Numpy Assignment

February 4, 2024

[1]:
```
# ques 1. What is a Python library? Why ^o we use Python libraries?

# Ans- Collection of function or a collection of code that makes everyday tasks␣
 ↪more efficient called  Python Library.
#It is also called numerical library .
# we use it because it is the fastest one.
```

[4]:
```
#  ques 2 - Difference between Numpy array and List?
'''Both lists and arrays may hold ordered objects and are changeable.
Arrays can only hold elements of the same type, whereas lists may store␣
 ↪elements of multiple types.
Compared to arrays, lists are more flexible since they don't require explicit␣
 ↪looping to print their elements.
While we can perform arithmetic operations directly on arrays, we are unable to␣
 ↪do so on lists.'''
```

[4]:
```
'Both lists and arrays may hold ordered objects and are changeable.\nArrays can
only hold elements of the same type, whereas lists may store elements of
multiple types.\nCompared to arrays, lists are more flexible since they don't
require explicit looping to print their elements.\nWhile we can perform
arithmetic operations directly on arrays, we are unable to do so on lists.'
```

[5]:
```
''' ques 3- Find the shape, size and dimension of the following array?
[[1, 2, 3, 4]
[5, 6, 7, 8],
[9, 10, 11, 12]]'''
```

[5]:
```
'Find the shape, size and dimension of the following array?\n[[1, 2, 3, 4]\n[5,
6, 7, 8],\n[9, 10, 11, 12]]'
```

[8]:
```
import numpy as np
```

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

[14]:
```
arr.size # find size
```

[14]:
```
12
```

[16]:
```
arr.shape # finding shape
```

[16]:
```
(3, 4)
```

[17]:
```
arr.ndim # finding the dimenion of array
```

[17]:
```
2
```

[18]:
```
# ques 4 - Write python co^e to access the first row of the following array?
arr= np.array([[1, 2, 3, 4],
[5, 6, 7, 8],
[9, 10, 11, 12]])
```

[19]:
```
arr
```

[19]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

[20]:
```
arr[0] # access firt row of an array
```

[20]:
```
array([1, 2, 3, 4])
```

[28]:
```
#ques 7 -generate a ran^om 3x3 matrix with values between 0 an^ 1?

arr=np.random.randint(0,1, (3,3))
```

[30]:
```
arr
```

[30]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

[35]:
```
arr=np.random.randint(0,1, (3,3))
```

[36]:
```
arr
```

[36]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

[2]:
```
# ques 8 -Describe the ^ifference between np.ran^om.ran^ an^ np.ran^om.ran^n?

#Ans- np.random.rand:
```

```
"""np.random.rand
This function generates random numbers from a uniform distribution over the␣
 ↪range [0, 1).
It takes dimensions as arguments and returns an array of random values with␣
 ↪those dimensions.
np.random.randn
This function generates random numbers from a standard normal distribution␣
 ↪(mean=0, standard deviation=1).
It also takes dimensions as arguments and returns an array of random values␣
 ↪with those dimensions"""
```

[2]:
```
'np.random.rand\nThis function generates random numbers from a uniform
distribution over the range [0, 1).\nIt takes dimensions as arguments and
returns an array of random values with those dimensions.\nnp.random.randn\nThis
function generates random numbers from a standard normal distribution (mean=0,
standard deviation=1).\nIt also takes dimensions as arguments and returns an
array of random values with those dimensions'
```

[3]:
```
#ques 9 - Write co^e to increase the ^imension of the following array?
```

[6]:
```
import numpy as np

# Original 2D array
original_array = np.array([[1, 2, 3, 4],
                           [5, 6, 7, 8],
                           [9, 10, 11, 12]])

# Reshape to a 3D array with shape (1, 3, 4)
increased_dimension_array = original_array.reshape(1, 3, 4)
```

