

3/5/2021

ASSIGNMENT-1

Basic data structures in Python,
NumPy and PANDAS

Name

Rishabh Sharma

Registration Number

20MAI0082

GitHub Link :- [Data-Mining/Assignment-1 at main · rishabh5197/Data-Mining \(github.com\)](https://github.com/rishabh5197/Data-Mining/Assignment-1)

Tuples

- Tuples is a type of data-structure which are immutable (which means that they are unchangeable).
- it supports only creation and reading of tuple and its element
- It is created in Round brackets()

Defining a tuple

```
In [1]: a=("ABC",2,3,"DEF",99)
        print(a)
```

```
('ABC', 2, 3, 'DEF', 99)
```

Printing the type of tuple

```
In [2]: print("Printing type of a :- \n",type(a))
```

```
Printing type of a :-
<class 'tuple'>
```

what is the length of the tuple

```
In [3]: print("Printing length of a :- \n",len(a))
```

```
Printing length of a :-
5
```

printing the element present at index-1 of tuple

```
In [4]: print("Printing index at 1 of a :- \n",a[1])
```

```
Printing index at 1 of a :-
2
```

printing the element present at last index of tuple

```
In [5]: print("Printing index at -1 of a :- \n",a[-1])
```

```
Printing index at -1 of a :-
99
```

Sliced Tuple from range 2 to 3 , though 4 is given it consider[a:n-1] where a is 2 and n is 4.

```
In [6]: print("Printing Slicing from 2 to 4 of a :- \n",a[2:4])
```

Printing Slicing from 2 to 4 of a :-
(3, 'DEF')

Negative Slicing

```
In [7]: print("Printing Slicing from -4 to -2 of a :- \n",a[-4:-2])
```

Printing Slicing from -4 to -2 of a :-
(2, 3)

storing a tuple value in a variable

```
In [8]: b=a[3]  
print("Storing value into a variable and then printing it \n",b)
```

Storing value into a variable and then printing it
DEF

printing Max and Min from tuple

```
In [9]: c=(99,25,18,28,35,650,1000,456,7,89,35)  
print("Max of new tuple c\n",max(c))  
print("Max of new tuple c\n",min(c))
```

Max of new tuple c
1000
Max of new tuple c
7

Printing all elements using For Loop

```
In [10]: a=("ABC",2,3,"DEF",99)  
for i in a:  
    print(i)
```

ABC
2
3
DEF
99

Printing all elements using For Loop with its index

```
In [11]: for i in range(len(a)):
         print(i,a[i])
```

```
0 ABC
1 2
2 3
3 DEF
4 99
```

Printing all elements using For Loop with its index

```
In [12]: for i,x in enumerate(a):
         print(i,x)
```

```
0 ABC
1 2
2 3
3 DEF
4 99
```

Sets

- A set is a collection which is both unordered and unindexed
- There is no duplication in the sets whereas the tuple and list may have
- it is passed in {} brackets

```
In [13]: a={1,2,3,4}
         print(type(a)) # printing Type of variable stored
         print(len(a)) # printing Length of set
```

```
<class 'set'>
4
```

```
In [14]: b=set()
         print(type(b))
```

```
<class 'set'>
```

Changing From Tuple Data-type to Set Data-type

```
In [15]: c=set((1,2,3))
         print(c)
```

```
{1, 2, 3}
```

Changing From List Data-type to Set Data-type

```
In [16]: d=set([1,2,3])  
print(d)
```

```
{1, 2, 3}
```

List

- it has the capacity to store multiple items in a single variable.
- It is Mutable which means changeable.
- It is created in Square[] brackets

```
In [17]: a=[1,2,3,"ABC"]           # Creating a list  
print(type(a))                   # Printing Type of a which is having list elements  
print(a)                         # printing whole list  
print(len(a))                   # printing the number of elements present in the list
```

```
<class 'list'>  
[1, 2, 3, 'ABC']  
4
```

