PW SKILLS

Assignment

NumPy

1. Create a NumPy array 'arr' of integers from 0 to 5 and print its data type.

2. Given a NumPy array 'arr', check if its data type is float64.

```
arr = np.array([1.5, 2.6, 3.7])
```

3. Create a NumPy array 'arr' with a data type of complex128 containing three complex numbers.

4. Convert an existing NumPy array 'arr' of integers to float32 data type.

5. Given a NumPy array 'arr' with float64 data type, convert it to float32 to reduce decimal precision.

6. Write a function array_attributes that takes a NumPy array as input and returns its shape, size, and data type.

7. Create a function array_dimension that takes a NumPy array as input and returns its dimensionality.

8. Design a function item_size_info that takes a NumPy array as input and returns the item size and the total size in bytes.

9. Create a function array_strides that takes a NumPy array as input and returns the strides of the array.

10. Design a function shape_stride_relationship that takes a NumPy array as input and returns the shape and strides of the array.

11. Create a function `create_zeros_array` that takes an integer `n` as input and returns a NumPy array of zeros with `n` elements.

12. Write a function `create_ones_matrix` that takes integers `rows` and `cols` as inputs and generates a 2D NumPy array filled with ones of size `rows x cols`.

13. Write a function `generate_range_array` that takes three integers start, stop, and step as arguments and creates a NumPy array with a range starting from `start`, ending at stop (exclusive), and with the specified `step`.

14. Design a function `generate_linear_space` that takes two floats `start`, `stop`, and an integer `num` as arguments and generates a NumPy array with num equally spaced values between `start` and `stop` (inclusive).

15. Create a function `create_identity_matrix` that takes an integer `n` as input and generates a square identity matrix of size `n x n` using `numpy.eye`.

16. Write a function that takes a Python list and converts it into a NumPy array.

17. Create a NumPy array and demonstrate the use of `numpy.view` to create a new array object with the same data.

18. Write a function that takes two NumPy arrays and concatenates them along a specified axis.

19. Create two NumPy arrays with different shapes and concatenate them horizontally using `numpy.concatenate`.

20. Write a function that vertically stacks multiple NumPy arrays given as a list.

21. Write a Python function using NumPy to create an array of integers within a specified range (inclusive) with a given step size.

22. Write a Python function using NumPy to generate an array of 10 equally spaced values between 0 and 1 (inclusive).

23. Write a Python function using NumPy to create an array of 5 logarithmically spaced values between 1 and 1000 (inclusive).

24. Create a Pandas DataFrame using a NumPy array that contains 5 rows and 3 columns, where the values are random integers between 1 and 100.

25. Write a function that takes a Pandas DataFrame and replaces all negative values in a specific column with zeros. Use NumPy operations within the Pandas DataFrame.

26. Access the 3rd element from the given NumPy array.

```
arr = np.array([10, 20, 30, 40, 50])
```

27. Retrieve the element at index (1, 2) from the 2D NumPy array.

```
arr_2d = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
```

28. Using boolean indexing, extract elements greater than 5 from the given NumPy array.

```
arr = np.array([3, 8, 2, 10, 5, 7])
```

29. Perform basic slicing to extract elements from index 2 to 5 (inclusive) from the given NumPy array.

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

30. Slice the 2D NumPy array to extract the sub-array `[[2, 3], [5, 6]]` from the given array.

```
arr_2d = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
```

31.Write a NumPy function to extract elements in specific order from a given 2D array based on indices provided in another array.

32. Create a NumPy function that filters elements greater than a threshold from a given 1D array using boolean indexing.

33. Develop a NumPy function that extracts specific elements from a 3D array using indices provided in three separate arrays for each dimension.

34. Write a NumPy function that returns elements from an array where both two conditions are satisfied using boolean indexing.

35. Create a NumPy function that extracts elements from a 2D array using row and column indices provided in separate arrays.

36. Given an array arr of shape (3, 3), add a scalar value of 5 to each element using NumPy broadcasting.

37. Consider two arrays arr1 of shape (1, 3) and arr2 of shape (3, 4). Multiply each row of arr2 by the corresponding element in arr1 using NumPy broadcasting.

38. Given a 1D array arr1 of shape (1, 4) and a 2D array arr2 of shape (4, 3), add arr1 to each row of arr2 using NumPy broadcasting.

39. Consider two arrays arr1 of shape (3, 1) and arr2 of shape (1, 3). Add these arrays using NumPy broadcasting.

40. Given arrays arr1 of shape (2, 3) and arr2 of shape (2, 2), perform multiplication using NumPy broadcasting. Handle the shape incompatibility.

41. Calculate column-wise mean for the given array:

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
```

42. Find maximum value in each row of the given array:

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
```

43. For the given array, find indices of maximum value in each column.

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
```

44. For the given array, apply custom function to calculate moving sum along rows.

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
```

45. In the given array, check if all elements in each column are even.

```
arr = np.array([[2, 4, 6], [3, 5, 7]])
```

46. Given a NumPy array arr, reshape it into a matrix of dimensions `m` rows and `n` columns. Return the reshaped matrix.

```
original_array = np.array([1, 2, 3, 4, 5, 6])
```

47. Create a function that takes a matrix as input and returns the flattened array.

```
input_matrix = np.array([[1, 2, 3], [4, 5, 6]])
```

48. Write a function that concatenates two given arrays along a specified axis.

```
array1 = np.array([[1, 2], [3, 4]])
array2 = np.array([[5, 6], [7, 8]])
```

49. Create a function that splits an array into multiple sub-arrays along a specified axis.

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

50. Write a function that inserts and then deletes elements from a given array at specified indices.

```
original_array = np.array([1, 2, 3, 4, 5])
indices_to_insert = [2, 4]
values_to_insert = [10, 11]
indices_to_delete = [1, 3]
```

51. Create a NumPy array `arr1` with random integers and another array `arr2` with integers from 1 to 10. Perform element-wise addition between `arr1` and `arr2`.

