In [3]:

```
''' *Q1) Given a 1D array,

return the first and last elements from the array.'''

"Ans="

import numpy as np

def get_elements(arr):
    '''
    INPUT: arr -> 1D numpy array

    OUTPUT elements -> tuple of first and last element.
    '''
        ###use indexing to get the values from a array


    first_element = arr[0]

    last_element =  arr[-1]

    return (first_element, last_element)


arr=[0, 1, 2, 3, 4, 5]

get_elements(arr)
```

Out[3]:

```
(0, 5)
```

In [8]:

```
'''*Q2) Given a 2D numpy array,

return array with its column reversed.'''

"Ans="

import numpy as np

def reverse_column(arr):
    '''
    INPUT: arr -> 2D array

    OUTPUT rev_arr -> 2D array
    '''
    ###use slicing of rows,colms
    rev_arr = arr[:,::-1]

    return rev_arr
arr1 = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8]])

reverse_column(arr1)
```

Out[8]:

```
array([[2, 1, 0],
       [5, 4, 3],
       [8, 7, 6]])
```

In [9]:

```
'''*Q3) Given an array of marks,

return the array only containing elements with marks > 40'''
```

```
"Ans="
import numpy as np

def filter_marks(marks):
    '''
    INPUT: marks -> 1D array

    OUTPUT: filtered_marks -> 1D array
    '''

    ### Step 1 Get the mask for marks > 40

    mask = marks > 40

    ### STEP 2 use the mask to filter the array

    filtered_array = marks[mask]

    return filtered_array
```

Out[9]:
```
array([1, 2, 3, 7, 8, 9])
```

In [9]:
```
'''*Q4) Given an array of marks,

return the array only containing elements with marks > 40'''

"Ans="

import numpy as np

def filter_marks(marks):
    '''
    INPUT: marks -> 1D array

    OUTPUT: filtered_marks -> 1D array
    '''

    ### Step 1 Get the mask for marks > 40

    mask = marks > 40

    ### STEP 2 use the mask to filter the array

    filtered_array = marks[mask]

    return filtered_array

marks=np.array([85, 18, 2, 57, 65, 44])

filter_marks(marks)
```

Out[9]:
```
array([85, 57, 65, 44])
```

In [10]:
```
'''*Q5)Given a 2D NumPy array of dimensions (n, n), remove all the odd elements from
and convert the remaining elements to a square matrix of dimensions (k, k).'''

"Ans="
```

```
import numpy as np
def new_mat(mat):
    '''mat -> A 2d numpy array
       Output -> A 2d numpy array is expected to be returned'''

    ## STEP 1: Filter out even elements from array (Masking)

    new_matrix = mat[(mat%2 == 0)]

    ## STEP 2: Get the length of new matrix (Recall that Masking will return 1D arra
    new_len = len(new_matrix)

    ## STEP 3: Find the shape using new_len

    new_shape = int(np.sqrt(new_len))

    ## STEP 4: Reshape the new matrix

    new_matrix = new_matrix.reshape(new_shape,new_shape)


    return new_matrix

mat=np.array([[7,4, 8, 5, 1],[ 8, 0, 11, 3, 5],[ 9, 6, 3, 5, 0],[ 1, 5 ,7 ,3 ,2],[ 4

new_mat(mat)
```

Out[10]:
```
array([[4, 8, 8],
       [0, 6, 0],
       [2, 4, 2]])
```

In [11]:
```
'''*Q4)Given a list of birds and their corresponding age,

calculate mean age of Crane bird (rounded off to 2 decimal points)'''

"Ans="
import numpy as np

def calculate_mean_age(birds, age):
    '''
    INPUT: birds, age -> 1D array

    OUPUT: mean_age -> float variable
    '''

    mean_age = None

    ## STEP1. mask

    mask = (birds == 'Cranes')


    ## STEP2. Get the age of crane birds

    crane_ages = age[mask]


    ## STEP 3. Calculate mean age of crane birds

    mean_age = np.mean(crane_ages)

    ## STEP 4. Round off the mean age to 2 decimal points
    mean_age = round(mean_age,2)


    return mean_age
```

In [ ]:

```
```