Indexing in List

```
In [18]: print(a[0])  
print(a[-1])  
print(a[0:2])  
print(a[-3:-1])
```

```
1  
ABC  
[1, 2]  
[2, 3]
```

Printing elements of the list

```
In [19]: for i in range(len(a)):  
print(i,".",a[i])
```

```
0 . 1  
1 . 2  
2 . 3  
3 . ABC
```

Replacing the element at the index-2

```
In [20]: a[2]=5  
print(a)
```

```
[1, 2, 5, 'ABC']
```

Replacing the element at the index-1

In [21]:

```
a[1]=10
print(a)
```

```
[1, 10, 5, 'ABC']
```

In [22]:

```
a=[1,2,3,"ABC","DEF"]
print(a)                # Printing the list
print(len(a))           # Printing the number of elements of the list
```

```
[1, 2, 3, 'ABC', 'DEF']
5
```

Appending to the list

In [23]:

```
a.append(10)
print(a)                # Printing the list
print(len(a))           # Printing the number of elements of the list
```

```
[1, 2, 3, 'ABC', 'DEF', 10]
6
```

In [24]:

```
a.append(["ABC",5])      # for appending multiple value we must mention it in a list and it will co
print(a)                # Printing the list
print(len(a))           # Printing the number of elements of the list
```

```
[1, 2, 3, 'ABC', 'DEF', 10, ['ABC', 5]]
7
```

In [25]:

```
a.extend("ABC")          # will treat each alphabet individually
a.extend(["XYZ","PQR"])  # Extending the elements into the list.
print(a)                # Printing the list
print(len(a))           # Printing the number of elements of the list
```

```
[1, 2, 3, 'ABC', 'DEF', 10, ['ABC', 5], 'A', 'B', 'C', 'XYZ', 'PQR']
12
```

Inserting elements at a given position

In [26]:

```
a.insert(2,"AAA")        # Inserting the elements at the position 2 and element "AAA"
print(a)                # Printing the list
print(len(a))           # Printing the number of elements of the list
```

```
[1, 2, 'AAA', 3, 'ABC', 'DEF', 10, ['ABC', 5], 'A', 'B', 'C', 'XYZ', 'PQR']
13
```

Popping Elements

In [27]:

```
a=["ABC",1,2,"DEF","abc"]  
b=a.pop()  
print("after popping with no parameters",a)
```

*# if no index is passed it will delete the last
printing the elements after popping*

after popping with no parameters ['ABC', 1, 2, 'DEF']

In [28]:

```
c=a.pop(0)  
print("after popping with parameter",a)
```

*# On 0 index only
printing the elements after popping*

after popping with parameter [1, 2, 'DEF']

Removing Elements

In [29]:

```
d=a.remove(2)  
print("After removing by passing index",a)
```

*# On elements
printing the elements after removing*

After removing by passing index [1, 'DEF']

Using Del

In [30]:

```
del a[1]  
print("After deleting",a)
```

*# it is performed on index only
printing the elements after deleting*

After deleting [1]

Printing Length of new list

In [31]:

```
a=[1,2,3,0,-1,-5,0,2,57,7,0,8,7,25]  
print(len(a))
```

14

Printing Min and Max in list

In [32]:

```
print(min(a))  
print(max(a))
```

-5

57

Reversed list using[::-1]

In [33]:

```
reverse=a[::-1]  
print(reverse)
```

[25, 7, 8, 0, 7, 57, 2, 0, -5, -1, 0, 3, 2, 1]

Sorted

In [34]:

```
sort=sorted(a)
print(sort)
```

```
[-5, -1, 0, 0, 0, 1, 2, 2, 3, 7, 7, 8, 25, 57]
```

Count Function

In [35]:

```
x=a.count(0)           # Counting number of times 0 present in the list
print(x)               # printing list
```

```
3
```

Dictionary

- It is mutable.
- it is used to store data values in key:value pairs.
- it is declared by {} brackets.

