Numpy Tasks

```
[ ]  #12th Aug
     #Indexing
```

```
mat = np.arange(0,100).reshape(10,10)
mat
```

```
array([[ 0,  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, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
[5]  row = 4
     col = 5
```

```
[6]  col
```

```
5
```

```
[7]  row
```

```
4
```

```
[9]  print(mat[row,col])
```

```
45
```

```
[10] mat
```

```
array([[ 0,  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, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

+ Code + Text

```
[11] mat[:]
```

```
array([[ 0,  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, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
[12] col = 6
```

```
[13] mat
```

```
array([[ 0,  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, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
[14] mat[7]#bydefault it represent to row
```

```
array([70, 71, 72, 73, 74, 75, 76, 77, 78, 79])
```

```
#with slices
col = 5
print(mat[:,col])#how to print column infor
```

```
[ 5 15 25 35 45 55 65 75 85 95]
```

```
[17] mat[col]#how to print rows
```

```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59])
```

```
[18] mat[:,4]#it will print 4th column
```

```
array([ 4, 14, 24, 34, 44, 54, 64, 74, 84, 94])
```

```
[19] mat[:,-1]
```

```
array([ 9, 19, 29, 39, 49, 59, 69, 79, 89, 99])
```

```
[20] mat[row ,:]
```

```
array([40, 41, 42, 43, 44, 45, 46, 47, 48, 49])
```

```
[22] mat[5,:]#it will print 4th row
```

```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59])
```

```
[23] mat[:,8]
```

```
array([ 8, 18, 28, 38, 48, 58, 68, 78, 88, 98])
```

```
[24] mat[:6]
```

```
array([[ 0,  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]])
```

```
[25] mat[4:]
```

```
array([[40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
[26] mat[:row]# it will print rows from 4th till last rows
```

```
array([[ 0,  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]])
```

```
[27] mat[:,col]
```

```
array([ 5, 15, 25, 35, 45, 55, 65, 75, 85, 95])
```

```
[28] mat[:col]#print till 4th columns
```

```
array([[ 0,  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]])
```

```
[30] mat
```

```
array([[ 0,  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, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
[29] mat[2:6,2:4]
```

```
array([[22, 23],
       [32, 33],
       [42, 43],
       [52, 53]])
```

```
[31] mat[1:2,2:4]
```

```
array([[12, 13]])
```

```
[32] mat[2:4,3:5]
```

```
array([[23, 24],
       [33, 34]])
```

```
[33] mat
```

```
array([[ 0,  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, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
[34] mat < 50
```

```
array([[ True,  True,  True,  True,  True,  True,  True,  True,  True
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True
         True],
       [False, False, False, False, False, False, False, False, False
        False],
       [False, False, False, False, False, False, False, False, False
        False],
       [False, False, False, False, False, False, False, False, False
        False],
       [False, False, False, False, False, False, False, False, False
        False],
       [False, False, False, False, False, False, False, False, False
        False]])
```

```
[35] mat[mat<50]#filter
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49
```

```
[37] mat[mat<=50]
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
```

```
[38] mat[mat==50]
```

```
array([50])
```

```
[36] mat[mat != 50]
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
       52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

```
[39] mat[mat>50]
```

```
array([51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

```
[107] # Create an array from a list
      a = np.array([1, 2, 3])
      print("Array a:", a)
```

```
Array a: [1 2 3]
```

```
[92] # Create an array with linearly spaced values
     c = np.linspace(0, 1, 5)
     print("Array c:", c)
```

```
Array c: [0.   0.25 0.5  0.75 1.  ]
```

```
[93] d = np.zeros((2, 3))  # 2x3 array of zeros
     print("Array d:\n", d)
```

```
Array d:
 [[0. 0. 0.]
 [0. 0. 0.]]
```

```
[94] e = np.ones((3, 2))  # 3x2 array of ones
     print("Array e:\n", e)
```

```
Array e:
 [[1. 1.]
 [1. 1.]
 [1. 1.]]
```

```
[95] f = np.eye(4)  # 4x4 identity matrix
     print("Identity matrix f:\n", f)
```

```
Identity matrix f:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

```
[96] #Array Manipulation Function
     a1 = np.array([1, 2, 3])
     reshaped = np.reshape(a1, (1, 3))  # Reshape to 1x3
     print("Reshaped array:", reshaped)
```

```
Reshaped array: [[1 2 3]]
```

```
[97] # Flatten an array
     f1 = np.array([[1, 2], [3, 4]])
     flattened = np.ravel(f1)  # Flatten to 1D array
     print("Flattened array:", flattened)
```

```
Flattened array: [1 2 3 4]
```


```
# Transpose an array
e1 = np.array([[1, 2], [3, 4]])
transposed = np.transpose(e1)
print("Transposed array:\n", transposed)
```

```
Transposed array:
 [[1 3]
 [2 4]]
```

```
[99] # Stack arrays vertically
     a2 = np.array([2, 3])
     b2 = np.array([4, 5])
     stacked = np.vstack([a2, b2])
     print("Stacked arrays:\n", stacked)
```

```
Stacked arrays:
 [[2 3]
 [4 5]]
```

```
[100] #mathematical Function
      # Add two arrays
      g = np.array([1, 2, 3, 4])
      added = np.add(g, 2)
      print("Added 2 to g:", added)
```

```
Added 2 to g: [3 4 5 6]
```

```
[102] # Square each element
      squared = np.power(g, 3)
      print("Squared g:", squared)
```

```
Squared g: [ 1  8 27 64]
```

```
      sqrt_val = np.sqr
      print("Square root of g:", sqrt_val)
```

