Problem Set 0

Computer Programming for Lawyers

Introduction

This is your first programming assignment. You must submit it, but it will not be graded. The primary goals of the tasks that follow are to acquaint you with the replit.com service and to ensure you are familiar with the basic features of Python. You should read Chapters 1, 3, and 4 of the book before you begin the problem set.

Another goal for this assignment is to help you better appreciate what this class will entail. In previous years, some students mentioned that the weekly problem sets were more than they bargained for. Unlike most assignments you have completed in law school, the problem sets in this class have correct answers, and it is hard to predict in advance how many hours will be required to find the correct answer to a particular problem.

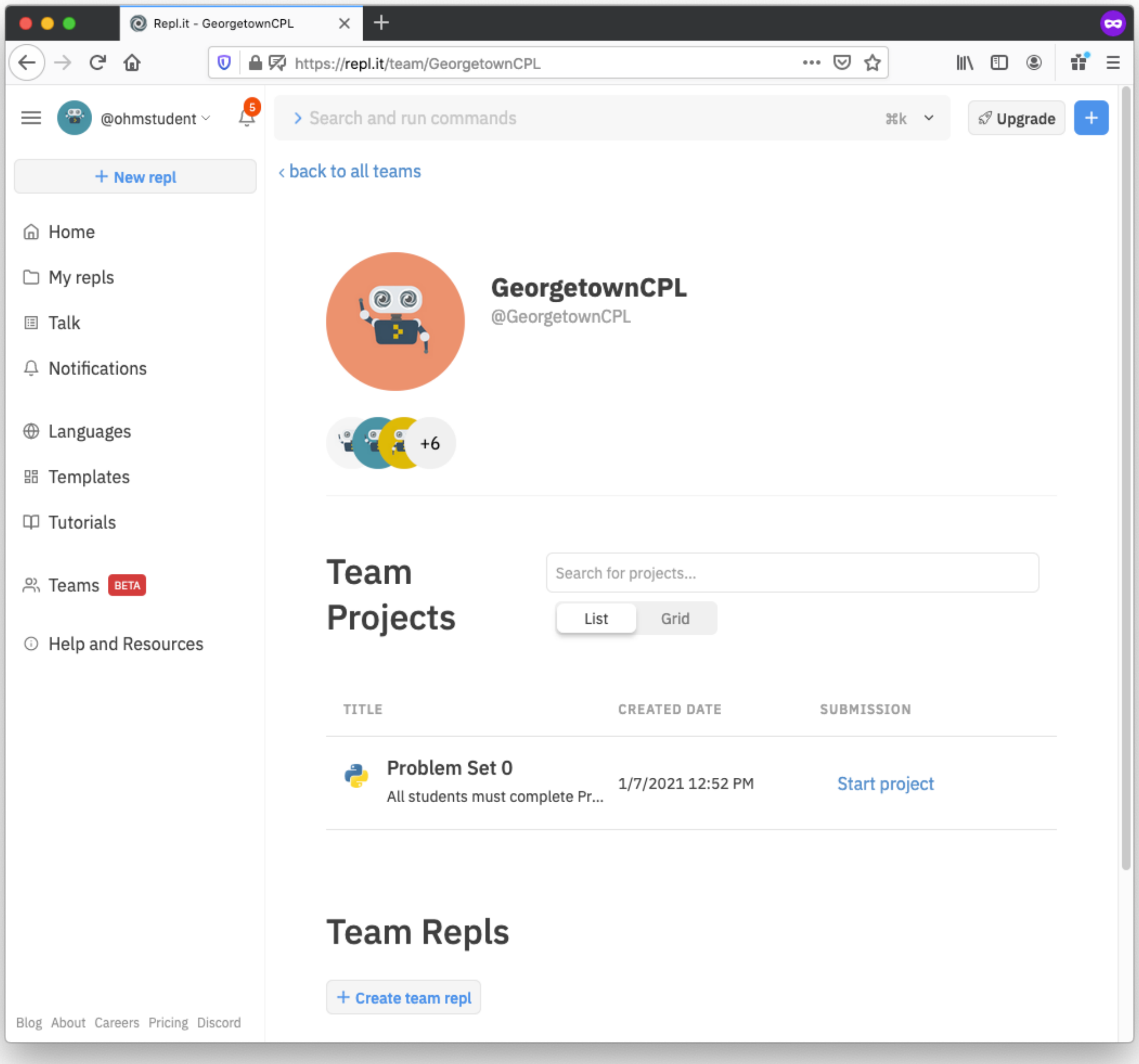
The key ingredient for success in this class is the ability to allot enough time each week to finish the problem sets. Students who have many other intensive competing obligations might struggle. To help you gauge what this means for you, we are assigning this mini-problem-set for you to try to solve before the start of the first lecture of this class. You must click the "Submit" button in replit.com with your answers before 10am on January 17th. The answer will be discussed in your first lab meeting on Thursday, January 19th.