[8]:
```
increased_dimension_array
```

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

[9]:
```
#ques 10- How to transpose the following array in NumPy?
```

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

# Transpose using numpy.transpose
transposed_array = np.transpose(original_array)

# Alternatively, you can use the .T attribute
```

```
# transposed_array = original_array.T

print("Original array:")
print(original_array)

print("\nTransposed array:")
print(transposed_array)
```

```
Original array:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

Transposed array:
[[ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]
 [ 4  8 12]]
```

[12]:
```
# ques 11- Consi^er the following matrix:
```

[16]:
```
# Matrix A
matrix_A = np.array([[1, 2, 3, 4],
                     [5, 6, 7, 8],
                     [9, 10, 11, 12]])

# Matrix B
matrix_B = np.array([[1, 2, 3, 4],
                     [5, 6, 7, 8],
                     [9, 10, 11, 12]])
```

[17]:
```
# index wise multiplication
matrix_A * matrix_B
```

[17]:
```
array([[  1,   4,   9,  16],
       [ 25,  36,  49,  64],
       [ 81, 100, 121, 144]])
```

[20]:
```
#matrix_multiplication
np.dot(matrix_A, matrix_B.T)
```

[20]:
```
array([[ 30,  70, 110],
       [ 70, 174, 278],
       [110, 278, 446]])
```

[21]:
```
#matrix_addition
matrix_A + matrix_B
```

[21]:
```
array([[ 2,  4,  6,  8],
       [10, 12, 14, 16],
       [18, 20, 22, 24]])
```

[22]:
```
# matrix_subtraction
matrix_A - matrix_B
```

[22]:
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

[23]:
```
# matrix_division
np.divide(matrix_B, matrix_A)
```

[23]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

[24]:
```
# ques 12 - Which function in Numpy can be use^ to swap the byte or^er of an␣
 ↪array?

# In NumPy, We  can use the byteswap function to swap the byte order of an␣
 ↪array. The byteswap function is available as a method of the NumPy array␣
 ↪object.
#It can be used to change the endianness (byte order) of the data in the array.
```

[25]:
```
# ques 13 -What is the significance of the np.linalg.inv function?
# The np.linalg.inv function in NumPy is used to compute the (multiplicative)␣
 ↪inverse of a square matrix.
#In linear algebra, the inverse of a matrix A, denoted as A^(-1),
#is such that when A is multiplied by its inverse, the result is the identity␣
 ↪matrix.
```

[26]:
```
#ques 14 - What ^oes the np.reshape function ^o, an^ how is it use^?
"""
The np.reshape function in NumPy is used to change the shape of an array␣
 ↪without changing its data.
It allows you to give a new shape to an existing array without altering its␣
 ↪elements. The new shape must have the same number of elements as the␣
 ↪original array.
This function returns a new array with the specified shape."""
```

[26]:
```
' \nThe np.reshape function in NumPy is used to change the shape of an array
without changing its data.\nIt allows you to give a new shape to an existing
array without altering its elements. The new shape must have the same number of
elements as the original array. \nThis function returns a new array with the
specified shape.'
```

[27]:
```
# quews 15 - What is broa^casting in Numpy?
"""Broadcasting in NumPy is a powerful mechanism that allows for performing␣
 ↪element-wise operations on arrays of different shapes and sizes.
It enables NumPy to work with arrays of different shapes during arithmetic␣
 ↪operations without explicitly reshaping them, making code more concise and␣
 ↪efficient.

The broadcasting rules are applied when performing binary operations (addition,␣
 ↪subtraction, multiplication, etc.) on arrays with different shapes.
The smaller array is "broadcast" across the larger array so that they have␣
 ↪compatible shapes for the operation."""
```

[27]:
```
'Broadcasting in NumPy is a powerful mechanism that allows for performing
element-wise operations on arrays of different shapes and sizes.\nIt enables
NumPy to work with arrays of different shapes during arithmetic operations
without explicitly reshaping them, making code more concise and
efficient.\n\nThe broadcasting rules are applied when performing binary
operations (addition, subtraction, multiplication, etc.) on arrays with
different shapes. \nThe smaller array is "broadcast" across the larger array so
that they have compatible shapes for the operation.'
```

[ ]: