Assignment

February 6, 2016

1 Function Exercises

Write a function, `sum_digits`, which takes an `int` and returns the sum of its digits.

Write another function that "wraps" `sum_digits` in that it calls `sum_digits` from within it. Call this function `diff_sum_digits`. Use that function to compute the input number minus the sum of digits of input number.

Writer another function that "wraps" `diff_sum_digits`. If `diff_sum_digits` returns a result that has more than one digit (either negative or positive) have it replace it with the sum of the digits of the result. Do this repeatedly until the result has just one digit, then display it.

Below you'll find an example of what we mean when we say "wraps" - this is not a decorator function or anything like that, just a function that calls another function inside of it.

```
In [1]: def example_base_func(x):
            "Returns the value of the input * -1"
            return -1 * x

        def wraps_example_base_func(x):
            temp_val = example_base_func(x)
            if temp_val < 0:
                return "trivial example"

        print(example_base_func(5))
        print(wraps_example_base_func(5))
```

```
-5
trivial example
```

Write a function `is_consonant` that takes a character and returns `True` if it is a consonant.

Use your function to create a new fuction `to_piglatin` that takes a word, moves all starting consonants to the end of the word, then adds ay to the end and returns the result. You may expect that the the input will be just one word.

Extra Credit: Have this function check whether or not the input is multiple words and return the whole sentence in pig latin. For this you may assume that a sentence always ends with a period and is always one sentence, never more than one.

2 Namespace Exercises

The following script defines a list of customer orders. Each item in the list is a tuple containing the name of the item, the number requested, and the price per item.

Fill in just the `process_order` function below without changing any other code. This function should remove one of the elements of the order list and print a nice message about it. Finally, make sure that the final print statement correctly displays the total price for the entire list.

```
In [ ]: total = 0

        process_order(x_list):
            pass

        x = [("oranges", 4, 3.22),("gummy bears",1,1.99),("sour bites", 3, 2.33), ("antacid", 1, 5.33)]
        while(len(x)>0):
            process_order(x)
        print("total price: ", total)
```

3 Functions as Objects

The following code defines a list of names and also contains a header for the function `best`. This function should take two arguments: a function, `score`, and a list of strings, `names`. Fill in the function so that it applies the `score` function to each string in `names` and returns the one with the highest score. If there are ties in the list, your function should return the first item with the maximum score.

```
In [ ]: def best(score, names):
            pass

        names = ["Ben", "April", "Zaber", "Alexis", "McJagger", "J.J.", "Madonna"]
```

- Next, define a function, `number_of_vowels`, that returns the number of vowels in a string. Use it with your `best` function to find the name in `names` with the most vowels.
- Next, pass a `lambda` function into your `best` function to find the name in `names` with the highest number of a's.

4 Exceptions Exercises

Refactor this code that attempts to compute a reciprocal so that it keeps prompting the user for a number repeatedly until the user successfully enters a valid number. If the user enters 0, the program should print that zero does not have a reciprocal, then terminate.

```
In [ ]: try:
            x = float(input("Enter a number: "))
            print("The reciprocal of your number is", 1/x)
        except ValueError:
            print("You did not enter a valid number.")
        except ZeroDivisionError:
            print("Zero does not have a reciprocal")
        except:
            print("something else went wrong.")
```

Write a function that takes a list of grades and a corresponding list of percentages and returns the weighted average of the grades. Your function should raise an exception if a percentage is less than 0 or greater than 100, and a different exception if the percentages do not add to 100.

Run your function on `grades1` with `weights1` and `grades2` with `weights2`, defined below. Catch the errors generated in each case and print a useful message for the user.

```
In [ ]: grades1 = [88,99,100,70]
        weights1 = [30, 30, 30, 5]

        grades2 = [78, 75, 80, 99]
        weights2 = [110, 10, -20]
```