Lab 1: Basic Python Programming

CPE232 Data Models

---

[1] Variable

1.1 Number Variable

In [19]:
```
num = 100 #integer variable
num2 = 12.5 #float variable
print(num)
print(num2)

print(num + num2)   #addition
print(num - num2)   #subtraction
print(num * num2)   #multiplication
print( num / num2)  #division
```

```
100
12.5
112.5
87.5
1250.0
8.0
```

1.2 String Variable

In [20]:
```
#string variable
string = "Data Models"
print(string) #print complete string

print("Hello " + string)    #print concatenated string
print(string[0])            #print first character of the string
print(string[:4])           #print first to 4th character of the string
print(string[5:])           #print 6th to last character of the string
print(string[1:4])          #print 2nd to 4th character of the string
print(string * 2)           #print string 2 time
```

```
Data Models
Hello Data Models
D
Data
Models
ata
Data ModelsData Models
```

1.3 Boolean Variable

In [21]:
```
#boolean variable
boolean = True
boolean2 = False
```

```
print(boolean)              #print boolean variable
print(not boolean)          #print opposite of boolean variable
print(boolean and boolean2) #print boolean and boolean2
print(boolean or boolean2)  #print boolean or boolean2
```

```
True
False
False
True
```

1.4 List Variable

In [22]:
```
#list variable
list = ["Data",20,123.23,40,50]
another_list = ["Models",60]

print(list)                 #print complete list
print(list[0])              #print first element of the list
print(list[1:3])            #print 2nd to 3rd element of the list
print(list[2:])             #print 3rd to last element of the list
print(another_list)         #print complete another_list
print(another_list * 2)     #print another_list two times
print(list + another_list)  #print concatenated list

list[0] = "CPE232"          #change first element of the list
print(list)                 #print complete list
```

```
['Data', 20, 123.23, 40, 50]
Data
[20, 123.23]
[123.23, 40, 50]
['Models', 60]
['Models', 60, 'Models', 60]
['Data', 20, 123.23, 40, 50, 'Models', 60]
['CPE232', 20, 123.23, 40, 50]
```

1.5 Tuple Variable

In [23]:
```
#tuple variable
tuple = ("Data",20,123.23,40,50)
another_tuple = ("Models",60)

print(tuple)                    #print complete tuple
print(tuple[0])                 #print first element of the tuple
print(tuple[1:3])               #print 2nd to 3rd element of the tuple
print(tuple[2:])                #print 3rd to last element of the tuple
print(tuple * 2)                #print tuple two times
print(tuple + another_tuple)    #print concatenated tuple
```

```
('Data', 20, 123.23, 40, 50)
Data
(20, 123.23)
(123.23, 40, 50)
('Data', 20, 123.23, 40, 50, 'Data', 20, 123.23, 40, 50)
('Data', 20, 123.23, 40, 50, 'Models', 60)
```

In [27]:
```
tuple[0]
```

Out[27]:
```
'Data'
```

In [150…
```
tuple[0] = "CPE232"       #trying to change first element of the tuple but it ca
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[150], line 1
----> 1 tuple[0] = "CPE232"       #trying to change first element of the tuple bu
t it cannot be changed so it gives error

TypeError: 'tuple' object does not support item assignment
```

In [151…
```
# tuple ไม่สามารถเปลี่ยนแปลงค่าได้ แต่ list สามารถเปลี่ยนแปลงค่าได้
```

1.6 Dictionary Variable

In [31]:
```
#dictionary variable
dictionary = {"name":"Alice","age":21}
another_dictionary = {}
another_dictionary["name"] = "Bob"
another_dictionary["age"] = 21

print(dictionary)                   #print complete dictionary
print(dictionary["name"])           #print value for specific key
print(dictionary.keys())            #print all the keys
print(dictionary.values())          #print all the values
print(dictionary.items())           #print all the items
print(another_dictionary)           #print complete another_dictionary
```

```
{'name': 'Alice', 'age': 21}
Alice
dict_keys(['name', 'age'])
dict_values(['Alice', 21])
dict_items([('name', 'Alice'), ('age', 21)])
{'name': 'Bob', 'age': 21}
```

