NumPy Assignments

December 18, 2024

[1]:
```
import numpy as np
```

[2]:
```
# 1. Write a NumPy program to create an array with the values 1, 7, 13, 105 and determine the size of the memory occupied by the array.

array1 = np.array([1, 7, 13, 105])
print("Memory size of array1:", array1.nbytes)
```

```
Memory size of array1: 32
```

[3]:
```
# 2. Element-wise comparison of two given arrays.

array2 = np.array([5, 10, 15, 20])
comparison_greater = array1 > array2
comparison_greater_equal = array1 >= array2
comparison_less = array1 < array2
comparison_less_equal = array1 <= array2
print("Greater:", comparison_greater)
print("Greater Equal:", comparison_greater_equal)
print("Less:", comparison_less)
print("Less Equal:", comparison_less_equal)
```

```
Greater: [False False False  True]
Greater Equal: [False False False  True]
Less: [ True  True  True False]
Less Equal: [ True  True  True False]
```

[4]:
```
# 3. Create an array of 10 zeros, 10 ones, 10 fives, 10 tens, 10 twenties and 10 fifties.

zeros = np.zeros(10)
ones = np.ones(10)
fives = np.full(10, 5)
tens = np.full(10, 10)
twenties = np.full(10, 20)
fifties = np.full(10, 50)
print("Zeros:", zeros)
print("Ones:", ones)
```

```
print("Fives:", fives)
print("Tens:", tens)
print("Twenties:", twenties)
print("Fifties:", fifties)
```

```
Zeros: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Ones: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Fives: [5 5 5 5 5 5 5 5 5 5]
Tens: [10 10 10 10 10 10 10 10 10 10]
Twenties: [20 20 20 20 20 20 20 20 20 20]
Fifties: [50 50 50 50 50 50 50 50 50 50]
```

[5]:
```
# 4. Create an array of integers from 30 to 70.

array4 = np.arange(30, 71)
print("Array from 30 to 70:", array4)
```

```
Array from 30 to 70: [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]
```

[6]:
```
# 5. Create an array of integers from 50 to 95.

array5 = np.arange(50, 96)
print("Array from 50 to 95:", array5)
```

```
Array from 50 to 95: [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]
```

[7]:
```
# 6. Create an array of all even integers from 20 to 80.

even_array = np.arange(20, 81, 2)
print("Even integers from 20 to 80:", even_array)
```

```
Even integers from 20 to 80: [20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52
54 56 58 60 62 64 66
 68 70 72 74 76 78 80]
```

[8]:
```
# 7. Create an array of all odd integers from 20 to 80.

odd_array = np.arange(21, 80, 2)
print("Odd integers from 20 to 80:", odd_array)
```

```
Odd integers from 20 to 80: [21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53
55 57 59 61 63 65 67
 69 71 73 75 77 79]
```

[9]:
```
# 8. Generate an array of 15 random numbers from 10 to 40.

random_array1 = np.random.randint(10, 41, size=15)
print("15 random numbers from 10 to 40:", random_array1)
```

```
15 random numbers from 10 to 40: [23 28 31 24 32 36 38 29 30 16 16 23 20 30 32]
```

[10]:
```
# 8. Generate an array of 15 random numbers from 10 to 40.

random_array1 = np.random.randint(10, 41, size=15)
print("15 random numbers from 10 to 40:", random_array1)
```

```
15 random numbers from 10 to 40: [25 23 24 11 34 21 22 36 16 24 16 10 34 22 14]
```

[11]:
```
# 9. Generate an array of 10 random numbers from 30 to 50.

random_array2 = np.random.randint(30, 51, size=10)
print("10 random numbers from 30 to 50:", random_array2)
```

```
10 random numbers from 30 to 50: [37 33 45 47 37 35 34 46 32 40]
```

[13]:
```
# 10. Generate an array of 20 random numbers from 50 to 90.

random_array3 = np.random.randint(50, 91, size=20)
print("20 random numbers from 50 to 90:", random_array3)
```

```
20 random numbers from 50 to 90: [64 78 76 55 68 55 86 67 75 52 76 70 90 81 79
78 74 59 80 69]
```

[14]:
```
# 11. Create two arrays & perform various mathematical operations.

array6 = np.array([1, 2, 3])
array7 = np.array([4, 5, 6])
print("Addition:", array6 + array7)
print("Subtraction:", array6 - array7)
print("Multiplication:", array6 * array7)
print("Division:", array6 / array7)
```

```
Addition: [5 7 9]
Subtraction: [-3 -3 -3]
Multiplication: [ 4 10 18]
Division: [0.25 0.4  0.5 ]
```

[15]:
```
# 12. Create a 3x4 matrix filled with values from 10 to 21.

matrix1 = np.arange(10, 22).reshape(3, 4)
print("3x4 matrix filled with values from 10 to 21:\n", matrix1)
```

```
3x4 matrix filled with values from 10 to 21:
 [[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]]
```

