Name: Chamara R.P.O Index no:190098M

In [ ]:
```
#1____________
for x in range(1,6):
    print(x," : ",x**2)
```

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

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

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

In [ ]:
```
#3____________
squares = [i**2 for i in range(1,6)]
for y,x in enumerate(squares):
    print(y+1," : ",x)
```

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

In [ ]:
```
#4____________
squares = [i**2 for i in range(1,6)]
for y,x in enumerate(squares):
    if not sympy.isprime(y+1):
        print(y+1," : ",x)
```

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

In [ ]:
```
#5 (a)____________
import numpy as np
A = np.array([[1,2],[3,4],[5,6]])
B = np.array([[7,8,9,0],[1,2,3,4]])
C = np.matmul(A,B)
print(C)
```

```
[[ 9 12 15  8]
 [25 32 39 16]
 [41 52 63 24]]
```

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

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

In [ ]:
```
#6____________
arr = np.random.randint(0,11,size=(5,7))
print('original Arr',arr)
sub_arr=arr[1:4,[0,1]]
print('Extracted array',sub_arr)
print('Size',np.shape(sub_arr))
```

```
original Arr [[ 0  5  3  7 10  5  3]
 [ 5  9  5  3  7  9  7]
 [ 6  4  6  2  3  4  1]
 [ 4  8  3  9  3 10  0]
 [ 7  2  4  1 10  0  6]]
Extracted array [[5 9]
 [6 4]
 [4 8]]
Size (3, 2)
```

In [ ]:
```
#7____________
p = np.array([[1,2,3],[4,5,6]])
q = np.array([7,8,9])
r = np.array([[1,2,3],[4,5,6],[7,8,9]])
s = np.array([[[1,2,3],[4,5,6],[7,8,9]],[[1,2,3],[4,5,6],[7,8,9]]])

print("p+q :", p+q)
print("size of operands :",np.shape(p)," and ",np.shape(q)," size of answer :",np.shap

print("p*q :",r*q)
print("size of operands :",np.shape(r)," and ",np.shape(q)," size of answer :",np.shap

print("p*r :",r*s)
print("size of operands :",np.shape(r)," and ",np.shape(s)," size of answer :",np.shap
```

```
p+q : [[ 8 10 12]
 [11 13 15]]
size of operands : (2, 3)  and  (3,)  size of answer : (2, 3)
p*q : [[ 7 16 27]
 [28 40 54]
 [49 64 81]]
size of operands : (3, 3)  and  (3,)  size of answer : (3, 3)
p*r : [[[ 1  4  9]
  [16 25 36]
  [49 64 81]]

 [[ 1  4  9]
  [16 25 36]
  [49 64 81]]]
size of operands : (3, 3)  and  (2, 3, 3)  size of answer : (2, 3, 3)
```

In [ ]:
```
#8(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 ) )
ones = np.ones((N,1))

X = np.append(x,ones,axis=1)
print(X)
```

```
[[0. 1.]
 [1. 1.]
 [2. 1.]
 [3. 1.]
 [4. 1.]
 [5. 1.]
 [6. 1.]
 [7. 1.]
 [8. 1.]
 [9. 1.]]
```

In [ ]:
```
#8(b)____________
from numpy import linalg
w = np.linalg.inv(X.T@X)@X.T@y
print(y)
```

```
[[-5.58257284]
 [ 3.75686758]
 [ 3.40064468]
 [ 5.10393068]
 [24.12341034]
 [10.24009956]
 [-2.49157364]
 [14.12190464]
 [23.20041794]
 [17.9936503 ]]
```

In [ ]:
```
#9(a)________
def init_estimate(S):
    n=-1
    while(S>1):
        a=S
        n+=1
        S = S/100
    est_root = (-190/(a+20)+10)*10**n
    return est_root
```