[2] Control Flow

2.1 IF ... ELIF ... ELSE

In [32]:
```
number = 123
number2 = 34

if number > number2:
    print("number is greater thanu number2")
elif number < number2:
    print("number is less than number2")
else:
    print("number is equal to number2")
```

```
number is greater thanu number2
```

[3] Loop

3.1 For Loop

In [33]:
```
#for loops
for num in range(0,10):
    print(num)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [34]:
```
#for loop with list

list = ["Alice","Bob","Charlie","Daisy"]

for name in list:
    print(name)
```

```
Alice
Bob
Charlie
Daisy
```

In [35]:
```
#continue in for loop

list = [1,23,7,"hello",True,1123,43,23,12]

for element in list:
    if type(element) != int:
        continue
    print(element)
```

```
1
23
7
1123
43
23
12
```

In [36]:
```
#break in for loop

list = [1,23,7,"hello",True,1123,43,23,12]

for element in list:
    if type(element) != int:
        break
    print(element)
```

```
1
23
7
```

3.2 While loop

In [37]:
```
#while loop

list = ["Alice","Bob","Charlie","Daisy"]
count = 0

while count < len(list):
    print(list[count])
    count += 1
```

```
Alice
Bob
Charlie
Daisy
```

In [38]:
```
#continue in while loop

list = [1,23,7,"hello",True,1123,43,23,12]
count = 0

while count < len(list):
    if type(list[count]) != int:
        count += 1
        continue
    print(list[count])
    count += 1
```

```
1
23
7
1123
43
23
12
```

In [39]:
```
#break in while loop

list = [1,23,7,"hello",True,1123,43,23,12]
count = 0

while count < len(list):
    if type(list[count]) != int:
        break
    print(list[count])
    count += 1
```

```
1
23
7
```

[4] Function

In [40]:
```
#define function
def function_name (arg1, arg2):
    return arg1 + arg2

#calling function
function_name(1,2)
```

Out[40]: 3

In [2]:
```
#define function with default argument
def function_with_default_arg(arg1, arg2 = 10, arg3 = 20 , arg4 = 30):
    return arg1 + arg2 + arg3 + arg4

result_1 = function_with_default_arg(1)
result_2 = function_with_default_arg(1,2,5)
result_3 = function_with_default_arg(1,2,5,10)

print(result_1)
print(result_2)
print(result_3)
```

```
61
38
18
```

In [41]:
```
#multiple agument
def function_with_multiple_arg(*args):
    print(args)
    print(type(args))
    sum = 0
    for num in args:
        sum += num

    return sum

function_with_multiple_arg(1,2,3,4,5)
```

```
(1, 2, 3, 4, 5)
<class 'tuple'>
```

Out[41]: 15

In [42]:
```
#lambda function
lambda_function = lambda arg1, arg2: arg1 + arg2

print(lambda_function(1,2))
```

```
3
```

[5] File Handling

5.1 Text File

In [43]:
```
with open("test.txt","w") as file:
    file.write("Hello World")
```

In [44]:
```
with open("test.txt","r") as file:
    print(file.read())
```

```
Hello World
```

5.2 CSV File

In [45]:
```
import csv
```

```
with open("test.csv","w",newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Name","Surname"])
        writer.writerow(["Alice","Johnson"])
        writer.writerow(["Bob","Smith"])
```

In [5]:
```
import csv

with open("test.csv","r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

```
['Name', 'Surname']
['Alice', 'Johnson']
['Bob', 'Smith']
```

[4] Libraries

4.1 Numpy

import numpy library

In [46]:
```
import numpy as np
```

ndarray initialization

Construct using python list

In [8]:
```
# pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\punch\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (1.26.3)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: C:\Users\punch\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
```

In [47]:
```
# 1d ndarray from 1d python list
list_a1=[1,2,3.5]
arr_a1=np.array(list_a1)
arr_a1
```

Out[47]:
```
array([1. , 2. , 3.5])
```

In [48]:
```
# 2d ndarray from 2d python list (list of list)
list_a2=[[1,2],[3,4],[5,6]]
arr_a2=np.array(list_a2)
arr_a2
```

Out[48]:
```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [49]:
```
list_a3=[[[1,2],[2,3]],[[3,4],[4,5]]]
arr_a3=np.array(list_a3)
arr_a3
```

Out[49]:
```
array([[[1, 2],
        [2, 3]],

