b[-1, :])

for row in b:
    print(row)

for element in b.flat:
    print(element)
```




```
import numpy as np

# Array Operations
a = np.array([20, 30, 40, 50])
b = np.arange(4)
c = a - b
print("a - b:", c)
print("b^2:", b**2)
print("10 * sin(a):", 10 * np.sin(a))
print("a < 35:", a < 35)

# Matrix Operations
A = np.array([[1, 1], [0, 1]])
B = np.array([[2, 0], [3, 4]])
print("A * B (Elementwise product):", A * B)
print("np.dot(A, B) (Matrix product):", np.dot(A, B))

# Another Matrix Sum Operations
b = np.arange(12).reshape(3, 4)
print("Sum of each column:", b.sum(axis=0))
print("Sum of each row:", b.sum(axis=1))

# Changing the Shape of a Matrix
print("Flattened array:", b.ravel())
try:
    B1 = b.reshape(4, 5)  # Raises ValueError
except ValueError as e:
    print("Error:", e)
```

```
# Stacking Operations
A1 = np.array([(3, 4, 5), (12, 6, 1)])
A2 = np.array([(1, 2, 6), (-4, 3, 8)])
D1 = np.vstack((A1, A2))
print("Vertical Stack D1:\n", D1)

D2 = np.hstack((A1, A2))
print("Horizontal Stack D2:\n", D2)

a = np.array([4., 2.])
b = np.array([3., 8.])
result_column_stack = np.column_stack((a, b))
print("Column-wise Stack:\n", result_column_stack)

# Indexing with Array of Indices
a = np.arange(12)**2
i = np.array([1, 1, 3, 8, 5])
result_i = a[i]
print("Indexing with array i:", result_i)

j = np.array([[3, 4], [9, 7]])
result_j = a[j]
print("Indexing with array j:\n", result_j)
```

```
43
210905189_Kushala@networklab:~/Desktop/210905189/Lab3$ python3 sample6.py
a - b: [20 29 38 47]
b^2: [0 1 4 9]
10 * sin(a): [ 9.12945251 -9.88031624  7.4511316  -2.62374854]
a < 35: [ True  True False False]
A * B (Elementwise product): [[2 0]
 [0 4]]
np.dot(A, B) (Matrix product): [[5 4]
 [3 4]]
Sum of each column: [12 15 18 21]
Sum of each row: [ 6 22 38]
Flattened array: [ 0  1  2  3  4  5  6  7  8  9 10 11]
Error: cannot reshape array of size 12 into shape (4,5)
Vertical Stack D1:
 [[ 3  4  5]
 [12  6  1]
 [ 1  2  6]
 [-4  3  8]]
Horizontal Stack D2:
 [[ 3  4  5  1  2  6]
 [12  6  1 -4  3  8]]
Column-wise Stack:
 [[4. 3.]
 [2. 8.]]
Indexing with array i: [ 1  1  9 64 25]
Indexing with array j:
 [[ 9 16]
 [81 49]]
210905189_Kushala@networklab:~/Desktop/210905189/Lab3$ python3 sample7.py
```

```
import numpy as np

# Mapping by Value
a = np.array([(3, 2, 9), (1, 6, 7)])
```

```
s1 = 0

for row in a:
    for col in row:
        s1 += col

print("Sum using value mapping:", s1)

# Mapping by Index
a = np.array([(3, 2, 9), (1, 6, 7)])
s = 0

for i in range(a.shape[0]):
    for j in range(a.shape[1]):
        s += a[i, j]

print("Sum using index mapping:", s)
```

```
210905189_Kushala@networklab:~/Desktop/210905189/Lab3$ python3 sample7.py
Sum using value mapping: 28
Sum using index mapping: 28
```