Data Science & Artificial Intelligence

Python For Data Science

Classes & Modules

Lecture No.- 03

By- Satya sir

Recap of Previous Lecture

- Set methods | Operations
- Dictionaries
- Functions
  - Recursion

Topics to be Covered

- Examples related to numpy module
- Collections module
- OS module
- Sys module.

Topic : Modules – datetime, math, numpy

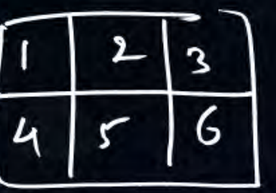
Example – 1

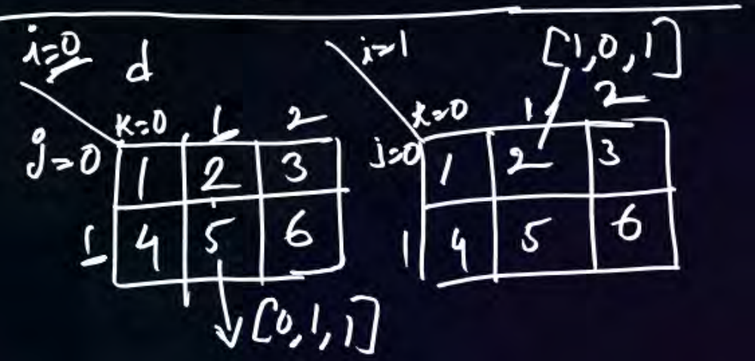
```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr) # [1,2,3,4,5]
print(np.__version__)
```

Example – 2

```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim) # 0
print(b.ndim) # 1
print(c.ndim) # 2
print(d.ndim) # 3
```

