1.How to convert a float (32 bits) array into an integer (32 bits) in place

In [4]:
```
import numpy as np
x = np.arange(15, dtype=np.float32)
x = x.astype(np.int32)
print(x)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

2.Create a 3x3 matrix with values ranging from 0 to 8

In [4]:
```
import numpy as np
x =  np.arange(0, 9).reshape(3,3)
print(x)
```

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

3.Find indices of non-zero elements from [1,2,0,0,4,0]

In [5]:
```
import numpy as np
nums = np.array([1,2,0,0,4,0])
print("Original array:")
print(nums)
print("Indices of elements equal to non-zero of the said array:")
result = np.where(nums == 0)[0]
print(result)
```

```
Original array:
[1 2 0 0 4 0]
Indices of elements equal to non-zero of the said array:
[2 3 5]
```

4.Create a 5x5 matrix with values 1,2,3,4 just below the diagonal

Eg: 0 0 0 0 0 1 0 0 0 0 0 2 0 0 0 0 0 3 0 0 0 0 0 4 0

In [6]:
```
import numpy as np
x = np.diag([1, 2, 3, 4])
print(x)
```

```
[[1 0 0 0]
 [0 2 0 0]
 [0 0 3 0]
 [0 0 0 4]]
```

5.Create a 10x10x10 array with random values

In [12]:
```
import numpy as np
arr = np.random.randint (100, size=(10,10,10))
arr
```

Out[12]:
```
array([[[34, 90, 49, 37, 53,  1, 16, 35, 10, 46],
        [35, 90, 50,  3, 20, 95, 97, 73, 75,  8],
        [82, 98, 52, 84, 73, 45, 88, 72, 44, 79],
        [52, 75,  6, 42, 23, 42, 76, 86, 25, 12],
        [33, 81,  0, 77, 91, 98, 32, 47, 70, 88],
        [92, 61,  0, 40, 71, 19, 40, 82, 35, 76],
        [15, 79,  8, 45, 75, 27, 86, 21, 12, 84],
        [52, 56, 97, 32, 64, 17, 16, 75, 91, 54],
        [ 9, 36, 75, 63, 98, 45, 90, 13, 79, 88],
        [78, 98, 37, 49, 61, 89, 78, 34, 24,  3]],

       [[51, 67, 70, 36, 90, 46, 87, 86, 31, 96],
        [83, 34, 91, 98, 38,  2, 40, 53, 83, 58],
        [ 9,  2, 94, 38, 98, 55,  2, 62, 81, 37],
        [66, 65, 86, 81, 38, 61, 35, 28, 82, 23],
        [83, 21, 32, 69, 26, 87, 49, 97, 21, 99],
        [14, 44, 99, 44, 76, 23, 30, 81, 63, 50],
        [54, 99, 22, 13, 86, 60, 56,  4, 22, 58],
        [35, 40, 19, 70, 81,  4, 89,  1, 18,  4],
        [79, 63,  1, 82,  4, 47, 34,  7, 12, 19],
        [98, 56, 35, 85, 81, 27, 35, 29, 49, 75]],

       [[94, 64, 80, 32, 87, 16, 37, 95, 26, 86],
        [37, 30,  2, 48, 80, 84, 65, 34, 21, 99],
        [64,  2, 88, 47,  7, 72, 43,  8,  2,  7],
        [ 3, 18, 43, 56, 95, 24, 95, 18,  7, 93],
        [28,  5, 64,  9, 51, 33, 54, 34,  8, 18],
        [20, 45, 81, 32, 27, 48, 54, 80, 98,  8],
        [35, 62, 35, 33, 46, 65, 37,  2, 17, 51],
        [50, 78, 26, 85, 97, 97, 52, 22, 84, 69],
        [60, 30,  2, 71, 93, 13,  3, 73, 49,  5],
        [79, 64, 48, 69, 20, 76,  3, 20, 63, 66]],

       [[43, 16, 80, 60, 48, 79, 61, 96, 37, 55],
        [ 2, 10, 13, 27, 97, 33, 16, 78, 10, 17],
        [15, 80, 50, 66, 20, 14, 43, 77, 91, 36],
        [ 9, 45, 86, 86, 13, 58, 79, 90, 16, 69],
        [65, 93, 12, 87, 32, 78, 52, 65, 10, 85],
        [59, 27, 78, 73, 93, 77, 11,  6, 15, 37],
        [57, 51, 43, 35, 13, 73, 15, 87, 30, 19],
        [63, 21, 57, 17, 55, 73,  2, 64, 69, 18],
        [ 3, 19, 45, 37, 25, 51, 34, 14, 56, 74],
        [33, 20,  7, 97, 92, 14, 36, 15, 97, 77]],