In [36]:

```
a={1:'ABC', 'A':"DEF", (1,2,3):[1,2,3], 2:"XYZ"}    # Declaring Dictionary
print(a)
print(type(a))
print(len(a))
```

```
{1: 'ABC', 'A': 'DEF', (1, 2, 3): [1, 2, 3], 2: 'XYZ'}
<class 'dict'>
4
```

In [37]:

```
a={1:"a", 2:"b", 3:"c", 4:"d", 5:"e", 6:"f"}
print(a)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'}
```

Update by assigning method

In [38]:

```
a[7]="g"
print(a)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f', 7: 'g'}
```

update by .update method

In [39]:

```
a.update({8:"h", 9:"i", 10:"j"})
print(a)
```



```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f', 7: 'g', 8: 'h', 9: 'i', 10: 'j'}
```

In [40]:

```
a={1:"ABC",2:"DEF",3:"GHI",4:"JKL"}  
print(a[1])  
print(a[4])
```

```
ABC  
JKL
```

list of dictionary

In [41]:

```
a=[{1:"ABC",2:"DEF",3:"GHI"},{4:"JKL",5:"MNO",6:"PQR"}]  
print(len(a))  
print(a[1][4])
```

```
2  
JKL
```

Deleting by key

In [42]:

```
a={1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'}  
del a[2]  
print(a)
```

```
{1: 'a', 3: 'c', 4: 'd', 5: 'e', 6: 'f'}
```

By pop method

In [43]:

```
a.pop(3)  # it is necessary to write key value  
print(a)
```

```
{1: 'a', 4: 'd', 5: 'e', 6: 'f'}
```

by popitem method

In [44]:

```
# popitem deletes the last item  
# And no argument is required in it..  
a.popitem()  
print(a)
```

```
{1: 'a', 4: 'd', 5: 'e'}
```

Printing Keys and Values

In [45]:

```
keys=list(a.keys())  
values=list(a.values())  
print(keys)  
print(values)
```

```
[1, 4, 5]  
['a', 'd', 'e']
```

Numpy

- importing numpy using alias name as np and printing its version

In [46]:

```
import numpy as np  
print(np.__version__)
```

1.19.2

In [47]:

```
import numpy as np  
a=np.array([1,2,3]) # Creating Numpy 1-d array  
print(a[0]) # Accessing element present at index-0  
print(a[1:]) # Accessing element present after index-1
```

1

[2 3]

Dot product of 2 numpy array

In [48]:

```
import numpy as np  
a=np.array([1,2,3])  
b=np.array([1,2,3])  
c=np.dot(a,b)  
print(c)
```

14

Addition of 2 arrays

In [49]:

```
import numpy as np  
# adding 2 arrays  
l1=np.array([1,2,3,4,5])  
l2=np.array([1,2,3,4,5])  
lout=l1+l2  
print(lout)
```

[2 4 6 8 10]

Division of 2 array

In [50]:

```
# division of numpy array  
import numpy as np  
a=np.array([1,2,3,4,5,6])  
a=a/10 # Used only for array  
print(a)
```

[0.1 0.2 0.3 0.4 0.5 0.6]

Multiplication of 2 array

In [51]:

```
# Multiplication in array.
import numpy as np
a=np.array([1,2,3,4,5,6])
a=a*5 # Used only for array
print(a)
```

[5 10 15 20 25 30]

In [52]:

```
# numpy with strings
a=np.array(["hello","world"])
print(a)
print(type(a))
print(a.size)
print(a.ndim)
print(a.shape)
print(a.itemsize)

# Storing String values in array

# printing its type
# printing its size
# printing its dimensions
# printing its shape
# printing size of single element in the array
```

```
['hello' 'world']
<class 'numpy.ndarray'>
2
1
(2,)
20
```

