Lab 1: Basic Python Programming

CPE232 Data Models

Owner : 66070501043 - Phoorin Chinphuad

---

[1] Variable

1.1 Number Variable

```
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

```
#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

```
#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

```
#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

```
#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)
```

```
tuple[0] = "CPE232"          #trying to change first element of the tuple but it cannot be changed so it gives error
```

1.6 Dictionary Variable

```
#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

```
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

```
#for loops
for num in range(0,10):
    print(num)
```

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

```
#for loop with list

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

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

```
Alice
Bob
Charlie
Daisy
```

```
#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
```

```
#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

```
#while loop

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

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

```
Alice
Bob
Charlie
Daisy
```

```
#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
```

```
#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

```
#define function
def function_name (arg1, arg2):
    return arg1 + arg2

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

```
3
```

```
#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
```

```
#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'>
15
```

```
#lambda function
lambda_function = lambda arg1, arg2: arg1 + arg2

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

```
3
```

[5] File Handling

5.1 Text File

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

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

```
Hello World
```

5.2 CSV File

```
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"])
```

```
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

```
import numpy as np
```

ndarray initialization

Construct using python list

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

```
array([1. , 2. , 3.5])
```

```
# 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
```

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

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

```
array([[[1, 2],
        [2, 3]],

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

or construct using some numpy classes and functions

```
np.zeros(5)
```

```
array([0., 0., 0., 0., 0.])
```

```
np.ones((3,4),dtype=float)
```

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

```
np.full((4,),999)
```

```
array([999, 999, 999, 999])
```

```
np.arange(3,10,2)
```

```
array([3, 5, 7, 9])
```

```
np.linspace(10,15,11)
```

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

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

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

```
np.random.randn(10)
```

```
array([-0.82222123,  0.7728091 ,  0.07231017,  0.58226338,  0.25735609,
        0.96683725,  1.56468755, -0.36298275, -0.69294433, -1.30699885])
```

ndarray properties

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

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

```
arr_a.ndim
```

```
2
```

```
arr_a.shape
```

```
(3, 4)
```

```
arr_a.dtype
```

```
dtype('int64')
```

```
arr_a.size
```

```
12
```

Reshaping & Modification

from this original ndarray

```
arr_a
```

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

try to convert into 3D array

```
arr_a.reshape((2,2,3))
```

```
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

```
arr_a.reshape((-1,6))
```

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

Would you like to try this?

```
arr_a.reshape((-1,5))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-127-286d5aa6424c> in <cell line: 0>()
----> 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: **To reshape the array into a shape with 5 columns, the total number of elements must be divisible by 5. However, 12 is not divisible by 5.**

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

```
np.append(arr_a,13)
```

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

```
np.append(arr_a,arr_a[0])
```

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

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

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

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

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

```
np.concatenate([arr_a,arr_a])
```

```
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]])
```

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

```
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

```
arr_a
```

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

try to access all element at the first row

```
arr_a[1]
```

```
array([5, 6, 7, 8])
```

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

```
arr_a[1][2]
```

```
7
```

```
arr_a[1,2]
```

```
7
```

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

```
arr_a[1,1:]
```

```
array([6, 7, 8])
```

```
1 arr_a[:2,1:]
```

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

sometimes you may specify some row number using list within indicing

```
1 arr_a[[1,2,1],1:]
```

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

Boolean slicing

based on this original array

```
1 arr_a
```

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

try to filter all elements which more than 5

```
1 arr_a>5
```

```
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

```
1 (arr_a>5)&(arr_a<10)
```

```
array([[False, False, False, False],
       [False,  True,  True,  True],
       [ True, False, False, False]])
```

Run the cell below and answer a question.

```
1 arr_a[(arr_a>5)&(arr_a<10)]
```

```
array([6, 7, 8, 9])
```

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

Ans: **The output consists of only the elements of arr_a that correspond to True in the combined boolean mask `(arr_a > 5) & (arr_a < 10)`.**

Try running the cell below.

```
1 arr_a[(arr_a>5) and (arr_a<10)]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-149-78eb1746bbfd> in <cell line: 0>()
----> 1 arr_a[(arr_a>5) and (arr_a<10)]

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: **The `and` operator in Python is a logical operator that works with single boolean values. While `arr_a > 5 and arr_a < 10` are both boolean arrays , not single boolean values.**

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

Ans: **To perform element-wise logical AND on NumPy arrays, try using the `&` operator instead of `and` >> `arr_a[(arr_a>5) & (arr_a<10)]`**

Basic operations

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

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

This is some operations for only 1 array