       [[86, 30,  1, 48, 12, 36, 39, 77, 49, 83],
        [13, 94, 18, 51, 24, 41, 60, 20, 40, 28],
        [92, 30, 50, 56, 66, 47, 18, 18, 51, 86],
        [61, 17, 65, 98, 34, 73, 15, 83,  8, 96],
        [26, 39, 78, 34, 94, 38, 34, 96, 73, 71],
        [63, 52, 92,  8, 92, 55, 26, 86, 71, 90],
        [76, 11, 13,  3, 23, 66, 66, 28, 87, 60],
        [17, 34, 47, 95,  9, 51, 46, 85, 12, 73],
        [68,  5, 81, 38, 78, 62, 55, 57, 80, 76],
        [16, 64, 55,  0, 76, 21, 37, 66, 31, 69]],

       [[46,  1, 30, 67, 77, 77, 65,  5, 26, 62],
        [22, 87, 15, 79, 32, 27, 82, 83, 16, 55],
        [63, 47, 19, 68, 17,  4, 45, 20, 24,  9],
        [93, 39, 84, 56, 86, 96, 99,  3,  7, 37],
        [72,  8, 80, 68, 62,  6, 75, 87, 51, 78],
        [36, 18, 15, 57, 76, 21, 49,  9, 21, 71],
        [59,  9, 22, 73, 41, 85, 38, 50, 14, 70],
        [27, 61, 70,  2, 86, 62, 81, 92,  9, 56],
        [68, 37, 78, 98,  1, 90, 23, 40, 66, 87],
        [80, 88, 35, 93, 88, 55, 23, 32,  7,  6]],

       [[83, 73, 67, 46, 38, 18, 27, 11, 32, 88],
        [40, 80, 42, 54, 14, 90, 44, 61, 55, 63],
        [25, 24, 21, 64, 19, 89, 96, 43, 47, 59],
        [71, 44, 10, 73, 37, 75, 30, 45, 42, 74],
        [31, 72, 64, 44, 13, 55, 10, 19, 28, 99],
        [61, 41, 36, 63,  9, 93, 29,  3, 96, 58],
        [33, 96, 86, 46, 26, 68, 91, 98, 82, 14],
        [50, 55,  9, 57, 10, 85, 51, 29, 63, 62],
        [72, 73, 32,  4, 97,  3, 87, 79, 83,  2],
        [76, 87, 79, 45, 98, 94, 64, 28, 73, 63]],

       [[49,  8, 30, 59, 11, 11,  3, 59, 58, 62],
        [45,  3, 44, 40,  4, 23, 70, 11, 62, 55],
        [59, 66, 65, 22, 44, 18, 89, 11, 44, 34],
        [13,  2, 99, 43, 17, 38,  3, 45, 82, 16],
        [69, 64,  9, 92, 11, 58, 44, 44, 34, 29],
        [71, 96, 72, 35, 76, 75, 26, 98, 32, 49],
        [44, 26, 24,  8, 76, 45, 40,  9, 88, 50],
        [70, 22, 62, 29, 75, 35, 86, 66, 48, 90],
        [79, 24, 44, 95, 59, 11, 81, 12, 31, 73],
        [46, 52, 33, 73, 23, 34, 42, 18, 15, 83]],

       [[59, 16, 37, 74, 94, 58, 43, 24, 22, 69],
        [94, 88, 94,  3, 19, 48, 71, 43, 16, 73],
        [26, 26, 71, 96, 17,  0, 30, 42, 33, 34],
        [70, 62, 38, 13, 78, 63,  9, 35, 11, 79],
        [12, 17, 24, 13, 56, 73, 28, 79, 27, 81],
        [13,  0, 71, 43, 71, 83, 54,  9, 70, 56],
        [51, 99, 49, 81, 57, 72, 13, 25, 18, 21],
        [ 1, 69,  7, 92, 40, 60, 58, 17, 34, 20],
        [48, 37, 79, 96,  3, 28, 52, 28, 46, 89],
        [82, 35, 80, 21, 96, 34, 87, 44, 56, 88]],