Beginning Problem Set 0 in replit.com

Before attempting this problem set, you should have set up a replit.com account (using your Georgetown Google account as your login), and joined the @GeorgetownCPL2023 replit.com Team by clicking the link provided in Canvas. If you haven't yet done this, go back and do all of the steps in the "Week 0 Assignments" document in Canvas.

To begin your work on Problem Set 0, do the following (which you will do every week):

1. Click "Teams" in the left pane of replit.com.
2. Click "GeorgetownCPL2023" in the list that appears in the right (it will probably be your only choice).
3. In the row labeled, "Problem Set 0," in the "Team Projects" list, click "Start project," as shown here:



TITLE	CREATED DATE	SUBMISSION
Problem Set 0 All students must complete Pr...	1/7/2021 12:52 PM	Start project

(Note: The @GeorgetownCPL Team name in the image above was from a prior semester. All semester, substitute @GeorgetownCPL2023 whenever you see this.)

Once you click this, you will be greeted with the soon-to-be-familiar replit.com coding environment. In the upper-left corner of the screen, confirm that you are working on a project called "GeorgetownCPL2023 / Problem Set 0-*your ID*". This is your individual copy of the Problem Set, and it is where you should write the code to satisfy the requirements below.

Your new project contains a number of files in the left pane. One of these files is called `pset0.pdf` and is a copy of this document you are reading now. Another key file is called `main.py`. Click this to read a brief description of the other files.

When you are done with all of the tasks below, click the `Submit` button in the upper-right corner, and the professor and TAs will know that you have finished your problem set. (You can click `Resubmit` if you want to overwrite your submission after making changes.)

Required and Optional Tasks

This problem set asks you to complete three tasks, each of which builds on the prior task, and each more difficult than the last. To satisfy the minimum requirements for this assignment, you must find the correct answer to Task One. Everybody should attempt to solve Task Two, but it is fine if you cannot solve it. Task Three is strictly optional: solve it only if you have the time and inclination.

Expectations

- This is an ungraded assignment. You are expected to attempt to solve at least Task One below, and submit it by clicking "Submit" no later than the start of the first lecture.
- This is an individual assignment. Do not seek help from anyone except the course instructors.
- You are permitted to use only the Python features we have covered so far. Specifically, you may not use if-statements or loops.
- You should use good programming style. Your reading mentions style rules and conventions throughout, and we will introduce you to more in the coming weeks. Starting next week, we will also provide you an official "style guide" for programs written in this course, but for now, just focus on the style points raised in your reading.

Summary of Tasks

- Write `login_checker1.py`
- Write `login_checker2.py` (Optional, but must be attempted.)
- Write `login_checker3.py` (Strictly optional.)

Important Warning

For each task, we provide some example interactions with the program on the command-line. The program you submit should match these interactions *exactly*. We are looking for correctness, not creativity -- your submissions should behave in a manner identical to our examples. Even the punctuation marks should match perfectly. Do not add any extra spaces or other decorations or prompts. We will grade your code with an automated Python program, and it will only register exact matches as correct.

Tasks

Task One: login_checker1.py

Write a program that prompts the user for a username and a password, receives them as inputs from the user, and outputs each in a complete sentence.

A sample interaction with your `login_checker1.py` program would appear as below. The words "Larry" and "foo" are entered by the user. It should not print in the output.

A sample interaction with your `login_checker1.py` program would appear as below. Input typed by the user is written in *italics*. Your program should read this data as user input - it should not print it as output.

```
Username: Larry
```

```
Password: foo
Hi Larry! Your password is foo.
```

Note that we should be able to give your program *any* name and password combination. It shouldn't work only with Larry/foo.

```
Username: Mary
Password: bar
Hi Mary! Your password is bar.
```

```
Username: Barry
Password: foo bar baz
Hi Barry! Your password is foo bar baz.
```

Place the code that solves this task in the file entitled `login_checker1.py` in the left pane. Don't click "Submit" after each Task. Wait until you are done with all of the work you are going to submit before clicking this button.

