Numpy Part 3

```
from typing import Concatenate
import numpy as np

x = np.arange(20)
print(x)
'''
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
'''

y = np.reshape(x, (4, 5))
print(y)
'''
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
'''

x = np.linspace(0, 50, 10, endpoint=False).reshape(5, 2)
print(x)
'''
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
'''

x = np.arange(1, 10).reshape(3, 3)
print(x)
'''
[[1 2 3]
 [4 5 6]
 [7 8 9]]
'''

# Transpose the array
# convert each row to column
# convert each column to row
transposed_x = x.T
print(transposed_x)
'''
[[1 4 7]
 [2 5 8]
 [3 6 9]]
'''

array = np.arange(10)
print(array)        # [0 1 2 3 4 5 6 7 8 9]

# delete elements from array
# syntax: delete(array, object)
# in this example will delete 2, 8 from the array
x = np.delete(array, [2, 8])
```

```
print(x)        # [0 1 3 4 5 6 7 9]

array = np.arange(1, 10).reshape(3, 3)
print(array)
'''
[[1 2 3]
 [4 5 6]
 [7 8 9]]
'''

# delete(array, value, axis)
# axis = 0: refers to the rows
# axos = 1: refers to the columns
# in this example will delete the first row
x = np.delete(array, 0, axis=0)
print(x)
'''
[[4 5 6]
 [7 8 9]]
'''

array = np.arange(1, 10).reshape(3, 3)
print(array)
'''
[[1 2 3]
 [4 5 6]
 [7 8 9]]
'''

# this example will delete the first, third column
x = np.delete(array, [0, 2], axis=1)
print(x)
'''
[[2]
 [5]
 [8]
'''

x = np.arange(0, 5)
print(x)        # [0 1 2 3 4]

# add element to the array
# syntax: append(array, value)
# append: used  to add elements at the end of the list
y = np.append(x, 10)
print(y)         # [ 0  1  2  3  4 10]


x = np.arange(1, 10).reshape(3, 3)
print(x)
'''
[[1 2 3]
 [4 5 6]
 [7 8 9]]
'''

print(x.shape)      # (3, 3)
```

```python
# append in 2D array
# append: used  to add elements at the end of the list
# axis = 0: refers to the rows
y = np.append(x, [[10, 11, 12]], axis=0)
print(y)
'''
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
'''

print(y.shape)      # (4, 3)

# append in 2D array
# append: used  to add elements at the end of the list
# axis = 1: refers to the column
f = np.append(x, [[22], [33], [44]], axis=1)
print(f)
'''
[[ 1  2  3 22]
 [ 4  5  6 33]
 [ 7  8  9 44]]
'''

x = np.array([1, 2, 5, 6, 7])
print(x)        # [1 2 5 6 7]

# insert in 1D array
# insert elements in the array in specific index
# syntax: insert(array, index, values)
x = np.insert(x, 2, [3, 4])
print(x)        # [1 2 3 4 5 6 7]

array = np.arange(1, 10).reshape(3, 3)
print(array)
'''
[[1 2 3]
 [4 5 6]
 [7 8 9]]
'''

# insert in 2D array
# axis = 0: refers to the rows
# syntax: insert(array, index, values, axis)
x = np.insert(array, 1, [4, 5, 6], axis=0)
print(x)
'''
[[1 2 3]
 [4 5 6]
 [4 5 6]
 [7 8 9]]
'''

# axis = 1: refers to the columns
# syntax: insert(array, index, values, axis)
print(array)
'''
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
'''

x = np.insert(array, 1, 5, axis=1)
print(x)
'''
[[1 5 2 3]
 [4 5 5 6]
 [7 5 8 9]]
'''

x = np.insert(array, 1, 2, axis=0)
print(x)
'''
[[1 2 3]
 [2 2 2]
 [4 5 6]
 [7 8 9]]
'''

# create 1D array
arr1 = np.array([1, 2])
print(arr1)     # [1 2]

# create 2D array
arr2 = np.arange(3, 7).reshape(2, 2)
print(arr2)
'''
[[3 4]
 [5 6]]
'''

# add to arrays vertically to each other
# means vertical stack
x = np.vstack((arr1, arr2))
print(x)
'''
[[1 2]
 [3 4]
 [5 6]]
'''

x = np.vstack((arr2, arr1))
print(x)
'''
[[3 4]
 [5 6]
 [1 2]]
'''

array1 = np.arange(0, 2).reshape(2, 1)
print(array1)
'''
[[0]
 [1]]
'''
```

```
array2 = np.arange(2, 6).reshape(2, 2)
print(array2)
'''
[[2 3]
 [4 5]]
'''

# add array1 to array2 horizontally
x = np.hstack((array1, array2))
print(x)
'''
[[0 2 3]
 [1 4 5]]
'''

array1 = np.arange(1, 3)
print(array1)       # [1 2]

# add array1 to array2 vertically
x = np.vstack((array1, array2))
print(x)
'''
[[1 2]
 [2 3]
 [4 5]]
'''

# used to split the array into sub-arrays
# syntax: split(array, num_of_sub_arrays)
# in the following example split the array into 3 sub-arrays
y = np.arange(1, 10).reshape(3, 3)
split_arrays = np.split(y, 3)
print(split_arrays)     # [array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8,
9]])]

# check for positive or negative infinity
# return True if exist infinity value
arr1 = np.array([1, 2, np.inf, 4, -np.inf])
inf_array = np.isinf(arr1)
print(inf_array)        # [False False  True False  True]

# check each element in the array is nan or not
# nan: means not a number
# if value is nan retuen True
array = np.array([1, 2, 3, 4, np.nan, 5])
nan_array = np.isnan(array)
print(nan_array)        # [False False False False  True False]

# concatenating arrays
arr1 = np.arange(1, 10)
arr2 = np.arange(10, 19)
concatenated_array = np.concatenate((arr1, arr2))
print(concatenated_array)       # [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
17 18 19]

# check if the arrays is equal to each other or not
# check all elements in array
# return True if the two arrays equal each other
element_wise_equal = np.array_equal(arr1, arr2)
print(element_wise_equal)       # False

# check if the arrays is equal to each other or not
# check each element in array, return the result in array
# return True if the two arrays equal each other
# Note That: number of elements in first array must equal number of elements in
second array
check_equal = np.equal(arr1, arr2)
print(check_equal)      # [False False False False False False False False False]

array1 = np.arange(5)
array2 = np.arange(5)
# check if the arrays is equal to each other or not
check_array = np.equal(array1, array2)
print(check_array)  # [ True  True  True  True  True]

# check if the arrays is equal to each other or not
check_array = np.array_equal(array1, array2)
print(check_array)      # True

# check if the arrays if not equal or equal
check_not_equal = np.not_equal(array1, array2)
print(check_not_equal)      # [False False False False False]

# check if the arrays if not equal or equal
check_not_equal = np.not_equal(arr1, arr2)
print(check_not_equal)      # [ True  True  True  True  True  True  True  True
True]
```