ForLoops

January 29, 2019

1 For loops

For loops are used to *iterate* through *elements* of an *object*, i.e. do something to elements of an object, one at a time. Python supports several styles of for loop:

1. **Classic (iterator) for loop:** An iteration index (usually i) is used to access the ith element in the object.
2. **For each loop:** This style accesses each element of an object using a keyword.
3. **Enumerator loop:** Need an iterator and the object? Use the *enumerate* keyword.

```
In [1]: # Classic loop example

        obj = [1,2,3,4,5]  # the object is a list with five elements
        result1 = []  # define an empty list to store the result calculation

        # iterate through obj, square each element, and store
        for i in range(len(obj)): # this says "loop through each element, obj[i],
        # where i=0:number of elements in obj"
            print('Iterator: ',i)
            element = obj[i]  # access the ith element in obj
            squared = element**2  # square the ith element in obj
            result1.append(squared)  # store the result
            print('Element: ',element)
            print('Squared: ',squared,'\n')

        print('obj: ', obj)
        print('result: ',result1)
```

```
Iterator:  0
Element:  1
Squared:  1

Iterator:  1
Element:  2
Squared:  4

Iterator:  2
```

```
Element:  3
Squared:  9

Iterator:  3
Element:  4
Squared:  16

Iterator:  4
Element:  5
Squared:  25

obj:  [1, 2, 3, 4, 5]
result:  [1, 4, 9, 16, 25]
```

```
In [2]: # For each example, using the same obj as before

        result2 = []

        for element in obj:
            squared = element**2  # square the ith element in obj
            result2.append(squared)  # store the result
            print('Element: ',element)
            print('Squared: ',squared,'\n')

        print('obj: ', obj)
        print('result: ',result2)
```

```
Element:  1
Squared:  1

Element:  2
Squared:  4

Element:  3
Squared:  9

Element:  4
Squared:  16

Element:  5
Squared:  25

obj:  [1, 2, 3, 4, 5]
result:  [1, 4, 9, 16, 25]
```

```
In [3]: # Enumerate example, using the same obj as before.
# This time, we'll store each object index as well

        result3 = []
        indices = []

        for index,element in enumerate(obj):
            squared = element**2
            result3.append(squared)
            indices.append(index)
            print('Iterator: ',index)
            print('Element: ',element)
            print('Squared: ',squared,'\n')

        print('obj: ', obj)
        print('result: ',result3)
        print('object indices: ',indices)
```

```
Iterator:  0
Element:  1
Squared:  1

Iterator:  1
Element:  2
Squared:  4

Iterator:  2
Element:  3
Squared:  9

Iterator:  3
Element:  4
Squared:  16

Iterator:  4
Element:  5
Squared:  25

obj:  [1, 2, 3, 4, 5]
result:  [1, 4, 9, 16, 25]
object indices:  [0, 1, 2, 3, 4]
```

1.1 A few more examples....

You can also use loops to iterate through keys of a dictionary, columns/rows of a Pandas dataframe, etc....

```
In [4]: # dictionary example
        fruit_dict = {'apples': 12,
                      'oranges': 10,
                      'bananas': 13}

        for fruit in fruit_dict:  # for each key in the dictionary...
            print('Dictionary key: ',fruit)
            print('Dictionary value associated with the current key: ',fruit_dict[fruit],'\n')
```

```
Dictionary key:  apples
Dictionary value associated with the current key:  12

Dictionary key:  oranges
Dictionary value associated with the current key:  10

Dictionary key:  bananas
Dictionary value associated with the current key:  13
```

```
In [5]: # Pandas example
        import pandas as pd

        # create a dataframe from a dictionary
        temp_dict = {'A': [1,2,3,4,5],
               'B': [6,7,8,9,10],
               'C': [11,12,13,14,15]}
        print('Dictionary:\n',temp_dict)
        df = pd.DataFrame.from_dict(data=temp_dict)
        print('DataFrame:\n',df,'\n')
        # NOTE: the far left column on the dataframe is a column of row indices

        # iterate through each column of the dataframe
        print('ITERATE THROUGH EACH COLUMN \n ------------------------')
        for col in df.columns:
            print('Column Name: ',col)
            print('Column:\n',df[col])  # a single column in a dataframe is called a series
        print('\n')

        # iterate through each row of the dataframe
        print('ITERATE THROUGH EACH ROW \n ------------------------')
        for idx in df.index:
            print('Row number: ',idx)
            print('Row:\n',df.loc[idx,:])
```

```
Dictionary:
 {'A': [1, 2, 3, 4, 5], 'B': [6, 7, 8, 9, 10], 'C': [11, 12, 13, 14, 15]}
DataFrame:
A   B   C
0  1   6  11
1  2   7  12
2  3   8  13
3  4   9  14
4  5  10  15

ITERATE THROUGH EACH COLUMN
 ------------------------
Column Name:  A
Column:
0    1
1    2
2    3
3    4
4    5
Name: A, dtype: int64
Column Name:  B
Column:
0     6
1     7
2     8
3     9
4    10
Name: B, dtype: int64
Column Name:  C
Column:
0    11
1    12
2    13
3    14
4    15
Name: C, dtype: int64


ITERATE THROUGH EACH ROW
 ------------------------
Row number:  0
Row:
A     1
B     6
C    11
Name: 0, dtype: int64
Row number:  1
Row:
```

```
A     2
B     7
C    12
Name: 1, dtype: int64
Row number:  2
Row:
A     3
B     8
C    13
Name: 2, dtype: int64
Row number:  3
Row:
A     4
B     9
C    14
Name: 3, dtype: int64
Row number:  4
Row:
A     5
B    10
C    15
Name: 4, dtype: int64
```