Python Programming

Homework 1

Instructor: Dr. Ionut Cardei

The purpose of this programming assignment is to:
1. practice writing Python programs that read input, compute a result, and displays it to the terminal
2. practice with control statements

Make sure you follow exactly all requirements listed in this document to get full credit.

Software Requirements

For this assignment we will use the **pylab** module. Installing this module manually is a bit tedious on Windows and Mac.
To keep it simple all students should download and install the **Anaconda Python 3.5 software** from https://www.continuum.io/downloads. This includes many of the modules we need in class and replaces the Python 3 installation we already have on the PC.
Be sure you pick the version (OS, 32 bit or a 64 bit version) based on your computer OS and CPU version. The whole software platform needs about 2 GB of hard disk space.
Before installing Anaconda make sure you remove the existing Python 3.x installation from your PC. For Windows users go to Control Panel / Programs and uninstall from there.

(For Linux it is much easier to install modules and one can avoid installing Anaconda. Read the note at the end of this file.)

From now on you will run IDLE by clicking on the **C:\Program Files\Anaconda3\Lib\idlelib\idle.bat** icon. Put a shortcut to this file on the desktop to simplify access to IDLE.

Don't forget to read the **Important Notes** at the end of this file.

Homework 1 has three problems.

Problem 1. Salary Computation

Write a Python program called **p1.py** that computes the salary for a worker that is paid weekly.

The **p1** program starts by displaying some instructions to the user.
Then it reads from the terminal (in this order!) the number of hours worked in a week (variable hours_worked, to convert to float), the hourly salary (rate_per_hour, float), and whether the worker has received a bonus (variable yes_no). If yes_no is 'y', then the program reads the bonus (in variable bonus, also converted to a float).

The program computes the total salary as the sum of the overtime salary, the non-overtime salary, and the bonus. At the end, the program displays the total salary and the overtime pay.

Any hours worked in a week above 40 will be paid with a salary rate that is 1.5 times the regular hourly rate.

Thus, if the number of hours worked is 45, the hour rate is $10/hour, and the bonus is $18, the overtime pay is (45 - 40) * 10 * 1.5 = 75, the non-overtime pay is 40 * 10 = 400, and the total salary is 75+400+18=$493.

With these parameters, the user session generates this output to the terminal, as follows. User input is shown in yellow and the ↵ character indicates the ENTER key typed by the user.

```
This program determines the weekly salary for an employee.
The salary is the sum of the hourly rate times the
hours worked, plus the bonus.
For work hours exceeding 40 per week, an overtime rate
of 1.5 is applied.
The user must indicate if the worker has received a
bonus by answering a y/n question.
Input consists of: hours worked, hourly rate, bonus.
The output is the total salary for this week.

Enter the number of hours worked this week: 45↵
Enter the salary rate per hour (do not include the '$' sign): 10↵
Did the worker get a bonus ? (y/n) y↵
Enter bonus: 18↵
The total salary is $493.00 (overtime pay $75.00)
```

Here is another example of the program's execution, with no overtime and no bonus:

```
This program determines the weekly salary for an employee.
The salary is the sum of the hourly rate times the
hours worked, plus the bonus.
For work hours exceeding 40 per week, an overtime rate
of 1.5 is applied.
The user must indicate if the worker has received a
bonus by answering a y/n question.
Input consists of: hours worked, hourly rate, bonus.
The output is the total salary for this week.

Enter the number of hours worked this week: 40↵
Enter the salary rate per hour (do not include the '$' sign): 20↵
Did the worker get a bonus ? (y/n) n↵
The total salary is $800.00 (overtime pay $0.00)
```

Instructions for Problem 1:

1. Start by understanding what the problem wants.
2. Write the code in a file **p1.py** using the IDLE program or an editor of your choice.

3. Use the input() Python function to read user input. Convert from string type to float type as needed.
4. Assume the user enters valid input, i.e. a number where one is expected. The program does not have to deal with error handling. (we do that in a later unit)
5. Ensure the program works correctly when run from IDLE. Test a few times.
6. Take a screenshot of the program running in IDLE, with all output displayed. On Windows, use the PrntScrn key or similar. For help, check out http://www.digitalcitizen.life/4-ways-take-screenshots-windows-8-81-using-built-tools
7. Create a Word document called **h1.doc**. Write the title "**Homework 1**", your full name, and the heading **"Problem 1**". Insert the content of the completed **p1.py** program and then insert the screenshot in the doc file.

   Although not required, the graders love **syntax highlighting** as it makes reading and understanding code much easier. Instead of simply copy-pasting from IDLE to Word or just inserting the Python file into the Word document, you could use http://www.planetb.ca/syntax-highlight-word or https://tohtml.com/python/ to generate the HTML text for the Python source and then copy-paste from the browser into the **h1.doc** file.

Problem 2. Quadratic Equations

A quadratic equation has the form:

$$a x^2 + b x + c = 0$$

The two solutions are given by the formula:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (1)$$

Write a program with a loop that :
a) solves quadratic equations with coefficients read from the terminal,
b) visualizes the corresponding quadratic function $y = ax^2 + bx + c$ using the ***pylab*** module.

Instructions
Write all code for this problem in a file **p2.py**.

1. The program consists of a main *while* loop (an infinite loop) in which the user is prompted to enter values for coefficients ***a***, ***b***, and ***c***. Assume the user types valid float numbers from the terminal. The program converts the input to float type and then uses formula (1) above to compute solutions x_1 and x_2, as follows:

a) if $b^2 - 4ac < 0$ then the solutions are complex numbers (i.e. not real) and the program displays the string "no real solutions",
b) if $b^2 - 4ac = 0$ then $x_1 = x_2$ and the program displays: "one solution: " followed by the value x_1.
c) if $b^2 - 4ac > 0$ then the solutions are distinct and the program displays "two solutions: " followed by the values of x_1 and x_2.

To keep the problem simple we can assume that the user **never** enters a value for **coefficient *a*** that is equal to 0.
To stop the loop and to end the program, the user types the enf-of-file key – CTRL-Z on Windows (CTRL-D on Linux/Mac/Unix) – when expected to enter coefficient ***a*** for a new iteration.

2. Within the main loop, after displaying the values of the real solutions (if any) the program must display the graphic of the quadratic function $y = ax^2 + bx + c$ on the [-5, 5] domain.
For that, use the ***pylab*** Python module.
Within a nested *while* loop populate a list of floats in variable **xs** with 100 float numbers between -5 and +5 as shown in the lecture and append to a list **ys** the matching function values:

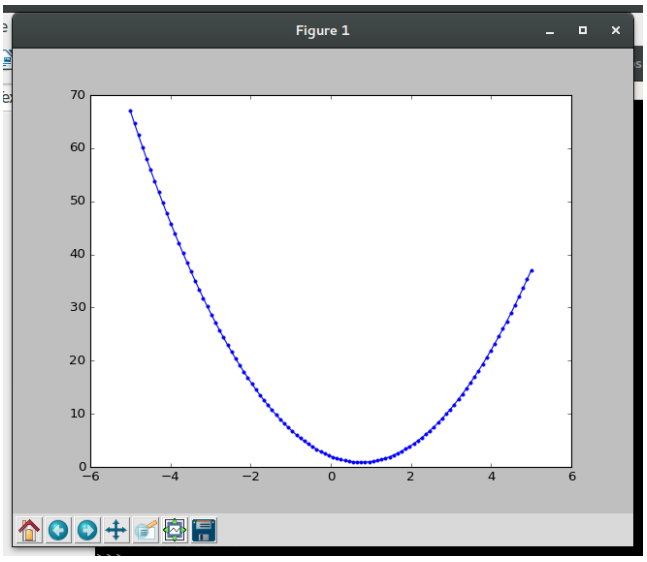
```
for each x in xs,
        append in ys the value of expression a*x*x + b * x + c.
```

(As an alternative to a loop, you could use the *pylab.linspace*() function to generate the *x* values.)

Then, use the ***pylab.plot***() function to generate the plot and the ***pylab.show***() function to display the graphic figure. Make sure the function line is displayed with a **blue line and dots**.

3. Insert the source code from **p2.py** into the h1.doc Word document. (use syntax highlighting if desired – remember, this formatting is not required)

4. Take a screenshot of the PC desktop showing the IDLE Python shell and the pylab figure window and paste it in the **h1.doc** file after the p2.py source code.
The screenshot may look similar to this one:



Here is a sample user session with the program:

```
Enter a: 1↵
Enter b: 2↵
Enter c: 1↵
one solution: -1.00000

Enter a: 3↵
Enter b: 0↵
Enter c: 1↵
no real solutions

Enter a: 1↵
Enter b: -1↵
Enter c: -6↵
two solutions: x1=-2.00000 x2=3.00000

(user types CTRL-Z on Windows to enter EOF and finish the program)
```

Problem 3. Computing Change (includes 10 extra credit points)

Write a program **p3.py** that computes the equivalent of a dollar amount in change using quarters, dimes, and pennies. **No nickels** are used for conversion.

The program reads from the terminal the dollar amount in a loop while not end-of-file. Inside the loop it computes how many quarters, dimes, and pennies make up the original dollar amount and then displays the change. The program should terminate if the user types an invalid string.

Here is a sample user session, with input highlighted in yellow:

```
Enter amount: 10↵
$10 makes 40 quarters, 0 dimes, and 0 pennies (40 coins), total amount in coins: $10.
Enter amount: 0.24↵
$0.24 makes 0 quarters, 2 dimes, and 4 pennies (6 coins), total amount in coins: $0.24.
Enter amount: 0↵
$0 makes 0 quarters, 0 dimes, and 0 pennies (0 coins), total amount in coins: $0.
Enter amount: 99.99↵
$99.99 makes 399 quarters, 2 dimes, and 4 pennies (405 coins), total amount in coins: $99.99.
Enter amount: 3.45↵
$3.45 makes 13 quarters, 2 dimes, and 0 pennies (15 coins), total amount in coins: $3.45.
Enter amount: 5.25↵
$5.25 makes 21 quarters, 0 dimes, and 0 pennies (21 coins), total amount in coins: $5.25.
-10.12↵
Invalid input.

(user types CTRL-Z on Windows to enter EOF and finish the program)
```

Instructions.

1. Write a heading "**Problem 3**" in file **h1.doc**.

2. Write the algorithm for this program in file **h1.doc**.

3. After that, write the **p3.py** program. Test it a few times.

4. Copy-paste source file p3.c to the **h1.doc** file, with syntax highlighting if possible.

5. Take a screenshot of the *p3* program **after it completed**, showing **at least 5** test cases, and with all output displayed. Make sure the IDLE (or python shell) window is visible, as we did before. Insert/paste the screenshot into the **h1.doc** file.

Submission Instructions

Convert the **h1.doc** file to PDF format (file **h1.pdf**) and upload the following files on Canvas by clicking on the Homework 1 link:

1. h1.pdf
2. p1.py
3. p2.py
4. p3.py

Grading:

Problem 1: 35%

- code correctness: 25%
- programming style: 5%
- screenshot showing the program running correctly: 5%

Problem 2: 35%

- code correctness: 25%
- programming style: 5%
- screenshot showing the program running correctly: 5%

Problem 3: 40% (10% extra credit !)

- algorithm: 5%
- code correctness: 25%
- programming style: 5%

- screenshot showing the program running correctly: 5%

IMPORTANT NOTES:

- A submission that does not follow the instructions 100% (i.e. perfectly) will not get full credit.
- Submit the **h1.pdf** PDF file and source files **p1.py**, **p2.py**, **p3.py**.
- Only submissions uploaded before the deadline will be graded.
- You have unlimited attempts to upload this assignment, but only the last one will be graded.

Alternative Instructions for Installing the Pylab Module in an Existing Python3 Installation, without Anaconda

This works for Linux and possibly Mac, but not on Windows.

The official name of this module is **matplotlib**. Its webpage is at http://matplotlib.org/.

Ubuntu Linux (and other Debian distributions) users:

```
sudo apt-get install python3-matplotlib
```

Fedora Linux (and similar) users:

```
sudo dnf install python3-matplotlib
```

or

```
sudo yum install python3-matplotlib
```