Python Basics: Practice Problems

In this homework, you're going to write code for a few problems.

You will practice these programming concepts we've covered in class: * Declaring and using variables. * Using mathematical operators. * Using string concatenation. * Storing data in lists. * Using built-in list functions (e.g., `max`, `min`, `sum`). * Using loops to go through data inside lists.

Deliverables

For each of the challenges listed below, you will create a new `.py` file and write code to solve the problem. For example, you would create `problem1.py` with your solution code to the first problem. Run the file from the command line to check your work.

Reminder: On your laptop, you can run the file from your command line with the following command:

```
python problem1.py
```

> **Hint:** Make sure you are printing something out with the `print` statement. Otherwise, you won't see any output from running your program!

Requirements:

- By the end of this, you should have six different `.py` files (one for each problem).
- If you attempt the bonus, you will have a seventh `.py` file.

Homework Problems

Problem 1: Can You Repeat Yourself Repeat Yourself?

Skill you're practicing: Writing loops.

For this problem, put your solution code into a file named `problem1.py`.

Create a string variable named `statement` and numerical variable named `num`. Underneath that, write a loop that prints the `statement` some `num` of times to the console.

Example 1:

```
statement = "Hello World"
num = 3

# Your loop here
```

Example 1, expected output:

```
Hello World
Hello World
Hello World
```

This should work, no matter the values of `statement` and `num`. For example, if you change the value of `num` to `4` instead of `3`, when you run your code in the terminal again, it should then print your statement four times instead of three. Likewise, if you change the value of `statement`, the text in the output should change.

Example 2:

```
statement = "Hi there"
num = 4

# Your loop here
```

Example 2, expected output:

```
Hi there
Hi there
Hi there
Hi there
```

Problem 2: I Got Chills, They're Multiplyin'

Skill you're practicing: Using mathematical operators and writing loops.

For this problem, put your solution code into a file named `problem2.py`.

Declare a number named `num` and a list of numbers named `num_list`. Put two or more numbers in `num_list` (our example below has a list length of `4`). Go through each element in `num_list` and multiply the number in the list by `num`, printing the resulting new list.

Example code:

```
num = 5
my_list = [1, 2, 3, 4]

# Your solution here
```

Expected output:

```
[5, 10, 15, 20]
```

This should work, no matter the values of `num` and `my_list`.

> **Hint:** You'll want to apply your knowledge of lists here. How do you access each item in a list? How do you assign a value at a specific location in the list? Refer to the class notes if you've forgotten exactly how to do these things.

Problem 3: REVERSE! — !ESREVER

Skill you're practicing: Using I/O, manipulating strings, and writing loops.

For this problem, put your solution code into a file named `problem3.py`.

Reverse a string manually, printing the result. Create a new variable storing an empty string and add the letters from the first string one by one. The `for` loop should iterate over the length of the string and access letters individually.

Example:

```
# <Input Prompt:> Enter a word or sentence, please.
# <User Input:> reverse_me

# Your expected program output:
em_esrever
```

> **Hint:** You can receive direct user input by using the `input` function. For example:

```
user_entry = input('Please enter your favorite number')
# user_entry now holds whatever the user typed in!
```

> **Note:** While there is an awesome shortcut to reverse strings, `s[::-1]`, don't use it — practice writing out the code instead.

Problem 4: Calc U Later

Skill you're Practicing: Using I/O and control flow.

For this problem, put your solution code into a file named `problem4.py`.

Create a simple calculator that first asks the user what method they would like to use (`add` for addition, `sub` for subtraction, `mult` for multiplication, or `div` for division), then asks the user for two numbers. Your program will print the result of the method with the two numbers. Here is a sample prompt:

```
# <Input Prompt:> What calculation would you like to do? (add, sub, mult, div)
# <User Input:> add
# <Input Prompt:> What is number 1?
# <User Input:> 3
# <Input Prompt:> What is number 2?
# <User Input:> 6

# Your expected program output:
Your result is 9. Calc U later!
```

> **Hint:** By default, Python sets user input to a string. If the user types `2` into the prompt, the value your variable holds will be the string `"2"`! To avoid that, you can use `int()`, as shown in the second example below.

```
# Wrong for numbers — Python saves a STRING!
# This will work for the user entering `"add"`, `"sub"`, `"mult"`, or `"div"`, but
not for numbers.
my_num = input("Please enter a number.")

# Correct for numbers — Python saves a NUMBER!
# This is how you should request the number from the user:
my_num = int(input("Please enter a number."))
```

Problem 5: Say Hello to My Little Friend

Skill you're practicing: Writing functions.

For this problem, put your solution code into a file named `problem5.py`.

Write a function called `say_hi` that prints out your favorite greeting!

> **Hint:** Remember that a function does not run its code until it is *called*. You can call a function by putting parentheses at the end of the function name, as in the example below:

```
# greeting_function gets written here.

# The following line CALLS or runs the function.
greeting_function()
```

Problem 6: Loops and Froot Loops

Skill you're practicing: Using lists, functions, and loops.

For this problem, put your solution code into a file named `problem6.py`.

Declare a list called `breakfast` and fill it with at least three examples of breakfast cereals (e.g., `"Wheaties"`, `"Froot Loops"`, etc.). Then, write a function called `cereal_time`. This function should loop through your `breakfast` list and print that each of the cereals you chose with `"are yummy!"`.

Example starter code:

```
breakfast = ["Froot Loops", "Wheaties", "Cap'n Crunch"]

# Your function

# Call your function.
```

Example Output:

```
Froot Loops are yummy!
Wheaties are yummy!
Cap'n Crunch are yummy!
```

BONUS: Problem 7: Pluralizer (Optional, But Fun!)

Skill you're practicing: String manipulation.

For this problem, put your solution code into a file named `problem7.py`.

If you take a look at the last problem's sample output, you'll notice that the last sentence, `Capt'n Crunch are yummy!`, doesn't quite make sense. What would make more sense is the phrase `Cap'n Crunch **is** yummy!`.

It would be better to look at whether or not the cereal name ends in an `s` and determine from there whether the rest of the sentence should be pluralized ("are") or singular ("is").

Your task is to alter the answer to the previous problem such that, if the last letter of the cereal string is `s`, it prints `are yummy!`, and if it ends in any other character, it prints `is yummy!`.

> **Hint:** You can access the last character in a string with the following code:

```
my_str = "Wheaties"
last_letter = my_str[-1]
```

Altered example output:

```
Froot Loops are yummy!
Wheaties are yummy!
Cap'n Crunch is yummy!
```

Last, But Not Least!

Take a deep breath. You've finished! You've earned some relaxation.