Intro to Python - Day 2

joseph.rejive

October 2018

1 Functions

Functions are blocks of code that execute only when you call them. Creating functions become useful when you have to repeat an operation multiple times.

```python
def my_function():
    print("This is my function")

print("Hello")
my_function()
```

To create a functions, we use the “def” keyword. Note that the next line is indented because it comes after a line ending in a colon (“:”). Let’s take a look at a more interesting function.

1.1 Arguments

```python
def make_list(length_of_list):
    my_list = []
    for number in range(length_of_list):
        my_list.append(number)
    return my_list

ordered_list = make_list(10)
print(ordered_list)
```

Arguments are data that we pass into a function. In this case, the argument we give to the function is “length_of_list”. Then inside the function, we use that variable to determine up to what number we want the “range” function to count. In other words, that variable determines the length of the list. The final line of the functions returns the list that we created. It’s because of the return statement that “ordered_list” (the variable outside the function) actually references the list we created. You can also add multiple arguments to a function:

```python
def make_list(length_of_list, name = "Tim"):
    my_list = []
    for number in range(length_of_list):
        my_list.append(name)
    return my_list

first_list = make_list(4)
print(first_list)
second_list = make_list(4, name = "Joe")
print(second_list)
```

In this example, our function accepts something called a “default argument”. This means that if we only provide one argument (the argument for length_of_list), the “name” argument will automatically be set to

"Tim". However, we can also specify what we want the default argument to be by saying name = "Joe". If your function includes these default arguments, make sure that they are the last arguments. For example, the following code will not work:

```python
def make_list(name = "Tim", length_of_list):
    my_list = []
    for number in range(length_of_list):
        my_list.append(name)
    return my_list

ordered_list = make_list(10)
```

Python throws an error: "SyntaxError: non-default argument follows default argument". Default arguments are arguments that we have to give values to when we call a function. Python throws an error in this scenario because "length_of_list" is a non-default argument and comes after "name", which is a default argument.

1.2 Mini Lab

Write a function that accepts two lists of integers of the same size as inputs. Your function should return a new list where each element is the sum of the elements from the other two lists.

For example:
Input: [3,5,2,8] and [5,9,1,0]
Output: [8,14,3,8]

2 Strings

A string is an array of characters. Here's an example:

```python
my_string = "Hello, how are you doing?"

print(len(my_string))

print(my_string[0])
print(my_string[1])

print(my_string.lower())
print(my_string.upper())

print(my_string.replace("o", "A"))
print(my_string.split(" "))
```

In this example, we can see that strings are similar to lists: the "len" function can find the length of a string, and strings can be indexed in the same way as lists. There are also many functions specific to strings. They are all self explanatory except for the split function. This functions takes the string and splits it into an array based off the argument we provide. Since we provided a space (" "), we created a list where each element is a word. Below are some useful string functions:

1. find(a) - returns the index of the first instance where "a" was found. If "a" isn't in the string, returns -1.
2. isdigit() - returns True if all the characters are digits.
3. replace(a,b) - replace all instances of the substring "a" with the substring "b".

4. split(a) - splits the string into a list on every instance of the substring “a”

   Ex: “Hello, how, are, you?”.split(“, ”) will return [“Hello”, “how”, “are”, “you?”]

It’s important to note that all string functions don’t modify the original string. Rather, they return a new, modified string.

One important aspect of strings is that they are immutable. This means once they have been created, they cannot be changed. Here’s an example:

```
my_string = "Hello, how are you doing?"
my_string[0] = "B"
```

In the above example, Python throws an error: “TypeError: ’str’ object does not support item assignment”. Since strings are immutable, we cannot change specific parts of it in the same way we modify lists.

2.1 String Slicing

String slicing allows you to easily take a substring (a portion) of any given string. Here’s an example:

```
my_string = "Python is Great"

print(my_string[0:5])
print(my_string[:5])

print(my_string[5:])

print(my_string[0:10:2]
print(my_string[::2])
```

Square brackets are used for string slicing. These brackets can take up to 3 numbers, separated by colons. Below are the rules for slicing:

1. When two numbers are provided (my_string[0:5]), the string is sliced from index 0 inclusive to index 5 non-inclusive.
2. When one number is provided after the colon (my_string[:5]), the string is sliced from the beginning to index 5 non-inclusive.
3. When one number is provided before the colon (my_string[5:]), the string is sliced from index 5 inclusive to the end.
4. When 3 numbers are provided (my_string[0:10:2]), the number is sliced from index 0 inclusive to index 10 non-inclusive and a multiplication factor of two is introduced. Normally, the string would be spliced to include indeces 0,1,2,3,4,5,6,7,8,9. But when the multiplication factor is 2, the new substring contains indeces 0,2,4,6,8.

2.2 Lab

Create a function that accepts a string that doesn’t contain punctuation and returns the length of the longest word.
Input: ”The best moment of my life happened yesterday”
Output: 9

3 Practice Problems

1. Create a function that accepts 2 strings ("a" and "b") and an integer ("n"). Your function should return True if string "a" contains the substring "b" "n" or more times. Otherwise, return False. (Hint: Search up all the Python string functions and see what's most useful).

   Example:
   a = "ABCD BCDE BCBC", b = "BC", n = 3
   Output: True

   Below is how to return a boolean (True/False):

   ```
   def my_func(a):
       if a == 1:
           return True
       return False
   ```

2. Create a function which accepts a string and removes all punctuation.
   Example:
   Input: "Hello, how are y'all doing?"
   Output: "Hello how are yall doing"

3. Create a function which takes in a string which contains multiple sentences. Using string slicing, your function should return the first sentence (including the period).

   Example:
   Input: "Hello, I am Tim. My favorite food is apples. I also like swimming."
   Output: "Hello, I am Tim."

4. Create a function which determines if a word is a palindrome. A palindrome is a word that is spelled the same forwards as it is backwards. Ignore punctuation in your code (i.e. "Racecar" and "rAceCar" are both palindromes).