Name: Treshan Ayesh

IndexNum: 190443T

In [ ]:

```
#Q1)
for i in range(1,6):
    print(i, ':', i**2)
```

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

In [ ]:

```
#Q2)
import sympy as sp
for i in range(1,6):
    if not sp.isprime(i):
        print(i, ':', i**2)
```

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

In [ ]:

```
#Q3)
squares = [i**2 for i in range(1,6)]
for index,val in enumerate(squares):
    print(index+1, ':', val)
```

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

In [ ]:

```
#Q4)
squares = [i**2 for i in range(1,6) if not sp.isprime(i)]
for i in squares:
    print(int(i**0.5), ':', i)
```

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

In [ ]:

```
#Q5) a)
import numpy as np
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 [ ]:

```
#Q5) b)
```

```
A = np.array([[1,2],[3,4],[5,6]])
B = np.array([[3,2],[5,4],[3,1]])
C = np.multiply(A,B)
print(C)
```

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

In [ ]:

```
#Q6)
A = np.random.randint(10,size = (5,7))
print(A)

sub_arr = A[1:4,0:2]
print('Sub array')
print(sub_arr)

print('size of sub array:', sub_arr.size)
```

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

In [ ]:

```
#Q7)
x = np.array([2,4,6])
A = np.array([[1,3,5],[2,4,6]])
print(A * x)
print(A + x)
print(A - x)
```

```
[[ 2 12 30]
 [ 4 16 36]]
[[ 3  7 11]
 [ 4  8 12]]
[[-1 -1 -1]
 [ 0  0  0]]
```

In [ ]:

```
#Q8)
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))

#a)
new_column = np.array([[1]]*N)
X = np.append(new_column,x, axis = 1)


#b)
XT = np.transpose(X)
```

```
C = np.matmul(np.matmul(np.linalg.inv(np.matmul(XT,X)),XT),y)
C
```

Out[ ]:

```
array([[11.27683125],
       [-1.18601575]])
```

In [ ]:

```
#Q9) a)
def square_root(num):
    n = 0
    while not  num // (10**n) < 10:
        n += 1
    a = num / (10**(2*n))
    return (-190/(a + 20) + 10)* (10**n)

#b)
def square_root_newton(num,precision):
    x = square_root(num)
    count = 0
    while (1) :
        count += 1
        root = 0.5 * (x + (num / x))
        if (abs(root - x) < precision) :
            break
        x = root

    return root

#c)
precision = 10**(-5)
nums = [64,75,100,1600]
for i in nums:
    print('square root of',i,':',square_root_newton(i,precision))
```

```
square root of 64 : 8.0
square root of 75 : 8.660254037844403
square root of 100 : 10.0
square root of 1600 : 40.0
```

In [ ]:

```
#Q10)
import cv2
image_1 = cv2.imread("gal_gaussian.png")
cv2.namedWindow('image', cv2.WINDOW_AUTOSIZE)
cv2.imshow('image',image_1)

blur = cv2.GaussianBlur(image_1, (5,5), 0)
cv2.namedWindow('image_blured', cv2.WINDOW_AUTOSIZE)
cv2.imshow('image_blured',blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:

```
#Q11)
import cv2
image_1 = cv2.imread("gal_sandp.png")
median = cv2.medianBlur(image_1,5)

cv2.namedWindow('image', cv2.WINDOW_AUTOSIZE)
cv2.imshow('image',image_1)
```

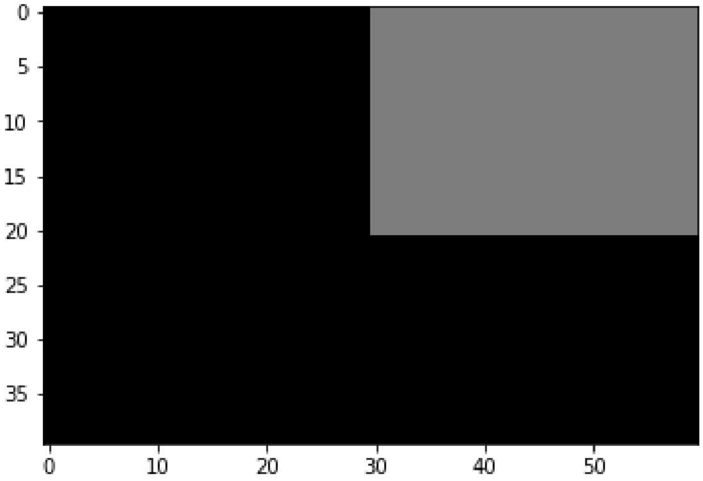
```
cv2.namedWindow('image_median', cv2.WINDOW_AUTOSIZE)
cv2.imshow('image_median',median)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:

```
#Q12)
import numpy as np
im = np.zeros((40,60), dtype = np.uint8)

im[0:21,30:61] = 125

cv2.namedWindow('image', cv2.WINDOW_AUTOSIZE)
cv2.imshow('image',im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:

```
import matplotlib.pyplot as plt
im = np.zeros((40,60), dtype = np.uint8)
im[0:21,30:61] = 125

fig,ax = plt.subplots()
ax.imshow(im, cmap = 'gray', vmin = 0, vmax = 255)
plt.show()
```



In [ ]:

```
#Q13)

no_of_channels = 3
color = (255,255,255)
array = np.full((40, 60, no_of_channels), color, dtype=np.uint8)

array[20:41,0:31] = (138, 33, 224)

cv2.imshow('image',array)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:

```
#Q14)
img = cv2.imread("tom_dark.jpg")
```

```
brightness = 40
new_arr = np.array(img) + brightness



cv2.imshow('image_original',img)
cv2.imshow('brightened', new_arr)
cv2.waitKey(0)
cv2.destroyAllWindows()
```