Ques 1: Create a simple Pandas Series from a list.

Ans1:

```
>>> import pandas
>>> from pandas import Series as s
>>> s1=s([1,2,3,4])
>>> s1
```

Output:

```
0  1
1  2
2  3
3  4
dtype: int64
```

Ques 2: Return the first and last values of the Series created above.

Ans2:

```
>>> s1[0:4:3]
```

Output:

```
0  1
3  4
dtype: int64
```

Ques3: Create a simple Pandas Series with your own labels i.e. index

Ans3:

```
>>> s2=s([10,20,30,40,50],index=['a','b','c','d','e'])
>>> s2
```

Output:

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

d 40

e 50

dtype: int64

Ques4: Access the values using your own index and print the value, also try –ve index.

Ans4:

```
>>> s2['d']
```

40

```
>>> s2[-3]
```

30

Ques5: Create a simple Pandas Series from a dictionary.

Ans5:

```
>>> s3=s({1:'a',2:'b',3:'c'})
>>> s3
```

Output:

1 a

2 b

3 c

dtype: object

Ques6: Create a Series using only calories intake data from user defined indexes "day1","day2", and "day3".

Ans6:

```
>>> s4=s([1500,2000,1300],index=['day1','day2','day3'])
>>> s4
```

Output:

day1 1500

day2 2000

day3 1300

dtype: int64

Ques7: Create a Series of heterogeneous data types and check the data type of the Series as well as individual items.

Ans7:

```
>>> s5=s([1,1.2,'a',7,'s',5.5])
>>> s5
```

Output:

```
0   1
1  1.2
2   a
3   7
4   s
5  5.5
dtype: object
>>> type(s5)
```

Output:

```
<class 'pandas.core.series.Series'>
>>> type(s5[4])
```

Output:

```
<class 'str'>
```

Ques8: Compute min, max, mean values of a Series.

Ans8:

```
>>> max(s2)
50
>>> min(s2)
10
```

```
>>> s.mean(s2)
30.0
```

Ques9: Compute the relative change percentage in values of a Series.

Ans9:

```
>>> df=s([1,2,3,4,5])
>>> df=(1+df.pct_change())
>>> df
0        NaN
1    2.000000
2    1.500000
3    1.333333
4    1.250000
dtype: float64
```

Ques10: Add items in a Series from another Series.

Ans10:

CASE 1:

```
>>> s1=s([1,2,3,4,5])
>>> s2=s([2,3,4,5,6])
>>> s1.add(s2)
```

Output:

```
0    3
1    5
2    7
3    9
4   11
dtype: int64
```

CASE 2:

```
>>> s1=s([1,2,3,4,5,6])
>>> s2=s([2,3,4,5])
>>> s1.add(s2)
```

Output:

```
0    3.0
1    5.0
2    7.0
3    9.0
4    NaN
5    NaN
dtype: float64
```

CASE3:

```
>>> s1=s([1,2,3,4,5,6])
>>> s2=s([2,3,4,5])
>>> s1.add(s2,None,0)
```

Output:

```
0    3.0
1    5.0
2    7.0
3    9.0
4    5.0
5    6.0
dtype: float64
```

Ques11: Create a Series of 20 items and print first 5 and last 5 elements of this Series. (use head and tail)

Ans11:

```
>>> S=s(['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t'])
>>> S.head(5)
```

Output:

```
0  a
1  b
2  c
3  d
4  e
dtype: object
>>> S.tail(5)
```

Output:

```
15  p
16  q
17  r
18  s
19  t
dtype: object
```

Ques12: Assign new index to existing series.

Ans12:

```
>>> s1=s([1,2,3,4,5])
>>> s1
0  1
1  2
2  3
```

```
>>> s1.index=['a','b','c','d','e']
>>> s1
a  1
b  2
c  3
```

3 4

4 5

dtype: int64

d 4

e 5

dtype: int64

Ques13: Reset the index of an existing Series and delete the existing index.

Ans13:

```
>>> s1=s([10,20,30,40,50],index=['a','b','c','d','e'])
>>> s2=s1.reset_index()
>>> s2
```

Output:

```
 index  0
0    a  10
1    b  20
2    c  30
3    d  40
4    e  50
>>> s2.drop('index',axis=1,inplace=True)
>>> s2
   0
0  10
1  20
2  30
3  40
4  50
```

Ques14: Sort the values of a Series in ascending and descending order and print.

Ans14:

```
>>> S=s([2,7,3,9,4,5])
>>> S.sort_values(ascending=False)
```

3 9

1 7

5 5

4 4

2 3

0 2

dtype: int64

>>> S.sort_values(ascending=True)

0 2

2 3

4 4

5 5

1 7

3 9

dtype: int64

Ques15: Print the number of occurrences of unique values in a series. (use value_counts).

Ans15:

>>> S=s([1,2,5,1,6,2,5,5,1])

>>> S.value_counts()

Output:

1 3

5 3

2 2

6 1

dtype: int64

Ques16: Create a Series of 10 integers, and later change its dtype to be float (use astype).

Ans16:

```
>>> s1=s([10,20,30,40,50])
>>> s1.astype(float)
```

Output:

```
0   10.0
1   20.0
2   30.0
3   40.0
4   50.0
dtype: float64
```

Ques17: Convert the Series you created above to numpy array (use to_numpy(), or array)

Ans17:

```
s1=s([10,20,30,40,50])
>>> s1.to_numpy()
```

Output:

```
array([10, 20, 30, 40, 50], dtype=int64)
```

Ques18: Delete an item from Series using single index.

Ans18:

```
>>> s1=s([10,20,30,40,50])
>>> s1.drop(2)
```

Output:

```
0   10
1   20
3   40
4   50
dtype: int64
```

Ques19: Find the number of items in a series. (use len or count).

Ans19:

```
>>> s1=s([10,20,30,40,50])
>>> s1.count()
```

Output:

5

Ques20: Append Series by assigning a value to a new index. (S[n]=v).

Ans20:

```
>>> s1=s([10,20,30,40,50])
>>> s1[5]=60
>>> s1
0  10
1  20
2  30
3  60
4  50
5  60
dtype: int64
```

Ques21: Check if a value is present in a Series. (use type cast to a set or check in values).

Ans21:

```
>>> s1=s([10,20,30,40,50])
>>> result=s1.isin([10,20,50])
>>> result
0   True
1   True
2  False
3  False
4   True
```

dtype: bool

Ques22: Print the index of a Series and also if all indexes appear only once. (use is_unique).

Ans22:

```
>>> s1=s([10,20,30,40,50])
>>> s1.index=['a','b','c','d','e']
>>> s1.index
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
>>> s1.is_unique
True
```

Ques23: Create two Series one with default index, other with index like 'a','b','c','d', etc. then access both the Series based on label and position (use iloc for index position, and loc for index labels)

Ans23:

```
>>> s1=s([1,2,3,4,5])
>>> s2=s([10,20,30,40,50],index=['a','b','c','d','e'])
>>> s1.iloc[0:5]
0  1
1  2
2  3
3  4
4  5
dtype: int64
>>> s2.loc['a':'e']
a  10
b  20
c  30
d  40
e  50
```

dtype: int64

Ques24: Try function at and iat on above problem and observe the difference in output with respect to loc and iloc

Ans24:

>>> s2.at['b']

20

>>> s2.iat[3]

40

As we can see in the above output when we use at function in the s2 series we have to use label index to have the output which is similar to loc on the other hand when we use iat function we have to give position index to have the output which is similar to iloc function. But we can't use at and iat functions to have the range of output.