EXP NO:1 PRACTICE BASIC PYTHON FUNCTIONS AND LIBRARIES FOR DATA SCIENCE PROJECT

AIM:

To implement the basic python functions and libraries for data science.

PROCEDURE:

Random Package:

The random module is part of the Python standard library and provides functions for generating random numbers. It is commonly used in simulations, games, and other scenarios where randomness is required.

Functions Used:

random.choice(sequence): Returns a randomly selected element from the given sequence.

random.randint(a, b): Returns a random integer between a and b, inclusive.

Statistics Package:

The statistics module is part of the Python standard library and offers functions for statistical operations on numeric data. It is helpful for calculating measures such as mean, median, mode, etc.

Functions Used:

statistics.mean(data): Calculates the mean (average) of a sequence of numeric values.

statistics.median(data): Computes the median (middle value) of a sequence of numeric values.

statistics.mode(data): Finds the mode (most common value) in a sequence of numeric values.

Math Package:

The math module is part of the Python standard library and provides mathematical functions and constants. It is useful for more advanced mathematical operations.

Functions Used:

math.sqrt(x): Returns the square root of x.

In [ ]:
```
# Datatypes
print("DATATYPES")
print()

x=5
print(x,type(x))
print()

x='Hello'
print(x,type(x))
print()

x=40.6
print(x,type(x))
print()

x=1j+4
print(x,type(x))
print()

x=True
print(x,type(x))
print()
```

```
DATATYPES

5 <class 'int'>

Hello <class 'str'>

40.6 <class 'float'>

(4+1j) <class 'complex'>

True <class 'bool'>
```

In [ ]:
```
x=[1,'Hello World',44.6,1+2j] # List is ordered , changeable and indexed . Allows D
print(x[2])
print(x,type(x))
print()

x=(1,'Hello World',44.6,1,1+2j) # Tuple is ordered , unchangeable and indexed . All
print(x,type(x))
print(x[2])
#x[2]=0 #Error occurs as Tuple is unchangeable
print()
x={1,2,1,33,44,2,67,877,332,44}  # Set is unordered , unchangeable and non-indexed
# print(x[2])  # Error occurs as Set is not indexed.
# x[2]=0       #Error occurs as Set is unchangeable
print(x,type(x))
print()
x={"Name":"Shabari","Age":"19","Place":"Svg"} # Dictionary is ordered and changeabl
print(x["Name"]) # Indexed using Key Values
x["Age"]="20"
```

```
print(x)
print()
```

```
44.6
[1, 'Hello World', 44.6, (1+2j)] <class 'list'>

(1, 'Hello World', 44.6, 1, (1+2j)) <class 'tuple'>
44.6

{1, 2, 67, 33, 332, 44, 877} <class 'set'>

Shabari
{'Name': 'Shabari', 'Age': '20', 'Place': 'Svg'}
```

In [ ]:

```
from numpy import random
import numpy as np
import statistics


#REPLACE WITH MEAN VALUES
y=random.randint(7,size=(10,6)) #create a 2d array of size 10 x 6. Range 0 to 7

print("MEAN")
for i in range(10):
 x=statistics.mean(y[i]) #find mean of each row
 print(x,end=" ")
 for j in range(6):
     if y[i][j]==0:  #if array value is 0 . fill with row mean that is x
         y[i][j]=x
print()
print(y)

#REPLACE WITH MEDIAN VALUES

y=random.randint(7,size=(10,6)) #create a 2d array of size 10 x 6. Range 0 to 7

print("MEDIAN")

for i in range(10):
 x=statistics.median(y[i]) #find median of each row
 print(x,end=" ")
 for j in range(6):
     if y[i][j]==0:  #if array value is 0 . fill with row median that is x
         y[i][j]=x
print()
print(y)
print()

#REPLACE WITH MODE VALUES

names = ['Shabari','Shaye','Keer','Sak','Pavi', 'Amir','Gokul',""]

namearray=np.empty((5, 5),dtype='U10')

for i in range(5):
```

```
    for j in range(5):
        namearray[i][j]=random.choice(names)

print()
print(namearray)

print("MODE")

for i in range(5):
    x=statistics.mode(namearray[i]) #mode gives the frequently used value in a row
    print(x,end=" ")
    for j in range(5):
     if namearray[i][j]==""  and x!="":
            namearray[i][j]=x
print()
print(namearray)
```

```
MEAN
3 2 2 3 3 4 3 3 3 4
[[3 6 3 6 2 4]
 [3 6 1 3 2 2]
 [5 2 1 3 3 2]
 [2 6 2 2 4 6]
 [4 2 1 2 3 6]
 [4 1 3 6 6 6]
 [6 4 2 3 5 2]
 [3 5 1 1 5 6]
 [5 1 4 4 6 2]
 [3 6 3 3 6 4]]
MEDIAN
1.0 4.0 1.5 3.0 2.0 1.0 4.0 1.5 3.5 3.0
[[1 4 4 1 1 1]
 [5 3 3 1 6 5]
 [1 1 4 2 6 1]
 [3 4 3 6 3 1]
 [4 2 2 2 3 2]
 [1 1 1 6 1 3]
 [5 6 5 3 4 2]
 [4 1 2 1 2 1]
 [5 4 3 1 3 6]
 [2 4 3 4 6 2]]


[['Shaye' 'Amir' 'Gokul' 'Keer' 'Gokul']
 ['Keer' 'Gokul' 'Amir' 'Pavi' '']
 ['Keer' '' 'Gokul' 'Pavi' 'Amir']
 ['Gokul' 'Pavi' '' '' 'Pavi']
 ['Sak' '' '' 'Shaye' 'Sak']]
MODE
Gokul Keer Keer Pavi Sak
[['Shaye' 'Amir' 'Gokul' 'Keer' 'Gokul']
 ['Keer' 'Gokul' 'Amir' 'Pavi' 'Keer']
 ['Keer' 'Keer' 'Gokul' 'Pavi' 'Amir']
 ['Gokul' 'Pavi' 'Pavi' 'Pavi' 'Pavi']
 ['Sak' 'Sak' 'Sak' 'Shaye' 'Sak']]
```

In [ ]:

```
from numpy import random
import random as rm

#implementation of loops,control flow and function

def setmat(mat):
    for i in range(6):
        for j in range(10):
            if(j%2==0):
                mat[i][j]=rm.randrange(1,49)
            else:
                mat[i][j]=rm.randrange(50,99)
    return mat

# 10 x 6 matrix
mat=[[0]*10]*6
mat=setmat(mat)
print(*mat,sep="\n",end="\n")
print()

x=random.randint(100,size=(10,6))
print(x)
print()

y=random.rand(10,6)
print(y)
print()

x=lambda a,b: a**b

print(x(random.randint(3,5),2))
```

```
[25, 79, 29, 56, 41, 50, 13, 76, 7, 60]
[25, 79, 29, 56, 41, 50, 13, 76, 7, 60]
[25, 79, 29, 56, 41, 50, 13, 76, 7, 60]
[25, 79, 29, 56, 41, 50, 13, 76, 7, 60]
[25, 79, 29, 56, 41, 50, 13, 76, 7, 60]
[25, 79, 29, 56, 41, 50, 13, 76, 7, 60]

[[75 22 20 92 95 92]
 [22 56 67 71 34 24]
 [95 13 53 33 22 89]
 [33 46 23 99 83 56]
 [50 99 46 76 57 91]
 [75 55 89 85 60 40]
 [85 93 87 14 62 33]
 [12 82 39 64 91 10]
 [46 68 18 14  7 15]
 [53 67 97 75 14 46]]

[[0.68335336 0.16359407 0.11714192 0.11352607 0.8071581  0.71513766]
 [0.77799079 0.69420095 0.19983748 0.8291813  0.9215405  0.07614824]
 [0.39860563 0.43345158 0.28204396 0.3832175  0.62381106 0.79384455]
 [0.09154151 0.7270594  0.22430312 0.42614028 0.25619172 0.53172393]
 [0.50473597 0.68794181 0.24946434 0.48247439 0.04271954 0.37715978]
 [0.58606121 0.99402316 0.53126003 0.28091322 0.8034968  0.97291772]
 [0.4609389  0.80493505 0.09603707 0.86439687 0.33914114 0.78158141]
 [0.62217398 0.72062511 0.49277955 0.37922285 0.67539659 0.59947363]
 [0.94625482 0.6358249  0.70699416 0.57672426 0.99114535 0.44850877]
 [0.62496736 0.14182768 0.46338506 0.65794606 0.05235585 0.48456386]]

16
```