dimension of an array

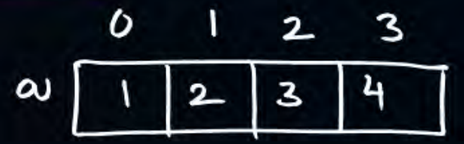


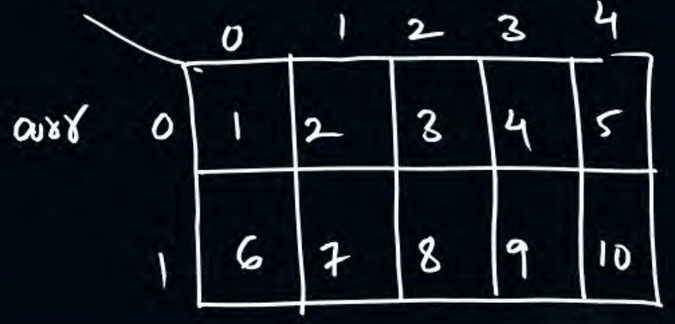


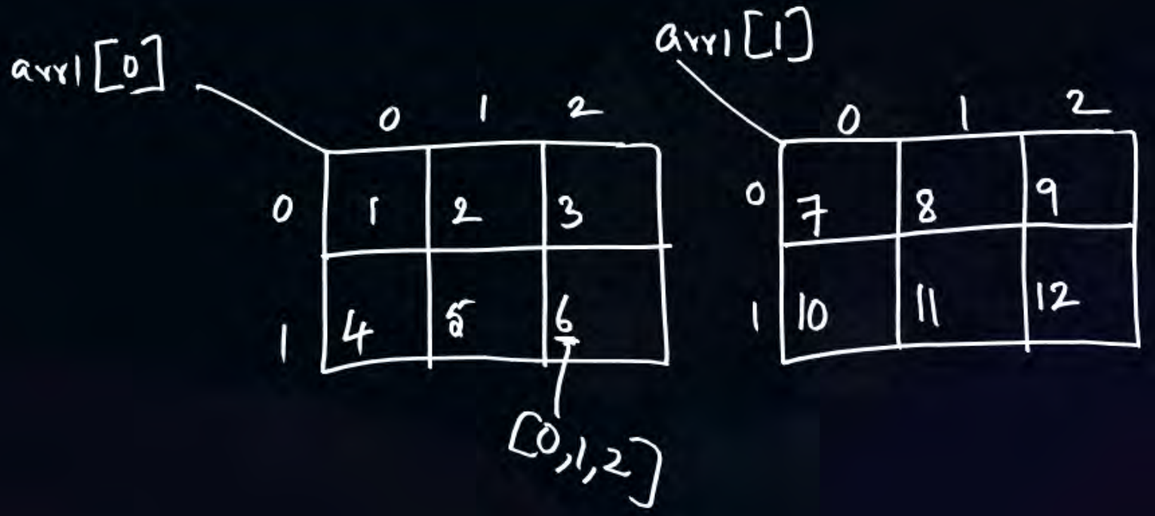
Topic : Modules – datetime, math, numpy

Example – 3

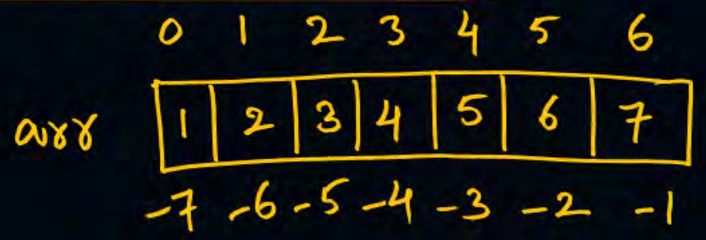
```
import numpy as np

a = np.array([1, 2, 3, 4])
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
arr1 = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(a[2] + a[3])  # 3+4 = 7
print('2nd element on 1st row: ', arr[0, 1])# 2
print('5th element on 2nd row: ', arr[1, 4])# 10
print(arr1[0, 1, 2]) # 6
```







Topic : Modules – datetime, math, numpy



Example – 4

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])


print(arr[1:5]) # [2,3,4,5]

print(arr[4:]) # [5,6,7]

print(arr[:4]) # [1,2,3,4]

print(arr[-3:-1]) # [5,6]

print(arr[1:5:2]) # [2, 4]

print(arr[::2]) # [1, 3, 5, 7]
```

Topic : Modules – datetime, math, numpy

Example – 4

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])

print(arr[4:])

print(arr[:4])

print(arr[-3:-1])

print(arr[1:5:2])

print(arr[::2])
```

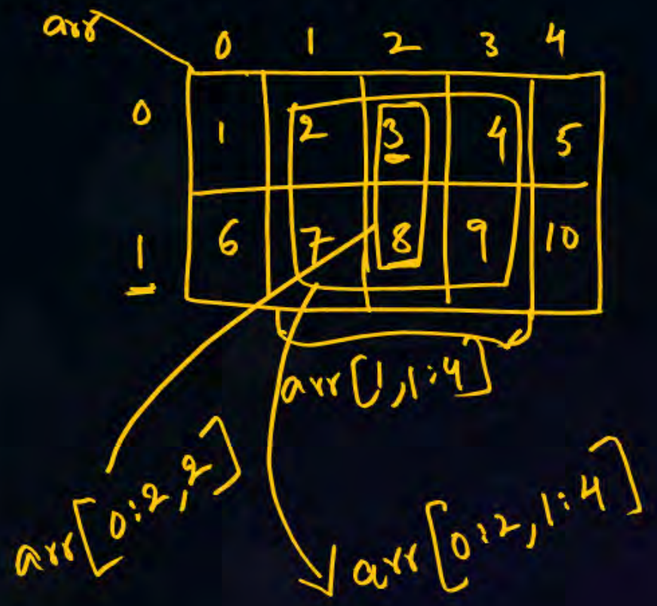
Example – 5

```
import numpy as np

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

print(arr[1, 1:4])

print(arr[0:2, 2])

print(arr[0:2, 1:4])
```

[7, 8, 9]

[3, 8]

[[2, 3, 4]
  [7, 8, 9]]



Topic : Modules – datetime, math, numpy

Example – 6

```
import numpy as np

arr = np.array([1, 2, 3, 4])
arr1 = np.array(['apple', 'banana', 'cherry'])
print(arr.dtype)
print(arr1.dtype)
```

int 64 → Processor configuration in bits (64-bit Processor)

U_x => Unicode type.

