Summer 2016 Bootcamp Problem Set 03

Function Created: upper(str1, str2)

- Code a function called upper that takes two strings and returns them in upper case. Notice that you are not returning one concatenated string. You are returning both strings as a tuple. You might want to google tuples or look up example code on stackoverflow.com. Both these options are great resources moving forward in your career.
- An example invocation of your function would look like this:

```
print upper("a", 'b')
print upper("woot", "this is great")
```

```
('A', 'B')
('WOOT', 'THIS IS GREAT')
```

- **Note:** The parentheses around the results are put there by Python automatically (not by you!) and indicate that you returned a tuple data structure. Also, you can use the method in Python strings called upper that returns the upper case of a given string.

Function Created: pad(filler, repeat)

- Code a function called pad that takes a filler string and returns that filler string repeated repeat times.
- This should be a one line function if you use the * operator with strings. Research this if you are not sure what this means.
- An example invocation of your function would look like this:

```
print pad('-', 10)
print pad('?', 20), pad(':',5), pad(';)', 5)
```

```
----------
???????????????????? ::::: ;);););););)
```

Summer 2016 Bootcamp Problem Set 03

Function Created: summer(number_list)

- Code a function called summer that accepts a number list and returns the sum of the numbers in the list.
- An example invocation of your function would look like this:

```
my_list = [10, 20, 30]
print summer(my_list)

print summer( [1, 2, 3, 4, 5] )

total = summer( [20, 200] )
print total
```

```
60
15
220
```

- Note the different ways of invoking our functions. Try changing the numbers in the list and testing that you get the correct result.

Function Created: adder(number_list, increment)

- Code a function called adder that accepts a number list and an increment and returns a new list (**not** the same list passed in) with each value in the original list incremented by the increment value.

```
beg_list = [10, 20, 30]
print "beg_list BEFORE function call", beg_list
end_list = adder(beg_list, 100)
print "beg_list AFTER function call", beg_list
print "end_list AFTER function call", end_list
```

```
beg_list BEFORE function call [10, 20, 30]
beg_list AFTER function call [10, 20, 30]
end_list AFTER function call [110, 120, 130]
```

- **Note**: The beg_list is unchanged after the function call. The end_list has the incremented values returned in a list.
- Once you have completed this, you might try manipulating the beg_list within the function instead of a new list as originally specified above. See how your output changes if you make this change and keep the invoking code as shown above.

Summer 2016 Bootcamp Problem Set 03

Function Created: return_dictionary(list_of_tuples)

- Code a function called return_dictionary that accepts a list of tuples as shown below and returns a dictionary.
- A dictionary is an association of key value pairs.
- The first element in the tuple will become the key and the second element will become the value.
- An example invocation of your function would look like this:

```
my_tuple_list = [('aa', 10), ('bb', 20), ('cc', 15)]

print return_dictionary(my_tuple_list)
# OR
print my_tuple_list, '--dict-->', return_dictionary(my_tuple_list)

{'aa': 10, 'cc': 15, 'bb': 20}
[('aa', 10), ('bb', 20), ('cc', 15)] --dict--> {'aa': 10, 'cc': 15, 'bb': 20}
```

- **Note**: Lists, tuples, sets, and dictionaries are very common data structures in Python. Lists are represented with straight brackets [], tuples/sets are represented with parentheses (), and dictionaries are represented with curly braces {}.
  Being able to use them and also being able to convert data from one data structure to another is very important. So please practice more with this as you are able.

Function Created: return_list_items(list_of_tuples)

- Code a function called return_list_items that accepts a list of tuples as shown below and returns a list of individual items.
- An example invocation of your function would look like this:

```
my_tuple_list = [('aapl', 93.40), ('goog', 675.22)]

print return_list_items(my_tuple_list)
# OR
print my_tuple_list, '==list==>', return_list_items(my_tuple_list)

[('aapl', 93.4), ('goog', 675.22)]
[('aapl', 93.4), ('goog', 675.22)] ==list==> ['aapl', 93.4, 'goog', 675.22]
```

- **Note**: Lists, tuples, sets, and dictionaries are very common data structures in Python. Being able to use them and also being able to convert data from one data structure to another is very important. So please practice more with this as you are able.

Summer 2016 Bootcamp Problem Set 03

Function Created: print_enumerated_stock_prices (stock_ticker_list, stock_price_list)

- Code a function called print_enumerated_stock_prices that accepts two lists; a stock ticker list and a separate stock price list.
- Do some research on **enumerate** and **zip** and use them to provide output that looks like the input and output below.
- An example invocation of your function would look like this:

```
stock_list = ["AA", "COKE", "LUV", ]
stock_price = [9.38, 140.59, 38.31]
print_enumerated_stock_prices(stock_list, stock_price)
```

```
 1 AA       9.38
 2 COKE   140.59
 3 LUV     38.31
```

- **Note**: We start the first stock ticker with an index of 1, not 0. Also, try to roughly match the format in terms of formatting.

Function Created: highlight_word(text, highlight)

- Code a function called highlight_word that accepts some text and a word to highlight.
- Anytime you find the word to highlight, make it more prominent by making it upper case.

```
print highlight_word('This is a test and only a test', "test")
```

```
This is a TEST and only a TEST
```

- There are many ways to do this. I would suggest **importing** and using the **re** module and then using re.split to "tokenize" or give you a list of the words in the sentence.

  You can then run through the words and decide if they need to be printed out as they are or in upper case.

  Look up an example of how to use the re.split method. It is just two lines of code to give you the list of the words in the sentence. re stands for regular expression which is a way to do text pattern matching in Python.

Summer 2016 Bootcamp Problem Set 03

Function Created: print_basic_stats(number_list)

- Code a function called basic_stats that accepts a number list and **returns a formatted string** of the various stats specified below.
- You should use a dictionary inside your function to store the various stats as you are calculating them. The curly braces in the formatted string result are from this dictionary being included in the output format string.
- If the list contains less than two numbers, return a formatted string containing a message as shown below. Use a dictionary for the error and error message as well just to be consistent in output.
- **Note**: Please only use the len function and the sort method in Python lists. Other than that calculate all the stats manually the old fashioned way by adding and dividing and so on as that will give you more coding practice.

  You can use the len function to get the number of items in number_list.

  You can also call number_list.sort() which "**works in place**" to order the list items. The "**works in place**" means that you don't have to specify:
  number_list = number_list.sort()

  Instead, you can just specify as follows and the sort method changes the list it is invoked on.
  number_list.sort()

  You have to be careful since not all methods in Python lists work "in place" and this is a common source of errors.

- An example invocation of your function would look like this:

```
number_list = [5]
print basic_stats(number_list)

number_list = [10, 2]
print basic_stats(number_list)

number_list = [5, 4, 3, 2, 1]
print basic_stat(number_list)
```

```
[5]             ---> {'error': 'List should contain more than one number.'}
[2, 10]         ---> {'min': 2, 'max': 10, 'median': 6.0, 'range': 8, 'total': 12, 'mean': 6.0}
[1, 2, 3, 4, 5] ---> {'min': 1, 'max': 5, 'median': 3.0, 'range': 4, 'total': 15, 'mean': 3.0}
```

Summer 2016 Bootcamp Problem Set 03

More Practice

- Remember to use Python Coding Bat or Learning Python the Hard Way if you would like more practice with coding.
- I will post the solutions and another problem set in a week after everyone has had a chance to attempt these problems.
- If you have other recommendations for coding practice resources, feel free to post to our Piazza site.
- Have fun! ☺