       [[3, 4],
        [4, 5]]])
```

or construct using some numpy classes and functions

In [50]:
```
np.zeros(5)
```

Out[50]:
```
array([0., 0., 0., 0., 0.])
```

In [51]:
```
np.ones((3,4),dtype=float)
```

Out[51]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [52]:
```
np.full((4,),999)
```

Out[52]:
```
array([999, 999, 999, 999])
```

In [16]:
```
np.arange(3,10,2)
```

Out[16]:
```
array([3, 5, 7, 9])
```

In [53]:
```
np.linspace(10,15,11)
```

Out[53]:
```
array([10. , 10.5, 11. , 11.5, 12. , 12.5, 13. , 13.5, 14. , 14.5, 15. ])
```

In [54]:
```
np.random.choice(['a','b'],9)
```

Out[54]:
```
array(['a', 'a', 'b', 'b', 'a', 'b', 'b', 'a', 'b'], dtype='<U1')
```

In [55]:
```
np.random.randn(10)
```

Out[55]:
```
array([ 1.45150399,  0.48203826,  2.38850795, -0.24804012, -0.22868819,
        1.79055569, -0.96285217, -0.53865576, -1.13936602,  1.45321947])
```

ndarray properties

In [56]:
```
list_a=[[1,2,3,4],[5,6,7,8],[9,10,11,12]]
arr_a=np.array(list_a)
arr_a
```

Out[56]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [57]:
```
arr_a.ndim # number of dimensions
```

Out[57]:
```
2
```

In [58]:
```
arr_a.shape # shape of the array
```

Out[58]:
```
(3, 4)
```

In [59]:
```
arr_a.dtype
```

Out[59]:
```
dtype('int32')
```

In [60]:
```
arr_a.size # number of elements in the array
```

Out[60]:
```
12
```

Reshaping & Modification

from this original ndarray

In [61]:
```
arr_a
```

Out[61]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

try to convert into 3D array

In [62]:
```
arr_a.reshape((2,2,3))
```

Out[62]:
```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

sometimes you may resize for same dimension where only known some dimension, insert -1 for unknown len

In [66]:
```
arr_a.reshape((-1,6))
```

Out[66]:
```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12]])
```

Would you like to try this?

In [67]:
```
arr_a
```

Out[67]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [ ]:
```
arr_a.reshape((-1,5))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[69], line 1
----> 1 arr_a.reshape((-1,5))

ValueError: cannot reshape array of size 12 into shape (5)
```

[Q1] From the above cell, explain in your own words why it worked or did not work.

Ans:

In [153…
```
#Q1 : เนื่องจากarr_a มี 12 ตัว จึงไม่สามารถแบ่งเป็น 5 ส่วนได้
```

Next, try to append any value(s) into exist 2darray

In [154…
```
np.append(arr_a,13)
```

Out[154…
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13])
```

In [155…
```
np.append(arr_a,arr_a[0])
```

Out[155…
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,  1,  2,  3,  4])
```