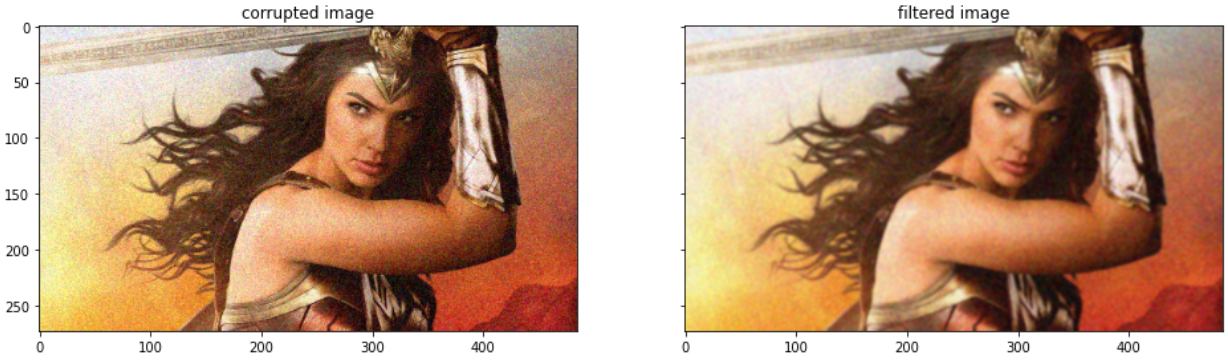
In [ ]:
```
#9(b)________
def raphson_method(x_n,S):

    x_n1 = x_n-((x_n**2) -S)/(2*x_n)
    if(abs(x_n1-x_n)>0.00001):
        return raphson_method(x_n1,S)
    else: return x_n1
```

In [ ]:
```
#9(C)________
x_n = init_estimate(64)
print("root of 64 :",raphson_method(x_n,64))
x_n = init_estimate(75)
print("root of 75 :",raphson_method(x_n,75))
x_n = init_estimate(100)
print("root of 100 :",raphson_method(x_n,100))
x_n = init_estimate(1600)
print("root of 1600 :",raphson_method(x_n,1600))
```

```
root of 64 : 8.000000000000094
root of 75 : 8.660254037844386
root of 100 : 10.0
root of 1600 : 40.0
```

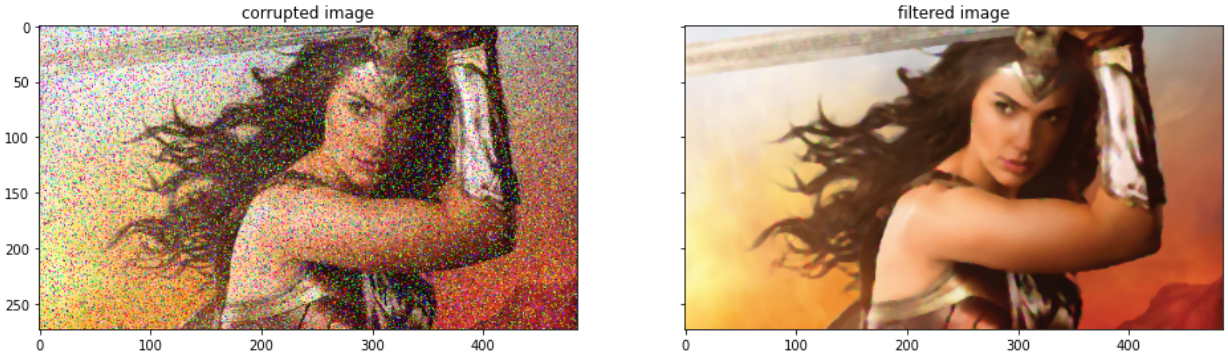
In [ ]:
```
#10_________
import cv2 as cv
import matplotlib.pyplot as plt
img = cv.imread("ex01\img\gal_gaussian.png")
img_rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(16,16))
blur = cv.GaussianBlur(img,(5,5),0)
blur_rgb = cv.cvtColor(blur, cv.COLOR_BGR2RGB)
ax1.imshow(img_rgb)
ax1.set_title("corrupted image")
ax2.imshow(blur_rgb)
ax2.set_title("filtered image")
```

