0121_more_list_comprehension_examples

February 17, 2019

1 Examples of using python list comprehensions as an alternative to loops

1.1 Example 1 - double the numbers

Standard loop approach:

```
In [1]: foo = [1, 2, 3, 4]
        bar = []

        for x in foo:
            bar.append(x * 2)

        print(bar)
```

```
[2, 4, 6, 8]
```

Using list comprehension

```
In [2]: foo = [1, 2, 3, 4]
        bar = [x * 2 for x in foo]
        print(bar)
```

```
[2, 4, 6, 8]
```

1.2 Example 2 - convert celsius to fahrenheit

This example calls a function from within the list comprehension.
Define the function:

```
In [3]: def convert_celsius_to_fahrenheit(deg_celsius):
            """
            Convert degress celsius to fahrenheit
            Returns float value - temp in fahrenheit
            Keyword arguments:
                def_celcius -- temp in degrees celsius
            """
            return (9/5) * deg_celsius + 32
```

Standard loop approach:

```
In [4]: #list of temps in degree celsius to convert to fahrenheit
        celsius = [39.2, 36.5, 37.3, 41.0]

        #standard for loop approach
        fahrenheit = []
        for x in celsius:
            fahrenheit.append(convert_celsius_to_fahrenheit(x))

        print('Using standard for loop: {}'.format(fahrenheit))
```

```
Using standard for loop: [102.56, 97.7, 99.14, 105.8]
```

Using list comprehension:

```
In [5]: fahrenheit = [convert_celsius_to_fahrenheit(x) for x in celsius]
        print('Using list comprehension: {}'.format(fahrenheit))
```

```
Using list comprehension: [102.56, 97.7, 99.14, 105.8]
```

1.3 Example 3 - convert the strings to different data types

This example also make use of the zip function. Zip allows you to iterate through two lists at the same time.

```
In [6]: inputs = ["1", "3.142", "True", "spam"]
        converters = [int, float, bool, str]

        values_with_correct_data_types = [t(s) for (s, t) in zip(inputs, converters)]
        print(values_with_correct_data_types)
```

```
[1, 3.142, True, 'spam']
```

1.4 Example 4 - Using if statements within a list comprehension

The example filters a list of file names to the python files only

```
In [7]: unfiltered_files = ['test.py', 'names.csv', 'fun_module.py', 'prog.config']

        # Standard loop form
        python_files = []
        # filter the files using a standard for loop
        for file in unfiltered_files:
            if file[-2:] == 'py':
                python_files.append(file)
        print('using standard for loop: {}'.format(python_files))

        #list comprehension
        python_files = [file for file in unfiltered_files if file[-2:] == 'py']

        print('using list comprehension {}'.format(python_files))
```

```
using standard for loop: ['test.py', 'fun_module.py']
using list comprehension ['test.py', 'fun_module.py']
```

1.5 Example 5 - List comprehension to create a list of lists

```
In [8]: list_of_lists = []

        # Standard loop form
        for i in range(5):
            sub_list = []
            for j in range(3):
                sub_list.append(i * j)
            list_of_lists.append(sub_list)

        print(list_of_lists)


        # List comprehension
        list_of_lists = [[i * j for j in range(3)] for i in range(5)]

        print(list_of_lists)
```

```
[[0, 0, 0], [0, 1, 2], [0, 2, 4], [0, 3, 6], [0, 4, 8]]
[[0, 0, 0], [0, 1, 2], [0, 2, 4], [0, 3, 6], [0, 4, 8]]
```

1.6 Example 6: Iterate over all items in a list of lists

The code converts a list of lists to a list of items We call this flattening the list.

```
In [9]: list_of_lists = [[8, 2, 1], [9, 1, 2], [4, 5, 100]]

        # Standard loop form
        flat_list = []
        for row in list_of_lists:
            for col in row:
                flat_list.append(col)

        print(flat_list)
        # List comprehension:
        flat_list = [item for sublist in list_of_lists for item in sublist]
        print(flat_list)
```

```
[8, 2, 1, 9, 1, 2, 4, 5, 100]
[8, 2, 1, 9, 1, 2, 4, 5, 100]
```