NumPy Refresher

Assignment 1

In [5]:
```
# set up notebook to display multiple output in one cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

print('The notebook is set up to display multiple output in one cell.')
```

```
The notebook is set up to display multiple output in one cell.
```

In [2]:
```
# Import NumPy

import numpy as np
```

Problem #1: Create and print out a 4 x 5 integer array and use formatted print statements to print out the following attributes for the array that you created. - size - shape - number of dimensions - data type - the length of each element of array in bytes

Sample Output

SOLUTION PROBLEM #1

In [12]:
```
arr = np.array([
    [2,7,3,8,1],
    [6,8,2,1,4],
    [2,1,1,8,8],
    [9,5,4,3,8]
])
arr
print("NumPy Array Attributes: \n")
print(f"Array size: {arr.size}")
print(f"Array shape: {arr.shape}")
print(f"Number of dimensions in array: {arr.ndim}")
print(f"Array data type: {arr.dtype}")
print(f"The byte length of each element: {arr.itemsize}")
```

Out[12]:
```
array([[2, 7, 3, 8, 1],
       [6, 8, 2, 1, 4],
       [2, 1, 1, 8, 8],
       [9, 5, 4, 3, 8]])
```

```
NumPy Array Attributes:

Array size: 20
Array shape: (4, 5)
Number of dimensions in array: 2
Array data type: int32
The byte length of each element: 4
```

Problem #2: Create and print out a 2 x 2 x 4 float array and use formatted print statements to print out the following attributes for the array that you created. - size - shape - number of dimensions - data type - the length of each element of array in bytes

Sample Output

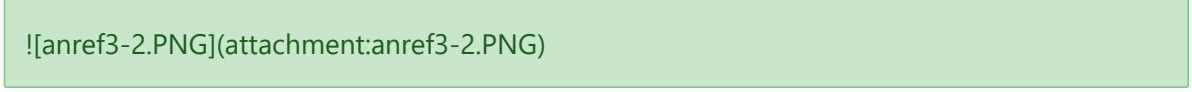
SOLUTION PROBLEM #2

In [21]:
```
arr = np.array([
    [
        [6.3,7.8,4.4,8.8],
        [3.2,3.1,6.7,5.5]
    ],
    [
        [9.7,8.6,1.4,5.2],
        [1.2,3.1,7.8,9.6]]
], float)
arr
print("NumPy Array Attributes: \n")
print(f"Array size: {arr.size}")
print(f"Array shape: {arr.shape}")
print(f"Number of dimensions in array: {arr.ndim}")
print(f"Array data type: {arr.dtype}")
print(f"The byte length of each element: {arr.itemsize}")
```

Out[21]:
```
array([[[6.3, 7.8, 4.4, 8.8],
        [3.2, 3.1, 6.7, 5.5]],

       [[9.7, 8.6, 1.4, 5.2],
        [1.2, 3.1, 7.8, 9.6]]])
NumPy Array Attributes:

Array size: 16
Array shape: (2, 2, 4)
Number of dimensions in array: 3
Array data type: float64
The byte length of each element: 8
```

Problem #3: Create a 4 x 6 integer array from a range between 10 to 106 such that the difference between each element is 4.

Desired Output



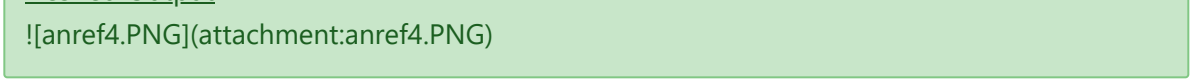
SOLUTION PROBLEM #3

In [25]:
```
np.arange(10,106,4).reshape(4,6)
```

Out[25]:
```
array([[ 10,  14,  18,  22,  26,  30],
       [ 34,  38,  42,  46,  50,  54],
       [ 58,  62,  66,  70,  74,  78],
       [ 82,  86,  90,  94,  98, 102]])
```

Problem #4: In the following code cell you will be provided with a NumPy array. Use that array to complete the following tasks: a. Return a array that contains the third column.
b. Return an array that contains the first through third row (including the third row).
c. Return an array that contains the elements found in the second through sixth row (including the sixth row) and the third through fifth column (including the fifth column)

Desired Output



SOLUTION PROBLEM #4

In [36]:
```
array = np.array([[8, 9, 4, 9, 0, 0, 1], [6, 8, 9, 4, 6, 3, 4,], [2, 2, 2, 1, 1, 9, 0
                  [4, 8, 9, 9, 0, 0, 0,], [5, 6, 7, 7, 6, 5, 4], [1, 2, 3, 4, 5, 6, 7]])

print("Starting Array")
print('\n', array)
print('\n Printing array of items in the third column from all rows \n')
print(array[:,2])
print('\n Printing array of items from the first through third row \n')
print(array[0:3,:])
print('\n Printing array of items from the second through sixth row and the third thro
print(array[1:7,2:5])
```

```
Starting Array

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

 Printing array of items in the third column from all rows

[4 9 2 9 7 3]

 Printing array of items from the first through third row

[[8 9 4 9 0 0 1]
 [6 8 9 4 6 3 4]
 [2 2 2 1 1 9 0]]

 Printing array of items from the second through sixth row and the third through fift
h column

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

Problem #5: Return an array of the odd rows and even columns from the Numpy array provided in the code cell below.

Desired Output

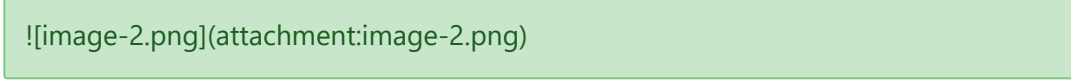
SOLUTION PROBLEM #5

In [37]:
```
sample_array = np.array([[4, 1, 5, 7, 9], [2, 4, 1, 1, 0], [9, 9, 3, 2, 6],
                         [0, 0, 7, 7, 7],  [5, 5, 5, 2, 1], [6, 2, 3, 8, 9]])

print("Starting Array")
print('\n', sample_array)
print("\n Printing array of odd rows and even columns")
print('\n', sample_array[::2,1::2])
```

```
Starting Array

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

 Printing array of odd rows and even columns

 [[1 7]
 [9 2]
 [5 2]]
```

Problem #6: Initialize the array below without using the np.array() function.

Desired Output



SOLUTION PROBLEM #6

In [54]:
```
arr = np.zeros([12,8], int)
arr[1:8,2:9] = 1
arr[3:7,4] = 7
arr[9:11,2:6]= 5
arr
```

Out[54]:
```
array([[0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 1, 1, 1, 1],
       [0, 0, 1, 1, 1, 1, 1, 1],
       [0, 0, 1, 1, 7, 1, 1, 1],
       [0, 0, 1, 1, 7, 1, 1, 1],
       [0, 0, 1, 1, 7, 1, 1, 1],
       [0, 0, 1, 1, 7, 1, 1, 1],
       [0, 0, 1, 1, 1, 1, 1, 1],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 5, 5, 5, 5, 0, 0],
       [0, 0, 5, 5, 5, 5, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0]])
```

Problem #7: a. How would you index the part of the array that is highlighted in blue below?

b. How would you index the part of the array that is highlighted in red below?

c. How would you index the part of the array that is highlighted in green below?

![image.png](attachment:image.png)

SOLUTION PROBLEM #7

In [127…
```
arr = np.arange(1,64).reshape(7,9)
arr

arr[1:3,1:5]

arr[1:4,7]
arr[4:,2]

np.rot90(arr[4:,4:7]).diagonal()
```

Out[127]:
```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18],
       [19, 20, 21, 22, 23, 24, 25, 26, 27],
       [28, 29, 30, 31, 32, 33, 34, 35, 36],
       [37, 38, 39, 40, 41, 42, 43, 44, 45],
       [46, 47, 48, 49, 50, 51, 52, 53, 54],
       [55, 56, 57, 58, 59, 60, 61, 62, 63]])
```

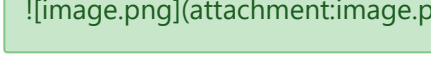
Out[127]:
```
array([[11, 12, 13, 14],
       [20, 21, 22, 23]])
```

Out[127]:
```
array([17, 26, 35])
```

Out[127]:
```
array([39, 48, 57])
```

Out[127]:
```
array([43, 51, 59])
```

Problem #8: a. Create and print out a 10 x 4 array to represent the **housing data set** found below.
b. Separate the **housing data set** into input and output data. Create formatted print statements to label your output.
c. Perform a 70 / 30 split on the **housing data set** ... i.e., 70% of the data is to be used as the train set and 30% of the data is to be used as a test set). Create formatted print statements to label your output.



SOLUTION PROBLEM #8

In [141…
```
housingArray = np.array([
    [2506,2,2.5,678900],
    [1550,44,1.5,295000],
    [2665,28,2.5,485000],
    [1691,46,1.5,320000],
    [2700,127,3,425000],
```

```
    [2196,64,2,326000],
    [6350,33,4.5,1200000],
    [3044,2,2.5,570000],
    [1657,61,1,301000],
    [2465,34,2.5,350000]
])

print(f"Input Data: \n{housingArray[:,0:3]}")
print(f"Ouput Data: \n{housingArray[:,3]}")
```

```
Input Data:
[[2.506e+03 2.000e+00 2.500e+00]
 [1.550e+03 4.400e+01 1.500e+00]
 [2.665e+03 2.800e+01 2.500e+00]
 [1.691e+03 4.600e+01 1.500e+00]
 [2.700e+03 1.270e+02 3.000e+00]
 [2.196e+03 6.400e+01 2.000e+00]
 [6.350e+03 3.300e+01 4.500e+00]
 [3.044e+03 2.000e+00 2.500e+00]
 [1.657e+03 6.100e+01 1.000e+00]
 [2.465e+03 3.400e+01 2.500e+00]]
Ouput Data:
[ 678900.  295000.  485000.  320000.  425000.  326000. 1200000.  570000.
  301000.  350000.]
```

Problem #9: In the following code cell you will be provided with several NumPy arrays. Use those arrays to complete the following tasks:

a. Create an array named "d" by vertically stacking array b onto array a. b. Create an array named "e" by horizontally stacking array c onto array d.

In [128…

```
a = np.array([[2, 2, 2], [3, 3, 3]])
b = np.array([[1, 1, 1], [4, 4, 4]])
c = np.array([[5, 5, 5], [6, 6, 6], [7, 7, 7], [8, 8, 8]])

print("Array a:")
print("\n", a)
print("\n", "Array b:")
print("\n", b)
print("\n", "Array c:")
print("\n", c)
```

```
Array a:

 [[2 2 2]
 [3 3 3]]

 Array b:

 [[1 1 1]
 [4 4 4]]

 Array c:

 [[5 5 5]
 [6 6 6]
 [7 7 7]
 [8 8 8]]
```

SOLUTION PROBLEM #9

In [132…
```
d = np.vstack([b,a])
d
```

Out[132]:
```
array([[1, 1, 1],
       [4, 4, 4],
       [2, 2, 2],
       [3, 3, 3]])
```

In [131…
```
e = np.vstack([c,d])
e
```

Out[131]:
```
array([[5, 5, 5],
       [6, 6, 6],
       [7, 7, 7],
       [8, 8, 8],
       [1, 1, 1],
       [4, 4, 4],
       [2, 2, 2],
       [3, 3, 3]])
```

In [ ]:
```

```