Programming Assignment 03 - While loops

Instructions

This programming assignment consists of **2 programming exercises**.

You have to:

1. **create** Python files on your computer (be careful of filenames)
2. **edit** them according to the assignment
3. **verify** on your computer that it works (run them and check the output in the shell)
4. **upload** the files to Gradescope (upload directly the `.py` files, **not** a `.zip` file)
5. **check** the auto-grader report on Gradescope
6. **go back to step 2** if necessary

The auto-grader will evaluate your code for a few test-cases. If some test-cases fail, the auto-grader should show you what is your code output, and what is the expected output.

The auto-grader will give you a score based on the test-cases, but a grader will manually evaluate your coding style after the deadline. Style represents 30% of the coding assignment grade.

Note:
The use of concepts that were not yet covered in class is prohibited.

Exercise 1 - Checking user input

Write a program (in the file `exercise1.py`) that does the following:

- **continuously** asks the user to **input a score** which is an integer between 0 and 100 (inclusive)...
- ...**until** the user enters a correct number (**display a message** if the user did not enter a correct number. see sample examples)
- then finally **prints the corresponding grade**

Table of correspondences score/grade:

Grade	Minimum score (inclusive)
A	95
A-	90
B+	87
B	83
B-	80
C+	77
C	73
C-	70
D+	67
D	63
F	0

Sample example 1 (the user input is in red, the printed output is in blue):

```
Score: 135
Invalid input. Please enter a score between 0 and 100.
Score: -25
Invalid input. Please enter a score between 0 and 100.
Score: 100
Your grade is A
```

Sample example 2:

```
Score: 87
Your grade is B+
```

Sample example 3:

```
Score: 101
Invalid input. Please enter a score between 0 and 100.
Score: 71
Your grade is C-
```

Exercise 2 - Square Root

Write a program (in the file `exercise2.py`) that does the following:

1. asks the user to **input a positive value** X
2. then, **prints the approximated value** Y **of the square root of** X according to Newton's method, up to the third decimal point

Newton's method for square root

Algorithm for estimating $Y \approx \sqrt{X}$:

1. Initialize Y to $X/2$
2. Loop until Y^2 is *close enough* to X:
   - Update Y with the **average** of Y and X/Y
   - Stop when Y^2 distance to X is less than 0.001

Example for $X = 10$

Steps for estimating $\sqrt{10}$:

1. $Y = \mathbf{5}$
   $Y^2 - X = 15 \rightarrow$ **need update**
2. $Y = (5 + 10/5)/2 = \mathbf{3.5}$
   $Y^2 - X = 2.25 \rightarrow$ **need update**
3. $Y = (3.5 + 10/3.5)/2 = \mathbf{3.178571428571429}$
   $Y^2 - X = 0.10331632653061362 \rightarrow$ **need update**
4. $Y = \mathbf{3.162319422150883}$
   $Y^2 - X = 0.00026412771269335167 \rightarrow$ **can stop**

Result: $\sqrt{10} \approx 3.162$

Restrictions

You are not allowed to use the `math` module.
You have to implement the Newton's method detailed above.

Sample examples (the user input is in red, the printed output is in blue, and the prompt is in black):

```
Enter a number: 78
Square root: 8.832
```

```
Enter a number: 12.3
Square root: 3.507
```

```
Enter a number: 9
Square root: 3.000
```