Numpy Part 4

```
import numpy as np

# ========================
# == Indexing & Slicing ==
# ========================
# - this used for pre-processing before enter the data into a model

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

# first way for indexing
# syntax: [row, column]
print(x[0, 0])      # 0
print([0, 1])       # [0, 1]
print(x[2, 2])      # 12
print(x[3, 4])      # 19

# another way for indexing
print(x[2] [4])     # 14
print(x[1] [0])     # 5

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

# update the array element
x [0, 0] = 54
print(x[0] [0])     # 54

print(x)
'''
[[54  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
'''

z = x[1:4, 2:4]
print(z)
'''
[[ 7  8]
 [12 13]
 [17 18]]
'''
```

```
# print all values of the array
z = x[: , :]
print(z)
'''
[[54  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
'''

# if end is not determined, will complete to the end of the array
# will print rows starting from row_with_index_1 to end
# will print colmns starting from column_with_index_2_to_end
z = x[1: , 2:]
print(z)
'''
[[ 7  8  9]
 [12 13 14]
 [17 18 19]]
'''

# if start is not determined, will start from index 0
z = x[:2, :3]
print(z)
'''
[[54  1  2]
 [ 5  6  7]]
'''

# copy array
# this is a method in numpy
z = np.copy(x)
print(z)
'''
[[54  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
'''

# this is a method in python
y = x[1:, 2:].copy()
print(y)
'''
[[ 7  8  9]
 [12 13 14]
 [17 18 19]]
'''

# extract diagonal elements
y = np.diag(x)
print(y)        # [54  6 12 18]
print(x)
'''
[[54  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
'''

# get the elements that exist on the top of the diagonal radius
z = np.diag(x, k=1)
print(z)        # [ 1  7 13 19]

z = np.diag(x, k=2)
print(z)        # [ 2  8 14]

# get the elements that exist under the diagonal radius
z = np.diag(x, k=-1)
print(z)        # [ 5 11 17]


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

# get the elements on array, without repeat elements
# discard the repeated elements
print(np.unique(x)) # [1 2 3 4 5 6 7 8 9]

# get the elements on array, without repeat elements
x = np.array([[1, 2, 3], [1, 2, 4,], [4, 5, 6]])
print(np.unique(x)) # [1 2 3 4 5 6]

# =========================================
# == slicing using comparision operators ==
# =========================================
x = np.arange(25).reshape(5, 5)
print(x)
'''
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
'''

# print all elements that bigger than 10 as boolean values
print(x > 10)
'''
[[False False False False False]
 [False False False False False]
 [False  True  True  True  True]
 [ True  True  True  True  True]
 [ True  True  True  True  True]]
'''
```

```
# print all elements that smaller than 10
print(x[x < 10])
'''
[0 1 2 3 4 5 6 7 8 9]
'''

# print elements that bigger than 10, and smaller than 17
print(x[(x > 10) & (x < 17)])
'''
[11 12 13 14 15 16]
'''

# change the value of elements that meet the condition, and change it = -1
x[(x > 10) & (x < 17)] = -1
print(x)
'''
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 -1 -1 -1 -1]
 [-1 -1 17 18 19]
 [20 21 22 23 24]]
'''

x = np.arange(1, 6)
y = np.array([6, 7, 2, 8, 4])

# get the intersection elements
print(np.intersect1d(x, y))     # [2 4]

# get the difference elements on array
# get the elements that exist on first array and not exist on the second array
print(np.setdiff1d(x, y))   # [1 3 5]

# get the union elements
# get the elements on the first array and elements on the second array
# union discard the repeated elements
print(np.union1d(x, y))     # [1 2 3 4 5 6 7 8]
```