Name: Bandara D.R.K.W.M.S.D

Index: 190071B

1. Write a program to display the squares of numbers form 1 to 5 as seen below

In [ ]:

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

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

1. Alter the code in 14 to print the square only of non-prime numbers. Use the isprime function in the sympy package for testing if a number is a prime.

In [ ]:

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

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

1. Use a comprehension to do 14.

In [ ]:

```
#3
squares = [i**2 for i in range(1,6)]
for i, i2 in enumerate(squares):
  print(i+1, ':', i2)
```

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

1. Use a comprehension to do 11.

In [ ]:

```
#4
squares = [i**2  for i in range(1,6) if not sympy.isprime(i)]
for i, i2 in enumerate(squares):
  print(i+1, ':', i2)
```

```
1 : 1
2 : 16
```

1. Using NumPy carry out the following computations

In [ ]:

```
#5
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)
d = np.array([[1,2],[3,4],[5,6]])
e = np.array([[3,2],[5,4],[3,1]])

print(c)
print(d*e)
```

```
[[ 9 12 15  9]
 [25 32 39 19]
 [41 52 63 29]]
[[ 3  4]
 [15 16]
 [15  6]]
```

1. Generate a 5×7 array of random integers in the interval [0,10] and extract the sub array consisting rows 2 to 4 and columns 1 and first two columns. What is the size of the resulting array?

In [ ]:

```
#6
a = np.random.randint(10, size=(5,7))
b = a[1:4,0]
c = a[:, 0:2]
print("5x7 matrix\n", a)
print("rows 2 to 4 and columns 1\n",np.reshape(b,(3,1)))
print(" first two columns\n", c)
```

```
5x7 matrix
 [[7 3 0 0 2 8 7]
 [7 4 5 1 3 0 3]
 [0 0 1 1 5 9 6]
 [6 7 8 1 5 6 2]
 [3 6 3 4 4 8 7]]
rows 2 to 4 and columns 1
 [[7]
 [0]
 [6]]
 first two columns
 [[7 3]
 [7 4]
 [0 0]
 [6 7]
 [3 6]]
```

1. Show three examples of broadcasting

In [ ]:

```
#7
a = np.array([[1,2,3,4]])
b = 3
arr = np.array([[1,3,5,6],[3,5,7,3],[4,5,7,3]])
c = a*b
d = a+b
e = arr + a
print("Matrix multiply with scaler\n", c)
print("Matrix add with scaler\n",d)
print("Add row vector to matrix\n", e)
```

```
Matrix multiply with scaler
 [[ 3  6  9 12]]
Matrix add with scaler
 [[4 5 6 7]]
Add row vector to matrix
 [[ 2  5  8 10]
 [ 4  7 10  7]
 [ 5  7 10  7]]
```

1. Consider the following code snippet 1

   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))

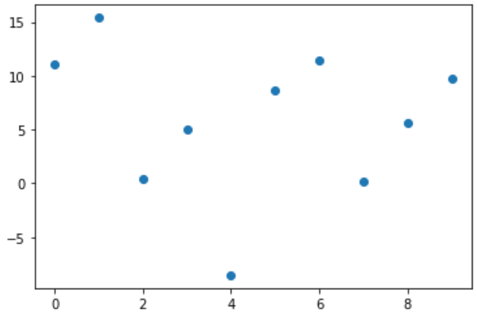
   (a) Append a column of ones to x to create X.

   (b) Compute [X TX] −1 X T y

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)
X = np.append(np.ones((N,1)),x,axis = 1)
w = np.linalg.inv(X.T @ X) @ X.T @ y
print(w)
```

```
[[ 7.1402003 ]
 [-0.27285181]]
```



1. Newtons - Raphsons method

In [ ]:

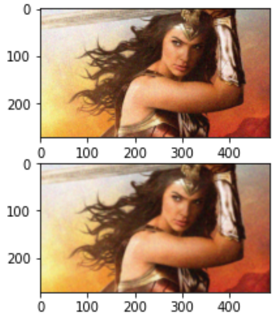
```
#9
def sqt(number, precision):

    n = 0
    while 10**(2*n) < number:
        n+=1
    n = n-1
    a = number/10**(2*n)
    s0 = (-190/(a+20)+10)*10**n

    k = 0
    while True:
        s1 = s0 - (s0**2-number)/(2*s0)
        if abs(s1**2 - number) < precision:
            break
        s0 =s1
    return s1

print("Square root of 64:",sqt(64, 10**(-5)))
print("Square root of 75:",sqt(75, 10**(-5)))
print("Square root of 100:",sqt(100, 10**(-5)))
print("Square root of 1600:",sqt(1600, 10**(-5)))
```

