lab1

April 10, 2024

[1]:

```
import numpy as np
```

1. Create a rank 2 (2D) array that resembles the matrix below. [[11 12 13 14] [15 16 17 18]]

[9]:

```
x1 = np.array([[11,12,13,14],[15,16,17,18]])
print(x1)
```

```
[[11 12 13 14]
 [15 16 17 18]]
```

1. Create an array with 4 rows and 3 columns of zeros. Your results should look as below: [[0. 0. 0.] [0. 0. 0.] [0. 0. 0.] [0. 0. 0.]]

2.Create an array of ones that has 3 rows and 4 columns. Your results should look as below: [[1. 1. 1. 1.] [1. 1. 1. 1.] [1. 1. 1. 1.]]

[14]:

```
x2 = np.zeros((4,3))
print(x2)

x3 = np.ones((3,4))
print(x3)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

3. Create an array containing integers 4 to 13 inclusive. Your results should look as below: [4 5 6 7 8 9 10 11 12 13]

[16]:

```
x4 = np.arange(4,14,1)
print(x4)
```

```
[ 4  5  6  7  8  9 10 11 12 13]
```

4. Create an array containing [0., 1.5, 3., 4.5]

[34]:

```
x5 = np.linspace(0,4.5,4)
print(x5)
```

```
[0.  1.5 3.  4.5]
```

5. Create a 2 by 2 array containing '4' in each position. Your results should look like this: [[4 4] [4 4]]

[21]:

```
x6 = np.full((2,2),4)
print(x6)
```

```
[[4 4]
 [4 4]]
```

6 . Create 2 matrices:i. Identity matrix of size 4 [[1. 0. 0. 0.] [0. 1. 0. 0.] [0. 0. 1. 0.] [0. 0. 0. 1.]] ii. Diagonal matrix with [10,12] as the diagonals [[10 0] [0 12]]

[25]:

```
x7 = np.eye(4)
print(x7)
x8 = np.diag([10,12])
print(x8)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
[[10  0]
 [ 0 12]]
```

7. Create a 3 by 3 array with random floats in [0, 10]. Your answer may be different because it is random but it should look something like this: [[6.3685612 0.61720883 8.93157783] [3.69927617 5.79879583 7.62145626] [7.21895112 4.02011535 4.48844787]]

[28]:

```
x9 = np.random.random((3,3))*10
print(x9)
```

```
[[6.03175075 2.58801226 6.7399004 ]
 [1.50701524 5.45266679 5.29012722]
 [4.12470517 9.35405781 3.04751922]]
```

8. Create a 3 by 3 array with random integers in [10, 20]. Your answer may be different because it is random but it should look something like this: [[19 11 13] [10 11 13] [11 14 10]]

[37]:

```
x10 = np.random.randint(10, 20, (3, 3))
print(x10)
```

```
[[16 10 14]
 [16 11 12]
 [19 18 17]]
```

1. Use this array for the following practice: myArray = np.array([[11,12,13], [14,15,16], [17,18,19]])a. Get a subarray of the first row and first 2 columns. Your results should look like this: [11 12]

b. Change all elements in 1st and second row to 0. Your results should look like this: [[0 0 0] [0 0 0] [17 18 19]]

```
[44]: x11 = np.array([[11,12,13], [14,15,16], [17,18,19]])
      print(x11[0,:2])
      x11[:2,:] = np.zeros((2,3))
      print(x11)
```

```
[11 12]
[[ 0  0  0]
 [ 0  0  0]
 [17 18 19]]
```

2. Create an array that contains [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20] and reverse the order.

```
[48]: x12 = np.arange(0,21,1)
      x12[::-1]
```

```
[48]: array([20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10,  9,  8,  7,  6,  5,  4,
              3,  2,  1,  0])
```

Use this array for the following practice: myArray = np.array([[11,12,13], [14,15,16]])Reshape the array to an array with 3 rows. Your results should look like this: [[11 12] [13 14] [15 16]]

```
[50]: x13 = np.array([[11,12,13], [14,15,16]])
      print(x13.reshape(3,2))
```

```
[[11 12]
 [13 14]
 [15 16]]
```

1. Find the square of every number in array
2. Find the square root of every number in array
3. Multiply the square of each number in array with its respective square root

```
[56]: x14 = np.arange(10)
      print(x14)
      print(x14**2)
      print(x14**0.5)
      print((x14**2) * (x14**0.5))
```

```
[0 1 2 3 4 5 6 7 8 9]
[ 0  1  4  9 16 25 36 49 64 81]
[0.         1.         1.41421356 1.73205081 2.         2.23606798
 2.44948974 2.64575131 2.82842712 3.        ]
[  0.           1.           5.65685425  15.58845727  32.
  55.90169944  88.18163074 129.64181424 181.01933598 243.        ]
```

1. Add a new row of elements containing 20, 21 and 22
2. Add a new column of elements containing 30, 40 and 50

```
[70]: x15 = np.array([[11,12,13], [14,15,16], [17,18,19]])
      x16 = np.array([[20,21,22]])
```

