$Name : Fernando I. A. M. D$

$Index No : 190172K$

In [ ]:
```
import cv2
import numpy as np
import sympy
import matplotlib.pyplot as plt
```

1

In [ ]:
```
for i in range(1,6):
    print(i,':',i**2)
```

```
1 : 1
2 : 4
3 : 9
4 : 16
5 : 25
```

2

In [ ]:
```
for i in range(1,6):
    if not sympy.isprime(i):
        print(i,':',i**2)
```

```
1 : 1
4 : 16
```

3

In [ ]:
```
squares = [i**2 for i in range(1,6)]
for i,j in enumerate(squares,start = 1):
    print(i,":",j)
```

```
1 : 1
2 : 4
3 : 9
4 : 16
5 : 25
```

4

In [ ]:
```
for i,j in enumerate(squares,start = 1):
    if not sympy.isprime(i):
        print(i,":",j)
```

```
1 : 1
4 : 16
```

5

In [ ]:
```
#####    a    #####
A = np.array([[1,2],
              [3,4],
              [5,6]])

B = np.array([[7,8,9,1],
              [1,2,3,4]])

C = np.matmul(A,B)
print(C)
```

```
[[ 9 12 15  9]
 [25 32 39 19]
 [41 52 63 29]]
```

In [ ]:
```
#####    b    #####
A = np.array([[1,2],
              [3,4],
              [5,6]])

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

C = A*B
print(C)
```

```
[[ 3  4]
 [15 16]
 [15  6]]
```

6

In [ ]:
```
A = np.random.randint(0,11,(5,7))
print(A)
B = A[1:4,:2]
print(B)
print("size of resulting array:",np.size(B))
```

```
[[ 2 10  5  0  5  6  0]
 [ 1  5  5  6  9  0  7]
 [ 7  1 10  4  4  3  8]
 [ 5  7  7  9  0  9  9]
 [ 7  9  2  5  1  3  8]]
[[1 5]
 [7 1]
 [5 7]]
size of resulting array: 6
```

7

In [ ]:
```
#Broadcasting examples
#1 Vertical
A = np.array([[1],          #4x1
              [2],
              [3],
              [4]])
B = np.array([[1,2,3,4],     #4x4
```

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

C = A+B
print(C)
```

```
[[ 2  3  4  5]
 [ 7  8  9 10]
 [12  4  5  6]
 [ 8  9 10 11]]
```

In [ ]:

```
#2 Horizonal
A = np.array([1,2,3])           #1x3
B = np.array([[1,2,3],          #3x3
              [4,5,6],
              [7,8,9]])

C = A+B
print(C)
```

```
[[ 2  4  6]
 [ 5  7  9]
 [ 8 10 12]]
```

In [ ]:

```
#3 Vertical and horizontal
A = np.array([1])               #1x1
B = np.array([[1,2,3],          #3x3
              [4,5,6],
              [7,8,9]])

C = A+B
print(C)
```

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

8

In [ ]:

```
#####    a    #####
m, c = 2 , -4
N = 10
x = np . linspace (0 , N-1, N) . reshape (N, 1 )
sigma = 10
y = m*x + c + np . random . normal (0 , sigma , (N, 1 ) )

x1 = np.ones(np.shape(x),int)
X = np.hstack((x1,x))
print(X)
```

```
[[1. 0.]
 [1. 1.]
 [1. 2.]
 [1. 3.]
 [1. 4.]
 [1. 5.]
 [1. 6.]
```

```
[1. 7.]
[1. 8.]
[1. 9.]]
```

In [ ]:

```
#####    b    #####
Z = np.matmul(np.matmul(np.linalg.inv(np.matmul(np.transpose(X),X)),np.transpose(X)),y)
print(Z)
```

```
[[ 4.85758636]
 [-0.23807311]]
```

9

In [ ]:

```
#####    a    #####
def squareRoot(x):
    n=0
    alpha=0
    while True:
        for a in range(1,101):
            y = a*(10**(2*n))
            if x == y:
                alpha = a
                break
        if alpha != 0 :
            break
        n=n+1

    rootx = ((-190/(alpha+20))+10)*(10**n)
    return rootx

#####    b    #####
def NewtonR(pres,a0,x):
    RS = (a0**2)-x
    RS1 = 2*a0
    a1 = a0 - (RS/RS1)
    if abs(a1-a0)<pres:
        return a1
    else:
        return NewtonR(pres,a1,x)
```

In [ ]:

```
#####    c    #####
x = [64,75,100,1600]
for i in x:
    a0 = squareRoot(i)
    print("S:",i,end=',')
    print("s0 =",a0,end=',')
    print("Newton-Raphson method:",NewtonR(0.00001,a0,i))  #precision = 10^-5
```

```
S: 64,s0 = 7.738095238095238,Newton-Raphson method: 8.000000000000094
S: 75,s0 = 8.0,Newton-Raphson method: 8.660254037844386
S: 100,s0 = 8.416666666666666,Newton-Raphson method: 10.0
S: 1600,s0 = 47.22222222222222,Newton-Raphson method: 40.0
```

10

In [ ]:

```
img = cv2.imread("gal_gaussian.png")
```

```
blur = cv2.GaussianBlur(img,(99,99),0)

cv2.namedWindow("image",cv2.WINDOW_AUTOSIZE)
cv2.imshow("image",img)
cv2.waitKey(0)
cv2.imshow("blurred image",blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

11

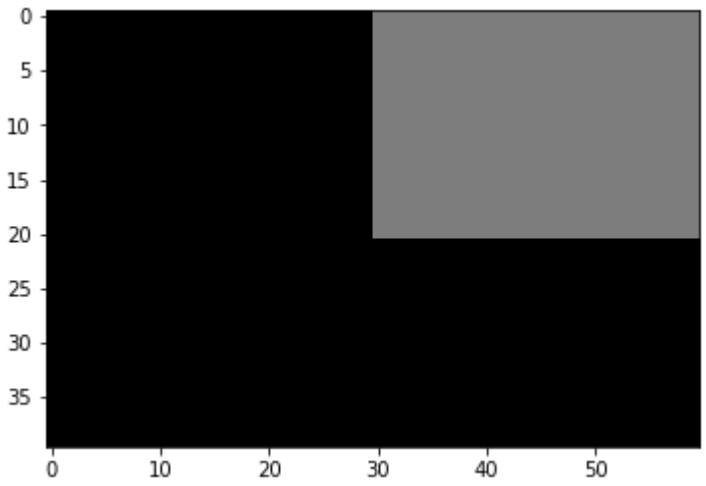
In [ ]:

```
img = cv2.imread("gal_sandp.png")
blur = cv2.medianBlur(img,5)

cv2.namedWindow("image",cv2.WINDOW_AUTOSIZE)
cv2.imshow("image",img)
cv2.waitKey(0)
cv2.imshow("blurred image",blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

12

In [ ]:

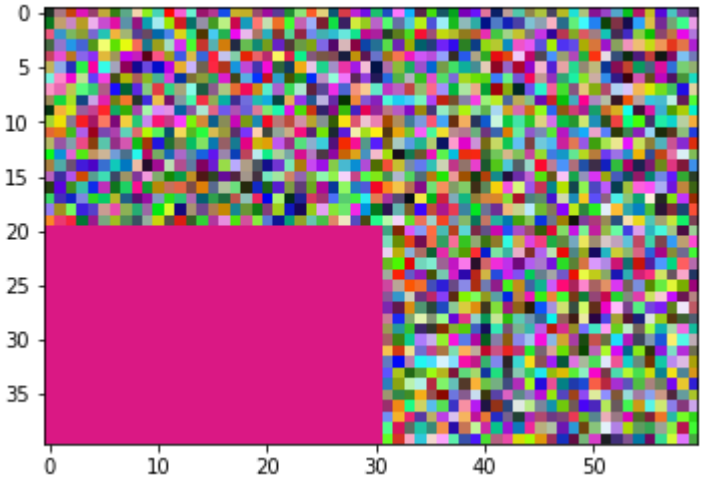
```
img = np.zeros((40,60),dtype=np.uint8)
img[0:21,30:] = 125

plt.imshow(img,cmap='gray',vmin=0,vmax=255)
plt.show()
```




13

In [ ]:

```
img = np.random.randint(0,255,(40,60,3),dtype=np.uint8)
img[20:,0:31,:] = (218,24,132)

plt.imshow(img,cmap='pink',vmin=0,vmax=255)
plt.show()
```




14

In [ ]:

```
img = cv2.imread("tom_dark.jpg")
bright = img+50

cv2.namedWindow("image",cv2.WINDOW_AUTOSIZE)
cv2.imshow("image",img)
cv2.waitKey(0)
cv2.imshow("bright image",bright)
cv2.waitKey(0)
cv2.destroyAllWindows()
```