PRACTICE QUESTIONS

1. What is series? Explain with the help of an example.

Ans. Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.

Example:

```
import pandas as pd # simple array
data =pd.Series([1,2,3,4,5])
print(data)
```

2. Write a suitable Python code to create an empty series.

Ans.

```
import pandas as pd
s=pd.Series()
print (s)
```

3. Write single line Pandas statement to declare a Pandas series named Packets having dataset as: [125, 92, 104, 92, 85, 116, 87, 90]

Ans. Packets = pd.Series([125, 92, 104, 92, 85, 116, 87, 90])

4. Write single line Pandas statement to declare a Pandas series named S having dataset as: (44,65,35,77,87,90)

Ans. S=pd.Series((44,65,35,77,87,90))

5. Write single line Pandas statement to declare a Pandas series named SR having dataset as: {1:'one',2:'two',3:'three'}

Ans. SR=pd.Series({1:'one',2:'two',3:'three'})

6. Write python code to create the Series EMP with following data (using Dictionary)

Code	
E1	Sanya
E2	Krish
E3	Rishav
E4	Deepak

Ans.

```
import pandas as pd
d={"E1":"Sanya","E2":"Krish","E3":Rishav","E4":"Deepak"}
EMP=pd.Series(d)
EMP.index.name="code"
```

```
print(EMP)
```

7. Write python series to print scalar value "100" 5 times with index values(1,2,3,4,5)

Ans.

```
import pandas as pd

s = pd.Series(100, index=[1, 2, 3, 4,5])

OR

s = pd.Series(100, [1, 2, 3, 4,5])

print (s)
```

8. Write python code to create the following series using Dictionary:

```
101  Harsh
102  Arun
103  Ankur
104  Harpahul
105  Divya
106  Jeet
```

Ans:

```
import pandas as pd

D={101:"Harsh",102:"Arun",103:"Ankur",104:"Harpahul",105:"Divya"
,106:"Jeet" }

s=pd.Series(D)

print(s)
```

9. Write a program to create a series by using given an array ['a','b','c','d'] and assign index values 100,101.....

Ans.

```
import pandas as pd

import numpy as np

data =np.array(['a','b','c','d'])

s =pd.Series(data,index=[100,101,102,103])

print(s)
```

Its output is as follows –

100 a

101 b

102 c

103 d

dtype: object

10. Give the output:

```
import pandas as pd
s =pd.Series(10, index=[5,6,7,9])
print(s)
```

Ans.

5 10

6 10

7 10

9 10

ACCESSING DATA

Accessing using head()

By default Series.head() function display top 5 rows. To print n no of top rows, pass n as parameter i.e. Series. head(n)

17. Write a code to create a series from empno list and show the first five rows empno = [101,102,103,104,105,106,107]

Ans.

```
import pandas as pd

p=pd.Series(empno)

print (p.head())
```

output:

0 101

1 102

2 103

3 104

4 105

18. Using the above series write a single line statement to show the first rows using head()

Ans. print(p.head(3))

output:

0 101

1 102

2 103

Accessing using tail()

By default Series.tail() function display last 5 rows. To print n no of last rows, pass n as parameter i.e. Series. tail(n)

19. Write a code to create a series from empno list and show the last five rows empno=[101,102,103,104,105,106,107]

Ans.

```
import pandas as pd

p=pd.Series(empno)

print (p.tail())
```

output:

2 103

3 104

4 105

5 106

6 107

20.Fill the missing statements

```
import pandas as pd L=[101,102,103,104,105,106,107]
_ = pd.Series(L) #statement 1
print (p.___(3)) #statement 2
```

output:

4 105
5 106
6 107

Ans.

```
p=pd.Series(L) #statement 1

print (p.tail(3)) #statement 2
```

Indexing

Pandas now supports three types of indexing.

(i) loc: is label based indexing.

