Assignment 3

Conditional Branching and Simple Repetition

Date Due: 11:59pm Friday Feb 2 **Total Marks: 21**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- Each question indicates what to hand in. You must give your document the name we prescribe for each question, usually in the form aNqM, meaning Assignment N, Question M.
- Make sure your name and student number appear at the top of every document you hand in. These conventions assist the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you.
- Do not submit folders, or zip files, even if you think it will help. It might help you, but it adds an extra step for the markers.
- Programs must be written in Python 3.5+.
- **Assignments must be submitted to Canvas.** There is a link on the course webpage that shows you how to do this.
- **Canvas will not let you submit work after the assignment deadline.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Read the purpose of each question. Read the Evaluation section of each question.

Question 1 (4 points):

Purpose: To practice slicing and using string methods

An **email address** is a sequence of characters with the following structure:

- it contains exactly one @ character
- there are 1 or more characters before the @. We call these characters the **username**
- There is at least one period (.) SOMEWHERE after the @
- IMMEDIATELY after the @ there are one or more characters that are NOT periods. We call all the characters after the @ the **fully qualified domain name (FQD)**, within the FQD there are zero or more **subdomain**s separated by periods, a **domain** before the last-period-after-the-@, and a **top-level domain** (i.e com, ca, etc) after the last period.

For this question, you will write a program that lets the user type in a valid email address, and reports the **username** and the **top-level domain** components of that address. Your program may assume that the user always types a correctly-formatted email address. You may also assume all top-level domains are no longer than 3 characters.

Hint: Tools such as slicing, negative indexing, and the string method `.find()` will probably be the most useful here.

Sample Runs

Here is one possible sample run. Green text was entered by the user, and the blue text is based on that input.

```
Enter email address: ash@pokemon.kanto.jp
Username: ash
First Subdomain/Domain: jp
```

Here is another one.

```
Enter email address: jeff.long@usask.ca
Username: jeff.long
First Subdomain/Domain: ca
```

What to Hand In

Hand in your solution in a file called `a3q1.py`.

Evaluation

- 3 marks for correctness
- 1 mark for code readability and style
- -1 mark if identifying information is missing (name, NSID, student number and instructor's name).

Question 2 (5 points):

Purpose: To practice using modules and simple conditionals

For this question, you'll write a program that checks whether a randomly generated integer is a perfect cube. A perfect cube is an integer for which the cube root is also an integer.

First, your program should import both the `random` and `math` modules. Make sure to use the import syntax as shown in the course readings.

Then, your program should generate a random number between 1 and 10000 (inclusive).

Finally, your program should display that number to the console. If the number is a perfect cube, an additional message should be displayed to say so. Use a conditional statement along with any functions from the `math` module that you think you need to do this.

Sample Run

Here is a possible sample run (output value of a variable is shown in blue text):

```
Your random number is: 5062
```

Here is another one.

```
Your random number is: 125
Amazing!  125 is a perfect cube!
```

What to Hand In

- A file called `a3q2.py` containing your finished program, as described above.

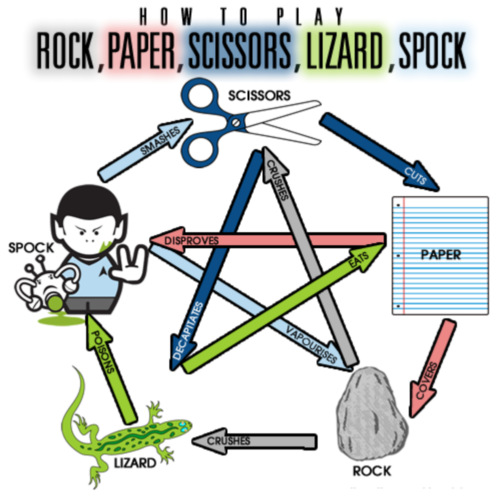
Evaluation

- 1 mark: The random and math modules are correctly imported
- 2 marks: A random number in the correct range is generated and displayed
- 2 marks: A message correctly reports whether the number is a perfect cube
- -1 mark if the student did not include their name, NSID, student number and instructor's name at the top of the submitted file.

Question 3 (8 points):

Purpose: To practice chained if-elif-else constructs.