[16]:
```
# 13. Create a 3x3 identity matrix.

identity_matrix = np.eye(3)
print("3x3 Identity matrix:\n", identity_matrix)
```

```
3x3 Identity matrix:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

[17]:
```
# 14. Find the number of rows and columns in a given matrix.

rows, cols = matrix1.shape
print("Number of rows:", rows, "Number of columns:", cols)
```

```
Number of rows: 3 Number of columns: 4
```

[18]:
```
# 15. Create a 5x5 zero matrix with elements on the main diagonal equal to 1,␣
 ↪2, 3, 4, 5.

zero_matrix = np.zeros((5, 5))
np.fill_diagonal(zero_matrix, [1, 2, 3, 4, 5])
print("5x5 zero matrix with diagonal elements 1 to 5:\n", zero_matrix)
```

```
5x5 zero matrix with diagonal elements 1 to 5:
 [[1. 0. 0. 0. 0.]
 [0. 2. 0. 0. 0.]
 [0. 0. 3. 0. 0.]
 [0. 0. 0. 4. 0.]
 [0. 0. 0. 0. 5.]]
```

[19]:
```
# 16. Create a 3x3x3 array filled with arbitrary values.

array8 = np.random.rand(3, 3, 3)
print("3x3x3 array filled with arbitrary values:\n", array8)
```

```
3x3x3 array filled with arbitrary values:
 [[[0.97540178 0.50729717 0.32517935]
  [0.31310021 0.17688811 0.18397219]
  [0.75246942 0.75160081 0.03044316]]

 [[0.48307182 0.28342164 0.48613565]
  [0.65038726 0.62660704 0.31540339]
```

```
  [0.3543918  0.62704435 0.83740412]]

 [[0.12079278 0.90675643 0.75492879]
  [0.37462359 0.72043366 0.03074496]
  [0.60030663 0.49136061 0.65341253]]]
```

[20]:
```
# 17. Create a 2x3x4 array filled with arbitrary values.

array9 = np.random.rand(2, 3, 4)
print("2x3x4 array filled with arbitrary values:\n", array9)
```

```
2x3x4 array filled with arbitrary values:
 [[[0.67241107 0.63115467 0.41596258 0.12719255]
  [0.70073131 0.77705145 0.06104228 0.54382396]
  [0.64667874 0.73521635 0.7488712  0.04952721]]

 [[0.8633295  0.06644852 0.28861425 0.76342016]
  [0.72233604 0.69118069 0.34897395 0.77384075]
  [0.02480684 0.0755945  0.86512406 0.11170709]]]
```

[21]:
```
# 18. Convert a list of numeric values into a one-dimensional NumPy array.

list_values = [1, 2, 3, 4, 5]
array10 = np.array(list_values)
print("One-dimensional NumPy array from list:", array10)
```

```
One-dimensional NumPy array from list: [1 2 3 4 5]
```

[22]:
```
# 19. Create a 3x3 matrix with values ranging from 2 to 10.

matrix2 = np.arange(2, 11).reshape(3, 3)
print("3x3 matrix with values from 2 to 10:\n", matrix2)
```

```
3x3 matrix with values from 2 to 10:
 [[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

[23]:
```
# 20. Create an array with values ranging from 12 to 38.

array11 = np.arange(12, 39)
print("Array with values from 12 to 38:", array11)
```

```
Array with values from 12 to 38: [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]
```

[24]:
```
# 21. Reverse an array.

array12 = np.array([1, 2, 3, 4, 5])
reversed_array = array12[::-1]
print("Reversed array:", reversed_array)
```

```
Reversed array: [5 4 3 2 1]
```

[25]:
```
# 22. Convert a data type into a floating type.

array13 = np.array([1, 2, 3], dtype=int)
float_array = array13.astype(float)
print("Converted to float type:", float_array)
```

```
Converted to float type: [1. 2. 3.]
```

[26]:
```
# 23. Convert a list array.

list_array = [1, 2, 3, 4, 5]
converted_array = np.array(list_array)
print("Converted list to NumPy array:", converted_array)
```

```
Converted list to NumPy array: [1 2 3 4 5]
```

[27]:
```
# 24. Get the element-wise remainder of an array of division.

array14 = np.array([10, 20, 30])
remainder_array = array14 % 3
print("Element-wise remainder of division by 3:", remainder_array)
```

```
Element-wise remainder of division by 3: [1 2 0]
```

[28]:
```
# 25. Get the element-wise remainder of an array of numbers from 20 to 50 which␣
 ↪are divisible by 3.

divisible_by_3 = np.arange(20, 51)
remainder_div3 = divisible_by_3[divisible_by_3 % 3 == 0] % 3
print("Element-wise remainder of numbers from 20 to 50 divisible by 3:",␣
 ↪remainder_div3)
```

```
Element-wise remainder of numbers from 20 to 50 divisible by 3: [0 0 0 0 0 0 0 0
0 0]
```