(a) A single label

21. Give the output:

```
import pandas as pd
s=pd.Series([10,20,30,40,50],index =[‘a’,’b’,’c’,0,1])
print(s.loc[‘a’])
print(s.loc[0])
```

Ans.

```
10
40
# in loc[0], 0 is interpreted as a label of the index. This is not an integer
  position along the index.
```

(b) A list of labels

22. Give the output:

```
import pandas as pd
s=pd.Series([10,20,30,40,50],index =[‘a’,’b’,’c’,0,1])
print(s.loc[[‘b’,’c’,1]])
```

Ans.

```
b 20
c 30
1 50
```

(c) A slice object with labels

A slice is a subset of series elements.

my_series[start:stop:step] where start is the index of the first element to include, stop is the index of the item to stop and step sets the interval

23. Give the output:

```
import pandas as pd
s=pd.Series([10,20,30,40,50],index =[`a',`b',`c',0,1])
print(s.loc[`a':`c'])
```

Ans.

```
a 10
b 20
c 30
```

Note: Both the start and the stop are included, when present in the label index

(ii) iloc: integer position based.

This series can also be indexed by position (using integers) even though it has string index entries! The first item is at key 0, and the last item is at key -1

(a) input an integer

24. Give the output:

```
import pandas as pd
s=pd.Series([10,20,30,40,50],index =[`a',`b',`c',0,1])
print(s.iloc[0])
```

Ans. 10

(b) input a list of integers

25. Write a program to print the values of 0,2,4 positions from Series s[10,20,30,40,50] using .iloc.

Ans.

```
import pandas as pd
s=pd.Series([10,20,30,40,50],index =[`a',`b',`c',0,1])
print(s.iloc[[0,2,4]])
```

output
a 10
c 30
1 50

26. Write a program to print the values of 0,2,4 positions from Series s[10,20,30,40,50] using .iloc.

Ans.

```
import pandas as pd

s=pd.Series([10,20,30,40,50],index =[`a',`b',`c',0,1])

print(s.iloc[[0,2,4]])
```

output

```
a 10

c 30

1 50
```

(c) input a slice object with ints

A slice is a subset of series elements. my_series[start:stop:step] where start is the index of the first element to include, stop is the index of the item to stop without including the stop value and step sets the interval.

27. Show the first 3 values from Series using iloc.

Ans.

```
import pandas as pd

s=pd.Series([10,20,30,40,50],index =[`a',`b',`c',0,1])

print(s.iloc[0:3])
```

```
a 10

b 20

c 30
```

(iii) with [ ]: Accessing Data from Series with Position and Using Label (index). By using [] you can take advantage of both .loc and .iloc. You can access the records using [] directly.

If index value is not defined you can pass default index value 0,1,2... to access the elements.

If string index values are defined then elements can be accessed by both ways: passing default index values (0,1,2...) or passing defined string index values.

If integer index values are defined, [] will work only as .loc i.e. elements can be accessed by passing defined index values only.

28. Give the output:

```
import pandas as pd
M=[15,-10,56,39,-90,15]
p=pd.Series(M)
print(p[0])
print(p[[0,3,4]])
```

Ans.

```
15
0 15
3 39
4 -90
```

29. Give the output of the following program:

```
import pandas as pd
s=pd.Series([10,20,30,40,50],index=[‘a’,’b’,’c’,’d’,’e’])
print(s[0]) #or    print(s.iloc[0])
print(s[‘a’:’c’]) #or print(s.loc[‘a’:’c’])
print(s[2]) #or print(s.iloc[s[2])
print(s[‘a’]) #or print(s.loc[‘a’])
```

Its output is as follows –

Ans.

```
10
a 10
b 20
c 30
30
10
```

30. **PRACTICE QUESTIONS**

(i) Write python code to create the following series

101 Harsh

102 Arun

103 Ankur

104 Harpahul

105 Divya

106 Jeet

(ii) Show details of 1st 3 employees using head function

(iii) Show details of last 3 employees using tail function

(iv) Show details of 1st 3 employees without using head function

(v) Show details of last 3 employee without using tail function

(vi) Show value of index no 102

(vii) Show 2nd to 4th records

(viii) Show values of index no=101,103,105

(ix) Show details of "Arun"

Ans.

(i) import pandas as pd
name=[`Harsh',`Arun',`Ankur',`Harpahul',`Divya',`Jeet']
p=pd.Series(name,index=[101,102,103,104,105,106])
print (p)

(ii) print (p.head(3))

(iii) print (p.tail(3))

(iv) print(p[:3]) or pirnt(p.loc[101:103]) or print(p.iloc[0:3]) or print(p[[101,102,103]])

(v) print (p[-3:]) or print(p[3:]) or print(p[[104,105,106]])

(vi) print(p[102]) or print(p.loc[102])

(vii) print(p[1:4])

(viii) print(p[[101,103,105]])

(ix) print(p[p= =`Arun'])

****