In [53]:

```
# numpy with float
import numpy as np
a=np.array([1.1,2.5,36.5,100.1])
print(a)
print(type(a))
print(a.size)
print(a.ndim)
print(a.shape)
print(a.itemsize)

# printing its type
# printing its size
# printing its dimensions
# printing its shape
# printing size of single element in the array
```

```
[ 1.1  2.5 36.5 100.1]
<class 'numpy.ndarray'>
4
1
(4,)
8
```

In [54]:

```
# numpy with int
import numpy as np

a=np.array([1,2,3,4,5])
print(a)
print(len(a)) # For length
print(type(a)) # For Type

print(a.size) # For size
print(a.ndim) #For dimensions
```

```
print(a.dtype) # For displaying data Types
print(a.shape) # For Displaying Shape in the form of Rows and columns(Rows,Col)
print(a.itemsize) # for displaying the size of single element in the array
```

```
[1 2 3 4 5]
5
<class 'numpy.ndarray'>
5
1
int32
(5,)
4
```

creating zeros array

```
In [55]: np.zeros((5,5))
```

```
Out[55]: array([[0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.]])
```

creating ones array

```
In [56]: np.zeros((3,3))
```

```
Out[56]: array([[0., 0., 0.],
               [0., 0., 0.],
               [0., 0., 0.]])
```

creating 2d array

```
In [57]: import numpy as np
a = np.array([[1,2,3],[4,5,6]])
print(a)
```

```
[[1 2 3]
 [4 5 6]]
```

```
In [58]: print(a.shape)      # printing Shape
```

```
(2, 3)
```

```
In [59]: print(a.ndim)      # Printing its dimension
```

```
2
```

creating 3-d array

In [60]:

```
import numpy as np
a = np.array([[[1, 2, 3], [4, 5, 6]], [[7,8,9], [10,11,12]]])
print(a)
```

```
[[[ 1  2  3]
  [ 4  5  6]]
```

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

In [61]:

```
print(a.shape)
```

```
(2, 2, 3)
```

In [62]:

```
print(a.ndim)
```

```
3
```

In [63]:

```
import numpy as np
a = np.array([28,99,1256,156,564,45,8723,12,564,86,123,21])
print(a)
```

```
[ 28  99 1256 156 564  45 8723  12 564  86 123  21]
```

Reshaping Array

In [64]:

```
a= a.reshape(4, 3)
print(a)
```

```
[[ 28  99 1256]
 [ 156 564  45]
 [8723  12 564]
 [ 86 123  21]]
```

In [65]:

```
import numpy as np
a = np.array([28,99,1256,156,564,45,8723,12,564,86,123,21])
print(a)
```

```
[ 28  99 1256 156 564  45 8723  12 564  86 123  21]
```

In [66]:

```
a = a.reshape(2, 3, 2)
print(a)
```

```
[[[ 28  99]
  [1256 156]
  [ 564  45]]
```

```
 [[8723  12]
  [ 564  86]
  [ 123  21]]]
```

```
In [67]: import numpy as np  
a = np.array([[1, 2, 3], [4, 5, 6]])  
print(a)
```

```
[[1 2 3]  
 [4 5 6]]
```

```
In [68]: a = a.reshape(-1)  
print(a)
```

```
[1 2 3 4 5 6]
```

concatenating 2 arrays

```
In [69]: import numpy as np  
a1 = np.array([1, 2, 3])  
a2 = np.array([4, 5, 6])  
a3 = np.concatenate((a1, a2))  
print(a3)
```

```
[1 2 3 4 5 6]
```

stacking 2 arrays

```
In [70]: import numpy as np  
a1 = np.array([1, 2, 3])  
a2 = np.array([4, 5, 6])  
a3 = np.stack((a1, a2),axis=1)  
print(a3)
```

```
[[1 4]  
 [2 5]  
 [3 6]]
```

