Index number : 190026T

Name : AHAMED M.I.I

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

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

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

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

In [ ]:
```
#3)
squares = [[i,i*i] for i in range(1,6)]
for i in squares:
    print(i[0], " : ", i[1])
```

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

In [ ]:
```
#4)
non_prime_squares = [[i,i*i] for i in range(1,6) if not sympy.isprime(i)]
for i in non_prime_squares:
    print(i[0], " : ", i[1])
```

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

In [ ]:
```
#5) 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 [ ]:
```
#5) b)
B = np.array([[3,2],
```

```
            [5,4],
            [3,1]])

AB = np.multiply(A,B)
print(AB)
```

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

In [ ]:

```
#6)
rand_array = np.random.randint(10, size=(5,7))
print("Random array \n", rand_array)
sub_array = rand_array[1:4, 0:2]
print("Sub array \n",sub_array)
print("Dimensions of the resulting array = ", sub_array.shape)
```

```
Random array
 [[3 0 0 7 6 2 2]
 [0 1 6 7 0 4 0]
 [3 7 8 0 7 3 3]
 [3 8 7 6 9 3 5]
 [6 1 1 8 3 7 2]]
Sub array
 [[0 1]
 [3 7]
 [3 8]]
Dimensions of the resulting array =  (3, 2)
```

In [ ]:

```
#7)
#scalar and one dimensional
A = np.array([1,2,3])
print("A \n", A)
x = 1
print("x = ", x)
print("A+x \n", A+x)

#scalar and two dimensional
B = np.array([[1,2,3], [4,5,6]])
print("B \n", B)
print("B+x \n", B+x)

#one dimensional and two dimensional
print("B+A \n", B+A)
```

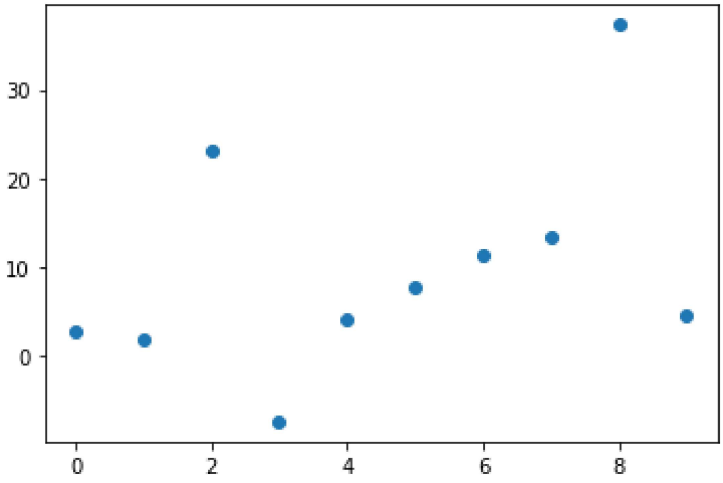
```
A
 [1 2 3]
x =  1
A+x
 [2 3 4]
B
 [[1 2 3]
 [4 5 6]]
B+x
 [[2 3 4]
 [5 6 7]]
B+A
 [[2 4 6]
 [5 7 9]]
```

In [ ]:

```
#8)
import matplotlib.pyplot as plt
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 ))
plt.scatter(x,y)

#8) a)
X = np.append(x, np.ones((N,1)), axis=1)
print("X \n", X)

#8) b)
X_T = X.transpose()
X_T_X = np.matmul(X_T, X)
X_T_X_inv = np.linalg.inv(X_T_X)
Y = np.matmul(np.matmul(X_T_X_inv, X_T), y)
print("Y \n",Y)
```

```
X
 [[0. 1.]
 [1. 1.]
 [2. 1.]
 [3. 1.]
 [4. 1.]
 [5. 1.]
 [6. 1.]
 [7. 1.]
 [8. 1.]
 [9. 1.]]
Y
 [[1.67833539]
 [2.26591878]]
```



In [ ]:

```
def newton_raphson(S, S_0, error):
    iters = 0
    while abs(S**0.5 - S_0) > error:
        S_0 = 0.5*(S_0 + S/S_0)
        iters += 1
    return S_0, iters

def get_initial(S):
    n = 0
    a = S
    if 0 <= a <= 100:
        return (-190/(a+20) + 10)*10**(n/2)
    if a < 0:
        while a < 0:
            a = a*100
            n = n - 2
    if a < 0:
        while a > 0:
```

```
            a = a/100
            n = n + 2
    return (-190/(a+20) + 10)*10**(n/2)
```

In [ ]:

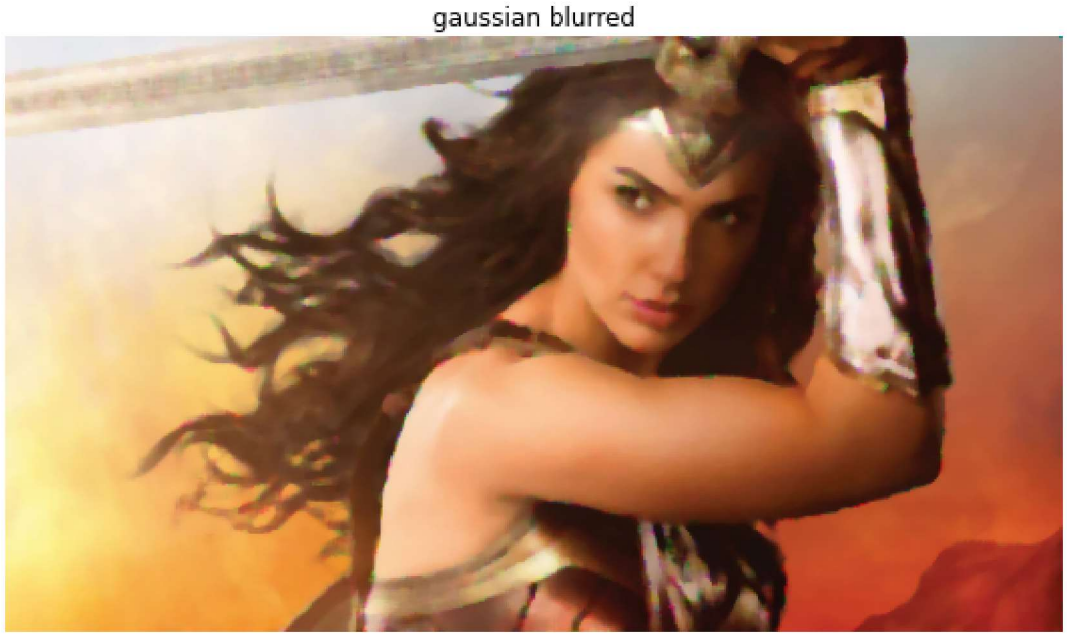
```
#9) c)
error = 0.00001
S = [64, 75, 100, 1600]
for i in S:
    S_0 = get_initial(i)
    root, iters = newton_raphson(i, S_0, error)
    print("Square root of ",i, " = ", root, "\n iterations = ", iters)
```

```
Square root of  64  =  8.000001227114023
 iterations =  2
Square root of  75  =  8.660254037949777
 iterations =  3
Square root of  100  =  10.000000059692617
 iterations =  3
Square root of  1600  =  40.00000777665428
 iterations =  5
```

In [ ]:

```
#10)
import cv2 as cv

im = cv.imread(r'E:\Aca\aca sem 4\Image Processing & Machine vision\exercises\exerci
im = cv.cvtColor(im, cv.COLOR_BGR2RGB)
assert im is not None

fig, ax = plt.subplots(2, figsize=(10,12))
ax[0].axis('off')
ax[0].title.set_text('orginal')
ax[0].imshow(im)

blur = cv.GaussianBlur(im,(5,5),0)

ax[1].axis('off')
ax[1].title.set_text('gaussian blurred')
ax[1].imshow(median)
```

Out[ ]: <matplotlib.image.AxesImage at 0x1b6e44161f0>



In [ ]:

```
#11)

im = cv.imread(r'E:\Aca\aca sem 4\Image Processing & Machine vision\exercises\exerci
im = cv.cvtColor(im, cv.COLOR_BGR2RGB)
assert im is not None
median = cv.medianBlur(im,5)

fig, ax = plt.subplots(2, figsize=(10,12))
ax[0].axis('off')
ax[0].title.set_text('orginal')
ax[0].imshow(im)

ax[1].axis('off')
ax[1].title.set_text('median blurred')
ax[1].imshow(median)
```

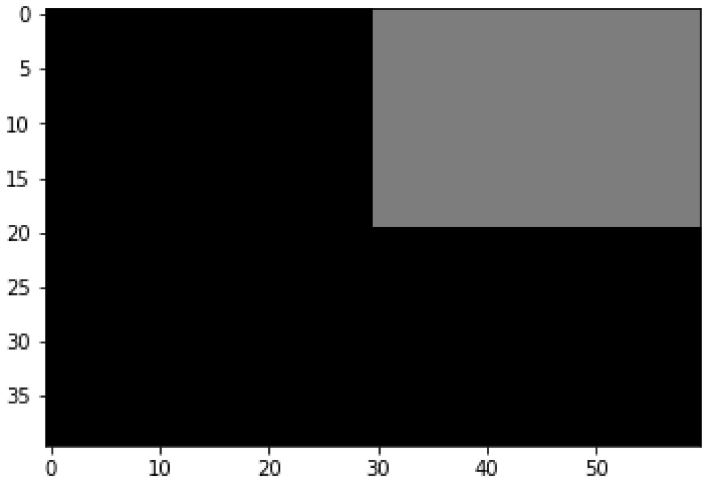
Out[ ]: <matplotlib.image.AxesImage at 0x1b6e47d47f0>

orginal

median blurred

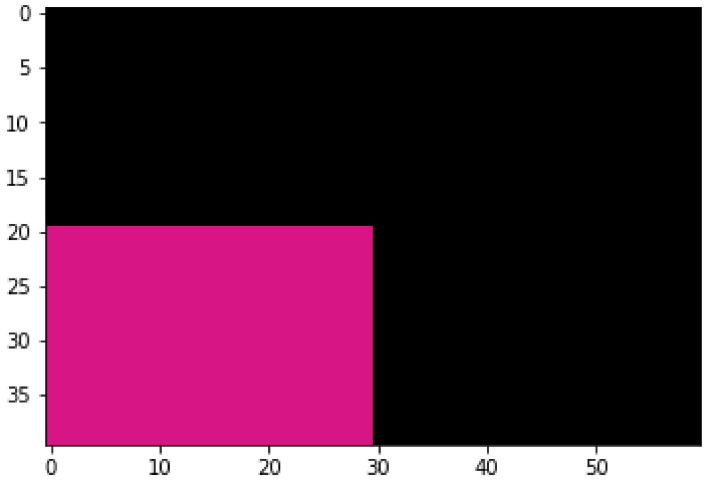
In [ ]:

```
#12)
im = np.zeros((40,60), dtype=np.uint8)
im[0:20, 30:60] = 125

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



In [ ]:

```
#13)
im = np.zeros([40,60,3])
im[20:40, 0:30, 0] = np.ones([20,30])*0.85
im[20:40, 0:30, 1] = np.ones([20,30])*0.09
im[20:40, 0:30, 2] = np.ones([20,30])*0.52

fig, ax = plt.subplots()
ax.imshow(im)
plt.show()
```



In [ ]:

```
#14)

im = cv.imread(r'E:\Aca\aca sem 4\Image Processing & Machine vision\exercises\exerci

fig, ax = plt.subplots(2, figsize=(10,15))
ax[0].axis('off')
ax[0].title.set_text('orginal')
ax[0].imshow(im, cmap = 'gray', vmin =0, vmax=255)

value = 50
hsv = cv.cvtColor(im, cv.COLOR_BGR2HSV)
h, s, v = cv.split(hsv)

lim = 255 - value
v[v > lim] = 255
v[v <= lim] += value

final_hsv = cv.merge((h, s, v))
brightned = cv.cvtColor(final_hsv, cv.COLOR_HSV2BGR)
```

```
ax[1].axis('off')
ax[1].title.set_text('brightned')
ax[1].imshow(brightned, cmap = 'gray', vmin =0, vmax=255)
```

Out[ ]: <matplotlib.image.AxesImage at 0x1b6e5f30e20>