APSSDC

Andhra Pradesh State Skill Development Corporation

(https://apssdc.in)

Python Overview & Data Manipulation with NumPy

In [1]:

```
1 print("Hello World")
```

```
Hello World
```

Literate Programming (http://www.literateprogramming.com/knuthweb.pdf)

1. **Programming Environment**
    - Jupyter Notebook Environment
    - Markdown format for documentation
        - Headings
        - Lists
2. **Python Overview**
    - input/output
    - Conditional Statements in python
        - if
        - if else
        - if-elif-else
    - List
    - Tuple
    - Dictionary
    - Files
    - Modules & Packages
3. **Data Manipulation with NumPy**
    - Introduction
    - NumPy Arrays
    - NumPy Basics
    - Math
    - Random
    - Indexing

Markdown Syntax

Shortcuts in Jupyter Notebook

- **ShiftEnter** - Execute the cell
- *EscM* - code cell to matkdown cell
- ***EscY*** - markdown cell to code cell
- **EscA** - insert the cell above the current cell
- **EscB** - Insert the cell below the current cell
- **EscDD** - Deleting
- **EscL** - Inserting Line Numbers for the Currret Cell

Python Overview

Heading2

Heading3

Heading6

Heading1

- Line1
- line2
  - Sub Point2
    - Sub Sub Point
  - Sub point2

1. Line1
2. Line2
   A. Line 2-1
   B. Line 2-2

In [4]:

```
1 name = input("Enter Your Name")
```

```
Enter Your Name123
```

In [5]:

```
1 print(name)
```

```
123
```

In [6]:

```
1 print(type(name))
```

```
<class 'str'>
```

In [7]:

```
name = int(input("Enter Your Name"))
```

```
Enter Your Name123
```

In [8]:

```
print(type(name))
```

```
<class 'int'>
```

Conditional Statements

- if

Syntax

```
if condition:
    print("Hello World")
```

- if-else

Syntax

```
if condition:
    print("Hello World")
    print("Hello World Again")
else:
    print("Failed Condition")
```

- if-elif-else

Syntax

```
if condition:
    print("Hello World")
    print("Hello World Again")
elif condition:
    print("Hello World Agani and Again")
    print("Final Hello World Again")
else:
    print("Failed Condition")
```

In [9]:

```
if True:
    print("Hello World")
```

```
Hello World
```

- int, float, complex, boolean, string
- List, tuple, Disctionary, Set

In [10]:

```
s1 = 's1'
s2 = "S2"
s3 = '''s3
s4
s5'''
s4 = """s6
s7
s8"""
```

In [11]:

```
bol1, bol2, bol3 = True, False, None

com = 3+5j
a = 5
b = 5.5
```

List

Which is stored Group of dissimilar data type of elements

- We can modify, update, remove existing elements in the list

In [12]:

```
li = [113, 5+6j, 52.5, 'APSSDC', True, None, ["Hey I'm a List"]]

print(li)
print(type(li))
```

```
[113, (5+6j), 52.5, 'APSSDC', True, None, ["Hey I'm a List"]]
<class 'list'>
```

In [13]:

```
li[0] = 115
print(li)
```

```
[115, (5+6j), 52.5, 'APSSDC', True, None, ["Hey I'm a List"]]
```

Tuple - ()

Which is stored Group of dissimilar data type of elements

- We can't modify, update, remove existing elements in the tuple

In [14]:

```
tup1 = (113, 5+6j, 52.5, 'APSSDC', True, None, ["Hey I'm a List"], ("Haha I'm a Tuple")

print(tup1)
```

```
(113, (5+6j), 52.5, 'APSSDC', True, None, ["Hey I'm a List"], "Haha I'm a Tu
ple")
```

In [15]:

```
tup1[0] = 115
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-fa9b6af20081> in <module>
----> 1 tup1[0] = 115

TypeError: 'tuple' object does not support item assignment
```

Dictionary

They are used to store data in `key:value` pairs

```
{"Key":"Value"}
```

In [16]:

```
dic1 = {'APSSDC':"Online Training"}

print(dic1)
```

```
{'APSSDC': 'Online Training'}
```

Files

In [20]:

```
f = open('textFile.txt', mode = 'r')
line = f.readline()
print(line)
print('***********')
data = f.read()
print(data)

print(type(data), type(line))
f.close()
```

```
APSSDC

***********
Online
Data
Analysis
Using
Python
Traning
Program
<class 'str'> <class 'str'>
```

In [21]:

```
f = open('textFile.txt', mode = 'r')
lines = f.readlines()
print(lines)


print(type(lines))
f.close()
```

```
['APSSDC\n', 'Online\n', 'Data\n', 'Analysis\n', 'Using\n', 'Python\n', 'Tra
ning\n', 'Program']
<class 'list'>
```

Modules & Packages

- Module is a single python file created for specific task
- Packages is a bunch of module avalaible on a single folder with specific task

In [22]:

```
import arthematic
```

moduleName.FuntionName

In [23]:

```
1 arthematic.addition(5, 10)
```

Out[23]:

15

In [24]:

```
1 arthematic.pi
```

Out[24]:

3.14

In [26]:

```
1 import arthematic as ar
2
3 ar.addition(50, 1100)
```

Out[26]:

1150

In [29]:

```
1 from arthematic import addition
2
3 addition(50, 500)
```

Out[29]:

550

In [30]:

```
1 from arthematic import addition as add
2
3 add(50000, 500)
```

Out[30]:

50500

```
import packageName.module as alais
from package import module
from package.module import method/class/attribute
from package.module import method/class/attribute as alais
```

Data Manipulation with NumPy

- Introduction
- NumPy Arrays
- NumPy Basics
- Math
- Random
- Indexing

[Numpy Official Website (https://numpy.org/)](https://numpy.org/)

In [31]:

```
1 import numpy as np
```

error:moduleNotFoundError

pip install PackageName

Numpy Array

It is used to store similar group of DataType of elements

Creating of Arrays

In [32]:

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

Out[32]:

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

In [33]:

```
1 type(arr1), type(arr2)
```

Out[33]:

```
(numpy.ndarray, numpy.ndarray)
```

In [35]:

```
arr1.dtype, arr2.dtype
```

Out[35]:

```
(dtype('int32'), dtype('int32'))
```

In [38]:

```
arr3 = np.array([1, 2, 3, 4, 5, 6], dtype = 'int8')
type(arr3), arr3.dtype
```

Out[38]:

```
(numpy.ndarray, dtype('int8'))
```

In [39]:

```
arr4 = np.array([1, 2, 3, 5.5, 6.6, 9.9, 8.8])

arr4, type(arr4), arr4.dtype
```

Out[39]:

```
(array([1. , 2. , 3. , 5.5, 6.6, 9.9, 8.8]), numpy.ndarray, dtype('float6
4'))
```

- strings , complex, float, int --> Strings
- complex, float, int --> Complex
- float, int --> float
- int -> int

In [40]:

```
arr5 = np.array([1, 2, 3, 5.5, 6.6, 9.9, 8.8, 4 + 5j])

arr5, type(arr5), arr5.dtype
```

Out[40]:

```
(array([1. +0.j, 2. +0.j, 3. +0.j, 5.5+0.j, 6.6+0.j, 9.9+0.j, 8.8+0.j,
        4. +5.j]),
 numpy.ndarray,
 dtype('complex128'))
```

In [41]:

```
arr6 = np.array([1, 2, 3, 5.5, 6.6, 9.9, 8.8, 4 + 5j, 'APSSDC'])

arr6, type(arr6), arr6.dtype
```

Out[41]:

```
(array(['1', '2', '3', '5.5', '6.6', '9.9', '8.8', '(4+5j)', 'APSSDC'],
       dtype='<U64'),
 numpy.ndarray,
 dtype('<U64'))
```

- a = 0
- b = 0.0
- c = ''
- li = []
- t1 = ()

In [42]:

```
zeros = np.zeros(50)

zeros
```

Out[42]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [43]:

```
zeros.shape
```

Out[43]:

```
(50,)
```

In [44]:

```
d2 = np.array([[1, 5], [5, 6]])

d2.shape
```

Out[44]:

```
(2, 2)
```

In [46]:

```
zeros = np.zeros((5, 2))

zeros
```

Out[46]:

```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

In [49]:

```
one = np.ones((5, 5), dtype = int)
one
```

Out[49]:

```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

In [56]:

```
one_complex = np.ones((500, 500), dtype = 'complex64')
one_complex
```

Out[56]:

```
array([[1.+0.j, 1.+0.j, 1.+0.j, ..., 1.+0.j, 1.+0.j, 1.+0.j],
       [1.+0.j, 1.+0.j, 1.+0.j, ..., 1.+0.j, 1.+0.j, 1.+0.j],
       [1.+0.j, 1.+0.j, 1.+0.j, ..., 1.+0.j, 1.+0.j, 1.+0.j],
       ...,
       [1.+0.j, 1.+0.j, 1.+0.j, ..., 1.+0.j, 1.+0.j, 1.+0.j],
       [1.+0.j, 1.+0.j, 1.+0.j, ..., 1.+0.j, 1.+0.j, 1.+0.j],
       [1.+0.j, 1.+0.j, 1.+0.j, ..., 1.+0.j, 1.+0.j, 1.+0.j]],
      dtype=complex64)
```

In [57]:

```
one_complex.shape
```

Out[57]:

```
(500, 500)
```

In [60]:

```
one = np.ones((5, 15), dtype = int)
one
```

Out[60]:

```
array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
```

In [61]:

```
one.shape
```

Out[61]:

```
(5, 15)
```

In [67]:

```
d3 = one.reshape(5, 5, 3)
```

In [64]:

```
d3.shape
```

Out[64]:

```
(5, 5, 3)
```

In [65]:

```
1 d3
```

Out[65]:

```
array([[[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]],

       [[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]],

       [[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]],

       [[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]],

       [[1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1],
        [1, 1, 1]]])
```

In [68]:

```
1 one.size
```

Out[68]:

```
75
```

In [69]:

```
1 one.nbytes
```

Out[69]:

```
300
```

In [70]:

```
1 one.dtype
```

Out[70]:

```
dtype('int32')
```

In [73]:

```
1 (32 * 75) // 8
```

Out[73]:

```
300
```