The game "rock-paper-scissors-lizard-spock" is an extension of the commonly known game "rock-paper-scissors". If you're not familiar with rock-paper-scissors, click here. Rock-paper-scissors-lizard-spock adds two additional moves to the basic game. The rules are summarized both in this educational video (click to view), and the image below. The arrows indicate which move beats which. For example, the arrow from paper to Spock indicates that paper disproves Spock (if one player plays paper, and the other plays Spock, the one that played paper wins).



There are 25 possible pairs of moves, so sometimes it's hard to remember them. You will write a computer program that will act as a referee. The program will ask for the moves made by the two players via console input and report which player won.

To solve this problem, you'll write a function that accepts the two moves made by the players as arguments, and returns the outcome (for more detail, see below). Separately, you'll write a main program to perform the console input to request player 1's move, and player 2's move, and then call the function to determine the outcome.

Note: The computer is not one of the players, the computer only determines who won given the moves that the two human players made.

Sample Run

Sample input and output (input typed by the user is shown in green text) for two different runs, the first showing a winner, the second showing a tie.

```
Enter move for player 1: spock
Enter move for player 2: rock
Player 1 wins!
```

```
Enter move for player 1: lizard
Enter move for player 2: lizard
It was a tie!
```

Advice on how to get started

(a) Write a function called `rock_paper_scissors_lizard_spock()` that has two parameters: the first parameter is the move made by player 1, the second parameter is the move made by player 2. The function should **return**:

- `1` when player 1 wins
- `2` when player 2 wins
- `0` if the game is tied

Hint: There's a bad way and a better way to do this question. You'll know you're doing it the bad way if your program is very long, and if you are tempted to copy/paste/edit to write your code.

The bad way looks at the problem from the point of view of *analyzing the 25 combinations of moves* by the players.

The better way consists of looking at the problem from the point of view of the *three possible outcomes*: a win for player 1, a win for player 2, or a tie. Try to aim for this approach.

(b) Write an appropriate docstring for your function in part (a).

(c) In the main program (after the function definition), write code to prompt for, and read, the players' moves from the console. You may assume that the user always enters one of the five strings: `'rock'`, `'paper'`, `'scissors'`, `'lizard'`, `'spock'`.

(d) Call the `rock_paper_scissors_lizard_spock()` function giving the two moves as arguments, and save its return value using a variable.

(e) Use the return value to print a message to the console indicating which player won, or whether it was a tie.

(f) Your program only has to referee one game. To referee another game, run the program again!

What to Hand In

- A file called `a3q3.py` containing your finished program, as described above.

Evaluation

- 1 mark: Your function takes two string arguments, one move chosen by each player, and returns the integer values 0, 1, or 2.
- 3 marks: Your function correctly uses a chained-if statement to determine the outcome of the game.
- 1 mark for good organization of your function logic (i.e. using the better way, and not the bad way)
- 1 mark for an acceptable docstring for `rock_paper_scissors_lizard_spock()`;
- 1 mark: Console input is present and external to `rock_paper_scissors_lizard_spock()`
- 1 mark: Game result is correctly displayed and is external to `rock_paper_scissors_lizard_spock()`
- -1 mark if identifying information is missing (name, NSID, student number and instructor's name).

Question 4 (4 points):

Purpose: To practice a simple loop and simple conditionals

For this question, you'll write a simple program that makes the user try to guess a secret number repeatedly until they get it right.

First, your program should select a random number between 1 and 10 (inclusive). You can use the `randint()` method from the `random` module to do this. This is the secret number that the user is trying to guess.

Then, your program should allow the user to input numbers as many times as needed in order to guess the secret number. If the user gets it wrong, the program should give a hint by telling the user whether their guess was too high or too low. If the user gets it right, the program should tell them so and then terminate.

Sample Run

Sample input and output (input typed by the user is shown in green text):

```
Guess a number from 1 to 10! 5
Too low!
Guess again: 7
You got it, the secret number was 7
```

What to Hand In

- A file called `a3q4.py` containing your finished program, as described above.

Evaluation

- 3 marks for correctness
- 1 mark for code readability and style
- -1 mark if the student did not include their name, NSID, student number and instructor's name at the top of the submitted file.