```
np.sqrt(arr_b)
```

```
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

```
arr_a-arr_b
```

```
array([[0, 0, 0, 0],
       [4, 4, 4, 4],
       [8, 8, 8, 8]])
```

```
np.add(arr_a,arr_b)
```

```
array([[ 2,  4,  6,  8],
       [ 6,  8, 10, 12],
       [10, 12, 14, 16]])
```

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

```
arr_a*3
```

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

```
1+arr_a**2
```

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

Try to play with 2 arrays with different shape

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

```
[1 2 3]
[[3]
 [5]
 [8]]
```

```
arr_c-arr_d
```

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

Basic aggregations

```
arr_a
```

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

```
arr_a.sum()
```

```
78
```

```
arr_a.mean()
```

```
6.5
```

```
arr_a.min()
```

```
1
```

```
arr_a.max()
```

```
12
```

```
arr_a.std() #standard division
```

```
3.452052529534663
```

ndarray axis

```
arr_a
```

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

```
arr_a.sum(axis=0)
```

```
array([15, 18, 21, 24])
```

```
arr_a.sum(axis=1)
```

```
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: **axis=0 : is performed column-wise. and axis=1 : is performed row-wise.**

4.2 Pandas

Series

```
import pandas as pd
import numpy as np
```

```
pd.Series(np.random.randn(6))
```

	0
0	0.637273
1	1.327869
2	-0.620961
3	-0.203078
4	0.378510
5	0.036953

dtype: float64

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

	0
a	-0.087000
b	-1.044680
c	-0.597356
d	0.285617
e	0.990374
f	0.968748

dtype: float64

Constructing Dataframe

Constructing DataFrame from a dictionary

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

```
df = pd.DataFrame(data=d)
df
```

	col1	col2
0	1	3
1	2	4

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

```
df2 = pd.DataFrame(data=d2)
df2
```

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

Constructing DataFrame from a List

```
marks_list = [85.10, 77.80, 91.54, 88.78, 60.55]
```

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

	Marks
0	85.10
1	77.80
2	91.54
3	88.78
4	60.55

Creating DataFrame from file

```
# Read csv file from path and store to df for create dataframe
df = pd.read_csv('nss15.csv')
```

```
df
```

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
0	150733174	7/11/2015	15.7762	V	5.0	Male	NaN	57.0	33.0	1.0	9.0	1267.0
1	150734723	7/6/2015	83.2157	S	36.0	Male	White	57.0	34.0	1.0	1.0	1439.0
2	150817487	8/2/2015	74.8813	L	20.0	Female	NaN	71.0	94.0	1.0	0.0	3274.0
3	150717776	6/26/2015	15.7762	V	61.0	Male	NaN	71.0	35.0	1.0	0.0	611.0
4	150721694	7/4/2015	74.8813	L	88.0	Female	Other	62.0	75.0	1.0	0.0	1893.0
...	...	...	...	...	...	...	...	...	...	...	...	...
234085	150968928	9/22/2015	15.7762	V	23.0	Male	Black	64.0	92.0	1.0	1.0	1141.0
234086	150965850	9/24/2015	83.2157	S	37.0	Female	NaN	64.0	31.0	1.0	1.0	4014.0
234087	150971407	9/26/2015	5.6748	C	13.0	Male	White	71.0	79.0	1.0	9.0	1211.0
234088	151026924	10/6/2015	5.6748	C	1.0	Male	White	53.0	75.0	1.0	1.0	4057.0
234089	15100638	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

234090 rows × 12 columns

Viewing DataFrame information

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

Check simple information

```
1 # Check dimension by .shape
2 df.shape
```

```
(234090, 12)
```

```
1 # Display the first 5 rows by default
2 df.head()
```

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
0	150733174	7/11/2015	15.7762	V	5.0	Male	NaN	57.0	33.0	1.0	9.0	1267.0
1	150734723	7/6/2015	83.2157	S	36.0	Male	White	57.0	34.0	1.0	1.0	1439.0
2	150817487	8/2/2015	74.8813	L	20.0	Female	NaN	71.0	94.0	1.0	0.0	3274.0
3	150717776	6/26/2015	15.7762	V	61.0	Male	NaN	71.0	35.0	1.0	0.0	611.0
4	150721694	7/4/2015	74.8813	L	88.0	Female	Other	62.0	75.0	1.0	0.0	1893.0