Example-7

```
import numpy as np

arr = np.array([1, 2, 3, 4], dtype='i4')
print(arr.dtype)
```

int 32

'i4' → 4 Bytes

Topic : Modules – datetime, math, numpy

Example – 8

```
import numpy as np

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

for x in arr: # for Each row
  for y in x: # for Each col in Each row
    print(y) end = ' ')
          # 1 2 3 4 5 6
```

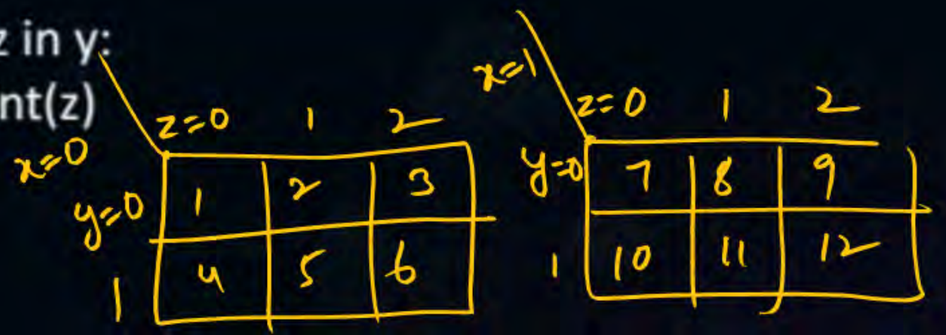
	y=0	1	2
x=0	1	2	3
x=1	4	5	6

1-Darray:

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

for i in a:
    if i%2==0:
        Print(i) # 4
                 # 6
```

Example-9

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

for x in arr:
  for y in x:
    for z in y:
      print(z)
```



Topic : Modules – datetime, math, numpy

Example – 10

```
import numpy as np

arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))
print(arr)

arr = np.concatenate((arr1, arr2), axis=0)
print(arr)

arr = np.concatenate((arr1, arr2), axis=1)
print(arr)
```

arr = [1,2,3,4,5,6]

arr = [[1,2,3] [4,5,6]]

rows

arr = [[1,4] [2,5] [3,6]]

cols

Topic : Modules – datetime, math, numpy

Example-11

```
import numpy as np

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

x = np.where(arr == 4) # array [3,5,6]

print(x)
```

Topic : collections Module

The Python collection module is defined as a container that is used to store collections of data, for example - list, dict, set, and tuple, etc.

It was introduced to improve the functionalities of the built-in collection containers.

Different types of collection modules include:

- NamedTuple
- OrderedDict
- DefaultDict
- Counters
- ChainMap
- DeQue
- UserDict
- UserList
- UserString

Topic : collections Module

A NamedTuple returns a tuple object with names for each position which the ordinary tuples lack.

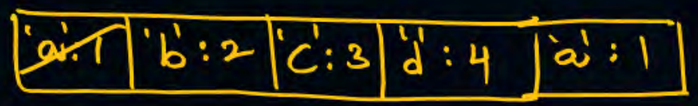
```
from collections import namedtuple

# Declaring namedtuple()
Student = namedtuple('Student',['name','age','DOB'])
                              name   age    DOB
S = Student('XYZ','16','25121996')

print (S[1])      #16  (or) Print( S.age )
print (S.name)    #XYZ
```

Topic : collections Module

An OrderedDict is a dictionary subclass that remembers the order in which keys were first inserted.

