EN2550 Homework 1 on Python and NumPy

Name - Ekanayake E.M.S.S.N.

Index no - 190164M

---

(01)

In [2]:
```
for i in range (1,6):
  print(str(i)+' : '+str(i*i))
```

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

(02)

In [3]:
```
import sympy
```

In [4]:
```
for i in range (1,6):
  if sympy.isprime(i)==False:
    print(str(i)+' : '+str(i*i))
```

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

(03)

In [5]:
```
squares = [i**2 for i in range (1,6)]
for a,b in enumerate(squares,1):
  print(a, ' : ', b)
```

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

(04)

In [6]:
```
notprime_squares = [(i,i**2) for i in range(1,6) if sympy.isprime(i)==False]
for j in notprime_squares:
  print(j[0], ' : ', j[1])
```

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

(05) a)

In [7]:
```
import numpy as np
```

In [8]:
```
A = np.array([[1,2],[3,4],[5,6]])
B = np.array([[7,8,9,1],[1,2,3,4]])
print(np.matmul(A,B))
```

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

b)

In [9]:
```
A = np.array([[1,2],[3,4],[5,6]])
B = np.array([[3,2],[5,4],[3,1]])
print(np.multiply(A,B))
```

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

(06)

In [10]:
```
array = np.random.randint(10,size=(5,7))
print(array,'\n')
subarray=array[1:4,0:2]
print(subarray,'\n')
subarray.shape
```

```
[[8 1 9 9 2 5 2]
 [5 2 9 1 0 8 1]
```

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

[[5 2]
 [3 7]
 [6 3]]
```

Out[10]: (3, 2)

(07)

i. adding a constant vector to each row of a matrix

(Adding vector *v* to each row of *x* to create *y*)

In [15]:
```
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = x + v  # Broadcasting happens because of mismatch in array dimension
print(y)
```

```
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

ii. Multiplying by a constant vector

In [19]:
```
A = np.array([[5, 7, 3, 1],[1,2,3,4],[0,2,4,6]])
B = np.array([1,2,0,3])

c = A * B
print (c)
```

```
[[ 5 14  0  3]
 [ 1  4  0 12]
 [ 0  4  0 18]]
```

iii. Compute outer products of vectors

In [20]:
```
v = np.array([1,2,3])
w = np.array([4,5])
# To compute an outer product, we first reshape v to be a column
# vector of shape (3, 1); we can then broadcast it against w to yield
# an output of shape (3, 2), which is the outer product of v and w:
# [[ 4  5]
#  [ 8 10]
#  [12 15]]
print(np.reshape(v, (3, 1)) * w)
```

```
[[ 4  5]
 [ 8 10]
 [12 15]]
```

(08) a)

In [11]:
```
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(np.ones((N,1)),x,axis=1)
print(X)
```

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

b)

In [12]:
```
print(np.matmul(np.matmul(np.linalg.inv(np.matmul(X.transpose(),X)),X.transpose()),y))
```

```
[[-3.58066565]
 [ 2.16125061]]
```

(10)

In [2]:
```
import cv2 as cv
```

```
import matplotlib.pyplot as plt

im = cv.imread(r'./Images/gal_gaussian.png')
img = im[:, :, ::-1]

blur = cv.GaussianBlur(im,(5,5),0)
blur2 = blur[:, :, ::-1]

fig, ax = plt.subplots()
ax.imshow(img)
plt.axis('off')
plt.show()

fig, ax = plt.subplots()
ax.imshow(blur2)
plt.axis('off')
plt.show()

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

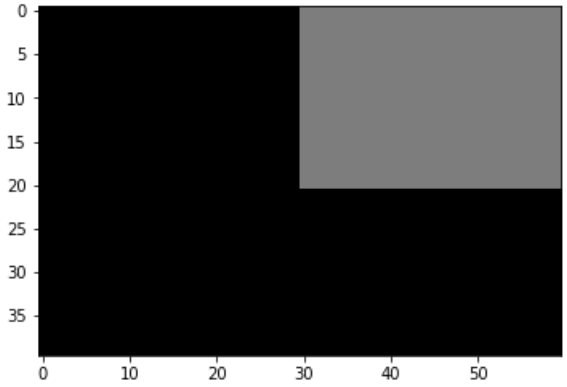
(11)

In [19]:
```
im = cv.imread(r'./Images/gal_sandp.png')
median = cv.medianBlur(im, 5)
img = median[:, :, ::-1]

fig, ax = plt.subplots()
ax.imshow(img)
plt.axis('off')
plt.show()
```

(12)

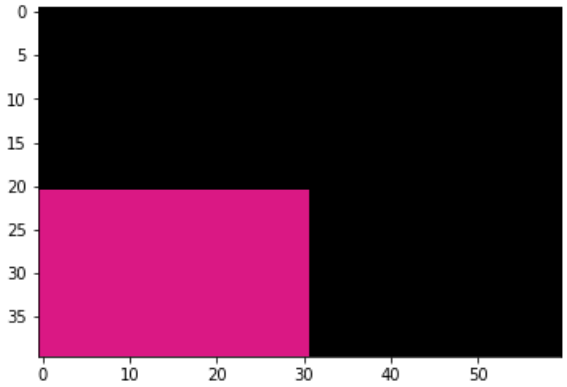
In [4]:
```
import numpy as np

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()
```



(13)

In [31]:
```
im = np.zeros((40,60,3), dtype = np.uint8)
im[21:,:31] = [218, 24, 132]

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



(14)

In [14]:
```
image= cv.imread(r'./Images/tom_dark.jpg')

new_image = np.zeros(image.shape, image.dtype)

for y in range(image.shape[0]):
    for x in range(image.shape[1]):
        for c in range(image.shape[2]):
            new_image[y,x,c] = np.clip(1.2*image[y,x,c] +40, 0, 255)

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