```
1 # Display the first 3 rows
2 df.head(3)
```

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
0	150733174	7/11/2015	15.7762	V	5.0	Male	NaN	57.0	33.0	1.0	9.0	1267.0
1	150734723	7/6/2015	83.2157	S	36.0	Male	White	57.0	34.0	1.0	1.0	1439.0
2	150817487	8/2/2015	74.8813	L	20.0	Female	NaN	71.0	94.0	1.0	0.0	3274.0

```
1 # Display the last 5 rows by default
2 df.tail()
```

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
234085	150968928	9/22/2015	15.7762	V	23.0	Male	Black	64.0	92.0	1.0	1.0	1141.0
234086	150965850	9/24/2015	83.2157	S	37.0	Female	NaN	64.0	31.0	1.0	1.0	4014.0
234087	150971407	9/26/2015	5.6748	C	13.0	Male	White	71.0	79.0	1.0	9.0	1211.0
234088	151026924	10/6/2015	5.6748	C	1.0	Male	White	53.0	75.0	1.0	1.0	4057.0
234089	15100638	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
1 # Overview information of dataframe
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234090 entries, 0 to 234089
Data columns (total 12 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   caseNumber     234090 non-null  int64  
 1   treatmentDate  234089 non-null  object 
 2   statWeight     234089 non-null  float64
 3   stratum        234089 non-null  object 
 4   age            234089 non-null  float64
 5   sex            234087 non-null  object 
 6   race           143318 non-null  object 
 7   diagnosis      234089 non-null  float64
 8   bodyPart       234089 non-null  float64
 9   disposition    234089 non-null  float64
 10  location       234089 non-null  float64
 11  product        234089 non-null  float64
dtypes: float64(7), int64(1), object(4)
memory usage: 21.4+ MB
```

Select column, multiple column, with condition

```
df.columns
```

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

```
#select single column
df['age']
```

	age
0	5.0
1	36.0
2	20.0
3	61.0
4	88.0
...	...
234085	23.0
234086	37.0
234087	13.0
234088	1.0
234089	NaN

234090 rows × 1 columns

dtype: float64

```
df.age
```

	age
0	5.0
1	36.0
2	20.0
3	61.0
4	88.0
...	...
234085	23.0
234086	37.0
234087	13.0
234088	1.0
234089	NaN

234090 rows × 1 columns

dtype: float64

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

	treatmentDate	statWeight	age	sex
0	7/11/2015	15.7762	5.0	Male
1	7/6/2015	83.2157	36.0	Male
2	8/2/2015	74.8813	20.0	Female
3	6/26/2015	15.7762	61.0	Male
4	7/4/2015	74.8813	88.0	Female
...	...	...	...	...
234085	9/22/2015	15.7762	23.0	Male
234086	9/24/2015	83.2157	37.0	Female
234087	9/26/2015	5.6748	13.0	Male
234088	10/6/2015	5.6748	1.0	Male
234089	NaN	NaN	NaN	NaN

234090 rows × 4 columns

Viewing the unique value

```
df.race.unique()
```

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

Describe

```
df['age'].describe()
```

	age
count	234089.000000
mean	31.323274
std	26.077750
min	0.000000
25%	10.000000
50%	23.000000
75%	51.000000
max	106.000000

dtype: float64

Select row with condition

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

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
0	150733174	7/11/2015	15.7762	V	5.0	Male	NaN	57.0	33.0	1.0	9.0	1267.0
1	150734723	7/6/2015	83.2157	S	36.0	Male	White	57.0	34.0	1.0	1.0	1439.0
3	150717776	6/26/2015	15.7762	V	61.0	Male	NaN	71.0	35.0	1.0	0.0	611.0
6	150713483	6/8/2015	15.7762	V	25.0	Male	Black	51.0	33.0	4.0	9.0	1138.0
7	150704114	6/14/2015	83.2157	S	53.0	Male	White	57.0	30.0	1.0	0.0	5040.0
...	...	...	...	...	...	...	...	...	...	...	...	...
234081	150953450	9/19/2015	5.6748	C	7.0	Male	White	57.0	80.0	4.0	0.0	3286.0
234084	150906288	8/27/2015	5.6748	C	14.0	Male	White	57.0	33.0	1.0	9.0	1329.0
234085	150968928	9/22/2015	15.7762	V	23.0	Male	Black	64.0	92.0	1.0	1.0	1141.0
234087	150971407	9/26/2015	5.6748	C	13.0	Male	White	71.0	79.0	1.0	9.0	1211.0
234088	151026924	10/6/2015	5.6748	C	1.0	Male	White	53.0	75.0	1.0	1.0	4057.0