```
Square root of 64: 8.000000000000094
Square root of 75: 8.660254037949775
Square root of 100: 10.000000059692617
Square root of 1600: 40.0
```

1. Use Gaussian smoothing to filter the noise in the image gal_gaussian.png shown in Fig. 1

In [ ]:

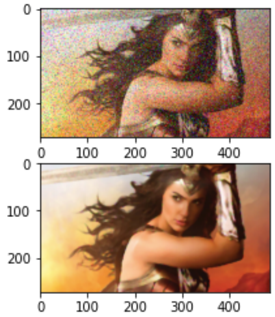
```
#10
import cv2
im = cv2.imread("Images/gal_gaussian.png")
blurImage = cv2.GaussianBlur(im,(5,5),0);
cv2.namedWindow("Image", cv2.WINDOW_AUTOSIZE)
cv2.imshow("Image", im)
cv2.waitKey(0)
cv2.imshow("Image", blurImage)
cv2.waitKey(0)
cv2.destroyAllWindows()
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
blurImage = cv2.cvtColor(blurImage, cv2.COLOR_BGR2RGB)
fig, ax = plt.subplots(2)
ax[0].imshow(im)
ax[1].imshow(blurImage)
plt.show()
```



1. Use median filtering to filter the salt and pepper noise in the image gal_sandp.png shown in Fig. 2.

In [ ]:

```
#11
im = cv2.imread("Images/gal_sandp.png")
assert im is not None
cv2.namedWindow("Image", cv2.WINDOW_AUTOSIZE)
cv2.imshow("Image", im)
cv2.waitKey(0)
medianImg = cv2.medianBlur(im,5)
cv2.imshow("Image", medianImg)
cv2.waitKey(0)
cv2.destroyAllWindows()
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
medianImg = cv2.cvtColor(medianImg, cv2.COLOR_BGR2RGB)
fig, ax = plt.subplots(2)
ax[0].imshow(im)
ax[1].imshow(medianImg)
plt.show()
```



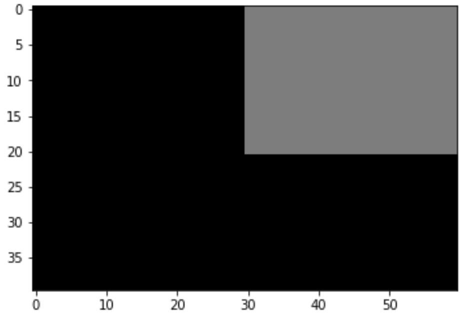
1. Create a 40×60 grayscale image and change the color of the top-right quarter to 125.

In [ ]:

```
#12
im = np.zeros((40,60),dtype = np.uint8)
im[0:21, 30:] = 125
cv2.namedWindow("Image", cv2.WINDOW_AUTOSIZE)
cv2.imshow("Image", im)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:

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



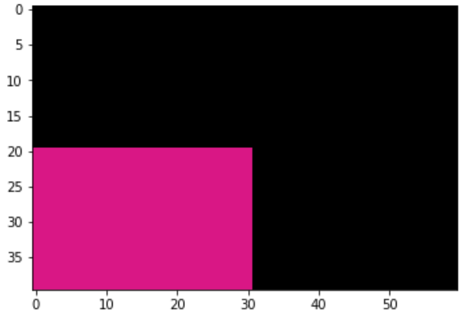
1. Create a 40×60 color image and change the color of the bottom-left quarter to “Barbie Pink”4

In [ ]:

```
#13
im = np.zeros((40,60,3),dtype = np.uint8)
im[20:, 0:31, 2] = 217
im[20:, 0:31, 1] = 23
im[20:, 0:31, 0] = 133
cv2.namedWindow("Image", cv2.WINDOW_AUTOSIZE)
cv2.imshow("Image", im)
cv2.waitKey(0)
cv2.destroyAllWindows()
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
```

In [ ]:

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



1. Increase the brightness of the image tom_dark.jpg shown in Fig. 3.

In [ ]:

```
#14
im = cv2.imread("Images/tom_dark.jpg")
assert im is not None
```

```
cv2.namedWindow("Image", cv2.WINDOW_AUTOSIZE)
cv2.imshow("Image", im)
cv2.waitKey(0)
im2 = im + 30
cv2.imshow("Image", im2)
cv2.waitKey(0)
cv2.destroyAllWindows()
fig, ax = plt.subplots(2)
ax[0].imshow(im)
ax[1].imshow(im2)
plt.show()
```