Loading...

```
Square root of g: [1.         1.41421356 1.73205081 2.        ]
```

```
[ ]  print(a1)
     print(g)
```

```
[108] a3 = np.array([1, 2, 3])
      dot_product = np.dot(a1, a)  # Dot product of a and g
      print("Dot product of a1 and a:", dot_product)
```

```
Dot product of a1 and a: 14
```

```
[110] #statistical Function
      s = np.array([2, 3, 4, 5])
      mean = np.mean(s)
      print("Mean of s:", mean)
```

```
Mean of s: 3.5
```

```
[114] std_dev = np.std(s)
      print("Standard deviation of s:", std_dev)
```

```
Standard deviation of s: 1.118033988749895
```

```
[115] minimum = np.min(s)
      print("Min of s:", minimum)
```

```
Min of s: 2
```

```
[116] maximum = np.max(s)
      print("Max of s:", maximum)
```

```
Max of s: 5
```

```
[117] #Linear Algebra Dunctions
      matrix = np.array([[1, 2], [3, 4]])
```

```
[118] determinant = np.linalg.det(matrix)
      print("Determinant of matrix:", determinant)
```

```
Determinant of matrix: -2.0000000000000004
```

```
[119] inverse = np.linalg.inv(matrix)
      print("Inverse of matrix:\n", inverse)
```

```
Inverse of matrix:
 [[-2.   1. ]
 [ 1.5 -0.5]]
```

```
#Random Sampling Functions
random_vals = np.random.rand(3)
print("Random values:", random_vals)
```

```
Random values: [0.03485484 0.2505857  0.41964462]
```

```
[121] np.random.seed(0)

      # Generate random values between 0 and 1
      random_vals = np.random.rand(3)
      print("Random values:", random_vals)
```

```
Random values: [0.5488135  0.71518937 0.60276338]
```

```
[123] rand_ints = np.random.randint(0, 20, size=5)
      print("Random integers:", rand_ints)
```

```
Random integers: [18  4  6 12  1]
```

```
np.random.seed(0)

# Generate random integers
rand_ints = np.random.randint(0, 20, size=5)
print("Random integers:", rand_ints)
```

```
Random integers: [12 15  0  3  3]
```

```
[126] #Boolean and Logical Functions
      logical_test = np.array([True, False, True])
      all_true = np.all(logical_test)
      print("All elements True:", all_true)
```

```
All elements True: False
```

```
[128] logical_test = np.array([False, False, False])
      all_true = np.all(logical_test)
      print("All elements True:", all_true)
```

```
All elements True: False
```

```
[129] any_true = np.any(logical_test)
      print("Any elements True:", any_true)
```

```
Any elements True: False
```

```
[132] #Set Operations
      set_a = np.array([2, 3, 4, 5])
      set_b = np.array([4, 5,7, 8])
      intersection = np.intersect1d(set_a, set_b)
      print("Intersection of a and b:", intersection)
```

```
Intersection of a and b: [4 5]
```

```
[133] union = np.union1d(set_a, set_b)
      print("Union of a and b:", union)
```

```
Union of a and b: [2 3 4 5 7 8]
```

```
#Array Attribute Functions
a = np.array([3, 4, 5])
shape = a.shape  # Shape of the array
size = a.size    # Number of elements
dimensions = a.ndim  # Number of dimensions
dtype = a.dtype   # Data type of the array

print("Shape of a:", shape)
print("Size of a:", size)
print("Number of dimensions of a:", dimensions)
print("Data type of a:", dtype)
```

```
Shape of a: (3,)
Size of a: 3
Number of dimensions of a: 1
Data type of a: int64
```

```
[137] #Other Functions
      a = np.array([3, 4, 5])
      copied_array = np.copy(a)
      print("Copied array:", copied_array)
```

```
Copied array: [3 4 5]
```

```
# Size in bytes of an array
array_size_in_bytes = a.nbytes
print("Size of a in bytes:", array_size_in_bytes)
```

```
Size of a in bytes: 24
```

```
[138] shared = np.shares_memory(a, copied_array)
print("Do a and copied_array share memory?", shared)
```

```
Do a and copied_array share memory? False
```