In [156…
```
np.append(arr_a,arr_a[0].reshape((1,-1)),axis=0)
```

Out[156…
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [ 1,  2,  3,  4]])
```

In [157…
```
np.append(arr_a,arr_a[:,0].reshape((-1,1)),axis=1)
```

Out[157…
```
array([[ 1,  2,  3,  4,  1],
       [ 5,  6,  7,  8,  5],
       [ 9, 10, 11, 12,  9]])
```

In [158…
```
np.concatenate([arr_a,arr_a])
```

Out[158…
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [159…
```
np.concatenate([arr_a,arr_a],axis=1)
```

Out[159…
```
array([[ 1,  2,  3,  4,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  5,  6,  7,  8],
       [ 9, 10, 11, 12,  9, 10, 11, 12]])
```

indexing & slicing

from this original array again

In [160…
```
arr_a
```

Out[160…
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

try to access all element at the first row

In [161…
```
arr_a[1] # แถวที่ 2
```

Out[161…
```
array([5, 6, 7, 8])
```

then you would like to access the second element from the first row

In [162…
```
arr_a[1][2] # แถวที่ 2 คอลัมน์ที่ 3
```

Out[162…
```
7
```

In [163…
```
arr_a[1,2] # แถวที่ 2 คอลัมน์ที่ 3
```

Out[163…
```
7
```

Next, try to access all element start from 1th in the first row

In [164…
```
arr_a[1,1:]
```

Out[164…
```
array([6, 7, 8])
```

In [165…
```
arr_a[:2,1:]
```

Out[165…
```
array([[2, 3, 4],
       [6, 7, 8]])
```

sometimes you may specify some row number using list within indicing

In [166…
```
arr_a[[1,2,1],1:]
```

Out[166…
```
array([[ 6,  7,  8],
       [10, 11, 12],
       [ 6,  7,  8]])
```

Boolean slicing

based on this original array

In [167…
```
arr_a
```

Out[167…
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

try to filter all elements which more than 5

In [168…
```
arr_a>5
```

Out[168…
```
array([[False, False, False, False],
       [False,  True,  True,  True],
       [ True,  True,  True,  True]])
```

Next, try to filter all elements which more than 5 and less than 10

In [169…
```
(arr_a>5)&(arr_a<10)
```

Out[169…
```
array([[False, False, False, False],
       [False,  True,  True,  True],
       [ True, False, False, False]])
```

Run the cell below and answer a question.

In [170…
```
arr_a[(arr_a>5)&(arr_a<10)]
```

Out[170…
```
array([6, 7, 8, 9])
```

[Q2] From the above cell, explain in your own words how the output came about?

Ans:

In [ ]:
```
# Q2 : การนำ element ที่มากกว่า 5 และน้อยกว่า 10 มาแสดง
```

Try running the cell below.

In [171…
```
arr_a[(arr_a>5) and (arr_a<10)]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[171], line 1
----> 1 arr_a[(arr_a>5) and (arr_a<10)] # ไม่สามารถใช้ and ได้

