Name: K. D. S. D. Kuruppu

Index no: 190338C

In [ ]:
```
#1)
for i in range(1, 6):
  print(f"{i} : {i**2}")
```

```
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(f"{i} : {i**2}")
```

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

In [ ]:
```
#3)
squares = {i:i**2 for i in range(1,6)}

for key in squares:
  print(f"{key} : {squares[key]}")
```

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

In [ ]:
```
#4)
squares = {i:i**2 for i in range(1,6) if not sympy.isprime(i)}

for key in squares:
  print(f"{key} : {squares[key]}")
```

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

In [ ]:
```
#5)
#a)
import numpy as np

m1 = np.array([[1,2],[3,4],[5,6]])
m2 = np.array([[7,8,9,1],[1,2,3,4]])

ans = np.matmul(m1, m2) #A@B

print(ans)
```

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

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

ans2 = np.multiply(A, B)

print(ans2)
```

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

In [ ]:
```python
#6)
arr1 = np.random.randint(0, 10, (5,7))

print(f"The original array:\n{arr1}\n")
print(f"The slice:\n{arr1[[2,3,4], :][:, [0,1]]}")
```

```
The original array:
[[9 0 4 6 5 7 2]
 [6 1 6 0 8 8 2]
 [2 4 8 4 4 7 7]
 [7 7 5 1 9 2 8]
 [0 6 2 5 5 6 2]]

The slice:
[[2 4]
 [7 7]
 [0 6]]
```

In [ ]:
```python
#7)
A = np.array([[1,2,3],[4,5,6]])
B = 5
C = A + B # a single value being broadcasted over an array
print(f"A:\n{A}\nB:\n{B}")
print(f"A+B:\n{C}\n")

B = np.array([10,20,30])
D = A + B # a single row vector being broadcasted over an array
print(f"A:\n{A}\nB:\n{B}")
print(f"A+B:\n{D}\n")

B = np.array([[10],[20]])
E = A + B # a single column vector being broadcasted over an array
print(f"A:\n{A}\nB:\n{B}")
print(f"A+B:\n{E}\n")
```

```
A:
[[1 2 3]
 [4 5 6]]
B:
5
A+B:
[[ 6  7  8]
 [ 9 10 11]]

A:
[[1 2 3]
 [4 5 6]]
B:
[10 20 30]
A+B:
[[11 22 33]
 [14 25 36]]

A:
[[1 2 3]
 [4 5 6]]
B:
[[10]
 [20]]
A+B:
[[11 12 13]
 [24 25 26]]
```

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 ))

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

print(f"x:\n{x}\n")
print(f"X:\n{X}")
```

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

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

In [ ]:
```python
#b)
ans = np.linalg.inv(X.T @ X) @ X.T @ y
print(f"Answer:\n{ans}")
```

```
Answer:
[[ 2.69822469]
 [-6.97878689]]
```

In [ ]:
```python
#9)
import math

def sqrt(x):
    if 1 <= x <= 100:
        sqrt_x = (-190/(x+20) + 10)
    elif x > 100:
        n = math.ceil( math.floor(np.log10(x)) / 2 )
        a = x / (10**(2*n))

        sqrt_x = (-190/(a+20) + 10) * (10**n)
    elif x < 1:
        n = math.floor( math.floor(np.log10(x)) / 2 )
        a = x / (10**(2*n))

        sqrt_x = (-190/(a+20) + 10) * (10**n)

    num_iter = 10

    for i in range(num_iter):
        sqrt_x = sqrt_x - ( ( sqrt_x**2 - x ) / (2*sqrt_x) )

    return sqrt_x

print(f"Square root of 64: {sqrt(64)}")
print(f"Square root of 75: {sqrt(75)}")
print(f"Square root of 100: {sqrt(100)}")
print(f"Square root of 1600: {sqrt(1600)}")
```

```
Square root of 64: 8.0
Square root of 75: 8.660254037844386
Square root of 100: 10.0
Square root of 1600: 40.0
```

In [ ]:
```python
#10)
import cv2 as cv

im = cv.imread("Images\gal_gaussian.png")
filter_im = cv.GaussianBlur(im, (5,5), 0)

cv.namedWindow('Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Image', im)
cv.waitKey(0)
cv.imshow('Image', filter_im)
cv.waitKey(0)
cv.destroyAllWindows()
```

In [ ]:
```python
#11)
im = cv.imread("Images\gal_sandp.png")
filter_im = cv.medianBlur(im, 5)

cv.namedWindow('Salt-and-Paper noise image', cv.WINDOW_AUTOSIZE)
cv.imshow('Salt-and-Paper noise image', im)
cv.waitKey(0)
cv.imshow('Salt-and-Paper noise image', filter_im)
cv.waitKey(0)
cv.destroyAllWindows()
```

In [ ]:
```python
#12)
gray_img = np.zeros((40, 60), dtype="uint8")

cv.namedWindow('Gray Image', cv.WINDOW_NORMAL)
cv.imshow('Gray Image', gray_img)
cv.waitKey(0)

gray_img[:20,30:60] += 125

cv.imshow('Gray Image', gray_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

In [ ]:
```python
#13)
colour_img = np.ones((40,60,3), dtype="float64")
colour_img[:,:,0] = 0.23
colour_img[:,:,1] = 0.48
colour_img[:,:,2] = 0.34

cv.namedWindow('Colour Image', cv.WINDOW_NORMAL)
cv.imshow('Colour Image', colour_img) #Amazon colour
cv.waitKey(0)

colour_img[20:40,0:30,0] = 0.85
colour_img[20:40,0:30,1] = 0.09
colour_img[20:40,0:30,2] = 0.52

cv.imshow('Colour Image', colour_img) #Amazon colour
cv.waitKey(0)
cv.destroyAllWindows()
```

In [ ]:
```
#14)
tom_img = cv.imread("Images\\tom_dark.jpg")

cv.namedWindow('Tom Image', cv.WINDOW_AUTOSIZE)
cv.imshow('Tom Image', tom_img)
cv.waitKey(0)

tom_img += 50

cv.imshow('Tom Image', tom_img)
cv.waitKey(0)
cv.destroyAllWindows()
```