       [[49,  2, 49, 41, 46,  5, 76,  0, 44, 40],
        [20, 76, 86, 89, 78, 36,  0, 64, 47, 26],
        [86, 74, 35, 33, 20, 27, 52, 78, 28, 26],
        [37, 47, 41, 89, 85,  0, 33, 67, 16, 55],
        [ 3, 40, 17, 28, 51,  1, 90, 33, 26, 99],
        [74, 67, 57, 46, 30, 83, 42, 47,  9, 61],
        [11, 64, 58, 97, 36, 89, 91, 99, 78, 19],
        [ 3, 11, 12, 41, 55, 93, 49,  4, 48,  8],
        [75, 44, 30, 17, 94, 66, 67, 72, 59,  9],
        [ 9, 41, 42, 53, 50, 76, 42, 24, 44,  6]]])
```

6.Create a 8x8 matrix and fill it with a checkerboard pattern

In [13]:
```
import numpy as np
x = np.ones((3,3))
print("Checkerboard pattern:")
x = np.zeros((8,8),dtype=int)
x[1::2,::2] = 1
x[::2,1::2] = 1
print(x)
```

```
Checkerboard pattern:
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

7.Create a 10x10 array with random values and find the minimum and maximum values

In [14]:
```
import numpy as np
x = np.random.random((10,10))
print("Original Array:")
print(x)
xmin, xmax = x.min(), x.max()
print("Minimum and Maximum Values:")
print(xmin, xmax)
```

```
Original Array:
[[0.39575334 0.788697   0.88605854 0.45305208 0.92048951 0.32558464
  0.60606814 0.96762341 0.72346675 0.33722525]
 [0.68692653 0.74007845 0.5722994  0.18955236 0.13766343 0.06707516
  0.74050962 0.70172019 0.71995817 0.02512975]
 [0.12818968 0.91831439 0.11218177 0.47484823 0.0479269  0.7211946
  0.51760631 0.59016386 0.89274372 0.42026247]
 [0.50467109 0.94749048 0.37158402 0.46153701 0.21314538 0.84066039
  0.55636933 0.47620417 0.70349767 0.26457729]
 [0.75746423 0.41160101 0.39049839 0.88506775 0.47785613 0.52829323
  0.96979276 0.37978935 0.62232318 0.82767191]
 [0.41478511 0.63120116 0.63686872 0.33297011 0.58249232 0.69815699
  0.05333901 0.63803334 0.69261052 0.46243293]
 [0.85605208 0.40567957 0.98202538 0.94396445 0.66001431 0.37851132
  0.71464016 0.34990725 0.22651779 0.53293069]
 [0.1570793  0.08630972 0.88494246 0.0581943  0.79036082 0.38654815
  0.92880424 0.6184045  0.86467035 0.30836243]
 [0.67888415 0.76002528 0.18797327 0.02783831 0.26178997 0.79614558
  0.57923132 0.65167432 0.98419003 0.61134935]
 [0.17357734 0.51598894 0.70231343 0.20059477 0.79493437 0.54603116
  0.28953961 0.53471524 0.48075074 0.38003381]]
Minimum and Maximum Values:
0.025129745645455892 0.9841900265284226
```

8.Create a 10x10 matrix with row values ranging from 0 to 9

In [16]:
```
import numpy as np
x = np.zeros((10,10))
print("Original array:")
print(x)
print("Row values ranging from 0 to 9.")
x += np.arange(10)
print(x)
```

```
Original array:
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Row values ranging from 0 to 9.
[[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]]
```

9.Consider two random matrices A anb B, check if they are equal.

In [18]:
```
A = np.random.randint(0,1,2)
B = np.random.randint(0,1,2)
equal = np.allclose(A,B)
print(equal)
```

```
True
```

10.Create a random vector of size 1000 and find the mean value

In [19]:
```
import numpy as np
x = np.random.randn(1000)
print("Average of the array elements:")
mean = x.mean()
print(mean)
```

```
Average of the array elements:
-0.024225354598720217
```

