homework0

August 28, 2019

1 Homework 0

1.0.1 Objectives

- Get set up on Jupyter
- Basic python operations
- Do not save work within the ml_practices folder
  - Create a folder for your homework assignments within your home directory and copy hw temples there

1.0.2 General References

- Python Built-in Functions
- Python Data Structures
- Python Lists
  - Reference 1 Lists
  - Reference 2 Lists
- Enumerated Lists
  - Example 1 using enumerate()
  - Example 2 using enumerate()
- Python Cheat Sheets

1.1 Lists

Create small lists using a standard for loop and python's list comprehension syntax

In [1]:
```
""" TODO:
Create an empty list, x and populate it with the first
15 even numbers, starting at 0, using a standard for
loop and the append() function
"""
x= []
for i in range(30):
    if i%2==0:
        x.append(i)
```

```
        """ TODO:
        Create and populate another list, y with the same even
        values instead using list comprehension
        """

        y= [i for i in range(30) if i% 2 == 0]

        """ TODO:
        Print the two lists, their lengths and verify they have
        the same values
        """
        print('x list: ', x,'\n','y list: ',y)
        assert len(x)==len(y), 'The two lists does not have the same length'
```

```
x list:  [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
 y list:  [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

1.1.1 Enumerated Lists

In [3]:
```
        """ TODO:
        Use the enumerate() function to print each element of
        the list x, alongside its index. For example, output
        something of a similar form:
            x[0]:  0
            x[1]:  2
            ...
        You may also refer to the example links referenced above
        for more details on enumerate().
        """

        for ind, val in enumerate(x):
            print('x[%d]: %d'%(ind, val))
```

```
x[0]: 0
x[1]: 2
x[2]: 4
x[3]: 6
x[4]: 8
x[5]: 10
x[6]: 12
x[7]: 14
x[8]: 16
x[9]: 18
x[10]: 20
x[11]: 22
```

```
x[12]: 24
x[13]: 26
x[14]: 28
```

1.1.2 Copying lists

In python, the variable used for a list, is actually a reference that points to where the list resides in memory. When using normal assignment sytnax to copy the contents of a list to create another list, both lists will reference the same memory location. Thus, a change to one variable will change both. Copies made using this syntax are refered to as "shallow" copies. For example,

```
L1 = [1 ,3 ,4]
L2 = L1 # shallow copy
L2[1] = 55
print(L1) # the change will be visible in L1 and L2
```

To create an independent copy of a list, i.e. a "deep" copy, use slicing or the `list()` function. This will designate independent separate memory space for the copied list.

```
L2 = L1[:]
L3 = list(L1)
```

You may refer to the references on lists provided above.

```
In [4]: """ TODO:
        Create a shallow copy of x, named x_shallow
        Changing x_shallow will change the original x list
        """
        x_shallow= x


        """ TODO:
        Create a deep copy of x, named x_deep
        Changing x_deep will NOT change the original x list
        """
        x_deep= list(x)

In [5]: # Simply run this cell
        x

Out[5]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

In [6]: # Simply run this cell
        x_deep[0] = 2002
        x_deep

Out[6]: [2002, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

```
In [7]: # Simply run this cell
        # x[0] will still be 0
        x

Out[7]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

In [8]: # Simply run this cell
        x_shallow[0] = 4004
        x_shallow

Out[8]: [4004, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

In [9]: # Simply run this cell
        # x[0] will be 4004
        x

Out[9]: [4004, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

1.2 Dictionaries

Create a simple dictionary, obtain the lists of key-value pairs, keys, and values.

```
In [14]: """ TODO:
         Create a dictionary with a few entries.
         """
         test= {'location': 'Norman',
                'temperature': 82,
                'humidity': 61,
                'wind speed': 3,
                'precipitation': 0,
                'wind direction': 'W'
               }

         # TODO: Simulataneously iterate over the key-value pairs in the dictionary
         # print them out
         for key, val in test.items():
             print('The key is %s, and value is "%s"'%(str(key), str(val)))


         # TODO: Only iterate over the keys and print them
         print("Keys:", test.keys())



         # TODO: Only iterate over the values and print them
         print("Values:", test.values())

The key is location, and value is "Norman"
The key is temperature, and value is "82"
```

```
The key is humidity, and value is "61"
The key is wind speed, and value is "3"
The key is precipitation, and value is "0"
The key is wind direction, and value is "W"
Keys: dict_keys(['location', 'temperature', 'humidity', 'wind speed', 'precipitation', 'wind di
Values: dict_values(['Norman', 82, 61, 3, 0, 'W'])
```