Python, Day 5.5: Finishing with lists, Functions

Andrew Bydlon

January 14, 2019

Some method functions for lists

Similar to the functions for strings, we have many for lists. They are also adjoined to the end of the list:

MyList.function()

Here are some options for function():

- append(element): Adjoins element to the end of the list.
- insert(pos,element): Adds element at position pos.
- index(element): Finds the position of the first element.
- remove(element): Eliminates the first occurence of element.

Some additional functions

- count(element): Yields the number of occurences of element.
- clear(): Empties the list of elements.
- sort(): Sorts the list from smallest to largest.You can include the reverse=True inside the parenthesis to sort from largest to smallest.
- reverse(): Reverses the order of elements.
- extend(sequence): Adds the sequence to the end of the given sequence.
- pop(pos): Removes and prints the element at position pos. If no pos is given, it removes the the last element from the list.

One major thing to note is that these functions don't return anything! They actually manipulate the list, unlike in the case of strings.

Functions!

A **function** is a block of code which can be executed within the program mulitple times.

It is of particular use for reducing the size of source code if you will be applying the same code to different pieces of data.

Syntax (Functions)

```
def FunctionName(var1, var2, var3,...):
    command1
    command2
    ReturnStatement # The output of the function call
```

This is usually declared at the beginning, and executed throughout the program.

Example (A function)

```
def MyFunction(integer, decimal, hometown):
    print(hometown, "is a very nice town. The product of your favorite \
    integer and decimal is", integer*decimal)

hometown = input("Enter your home town. ")
integer = int(input("Enter your favorite integer. "))
decimal = float(input("Enter your favorite decimal. "))

MyFunction(integer, decimal, hometown)
```

Example (Alternative Approach)

```
def EarlierFunctionMod():
    hometown = input("Enter your home town. ")
    integer = int(input("Enter your favorite integer. "))
    decimal = float(input("Enter your favorite decimal. "))

    print(hometown, "is a very nice town. The product of your favorite \
    integer and decimal is", integer*decimal)

EarlierFunctionMod()
```

Quick note about return

By default, the functions don't actually yield any data. They are merely executing the commands within.

If you want to output a value given a function, you can do that with return.

Example

```
def FunctionName(var1, var2, var3,...):
    command1
    command2
    return SomeVariable # The output of the function call
```

Different Types of Variables

Just a quick linguistic note: There are 2 different types of variables for functions: **local** and **global** variables.

- A local variable is one that is declared within a function. This is important to note because you **CAN NOT** call a local variable outside of a function.Its value will be removed from RAM at the end of the function.
- A global variable is one declared outside of all functions (with no tabs). This can be accessed within functions or outside of functions. You can use the global modifier to make variables globally defined.

It is important to track the **scope** of your variable, knowing what can see a given piece of data.

Example

```
> > > def MyFactorial(n):
        Local = 1
        for i in range(1,n+1):
            Local *= i
        global Global # Declares Global of global type
        Global=5
        return Local
> > > MyFactorial(10)
3628800
> > > 2*3*4*5*6*7*9*8*10
3628800
> > > Local
Traceback (most recent call last):
NameError: name 'Local' is not defined
> > > Global
5
```

Functions within Functions

As soon as a function is declared, it can be called within other functions:

Syntax

```
def function1(var1,var2):
    commands
    print(Something)

OtherCommands

def function2(var3,var4):
    function1(var3,var4)
    function1(var4,var3)

function2(2,5)
```

Example

Example (Functionception)

```
def MyFactorial(n):
    local = 1
    for i in range(1,n+1):
        local *= i
    return local

def MySquare(n):
    return n**2

def main():
    X = int(input("Enter an integer: "))
    print(MySquare(MyFactorial(X)))
    print(MySquare(logarithm(X)))

main()
```

Assignment 9

Write a program with the following two functions:

1. ListElementTracker: Takes as input a list of integers MyList and an integer MyInteger, and returns a list of all of the positions of MyInteger in MyList. **hint:** It may be useful to use a for loop with a locally defined new copy of MyList.
2. ListReverser: Takes a list and reverses its order.

Then take the 2 pieces of info from the user. Print the result of applying ListElementTracker to their list, and then ListElementTracker applied to their reversed list.

Upload your .py file when you finish.