127729 rows × 12 columns

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

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
8	150736558	7/16/2015	83.2157	S	98.0	Male	Black	59.0	76.0	1.0	1.0	1807.0
63	150418623	1/12/2015	15.0591	V	97.0	Male	Other	62.0	75.0	4.0	1.0	4076.0
97	150700375	6/28/2015	83.2157	S	85.0	Male	NaN	59.0	92.0	1.0	0.0	478.0
131	150940801	9/14/2015	15.7762	V	96.0	Male	NaN	62.0	75.0	1.0	5.0	1807.0
177	160110774	12/19/2015	85.7374	S	81.0	Male	White	59.0	82.0	1.0	1.0	3278.0
...	...	...	...	...	...	...	...	...	...	...	...	...
233609	150117332	1/2/2015	78.5926	S	82.0	Male	NaN	59.0	76.0	4.0	9.0	3223.0
233867	150822235	7/25/2015	15.7762	V	93.0	Male	NaN	57.0	79.0	4.0	5.0	1679.0
233932	151029869	6/17/2015	74.8813	L	85.0	Male	NaN	71.0	36.0	4.0	1.0	4076.0
233985	151015969	10/6/2015	83.2157	S	82.0	Male	NaN	62.0	75.0	1.0	5.0	1807.0
234052	150827615	7/30/2015	15.7762	V	85.0	Male	NaN	59.0	33.0	1.0	1.0	1884.0

4407 rows × 12 columns

Select row with .iloc

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

	caseNumber	treatmentDate	statWeight	stratum	age	sex	race	diagnosis	bodyPart	disposition	location	product
10	150734952	7/4/2015	15.7762	V	20.0	Male	Black	59.0	82.0	1.0	1.0	1894.0
11	150821622	7/20/2015	83.2157	S	20.0	Female	White	57.0	36.0	1.0	9.0	1267.0
12	150713631	7/4/2015	15.7762	V	11.0	Male	NaN	60.0	88.0	1.0	0.0	3274.0
13	150666343	6/27/2015	15.7762	V	26.0	Female	White	62.0	75.0	1.0	1.0	1807.0
14	150748843	7/16/2015	37.6645	L	33.0	Male	Asian	53.0	93.0	1.0	1.0	4057.0

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

	caseNumber	treatmentDate	statWeight	stratum	age
0	150733174	7/11/2015	15.7762	V	5.0
1	150734723	7/6/2015	83.2157	S	36.0
2	150817487	8/2/2015	74.8813	L	20.0
3	150717776	6/26/2015	15.7762	V	61.0
4	150721694	7/4/2015	74.8813	L	88.0

Select column and row with .loc

234085 150968928 9/22/2015 15.7762 V 23.0

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

	treatmentDate	statWeight	stratum	age	sex	race	diagnosis
0	7/11/2015	15.7762	V	5.0	Male	NaN	57.0
1	7/6/2015	83.2157	S	36.0	Male	White	57.0
2	8/2/2015	74.8813	L	20.0	Female	NaN	71.0
3	6/26/2015	15.7762	V	61.0	Male	NaN	71.0
4	7/4/2015	74.8813	L	88.0	Female	Other	62.0
5	7/2/2015	5.6748	C	1.0	Female	White	71.0
6	6/8/2015	15.7762	V	25.0	Male	Black	51.0

```
# select row by condition

df.loc[df['age']>80, ['treatmentDate', 'age']]
```

	treatmentDate	age
4	7/4/2015	88.0
8	7/16/2015	98.0
39	5/3/2015	88.0
46	4/15/2015	91.0
63	1/12/2015	97.0
...	...	...
233985	10/6/2015	82.0
234047	8/4/2015	83.0
234049	6/16/2015	85.0
234052	7/30/2015	85.0
234069	8/29/2015	83.0

14171 rows × 2 columns

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

Ans: **.iloc is Integer-based indexing (position-based). While .loc is Label-based indexing (name-based).**