The only difference between dict() and OrderedDict() lies in their handling of key order in Python.

```
from collections import OrderedDict

od = OrderedDict()
od['a'] = 1
od['b'] = 2
od['c'] = 3
od['d'] = 4

print('Before Deleting')
for key, value in od.items():
    print(key, value)

od.pop('a')

od['a'] = 1

print('\nAfter re-inserting')
for key, value in od.items():
    print(key, value)
```



{'b':2, 'c':3, 'd':4, 'a':1}

Topic : collections Module

Python Defaultdict is a sub-class of the dictionary class that returns a dictionary-like object.

The functionality of both dictionaries and defaultdict is almost the same except for the fact that defaultdict never raises a KeyError.

It provides a default value for the key that does not exist.

```
from collections import defaultdict

# Create a defaultdict with a default value of an empty list
my_dict = defaultdict(list)

# Add elements to the defaultdict
my_dict['fruits'].append('Orange')
my_dict['vegetables'].append('Tomato')

print(my_dict)
```

O/P: {'fruits': ['Orange'], 'vegetables': ['Tomato']}

```
Print(my_dict['Groceries']) # {'Groceries': None}
```

Topic : collections Module

A counter is a sub-class of the dictionary.

It is used to keep the count of the elements in an iterable in the form of an unordered dictionary where the key represents the element in the iterable and value represents the count of that element in the iterable.

```
from collections import Counter

# With sequence of items
print(Counter(['B','B','A','B','C','A','B','B','A','C']))
        OR
# with dictionary
print(Counter({'A':3, 'B':5, 'C':2}))
        OR
# with keyword arguments
print(Counter(A=3, B=5, C=2))
```

{'A', 'A', 'A', 'B', 'B', 'B', 'B', 'B', 'C', 'C'}

Topic : collections Module

A ChainMap encapsulates many dictionaries into a single unit and returns a list of dictionaries.

```
from collections import ChainMap


d1 = {'a': 1, 'b': 2}
d2 = {'c': 3, 'd': 4}
d3 = {'e': 5, 'f': 6}

# Defining the chainmap
c = ChainMap(d1, d2, d3)

print(c)
```

{'a':1, 'b':2, 'c':3, 'd':4, 'e':5, 'f':6}

Topic : collections Module

Deque (Doubly Ended Queue) is the optimized list for quicker append and pop operations from both sides of the container.

It provides O(1) time complexity for append and pop operations as compared to list with O(n) time complexity.

```
from collections import deque

# initializing deque
de = deque([1,2,3])

de.append(4)
```


```
print ("The deque after appending at right is : ")
print (de)

de.appendleft(6)
```


```
print ("The deque after appending at left is : ")
print (de)
```

Topic : os Module

Python has a built-in os module with methods for interacting with the operating system, like creating files and directories, management of files and directories, input, output, environment variables, process management, etc.

Method	Description
os._exit()	Exits the process with the specified status
os.abort()	Terminates a running process immediately
os.chdir()	Change the current working directory
os.fdopen()	Returns an open file object connected to a file
os.fork()	Forks a child process
os.getpid()	Returns the process id of the current process
os.getppid()	Returns the parent process id of the current process
os.getpriority()	Returns the scheduling priority of a process, process group, or user

Topic : sys Module

- The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter.

- The sys modules provide variables for better control over input or output. We can even redirect the input and output to other devices. This can be done using three variables –

1. stdin
2. stdout
3. stderr

Topic : sys Module

```
import sys
for line in sys.stdin:
    if 'x' == line.rstrip():
        break
    print(f'Input : {line}')

print("Exit")
```

```
o/p:  Welcome
      To
      Sys module
      x
```

o/p: Welcome To Sys module

```
import sys
sys.stdout.write('GATE')
```

GATE

Module object method

```
import sys
def print_to_stderr(*a):
    print(*a, file = sys.stderr)

print_to_stderr("Hello World")
```

2 mins Summary

- numpy module
- Collections module
- Sys module
- OS module

Tomorrow last class : Practice Session : Complete discussed Syllabus.

THANK - YOU