ValueError: The truth value of an array with more than one element is ambiguous.
Use a.any() or a.all()
```

[Q3] Explain in your own words why the above cell gives an error.

Ans:

In [98]:
```
# Q3 : เนื่องจากไม่สามารถใช้ and ได้ ต้องใช้ & แทน
```

[Q4] And what should be written instead so that the code is error-free?

Ans:

In [173…
```
# Q4 :
arr_a[(arr_a>5)&(arr_a<10)]
```

Out[173…
```
array([6, 7, 8, 9])
```

Basic operations

In [100…
```
list_b=[[1,2,3,4],[1,2,3,4],[1,2,3,4]]
arr_b=np.array(list_b)
arr_b
```

Out[100…
```
array([[1, 2, 3, 4],
       [1, 2, 3, 4],
       [1, 2, 3, 4]])
```

This is some operations for only 1 array

In [101…
```
np.sqrt(arr_b)
```

Out[101…
```
array([[1.        , 1.41421356, 1.73205081, 2.        ],
       [1.        , 1.41421356, 1.73205081, 2.        ],
       [1.        , 1.41421356, 1.73205081, 2.        ]])
```

This is some operations for 2 arrays with the same shape

In [102…
```
arr_a-arr_b
```

Out[102…
```
array([[0, 0, 0, 0],
       [4, 4, 4, 4],
       [8, 8, 8, 8]])
```

In [ ]:
```
np.add(arr_a,arr_b)
```

Next, try to operate with 1 array and one numeric variable

In [103…
```
arr_a*3
```

Out[103…
```
array([[ 3,  6,  9, 12],
       [15, 18, 21, 24],
       [27, 30, 33, 36]])
```

In [104…
```
1+arr_a**2
```

Out[104…
```
array([[  2,   5,  10,  17],
       [ 26,  37,  50,  65],
       [ 82, 101, 122, 145]])
```

Try to play with 2 arrays with different shape

In [105…
```
arr_c=np.array([1,2,3])
arr_d=np.array([[3],[5],[8]])
```

In [107…
```
arr_c-arr_d
```

Out[107…
```
array([[-2, -1,  0],
       [-4, -3, -2],
       [-7, -6, -5]])
```

Basic aggregations

In [108…
```
arr_a
```

Out[108…
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [109…
```
arr_a.sum()
```

Out[109…
```
78
```

In [110…
```
arr_a.mean()
```

Out[110…
```
6.5
```

In [111…
```
arr_a.min()
```

Out[111…
```
1
```

In [112…
```
arr_a.max()
```

Out[112…
```
12
```

In [113…
```
arr_a.std()
```

Out[113…
```
3.452052529534663
```

ndarray axis

In [114…
```
arr_a
```

Out[114…
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [115…
```
arr_a.sum(axis=0)
```

Out[115…
```
array([15, 18, 21, 24])
```

In [116…
```
arr_a.sum(axis=1)
```

Out[116…
```
array([10, 26, 42])
```

[Q5] Summarize the value of the argument *axis*, what is the value for row-wise summation and column-wise summation, respectively?

Ans:

In [119…
```
# Q5 :
# axis=0 คือ แนวตั้ง และ axis=1 คือ แนวนอน
# ดังนั้น การใช้ axis=0 จะทำการบวกค่าในแนวตั้ง และ axis=1 จะทำการบวกค่าในแนวนอน
```

4.2 Pandas

Series

In [120…
```
import pandas as pd
import numpy as np
```

In [121…
```
pd.Series(np.random.randn(6))
```

Out[121…
```
0    2.203836
1    1.454827
2   -1.212534
3    0.982740
4   -1.285504
5   -0.851363
dtype: float64
```

In [122…
```
pd.Series(np.random.randn(6), index=['a','b','c','d','e','f'])
```

Out[122…
```
a    1.156450
b    0.627432
c   -0.323126
d   -0.843758
e   -0.099458
f   -0.437170
dtype: float64
```

Constructing Dataframe

Constructing DataFrame from a dictionary

In [123…
```
d = {'col1':[1,2], 'col2': [3,4]}
```

In [124…
```
df = pd.DataFrame(data=d)
df
```

Out[124…

	col1	col2
0	1	3
1	2	4

In [125…
```
d2 = {'Name':['Joe','Nat','Harry','Sam','Monica'],
      'Age': [20,21,19,20,22]}
```

In [126…
```
df2 = pd.DataFrame(data=d2)
df2
```

Out[126…

	Name	Age
0	Joe	20
1	Nat	21
2	Harry	19
3	Sam	20
4	Monica	22

Constructing DataFrame from a List

In [127…
```
marks_list = [85.10, 77.80, 91.54, 88.78, 60.55]
```

In [128…
```
df3 = pd.DataFrame(marks_list, columns=['Marks'])
df3
```

Out[128…

	Marks
0	85.10
1	77.80
2	91.54
3	88.78
4	60.55

Creating DataFrame from file

In [129…
```
# Read csv file from path and store to df for create dataframe
df = pd.read_csv('test2.csv')
```

In [130…
```
df
```

Out[130…

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagno
0	150733174	7/11/2015	15.7762	V	5	Male	NaN	
1	150734723	7/6/2015	83.2157	S	36	Male	White	
2	150817487	8/2/2015	74.8813	L	20	Female	NaN	
3	150717776	6/26/2015	15.7762	V	61	Male	NaN	
4	150721694	7/4/2015	74.8813	L	88	Female	Other	
…	…	…	…	…	…	…	…	
334834	150739278	5/31/2015	15.0591	V	7	Male	NaN	
334835	150733393	7/11/2015	5.6748	C	3	Female	Black	
334836	150819286	7/24/2015	15.7762	V	38	Male	NaN	
334837	150823002	8/8/2015	97.9239	M	38	Female	White	
334838	150723074	6/20/2015	49.2646	M	5	Female	White	

334839 rows × 12 columns

Viewing DataFrame information

