FUNCTIONS.

Write the function of factorial of a number.

```
number = int(input('Enter your number'))
    if number == 0:
        print ('Factorial is 1')
    else:
        for i in range (1, number+1): //1, 6
            factorial = factorial * i
        printf("The factorial of', number, 'is', factorial)
```

TO call a function:

```
def factorial_value(num):
    factorial = 1
    if num == 0:
        return factorial
    else:
        for i in range (1, num+1):
            factorial = factorial * i
        return factorial
```

Numerical Python: numpy

Numpy arrays have 2 advantages over "list" & "tuples".

(i) Numpy arrays allows several mathematical operations to be performed on them in contrast to list & tuples.

(ii) Operations on numpy is way faster compared to list & tuples.

```
np_array = np.array([1,2,3,4])
print(np_array)
[1,2,3,4]
type np_array  // numpy.ndarray
```

1-dimensional Array:-

```
a = np.array([1,2,3,4])
a.shape  // (4,) means 4 columns, 1 row
```

2-dimensional Arrays.

```
b = np.array([[1,2,3,4]])
~~b = np.array([1,2,3,4])~~
b = np.array([[1,2,3,4],[5,6,7,8]])
print(b)  // [[1 2 3 4]
          //  [5 6 7 8]]   Array of 2 rows of 4 col.
```

Declaring data type in numpy arrays.

Initiating Arrays.

```
c = np.array([[1,2,3,4),(5,6,7,8)], dtype=float)
x = np.zeros(4,5)
print(x)  //  [[0. 0. 0. 0. 0.]
          //   [0. 0. 0. 0. 0.]
          //   [0. 0. 0. 0. 0.]
          //   [0. 0. 0. 0. 0.]]
```

Array of ones Value.

```
y = np.ones((3,3))
```

// [[1. 1. 1.]
[1. 1. 1.]
[1. 1. 1.]]

Array of particular value.

```
z = np.full((5,4),5)
print(z)
```

// [[5 5 5 5]
[5 5 5 5]
[5 5 5 5]
[5 5 5 5]]

Identity Matrix

```
c = np.eye(3)
print(c)
```

// [[1 0 0]
[0 1 0]
[0 0 1]]

Array of Random Values:

```
b = np.random.random((3,4))
print(b)
```

// [[0.021 0.011 0.112]
[0.01 0.22 0.411]
[0.031 0.252 0.560]]

Array of random int values with Specific range:

```
d = np.random.randint(3,5)
d = np.random.randint(10,100,(3,5))
print(d)
```

lower 10, Upper 100
(3rows, 5col)

[[11 59 68 54 87]
[96 16 58 86 83]
[87 59.99 42 79]]

Array of Evenly Spaced values. Specifying the number of values required.

```
a = np.linspace(10,30,6)
print(a)  // [10.,14.,18.,22.,26.,30.]
```

Array of Evenly Spaced values, specifying the step.

```
e = np.arange(10,30,5)
print(e)  // [10.,15.,20.,25.]
```

Converting Lists to Array.

```
list_1 = [10,20,30,50]
np_array = np.asarray(list_1)
print(np_array) // [10 20 30 50]
type(np_array) // numpy.ndarray.
```

Analyzing an Array.

Shape of Array = c.shape
No of Dimension of Array = c.ndim
No of Element of Array = c.size
Check datatype of values in the Array = c.dtype.

Mathematic operations of an Np Array.
(+, -, *, /) OR np.add(a,b)
np.subtract(a,b)
np.multiply(a,b)
np.divide(a,b).

Array Manipulation.

```
array = np.random.randint(0,10,(2,3))
trans = np.transpose(array)
```

OR trans2 = array.T

Reshape Arrays.

```
b = a.reshape(3,2)
```

.... from (2,3) shape

...frame(dataset.data, columns = dataset.feature_names)

Pandas-

import pandas as pd

	FileName	Data Type
1.	data	~~Nested~~ Jason
2	nestedjason	Nested Jason
3	numbertext	txt
4	text	txt
5	try	XLS.

Key Syntax for Checking DataFrames-

1) df.shape
2) df.head(15) // (506, 13) (Rows, Col.)
3)

```
df = pd.read_csv(path)
df = pd.Read_excel(path)
df = pd.read_json(path)
```

Exporting a DataFrame to csv

~~df = pd.read_~~ df = pd.to_csv("filename")