Lab 3 – CSE 101 (Fall 2020)

Objectives

The primary objectives are:

- To learn some debugging features of **PyCharm**.
- To gain additional practice with if statements to solve problems.

1. Debugger in PyCharm

Download debug_test.py and open the file in **PyCharm**.

1. Set a *breakpoint* on each line that says "set a breakpoint on this line". You set a breakpoint by clicking immediately right of the line number in your program file in PyCharm.
2. Now right-click with your mouse on `debug_test.py` program and start your `Debug 'debug_test'`. Note that it will stop at the next breakpoint and you can resume the execution by clicking on the green arrow at the lower-left corner of PyCharm window. As you continue by clicking on the resume button (green arrow), see what it shows in the two lower windows labeled: **Frames** and **Variables**. Also watch the values displayed at the end of each line it executed.
3. When it is all done executing the program, you can click on the **Console** button (just above the **Frames** window) to see the result of your program.
4. Try to use this debugging capability of **PyCharm** often so that you will be able to use it when you are experiencing difficulty understanding/debugging your program. It will be particularly helpful when your program gets larger and more complex.

If you are interested in learning more about debugging, check out this tutorial: https://www.jetbrains.com/help/pycharm/part-1-debugging-python-code.html

2. Functions with conditionals

a. In this problem, you will write a function that uses `if-elif` conditional structures and the string method `s.endswith` to satisfy the following requirements:

1. Create a file named `fileext.py`. Define a function named `fileType`. This function takes one string parameter and decides whether it is a Python file, Java file, or something else and generates an appropriate output to the computer screen. A Python filename ends with `.py`; a Java filename ends with `.java`; if neither, then it contains something other than a Python or Java program.
2. Now write code that that takes a file name as user input and calls `fileType` to test your implementation.
3. Make sure your program outputs the correct value.

b. Create a file named `gpa.py`. Define a function named `gradePoint` that will take a string named `grade` as an argument. You can assume the string is one character long and is either ’A’, ’B’, ’C’, ’D’, or ’F’. The value returned by your function should be the point value of the `grade` passed in, where A is worth 4 points, B is worth 3, C is worth 2, D is worth 1, and F is 0.

```
>>> gradePoint('A')

4

>>> gradePoint ('B')

3
```

c. Modify `gradePoint` so it will handle + and - grades by adding or subtracting 0.3 points. For example, a B+ is worth 3.3 points, and a C- is 1.7 points.

```
>>> gradePoint('A-')

3.7

>>> gradePoint('B+')

3.3
```

Suggestion: You could just add a bunch of `elif` clauses to test each grade separately, but a simpler design is to use a call to `grade.startswith` to figure out the value of the letter grade, then use `grade.endswith` to see if you should add or subtract 0.3 points.

3. Submit `fileext.py` (2 points) and `gpa.py` (3 points) on Blackboard.