(.shape, .head, .tail, .info, select column, .unique, .describe, select low with .loc and .iloc)

Check simple information

In [131…
```
# Check dimension by .shape
df.shape
```

Out[131…
```
(334839, 12)
```

In [132...
```
# Display the first 5 rows by default
df.head()
```

Out[132...

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	b
0	150733174	7/11/2015	15.7762	V	5	Male	NaN	57	
1	150734723	7/6/2015	83.2157	S	36	Male	White	57	
2	150817487	8/2/2015	74.8813	L	20	Female	NaN	71	
3	150717776	6/26/2015	15.7762	V	61	Male	NaN	71	
4	150721694	7/4/2015	74.8813	L	88	Female	Other	62	

In [133...
```
# Display the first 3 rows
df.head(3)
```

Out[133...

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	b
0	150733174	7/11/2015	15.7762	V	5	Male	NaN	57	
1	150734723	7/6/2015	83.2157	S	36	Male	White	57	
2	150817487	8/2/2015	74.8813	L	20	Female	NaN	71	

In [134...
```
# Display the last 5 rows by default
df.tail()
```

Out[134...

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagno
334834	150739278	5/31/2015	15.0591	V	7	Male	NaN	
334835	150733393	7/11/2015	5.6748	C	3	Female	Black	
334836	150819286	7/24/2015	15.7762	V	38	Male	NaN	
334837	150823002	8/8/2015	97.9239	M	38	Female	White	
334838	150723074	6/20/2015	49.2646	M	5	Female	White	

In [135...
```
# Overview information of dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 334839 entries, 0 to 334838
Data columns (total 12 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   caseNumber     334839 non-null  int64  
 1   treatmentDate  334839 non-null  object 
 2   statWeight     334839 non-null  float64
 3   stratum        334839 non-null  object 
 4   age            334839 non-null  int64  
 5   sex            334837 non-null  object 
 6   race           205014 non-null  object 
 7   diagnosis      334839 non-null  int64  
 8   bodyPart       334839 non-null  int64  
 9   disposition    334839 non-null  int64  
 10  location       334839 non-null  int64  
 11  product        334839 non-null  int64  
dtypes: float64(1), int64(7), object(4)
memory usage: 30.7+ MB
```

Select column, multiple column, with condition

In [136…
```
df.columns
```

Out[136…
```
Index(['caseNumber', 'treatmentDate', 'statWeight', 'stratum', 'age', 'sex',
       'race', 'diagnosis', 'bodyPart', 'disposition', 'location', 'product'],
      dtype='object')
```

In [137…
```
#select single column
df['age']
```

Out[137…
```
0          5
1         36
2         20
3         61
4         88
          ..
334834     7
334835     3
334836    38
334837    38
334838     5
Name: age, Length: 334839, dtype: int64
```

In [138…
```
df.age
```

Out[138…
```
0          5
1         36
2         20
3         61
4         88
          ..
334834     7
334835     3
334836    38
334837    38
334838     5
Name: age, Length: 334839, dtype: int64
```

In [139…
```
#select multiple column
df[['treatmentDate','statWeight','age','sex']]
```

Out[139…

	treatmentDate	statWeight	age	sex
0	7/11/2015	15.7762	5	Male
1	7/6/2015	83.2157	36	Male
2	8/2/2015	74.8813	20	Female
3	6/26/2015	15.7762	61	Male
4	7/4/2015	74.8813	88	Female
…	…	…	…	…
334834	5/31/2015	15.0591	7	Male
334835	7/11/2015	5.6748	3	Female
334836	7/24/2015	15.7762	38	Male
334837	8/8/2015	97.9239	38	Female
334838	6/20/2015	49.2646	5	Female

334839 rows × 4 columns

Viewing the unique value

In [140…
```
df.race.unique()
```

Out[140…
```
array([nan, 'White', 'Other', 'Black', 'Asian', 'American Indian'],
      dtype=object)
```

Describe

In [141…
```
df['age'].describe()
```

Out[141…
```
count    334839.000000
mean         31.385451
std          26.105098
min           0.000000
25%          10.000000
50%          23.000000
75%          51.000000
max         107.000000
Name: age, dtype: float64
```

Select row with condition

In [142…
```
#select by condition
df[df['sex'] == 'Male']
```

Out[142...

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis
0	150733174	7/11/2015	15.7762	V	5	Male	NaN	57
1	150734723	7/6/2015	83.2157	S	36	Male	White	57
3	150717776	6/26/2015	15.7762	V	61	Male	NaN	71
6	150713483	6/8/2015	15.7762	V	25	Male	Black	51
7	150704114	6/14/2015	83.2157	S	53	Male	White	57
...	...	...	...	...	...	...	...	..
334824	150607827	5/27/2015	5.6748	C	1	Male	White	71
334825	150600190	5/28/2015	80.8381	S	5	Male	NaN	56
334833	150747217	7/24/2015	83.2157	S	2	Male	NaN	62
334834	150739278	5/31/2015	15.0591	V	7	Male	NaN	59
334836	150819286	7/24/2015	15.7762	V	38	Male	NaN	71

182501 rows × 12 columns

In [143...