52. Generate a NumPy array `arr1` with sequential integers from 10 to 1 and another array `arr2` with integers from 1 to 10. Subtract `arr2` from `arr1` element-wise.

53. Create a NumPy array `arr1` with random integers and another array `arr2` with integers from 1 to 5. Perform element-wise multiplication between `arr1` and `arr2`.

54. Generate a NumPy array `arr1` with even integers from 2 to 10 and another array `arr2` with integers from 1 to 5. Perform element-wise division of `arr1` by `arr2`.

55. Create a NumPy array `arr1` with integers from 1 to 5 and another array `arr2` with the same numbers reversed. Calculate the exponentiation of `arr1` raised to the power of `arr2` element-wise.

56. Write a function that counts the occurrences of a specific substring within a NumPy array of strings.

```
arr = np.array(['hello', 'world', 'hello', 'numpy', 'hello'])
```

57. Write a function that extracts uppercase characters from a NumPy array of strings.
arr = np.array(['Hello', 'World', 'OpenAI', 'GPT'])

```
arr = np.array(['Hello', 'World', 'OpenAI', 'GPT'])
```

58. Write a function that replaces occurrences of a substring in a NumPy array of strings with a new string.

```
arr = np.array(['apple', 'banana', 'grape', 'pineapple'])
```

59. Write a function that concatenates strings in a NumPy array element-wise.

```
arr1 = np.array(['Hello', 'World'])
arr2 = np.array(['Open', 'AI'])
```

60. Write a function that finds the length of the longest string in a NumPy array.
arr = np.array(['apple', 'banana', 'grape', 'pineapple'])

```
arr = np.array(['apple', 'banana', 'grape', 'pineapple'])
```

61. Create a dataset of 100 random integers between 1 and 1000. Compute the mean, median, variance, and standard deviation of the dataset using NumPy's functions.

62. Generate an array of 50 random numbers between 1 and 100. Find the 25th and 75th percentiles of the dataset.

63. Create two arrays representing two sets of variables. Compute the correlation coefficient between these arrays using NumPy's `corrcoef` function.

64. Create two matrices and perform matrix multiplication using NumPy's `dot` function.

65. Create an array of 50 integers between 10 and 1000. Calculate the 10th, 50th (median), and 90th percentiles along with the first and third quartiles.

66. Create a NumPy array of integers and find the index of a specific element.

67. Generate a random NumPy array and sort it in ascending order.

68. Filter elements >20 in the given NumPy array.

```
arr = np.array([12, 25, 6, 42, 8, 30])
```

69. Filter elements which are divisible by 3 from a given NumPy array.

```
arr = np.array([1, 5, 8, 12, 15])
```

70. Filter elements which are ≥ 20 and ≤ 40 from a given NumPy array.

```
arr = np.array([10, 20, 30, 40, 50])
```

71. For the given NumPy array, check its byte order using the `dtype` attribute byteorder.

```
arr = np.array([1, 2, 3])
```

72. For the given NumPy array, perform byte swapping in place using `byteswap()`.

```
arr = np.array([1, 2, 3], dtype=np.int32)
```

73. For the given NumPy array, swap its byte order without modifying the original array using `newbyteorder()`.

```
arr = np.array([1, 2, 3], dtype=np.int32)
```

74. For the given NumPy array and swap its byte order conditionally based on system endianness using `newbyteorder()`.

```
arr = np.array([1, 2, 3], dtype=np.int32)
```

75. For the given NumPy array, check if byte swapping is necessary for the current system using `dtype` attribute `byteorder`.

```
arr = np.array([1, 2, 3], dtype=np.int32)
```

76. Create a NumPy array `arr1` with values from 1 to 10. Create a copy of `arr1` named `copy_arr` and modify an element in `copy_arr`. Check if modifying `copy_arr` affects `arr1`.

77. Create a 2D NumPy array `matrix` of shape (3, 3) with random integers. Extract a slice `view_slice` from the matrix. Modify an element in `view_slice` and observe if it changes the original `matrix`.

78. Create a NumPy array `array_a` of shape (4, 3) with sequential integers from 1 to 12. Extract a slice `view_b` from `array_a` and broadcast the addition of 5 to view_b. Check if it alters the original `array_a`.

79. Create a NumPy array `orig_array` of shape (2, 4) with values from 1 to 8. Create a reshaped view `reshaped_view` of shape (4, 2) from orig_array. Modify an element in `reshaped_view` and check if it reflects changes in the original `orig_array`.

80. Create a NumPy array `data` of shape (3, 4) with random integers. Extract a copy `data_copy` of elements greater than 5. Modify an element in `data_copy` and verify if it affects the original `data`.

81. Create two matrices A and B of identical shape containing integers and perform addition and subtraction operations between them.

82. Generate two matrices `C` (3x2) and `D` (2x4) and perform matrix multiplication.

83. Create a matrix `E` and find its transpose.

84. Generate a square matrix `F` and compute its determinant.

85. Create a square matrix `G` and find its inverse.