11.Create random vector of size 100 and replace the maximum value by 0

In [21]:
```
import numpy as np
x = np.random.random(100)
print("Original array:")
print(x)
x[x.argmax()] = 0
print("Maximum value replaced by 0:")
print(x)
```

```
Original array:
[0.42238695 0.03323889 0.68518404 0.82935629 0.97756211 0.26752965
 0.49859228 0.7707461  0.90262044 0.02802842 0.62524887 0.1556384
 0.81176747 0.04060601 0.47773929 0.31408669 0.07215969 0.10982873
 0.09276426 0.03201236 0.7671518  0.67885215 0.75078488 0.30995164
 0.00974549 0.0788941  0.6048819  0.83228565 0.03145178 0.06040011
 0.88903412 0.68467034 0.81300111 0.15657866 0.04253902 0.49235145
 0.391351   0.15173634 0.32090366 0.39218347 0.98444801 0.19512773
 0.90489667 0.57511965 0.09607728 0.89836491 0.00955427 0.92918137
 0.77797181 0.90002888 0.29684779 0.87914534 0.08434117 0.99207153
 0.67914408 0.3214814  0.43071329 0.64357308 0.57546901 0.57891252
 0.21286047 0.12376767 0.50917229 0.4620497  0.74968235 0.76584263
 0.14370353 0.39685273 0.85879901 0.12671881 0.39371539 0.7413362
 0.95430498 0.82281988 0.54196016 0.57510442 0.55111896 0.47610625
 0.13191877 0.60913084 0.37970786 0.94764139 0.82194114 0.40635278
 0.18194888 0.44738155 0.68970438 0.24955781 0.70080036 0.38922842
 0.7804126  0.68056021 0.27544941 0.30682034 0.02281206 0.14021512
 0.39537599 0.32283809 0.91827849 0.35820252]
Maximum value replaced by 0:
[0.42238695 0.03323889 0.68518404 0.82935629 0.97756211 0.26752965
 0.49859228 0.7707461  0.90262044 0.02802842 0.62524887 0.1556384
 0.81176747 0.04060601 0.47773929 0.31408669 0.07215969 0.10982873
 0.09276426 0.03201236 0.7671518  0.67885215 0.75078488 0.30995164
 0.00974549 0.0788941  0.6048819  0.83228565 0.03145178 0.06040011
 0.88903412 0.68467034 0.81300111 0.15657866 0.04253902 0.49235145
 0.391351   0.15173634 0.32090366 0.39218347 0.98444801 0.19512773
 0.90489667 0.57511965 0.09607728 0.89836491 0.00955427 0.92918137
 0.77797181 0.90002888 0.29684779 0.87914534 0.08434117 0.
 0.67914408 0.3214814  0.43071329 0.64357308 0.57546901 0.57891252
 0.21286047 0.12376767 0.50917229 0.4620497  0.74968235 0.76584263
 0.14370353 0.39685273 0.85879901 0.12671881 0.39371539 0.7413362
 0.95430498 0.82281988 0.54196016 0.57510442 0.55111896 0.47610625
 0.13191877 0.60913084 0.37970786 0.94764139 0.82194114 0.40635278
 0.18194888 0.44738155 0.68970438 0.24955781 0.70080036 0.38922842
 0.7804126  0.68056021 0.27544941 0.30682034 0.02281206 0.14021512
 0.39537599 0.32283809 0.91827849 0.35820252]
```

12.Create a 5X2 integer array from a range between 100 to 200 such that the difference between each element is 10

In [22]:
```
import numpy

print("Creating 5X2 array using numpy.arange")

sampleArray = numpy.arange(100, 200, 10)
sampleArray = sampleArray.reshape(5,2)

print()
print (sampleArray)
```

```
Creating 5X2 array using numpy.arange

[[100 110]
 [120 130]
 [140 150]
 [160 170]
 [180 190]]
```

13.Following is the provided numPy array. Return array of items by taking the third column from all rows

sampleArray = numpy.array([[11 ,22, 33], [44, 55, 66], [77, 88, 99]])

In [23]:
```
import numpy

sampleArray = numpy.array([[11 ,22, 33], [44, 55, 66], [77, 88, 99]])
print("Printing Input Array")
print(sampleArray)

print("\n Printing array of items in the third column from all rows")
newArray = sampleArray[...,1]
print(newArray)
```

```
Printing Input Array
[[11 22 33]
 [44 55 66]
 [77 88 99]]

 Printing array of items in the third column from all rows
[22 55 88]
```

14.Return array of odd rows and even columns from below numpy array

sampleArray = numpy.array([[3 ,6, 9, 12], [15 ,18, 21, 24], [27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])

In [24]:
```
import numpy
sampleArray = numpy.array([[3 ,6, 9, 12], [15 ,18, 21, 24],
[27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])

print("Printing Input Array")
print(sampleArray)

print("\n Printing array of odd rows and even columns")
newArray = sampleArray[::2, 1::2]
print(newArray)
```

```
Printing Input Array
[[ 3  6  9 12]
 [15 18 21 24]
 [27 30 33 36]
 [39 42 45 48]
 [51 54 57 60]]

 Printing array of odd rows and even columns
[[ 6 12]
 [30 36]
 [54 60]]
```

