Problem Set 1, CSCI1101 (Sections 1–3), Fall 2017

Due at: 6PM, Friday, September 22nd 2017, on GitHub.

Note on the format of your submission: For each part below, we have created a separate Python file in your repository. Please note that each of your files must start with the following lines, followed by your program/code (please don't copy these from the pdf file because the characters used in a pdf file are different from the characters used in a Python file (type these lines into your file please):

```
# pylint: disable=w,c
# Problem Set 1, Part I (or Part II if it's for part II, etc.)
# Name: your last name, your first name
# Collaborators: last name1, first name1; last name2, first name2; etc.
```

–Please don't hesitate to collaborate with anyone you want (or post your questions to Piazza). However, when it comes to sitting down and writing your code, please write your own code.

–If you encounter any problems in completing this assignment or in the submission process, don't hesitate to ask for help (come to the office hours, and/or post your questions to Piazza).

Part I

In the Python script, named `part_i.py` in your repository, define a function, named `weird_number`, which has no parameters and returns the weird number defined in the **Hint** below. In your main program (that is outside the function definition), call this function, assign the value it returns to a variable, and then print the result in the shell (e.g., a phrase like `The weird number is ...`).

Hint: The `weird_number` is defined as the value of this fraction: $\frac{1+\sqrt{14}}{3}$

To compute the square root, you might want to use the `sqrt` function from the `math` module. Don't forget to import the module at the top of your program.

Part II

In your repository, there is a Python file, named `mystery.py`, in which we have defined a function, named `gen_adjective`. Every call to this function will return a string to the caller. The value of the string is an adjective, randomly selected from a list of different adjectives. So, the value of the returned string may change from one function call to another. Write a short program in the Python file, named `part_ii.py`, that asks the user for their name and prints the following phrase

`Hi, name! You look adjective today!`

where `name` should be replaced by the actual name the user has entered, and `adjective` should be the random value that the `gen_adjective` function returns. Note that there's no space before the exclamation marks in the sentence you print! For example, if the user's name is Sam, the program might print `Hi, Sam!  You look great today!` Please don't change anything in the `mystery.py` file; you write your function in `part_ii.py` and call the `gen_adjective` function from `mystery.py` (of course you should import the `mystery` module).

Part III

In your repository, in the Python script named `part_iii.py`, define a function, named `is_even`, that has an integer parameter and returns `True` if the input integer is an even number (i.e., it is divisible by 2); otherwise, it returns `False`. In your main program, call this function once with 20 as its argument, and once with 3 and print the results in the shell. In this part, using any `if` statements is not allowed because we don't know what they are yet.

Hint: These two operators might be helpful: the modulo (`%`) operator and the equality (`==`) operator (a friendly reminder that even numbers are divisible by 2 while odd numbers are not). Note that the first line in your function definition should look like the line below (feel free to replace `input_int` with any name you like):

```
def is_even(input_int):
```

Part IV (Quadratic Equation Solver)

In the Python file named `part_iv.py` in your repository, write a Python program to find the solutions to the quadratic equation $ax^2 + bx + c = 0$. The two solutions can be calculated as the following:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

For this problem, we assume that there are two solutions; in other words, we assume that a is not zero and that the discriminant (the part under the radical) is ≥ 0. We'll learn how to handle other cases later in the semester.

First, we define a function, named `quadratic_solver`, with three parameters, `a`, `b`, and `c`, that solves the quadratic equation and returns its two solutions. In the body of the function, we complete the following steps:

1. Evaluate the discriminant first and assign it to a variable.
2. Take the square root of the discriminant variable we just defined. Assign the result to a variable.
3. Define a variable, named `denominator`, that is equal to $2a$.
4. Define `numerator1` as `-b + <variable defined in step 2>`.
5. Define `numerator2` as `-b - <variable defined in step 2>`.
6. The ratio of `numerator1` over `denominator` will give us the value of x_1.
7. The ratio of `numerator2` over `denominator` will give us the value of x_2.
8. At the end, the function returns these two values (see **"note on returning multiple values in a function"** below for how to do this).

Then, in the main program outside the function definition, we prompt the user for a, b, and c (that can be real numbers and not necessarily integers). Use `raw_input` and don't forget to cast its result to an appropriate

type. We then call the `quadratic_solver` function, and pass these numbers to it as arguments, and assign the function outputs to two variables. We then print these two variables in the shell. You should always have a prompt for input, explaining what is desired (e.g., `Please enter the value for a:`). And the output should likewise be labeled (e.g., `The first solution is ...`), don't simply print the two values that the function returns.

Finally, test your program several times, with different sets of data of your choosing. When your test cases work, try this: $a = 2, b = -1.2, c = -6.3$. The answers should be approximately -1.5 and 2.1 (of course the order doesn't matter). The answers for $a = 1, b = 0, c = -1$ should be 1 and -1. Please note that when you're testing your program, for a lot of input triplets (a, b, c), you might get an error because the discriminant can be negative if these values are not chosen carefully. So, please choose the values of your a, b, and c carefully to avoid errors when testing your program (if $a > 0$ and $c \leq 0$, the discriminant is guaranteed to be positive, and you shouldn't get any errors).

Note on returning multiple values in a function

So far, we have seen functions that return only one single value (or some that don't return anything; i.e., return `None`). We should note that functions can return multiple values as well. Let's take a look at the following program (please write this piece of code in a separate file for your own practice (you don't need to submit this file) to see the results firsthand rather than simply reading the code here):

```
def new_function():
    rv1 = 1
    rv2 = 'CS'
    return rv1, rv2


result_num, result_str = new_function()
print 'The first return value is', result_num
print 'The second return value is', result_str
print 'The two return values are', result_str, result_num
```

So the lesson here is that we can separate multiple values using comma in the function's return statement (in the example above, we separated `rv1` and `rv2` using a `,`). When we call this function in the main program, it returns all the values listed in its return statement. When calling the function, if we want to assign these return values to some variables for later use, we can just put the names of the variables, separated by comma, on the left-hand side of the assignment operator (`=`) (see the first line in the main program above). So, the line with the function call above is equivalent to writing

```
result_num, result_str = 1, 'CS'
```

which, in turn, is equivalent to:

```
result_num = 1
result_str = 'CS'
```