```
#select by multiple condition
df[(df['sex'] == 'Male') & (df['age'] > 80)]
```

Out[143...

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis
8	150736558	7/16/2015	83.2157	S	98	Male	Black	59
63	150418623	1/12/2015	15.0591	V	97	Male	Other	62
97	150700375	6/28/2015	83.2157	S	85	Male	NaN	59
131	150940801	9/14/2015	15.7762	V	96	Male	NaN	62
177	160110774	12/19/2015	85.7374	S	81	Male	White	59
...	...	...	...	...	...	...	...	..
334616	160104368	12/30/2015	74.8813	L	86	Male	Other	71
334677	151115099	11/4/2015	16.5650	V	83	Male	NaN	63
334699	150633387	5/29/2015	74.8813	L	84	Male	NaN	53
334701	150515945	4/27/2015	97.9239	M	86	Male	NaN	57
334785	150733286	7/11/2015	15.7762	V	86	Male	White	71

6379 rows × 12 columns

Select row with .iloc

In [144…
```
# select row by .iloc
df.iloc[10:15]
```

Out[144…

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	I
10	150734952	7/4/2015	15.7762	V	20	Male	Black	59	
11	150821622	7/20/2015	83.2157	S	20	Female	White	57	
12	150713631	7/4/2015	15.7762	V	11	Male	NaN	60	
13	150666343	6/27/2015	15.7762	V	26	Female	White	62	
14	150748843	7/16/2015	37.6645	L	33	Male	Asian	53	

In [145…
```
# select column by .iloc
df.iloc[:,[0,1,2,3,4]]
```

Out[145…

	caseNumber	treatmentDate	statWeight	stratum	age
0	150733174	7/11/2015	15.7762	V	5
1	150734723	7/6/2015	83.2157	S	36
2	150817487	8/2/2015	74.8813	L	20
3	150717776	6/26/2015	15.7762	V	61
4	150721694	7/4/2015	74.8813	L	88
…	…	…	…	…	…
334834	150739278	5/31/2015	15.0591	V	7
334835	150733393	7/11/2015	5.6748	C	3
334836	150819286	7/24/2015	15.7762	V	38
334837	150823002	8/8/2015	97.9239	M	38
334838	150723074	6/20/2015	49.2646	M	5

334839 rows × 5 columns

Select column and row with .loc

In [146…
```
# select column and low by .loc
df.loc[:6,'treatmentDate':'diagnosis']
```

Out[146…

	treatmentDate	statWeight	stratum	age	sex	race	diagnosis
0	7/11/2015	15.7762	V	5	Male	NaN	57
1	7/6/2015	83.2157	S	36	Male	White	57
2	8/2/2015	74.8813	L	20	Female	NaN	71
3	6/26/2015	15.7762	V	61	Male	NaN	71
4	7/4/2015	74.8813	L	88	Female	Other	62
5	7/2/2015	5.6748	C	1	Female	White	71
6	6/8/2015	15.7762	V	25	Male	Black	51

In [147…

```
# select row by condition
df.loc[df['age']>80, ['treatmentDate', 'age']]
```

Out[147…

	treatmentDate	age
4	7/4/2015	88
8	7/16/2015	98
39	5/3/2015	88
46	4/15/2015	91
63	1/12/2015	97
…	…	…
334701	4/27/2015	86
334784	7/7/2015	82
334785	7/11/2015	86
334815	10/28/2015	85
334819	1/13/2015	85

20422 rows × 2 columns

[Q6] What is the difference between .iloc and .loc?

Ans:

In [ ]:

```
# Q6 :
# .iloc ใช้สำหรับการเลือกแถวและคอลัมน์โดยใช้ตำแหน่ง index ในการเลือก
# .loc ใช้สำหรับการเลือกแถวและคอลัมน์โดยใช้ชื่อ index ในการเลือก
```

Punchaya Chancharoen

65070507236