Transpose

```
In [71]: print(a3.transpose())
```

```
[[1 2 3]  
 [4 5 6]]
```

Pandas

- importing pandas as pd here pd is used as alias name

```
In [72]: import numpy as np  
import pandas as pd  
a=np.array([1,2,3,4,5])
```

```
a=pd.Series(a)
print(a)
print(type(a))
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int32
<class 'pandas.core.series.Series'>
```

creating a series

In [73]:

```
a=np.array(["Hello","world","1",2,3])
a=pd.Series(a)
print(a)
print(type(a))
```

```
0    Hello
1    world
2         1
3         2
4         3
dtype: object
<class 'pandas.core.series.Series'>
```

change of index

In [74]:

```
import numpy as np
import pandas as pd
a=np.array(["Hello","world","1",2,3])
a=pd.Series(a,index=[101,102,103,104,105])
print(a)
print(type(a))
```

```
101    Hello
102    world
103         1
104         2
105         3
dtype: object
<class 'pandas.core.series.Series'>
```

In [75]:

```
import numpy as np
import pandas as pd
newlis=[]
for i in range(len(a)):
    newlis.append(i+100)
print(newlis)
print(pd.Series(["Hello","world",1,2,3],index=newlis))
```

```
[100, 101, 102, 103, 104]
```

```
100 Hello
101 world
102 1
103 2
104 3
dtype: object
```

dictionary for pandas series

```
In [76]: import pandas as pd
print(pd.Series({"A":100,"B":200}))
```

```
A 100
B 200
dtype: int64
```

using same values but multiple index

```
In [77]: print(pd.Series(10,index=[1,2,3,4]))
```

```
1 10
2 10
3 10
4 10
dtype: int64
```

Creating a dataframe

```
In [78]: import pandas as pd
lis=[1,2,3,4,5,6,7]
a=pd.DataFrame(lis)
print(a)
print(type(a))
```

```
0
0 1
1 2
2 3
3 4
4 5
5 6
6 7
<class 'pandas.core.frame.DataFrame'>
```

Giving Column name

```
In [79]: import pandas as pd
a=[["Per1",101],
    ["Per2",102],
    ["Per3",103]]
```



```
b=pd.DataFrame(a,columns=["Name","Number"],index=[1,2,3])  
print(b)
```

	Name	Number
1	Per1	101
2	Per2	102
3	Per3	103

using list to create a dataframe

In [80]:

```
import pandas as pd  
a={"Name":["N1","N2","N3"],  
   "Number":[101,102,103]}  
b=pd.DataFrame(a,index=[1,2,3])  
print(b)
```

	Name	Number
1	N1	101
2	N2	102
3	N3	103

using dictionary to create a dataframe

In [81]:

```
import pandas as pd  
a={"Name":"Per1","Age":50},  
   {"Name":"Per2","Age":10},  
   {"Name":"Per3","Age":40}]  
print(pd.DataFrame(a,index=[1,2,3]))
```

	Name	Age
1	Per1	50
2	Per2	10
3	Per3	40

using series to create a dataframe

In [82]:

```
import pandas as pd  
a={'Name':pd.Series([1,2,3,4,5]),  
   "ID":pd.Series([101,203,405,110,36])}  
print(pd.DataFrame(a))
```

	Name	ID
0	1	101
1	2	203
2	3	405
3	4	110
4	5	36

storing dataframe into file

In [83]:

```
import pandas as pd
```

```
a=[{"Name":"Per1","Age":50},
    {"Name":"Per2","Age":10},
    {"Name":"Per3","Age":40}]
df=pd.DataFrame(a,index=[1,2,3])
df.to_csv("data.csv",index=False)
df.to_json("data.json")
```

reading a file

In [84]:

```
import pandas as pd
data=pd.read_csv("data.csv")
print(pd.DataFrame(data))
```

```
   Name  Age
0  Per1   50
1  Per2   10
2  Per3   40
```

using iloc

In [85]:

```
print(data.iloc[2,1])
print(data.iloc[1,0])
```

```
40
Per2
```

using loc

In [86]:

```
a=data.loc[(data["Name"]=='Per1') | (data["Age"]==10)]
print(a)
```

```
   Name  Age
0  Per1   50
1  Per2   10
```