15.Create a result array by adding the following two NumPy arrays. Next, modify the result array by calculating the square of each element

arrayOne = numpy.array([[5, 6, 9], [21 ,18, 27]]) arrayTwo = numpy.array([[15 ,33, 24], [4 ,7, 1]])

In [25]:
```
import numpy
arrayOne = numpy.array([[5, 6, 9], [21 ,18, 27]])
arrayTwo = numpy.array([[15 ,33, 24], [4 ,7, 1]])
resultArray  = arrayOne + arrayTwo

print("addition of two arrays is \n")
print(resultArray)

for num in numpy.nditer(resultArray, op_flags = ['readwrite']):
   num[...] = num*num
print("\nResult array after calculating the square root of all elements\n")
print(resultArray)
```

```
addition of two arrays is

[[20 39 33]
 [25 25 28]]

Result array after calculating the square root of all elements

[[ 400 1521 1089]
 [ 625  625  784]]
```

16.Split the array into four equal-sized sub-arrays

Note: Create an 8X3 integer array from a range between 10 to 34 such that the difference between each element is 1 and then Split the array into four equal-sized sub-arrays.

In [26]:
```
import numpy

print("Creating 8X3 array using numpy.arange")
sampleArray = numpy.arange(10, 34, 1)
sampleArray = sampleArray.reshape(8,3)
print (sampleArray)

print("\nDividing 8X3 array into 4 sub array\n")
subArrays = numpy.split(sampleArray, 4)
print(subArrays)
```

```
Creating 8X3 array using numpy.arange
[[10 11 12]
 [13 14 15]
 [16 17 18]
 [19 20 21]
 [22 23 24]
 [25 26 27]
 [28 29 30]
 [31 32 33]]

Dividing 8X3 array into 4 sub array

[array([[10, 11, 12],
       [13, 14, 15]]), array([[16, 17, 18],
       [19, 20, 21]]), array([[22, 23, 24],
       [25, 26, 27]]), array([[28, 29, 30],
       [31, 32, 33]])]
```

17.Sort following NumPy array

Case 1: Sort array by the second row Case 2: Sort the array by the second column sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])

In [27]:
```
import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)

sortArrayByRow = sampleArray[:,sampleArray[1,:].argsort()]
print("Sorting Original array by secoond row")
print(sortArrayByRow)

print("Sorting Original array by secoond column")
sortArrayByColumn = sampleArray[sampleArray[:,1].argsort()]
print(sortArrayByColumn)
```

```
Printing Original array
[[34 43 73]
 [82 22 12]
 [53 94 66]]
Sorting Original array by secoond row
[[73 43 34]
 [12 22 82]
 [66 94 53]]
Sorting Original array by secoond column
[[82 22 12]
 [34 43 73]
 [53 94 66]]
```

18.Print max from axis 0 and min from axis 1 from the following 2-D array.

sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])

In [28]:
```
import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)

minOfAxisOne = numpy.amin(sampleArray, 1)
print("Printing amin Of Axis 1")
print(minOfAxisOne)

maxOfAxisOne = numpy.amax(sampleArray, 0)
print("Printing amax Of Axis 0")
print(maxOfAxisOne)
```

```
Printing Original array
[[34 43 73]
 [82 22 12]
 [53 94 66]]
Printing amin Of Axis 1
[34 12 53]
Printing amax Of Axis 0
[82 94 73]
```

19.Delete the second column from a given array and insert the following new column in its place.

sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]]) newColumn = numpy.array([[10,10,10]])

In [29]:
```
import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)

print("Array after deleting column 2 on axis 1")
sampleArray = numpy.delete(sampleArray , 1, axis = 1)
print (sampleArray)

arr = numpy.array([[10,10,10]])
print("Array after inserting column 2 on axis 1")
sampleArray = numpy.insert(sampleArray , 1, arr, axis = 1)
print (sampleArray)
```

```
Printing Original array
[[34 43 73]
 [82 22 12]
 [53 94 66]]
Array after deleting column 2 on axis 1
[[34 73]
 [82 12]
 [53 66]]
Array after inserting column 2 on axis 1
[[34 10 73]
 [82 10 12]
 [53 10 66]]
```

20.Create two 2-D arrays and Plot them using matplotlib

In [31]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([1,3,5,7,9])
y = np.array([2,6,7,8,9])

plt.title("Graph")
plt.xlabel("x_axis")
plt.ylabel("y_axis")
plt.plot(x,y, "*k--")
plt.show()
```


In [ ]: