[41]:
```
passc(21,23)
```

```
Disqualified
```

8 NUMPY

It is a scientific package used for computing in Python. It provides a Python Library that is used for working with multidimensional arrays.

8.1 Array

An array is a grid of values and it contains information about the raw data.

[42]:
```
import numpy as np
```

[43]:
```
# 0-D Array(Scalar)
a=21
b=np.array([22])
print(b)
```

```
[22]
```

[44]:
```
# 1-D Array (it contains 0-D array as elements)
d=np.array([1,2,3])
print(d)
```

```
[1 2 3]
```

[45]:
```
# 2-D Array(it contains 1-D Array as elements)
e=np.array([4,5,6])
print(np.array([d,e]))
```

```
[[1 2 3]
 [4 5 6]]
```

[46]:
```
# Create 3 by 3 matrix
print(np.array([[1,2,3],[4,5,6],[7,8,9]]))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

[47]:
```
# Task (create a 2 by 4 matrix)
b=np.array([1,2,3,4])
c=np.array([0,1,2,3])
print(np.array([b,c]))
```

```
[[1 2 3 4]
 [0 1 2 3]]
```

[48]:
```
# Sum of matrices
A=np.array([[1,2],[1,5]])
B=np.array([[0,3],[3,2]])
print(A)
print(B)
print(A+B)
```

```
[[1 2]
 [1 5]]
[[0 3]
 [3 2]]
[[1 5]
 [4 7]]
```

[49]:
```
# Matrix Multiplication
print(np.matmul(A,B))
```

```
[[ 6  7]
 [15 13]]
```

[50]:
```
# Componentwise Multilpication(when order of matrices are same.)
x=A*B
print(x)
```

```
[[ 0  6]
 [ 3 10]]
```

[51]:
```
# Scalar multiplication
b=2*A
print(b)
```

```
[[ 2  4]
 [ 2 10]]
```

[52]:
```
# Order of Matrix
print(np.shape(b))
```

```
(2, 2)
```

[53]:
```
# Special matrices
```

[54]:
```
# Matrix of all entry 1
c=np.ones([2,3],dtype=int)
print(c)
```

```
[[1 1 1]
 [1 1 1]]
```

[55]:
```
# Matrix of all entry as 0
d=np.zeros([3,4],dtype=int)
print(d)
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

[56]:
```
# Identity Matrix
i=np.identity((3),dtype=int)
print(i)
```

```
[[1 0 0]
 [0 1 0]
 [0 0 1]]
```

[57]:
```
# Diagonal Matrix
e=np.diag([1,2,3,4])
print(e)
```

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
```

[58]:
```
# Determinant of a matrix
print(np.linalg.det(e))
```

```
23.999999999999993
```

[59]:
```
# Transpose of a Matrix
print(np.transpose(d))
```

```
[[0 0 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]]
```

[60]:
```
# Maximum entry in a matrix
print(np.max(e))
```

```
4
```

[61]:
```
# minimum entry in a matrix
print(np.min(e))
```

0

[62]:
```
s=np.array([[1,2,3],[0,5,6],[3,0,2]])
print(s)
```

```
[[1 2 3]
 [0 5 6]
 [3 0 2]]
```

[63]:
```
# Extraction of any entry from a matrix
s[1,2]
```

[63]: 6

[64]:
```
# To get any column of a matrix
s[:,1]
```

[64]: array([2, 5, 0])

[65]:
```
# To get any row of a matrix
s[1,:]
```

[65]: array([0, 5, 6])

[66]:
```
import numpy as np
```

8.1.1 Creates an array with evenly spaced values between the specified start, end, and increment values.

[67]:
```
x1=np.arange(1,10,1)
print(x1)
```

```
[1 2 3 4 5 6 7 8 9]
```

8.1.2 Creates an array with evenly spaced values between specified start and end values, using a specified number of elements

[68]:
```
x2=np.linspace(1,10,12)
print(x2)
```

```
[ 1.          1.81818182  2.63636364  3.45454545  4.27272727  5.09090909
  5.90909091  6.72727273  7.54545455  8.36363636  9.18181818 10.        ]
```

8.2 Random Integers

[69]:
```
np.random.rand(2,4)
```

[69]:
```
array([[0.26995818, 0.23300628, 0.97685512, 0.81074431],
       [0.67427581, 0.30449314, 0.34963785, 0.10362583]])
```

[70]:
```
np.random.randint(10,size=(2,4))
```

[70]:
```
array([[5, 0, 7, 2],
       [2, 0, 9, 0]])
```

[6]:
```
np.random.randint(7,10,size=(2,4))
```

[6]:
```
array([[8, 9, 7, 7],
       [8, 7, 8, 7]])
```

8.3 Meshgrid

[72]:
```
x3=np.array([1,2,3])
x4=np.array([-1,0,1])
X,Y=np.meshgrid(x3,x4)
print(x3,x4,X,Y)
```

```
[1 2 3] [-1  0  1] [[1 2 3]
 [1 2 3]
 [1 2 3]] [[-1 -1 -1]
 [ 0  0  0]
 [ 1  1  1]]
```

[74]:
```
Z=X**3+Y**2+2*X*Y+8
print(Z)
```

```
[[ 8 13 30]
 [ 9 16 35]
 [12 21 42]]
```

8.4 Now find the value of function $Z = X + 10Y$ over $X \in [0, 5]$ and $Y \in [0, 5]$ with increment 1

[9]:
```
x=np.arange(0,6,1)
y=np.arange(0,6,1)
m,n=np.meshgrid(x,y)
Z=n+10*m
print(Z)
```

```
[[ 0 10 20 30 40 50]
 [ 1 11 21 31 41 51]
 [ 2 12 22 32 42 52]
 [ 3 13 23 33 43 53]
```

```
 [ 4 14 24 34 44 54]
 [ 5 15 25 35 45 55]]
```

8.5 Slicing sub arrays from a Matrix

[10]:
```
Z[:3,:3]
```

[10]:
```
array([[ 0, 10, 20],
       [ 1, 11, 21],
       [ 2, 12, 22]])
```

[11]:
```
Z[1:3,2:5]
```

[11]:
```
array([[21, 31, 41],
       [22, 32, 42]])
```

[12]:
```
Z[::2,::2]
```

[12]:
```
array([[ 0, 20, 40],
       [ 2, 22, 42],
       [ 4, 24, 44]])
```

[13]:
```
Z[1:5:2,2:6:2]
```

[13]:
```
array([[21, 41],
       [23, 43]])
```

8.6 Constructing matrices with 1's on subdiagonal or superdiagonal

[14]:
```
A=np.eye(3,dtype=int)
B=np.eye(3,k=1,dtype=int)
C=np.eye(3,k=-1,dtype=int)
A,B,C
```

[14]:
```
(array([[1, 0, 0],
        [0, 1, 0],
        [0, 0, 1]]),
 array([[0, 1, 0],
        [0, 0, 1],
        [0, 0, 0]]),
 array([[0, 0, 0],
        [1, 0, 0],
        [0, 1, 0]]))
```