In [87]:

```
a=data.loc[(data["Name"]=='Per1') & (data["Age"]==10)]
print(a)
```

```
Empty DataFrame
Columns: [Name, Age]
Index: []
```

In [88]:

```
a=data.loc[(data["Name"]=='Per1') & (data["Age"]==50)]
print(a)
```

```
   Name  Age
0  Per1   50
```

Using Rename

```
In [89]: a=a.rename(columns={"Name":"Names of People","Age":"Age of people"})
print(a)
```

```
Names of People Age of people
0      Per1      50
```

Using Append

```
In [90]: import pandas as pd
a=pd.DataFrame([[101,201],[301,401]],index=[1,2],columns=[1,2])
b=pd.DataFrame([[11,21],[12,22],[13,23]],index=[2,3,4],columns=[1,2])
c=a.append(b)
print(c)
```

```
1 2
1 101 201
2 301 401
2 11 21
3 12 22
4 13 23
```

using concat

```
In [91]: d=pd.concat([a,b],axis=1)# by-default axis is 0
print(d)
```

```
1 2 1 2
1 101.0 201.0 NaN NaN
2 301.0 401.0 11.0 21.0
3 NaN NaN 12.0 22.0
4 NaN NaN 13.0 23.0
```

```
In [92]: d=pd.concat([a.reset_index(drop=True),b])
print(d)
```

```
1 2
0 101 201
1 301 401
2 11 21
3 12 22
4 13 23
```

```
In [93]: import pandas as pd
import numpy as np

df1=pd.DataFrame({"A":range(1,6),
                  "B":["Gujarat","Maharashtra","UP","Rajasthan","MP"],
                  "c":np.random.randint(0,25,size=5)}) # Here size refers to no of quantity
df1
```

```
Out[93]:
```

A	B	c
---	---	---

	A	B	c
0	1	Gujarat	4
1	2	Maharastra	21
2	3	UP	10
3	4	Rajasthan	4
4	5	MP	18

In [94]: `df2=pd.DataFrame([[2,"Medium"],[4,"Low"],[5,"High"]],columns=["A","Product"])
df2`

Out[94]:

	A	Product
0	2	Medium
1	4	Low
2	5	High

Using Merge

In [95]: `# Joins
print("This acts like a join")
df=(df1.merge(df2,on="A",how="left"))
df`

This acts like a join

Out[95]:

	A	B	c	Product
0	1	Gujarat	4	NaN
1	2	Maharastra	21	Medium
2	3	UP	10	NaN
3	4	Rajasthan	4	Low
4	5	MP	18	High

In [96]: `df=(df1.merge(df2,on="A",how="right"))
df`

Out[96]:

	A	B	c	Product
0	2	Maharastra	21	Medium
1	4	Rajasthan	4	Low
2	5	MP	18	High

In [97]:

```
df=df2.merge(df1,on="A",how="left")
df
```

Out[97]:

	A	Product	B	c
0	2	Medium	Maharastra	21
1	4	Low	Rajasthan	4
2	5	High	MP	18

Using Merge

In [98]:

```
import numpy as np
import pandas as pd
df1=pd.DataFrame({"col1":[2,4,6,8],
                  "col2":[11,13,15,17]},index=[101,102,103,104])
df2=pd.DataFrame({"col3":[22,24,26,28],
                  "col4":[31,33,35,37]},index=[101,103,104,105])
df=df1.join(df2)
df
```

Out[98]:

	col1	col2	col3	col4
101	2	11	22.0	31.0
102	4	13	NaN	NaN
103	6	15	24.0	33.0
104	8	17	26.0	35.0

In [99]:

```
df=df2.join(df1)
df
```

Out[99]:

	col3	col4	col1	col2
101	22	31	2.0	11.0
103	24	33	6.0	15.0
104	26	35	8.0	17.0
105	28	37	NaN	NaN