Out[ ]:
```
Text(0.5, 1.0, 'filtered image')
```



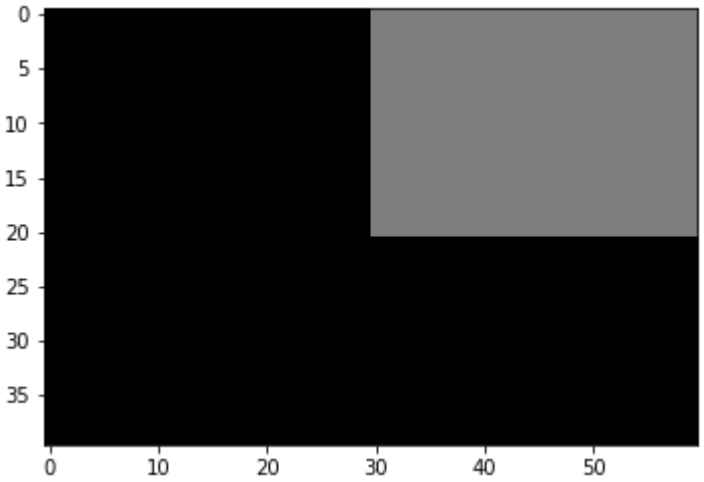
In [ ]:
```
img = cv.imread("ex01\img\gal_sandp.png")
img_rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(16,16))
median = cv.medianBlur(img,5)
median_rgb = cv.cvtColor(median, cv.COLOR_BGR2RGB)
ax1.imshow(img_rgb)
ax1.set_title("corrupted image")
ax2.imshow(median_rgb)
ax2.set_title("filtered image")
```

Out[ ]:
```
Text(0.5, 1.0, 'filtered image')
```



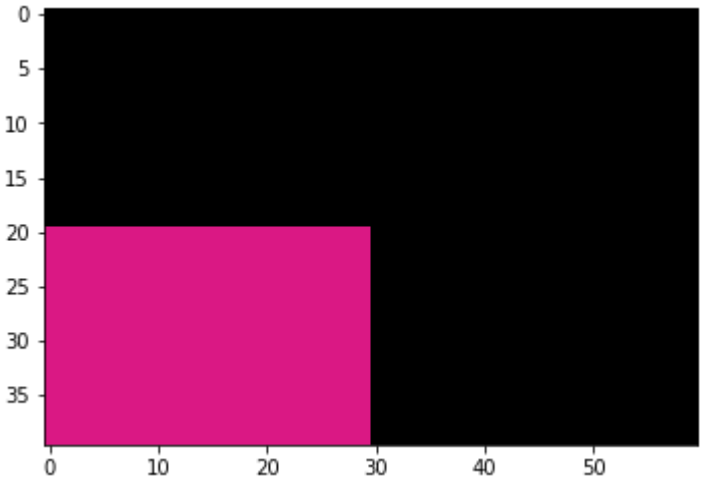
In [ ]:
```
img = np.zeros((40,60),dtype=np.uint8)
img[0:21,30:61]=125
img_gray = cv.cvtColor(img, cv.IMREAD_GRAYSCALE)
plt.imshow(img_gray)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x221f5ecd6c0>
```



In [ ]:
```
img = np.zeros((40,60,3),dtype=np.uint8)
img[20:41,0:30]=[132,24,218]
img_rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.imshow(img_rgb)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x221f5f391e0>
```



In [ ]:
```
img = cv.imread("ex01\img\_tom_dark.jpg")
beta = 70 # change value between 0-100
print(np.shape(img))
new_img = np.zeros(np.shape(img),dtype=np.uint8)
for x in range(np.shape(img)[0]):
    for y in range(np.shape(img)[1]):
        for z in range(np.shape(img)[2]):
            new_img[x,y,z] = np.clip(img[x,y,z]+beta,0,255)
```

```
(473, 710, 3)
```

In [ ]:
```
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(16,16))
ax1.imshow(new_img)
ax1.set_title("brighter image")
ax2.imshow(img)
ax2.set_title("dark image")
```

Out[ ]:
```
Text(0.5, 1.0, 'dark image')
```


brighter image
0
100
200
300
400
0
100
200
300
400
500
600
700


dark image
0
100
200
300
400
500
600
700