```
x17 = np.concatenate([x15,x16])
print(x17)
x18 = np.array([[30],[40],[50]])
print(np.hstack([x15,x18]))
```

```
[[11 12 13]
 [14 15 16]
 [17 18 19]
 [20 21 22]]
[[11 12 13 30]
 [14 15 16 40]
 [17 18 19 50]]
```

1. Add 1 column of 1 to this array: myArray = np.zeros((2,2))
2. Add 2 rows of 2 to the answer from part 1
3. Remove the last column
4. Remove the last row

[85]:

```
x19 = np.zeros((2,2))
print(np.hstack([x19, np.array([[1],[1]])]))
x20 = np.hstack([x19, np.array([[1],[1]])])
x21 = np.full((2,3),2)
print(x21)
print(np.concatenate([x20,x21]))
x22 = np.concatenate([x20,x21])
print(x22[:,0:-1])
print(x22[0:-1])
```

```
[[0. 0. 1.]
 [0. 0. 1.]]
[[2 2 2]
 [2 2 2]]
[[0. 0. 1.]
 [0. 0. 1.]
 [2. 2. 2.]
 [2. 2. 2.]]
[[0. 0.]
 [0. 0.]
 [2. 2.]
 [2. 2.]]
[[0. 0. 1.]
 [0. 0. 1.]
 [2. 2. 2.]]
```

Remove the elements from the middle column of this array: myArray = np.matrix([[1, 2, 3], [4, 5, 6], [9, 8, 7]])Your codes should look like this: [[1 3] [4 6] [9 7]]

[2]:

```
x23 = np.matrix([[1, 2, 3], [4, 5, 6], [9, 8, 7]])
x23
```

```
# x23[:,0::2]
np.delete(x23, 1, 1)
# arr = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# arr
# array([[ 1,  2,  3,  4],
#        [ 5,  6,  7,  8],
#        [ 9, 10, 11, 12]])

# np.delete(arr, 1, 0)
# array([[ 1,  2,  3,  4],
#        [ 9, 10, 11, 12]])
```

[2]:
```
matrix([[1, 3],
        [4, 6],
        [9, 7]])
```

Replace all odd numbers in the given array with -1 Start with: exercise_1 = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])Desired output: [0, -1, 2, -1, 4, -1, 6, -1, 8, -1]

[4]:
```
exercise_1 = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

exercise_1[exercise_1 % 2 == 1] = -1
exercise_1
```

[4]:
```
array([ 0, -1,  2, -1,  4, -1,  6, -1,  8, -1])
```

Convert a 1-D array into a 2-D array with 3 rows Start with: exercise_2 = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])Desired output: [[0, 1, 2] [3, 4, 5] [6, 7, 8]]

[89]:
```
exercise_2 = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
print(exercise_2.reshape(3,3))
```

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

Add 202 to all the values in given arrayStart with: exercise_3 = np.arange(4).reshape(2,-1)Desired output: [[202, 203] [204, 205]]

[90]:
```
exercise_3 = np.arange(4).reshape(2,-1)
print(exercise_3 + 202)
```

```
[[202 203]
 [204 205]]
```

Generate a 1-D array of 10 random integers. Each integer should be a number between 30 and 40 (inclusive)Sample of desired output: [36, 30, 36, 38, 31, 35, 36, 30, 32, 34]

[97]:

```
exercise_4 = np.random.randint(30, 41, 10)
print(exercise_4)
```

```
[30 36 32 36 38 40 34 34 32 35]
```

Find the positions of:

```
elements in x where its value is more than its corresponding element in y, and
elements in x where its value is equals to its corresponding element in y.
```

Start with these: x = np.array([21, 64, 86, 22, 74, 55, 81, 79, 90, 89]) y = np.array([21, 7, 3, 45, 10, 29, 55, 4, 37, 18])Desired output: (array([1, 2, 4, 5, 6, 7, 8, 9]),) and (array([0]),)

[7]:

```
x = np.array([21, 64, 86, 22, 74, 55, 81, 79, 90, 89])
y = np.array([21, 7, 3, 45, 10, 29, 55, 4, 37, 18])
morethan = np.where(x > y)
equal= np.where(x == y)
print("Positions where x is more than y:", morethan)
print("Positions where x is equal to y:", equal)
```

```
Positions where x is more than y: (array([1, 2, 4, 5, 6, 7, 8, 9]),)
Positions where x is equal to y: (array([0]),)
```

Extract the first four columns of this 2-D array Start with this: exercise_6 = np.arange(100).reshape(5,-1)Desired output: [[0 1 2 3] [20 21 22 23] [40 41 42 43] [60 61 62 63] [80 81 82 83]]

[115]:

```
exercise_6 = np.arange(100).reshape(5,-1)
print(exercise_6[:,0:4:1])
```

```
[[ 0  1  2  3]
 [20 21 22 23]
 [40 41 42 43]
 [60 61 62 63]
 [80 81 82 83]]
```