Hint one: To receive input from the user, use the `input()` function as described in Chapter 4 of the book (*Making Programs Interactive*).

Hint two: You will need to build up the string-to-be-output as one multi-part string using concatenation. See Chapter 3 (*Concatenation and Type Casting*) of the textbook for details. Then, use the `print()` command (Chapter 4, *Printing*) to print the string to the screen.

Task Two: login_checker2.py

You should attempt to solve task two of this assignment, but do not fret if you cannot. Just allocate a set amount of time you have available (we suggest an hour or two) and see if you can come up with the solution.

Write a program that prompts the user for a username and a password, receives each as inputs, tests whether the inputted username and password match a particular pre-specified combination, and outputs whether or not the password matches, i.e. whether access has been granted.

For this task, the only username and password combination that should be matched are:

Username	Password
Larry	coding_is_fun

Various sample interactions with your `login_checker2.py` program would appear as below.

```
Username: Larry
Password: foo
Access granted: False.
```

```
Username: Larry
Password: coding_is_fun
Access granted: True.
```

```
Username: Mary
Password: coding_is_fun
Access granted: False.
```

```
Username: Larry
Password: coding_is_fun
Access granted: False.
```

If you can solve this task, place your code in the file entitled `login_checker2.py`. Even if you cannot come up with the solution, leave your answer in that file when you submit. Place a comment at the top of your code letting us know that you didn't come up with the correct answer.

Note: If you have coded before, or if you have read ahead in the book, you might be tempted to solve this task using an `if` statement. You may not use `if` (or anything else that isn't included in Chapters 1 to 4 of the book) to solve this task.

Additional Hints for Task Two

Because Task Two is significantly more difficult than Task One, here are some hints if you get stuck. Try to solve Task Two before you look at these.

1. Remember that to test whether two expressions are equal in Python, use `==` (two equals signs). This returns a boolean value (i.e. `True` or `False`).
2. You can save the output boolean value of an equality test to a variable for later use.
3. You will need to use a compound boolean operator as described in Chapter 3 (*Compound Operators*) of the textbook. You can assign the output of a compound operator to another variable.
4. You can print a boolean value directly using the `print()` command, but if you want to concatenate that value to a string, you need to cast that boolean value to the string type, as described in Chapter 4 (*Concatenation and Type Casting*) of the textbook.

Task Three: login_checker3.py

The third task of this assignment is strictly optional. If you solved the first two tasks easily, are enjoying these tasks, or just want to challenge yourself further, then attempt this task, which is a small extension of Task Two.

Write a program that prompts the user for a username and a password, receives them as inputs, tests whether the inputted username and password matches any one of *four* pre-specified combinations, and outputs whether or not access has been granted. The four username/password combinations your code should authenticate are:

Username	Password
Larry	coding_is_fun
Falken	Joshua
Setec Astronomy	too many secrets
Neo	False

Various sample interactions with your `login_checker3.py` program would appear as below.

```
Username: Larry
Password: foo
Access granted: False.
```

```
Username: Larry
Password: coding_is_fun
Access granted: True.
```

```
Username: Falken
Password: Joshua
Access granted: True.
```

```
Username: Neo
Password: False
Access granted: True.
```

```
Username: Neo
Password: too  many  secrets
Access granted: False.
```

If you can solve this task, place your code in the file entitled `login_checker3.py` . Even if you cannot come up with the solution, leave your answer in that file when you submit. Place a comment at the top of your code letting us know that you didn't come up with the correct answer.

Once again: do not use `if` to solve this.

Hints? If you're going to solve Task Three, you're on your own. No hints for this one!

Self-Assessment

Once you are done with this assignment, use this rough guide to conduct a self-assessment of your ability to thrive in this class. Although we doubt we calibrated the difficulty level of this assignment with precision, our goal was to make Task Three alone about as difficult as one-third of a typical problem set, given what you know at this stage in the semester. (Tasks One and Two are calibrated to be less difficult than one-third of a typical problem set.)

If you found even Task One to be impossibly difficult, you might reconsider whether this is the class for you. That