ENGR 13300 – HW6 PY2
Python 2: User-Defined Functions and Conditional Statements

Individual Tasks

Guidelines for Tasks 6–7:

1. These two tasks are individual assignments. You may seek help from classmates, the instructional team or others but the work you submit should be your own. If you collaborate with others and use information developed together or by someone else, ALWAYS document and reference that material.
2. Each individual is responsible for submitting their own individual assignment to Gradescope.

Task 6 (of 7) [Individual] - Autograded

Learning Objectives: Create and implement user-defined functions in Python; Construct logic operations using comparison, membership, and logical operators to generate control flow statements in Python; Utilize conditional if-elif-else statements while programming in Python, as well as previously learned objectives such as: Output data from a function to the screen in Python; Apply course code standard in development of Python scripts; Modularize and comment code in Python for readability and reusability.

Background:
You have graduated from Purdue and are now working as an engineer for Photon Optics, Inc. You have been given the task of modeling a simple light control system using experimental optical media for a new medical imaging method. The model will determine the behavior of light passing through two optical media. Your team needs you to code a proof-of-concept model for a ray-tracing program using Python and develop a corresponding flow diagram as a working document for your team to understand the program. The model will later be integrated within a larger Python program modeling the imaging behavior. For this proof-of-concept, you only have to deal with one ray of light and two optical media that are not air. You have decided to use Snell's Law which should be accurate enough for this proof-of-concept.

Snell's Law states that the behavior of a ray of light as it passes through to another medium is a function of the indices of refraction of the media (n_1 and n_2), and the angles with which the light enters or leaves each medium (θ_1 and θ_2):

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \qquad \text{(Equation 1)}$$

You draw a diagram indicating a ray of light that starts in medium 1, passes the interface and then continues in medium 2 if there is no total internal reflection. You come up with something like this:

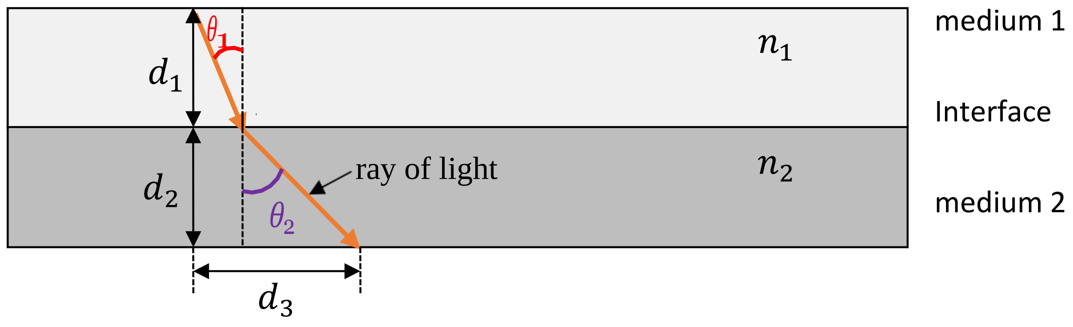

Figure 1. *Diagram of optical media.*

Relating this to Snell's Law, you see that the angle exiting an optical interface (where one medium meets another) can be found from Equation 1, as long as you know the incoming angle and the two indices of refraction. Additionally, you notice immediately a situation in which an error (hint: arithmetic error) could occur in finding the angle leaving the interface. This situation appears when light leaves a medium with a higher refractive index into a medium with a lower refractive index, which might be reflected back instead of refracting through the second medium when the incoming angle reaches a certain value (the largest incoming angle before this error occurs is called the *critical angle*); this situation is defined as total internal reflection.
In simpler words, if n1>n2, there exists a critical angle (θ_c) and if the incoming angle (θ_1) goes beyond the critical angle, Total Internal Reflection (TIR) takes place.

$$n_1 \sin\theta_c = n_2 \sin 90^{\circ} \qquad \text{(Equation 2)}$$

For your application, you need to be able to handle one optical interface with two media as described in Figure 1. You need to determine if there is refraction or total internal reflection assuming n_2, d_1 and d_2 are given and θ_1 and n_1 are inputs. Based on this determination, your program should display a message indicating the behavior of the light ray. Also, when appropriate, the program should output the second angle θ_2 , the ending distance d_3 where the light ray will fall, and the critical angle (θ_c). All outputs must be formatted so that only one decimal is shown.

$$d_3 = d_1 \tan\theta_1 + d_2 \tan\theta_2 \qquad \text{(Equation 3)}$$

Save a flowchart indicating the logic needed for the task in a PDF file and write a Python script that implements the logic in the flowchart.

You must make use of the following three user-defined functions:

1. `calc_crit_angle(n1, n2)` to calculate and return the critical angle θ_c in degrees
2. `calc_exit_angle(n1, n2, theta_in)` to calculate and return the angle θ_2 in degrees
3. `calc_dist(d1, theta_1, d2, theta_2)` to calculate and return the distance d_3 in cm

The angle parameters should also be assumed to be in degrees. You must ask the user for the incoming angle, θ_1, and the refractive index of medium 1, n_1. Use the `input()` function to do so (don't use the input() function for anything else).

Let $n_2 = 1.44$, $d_1 = 3.8$ cm and $d_2 = 9.1$ cm. Initialize these values in the initialization part (in the beginning) of your script.

Example 1: Assuming the following input values, $\theta_1 = 22.0°$, $n_1 = 1.0$, the outputs of your flowchart and your Python program should give the following message:

```
Enter the incoming angle in degrees: 22

Enter the refractive index of medium 1 [unitless]: 1
The light ray refracts with a leaving angle of 15.1°.
The transverse distance traveled by the light ray is 4.0 cm.
```

Example 2: Assuming the following input values, $\theta_1 = 45°$, $n_1 = 2.4$, the outputs of your flowchart and your Python program should give the following message:

```
Enter the incoming angle in degrees: 45

Enter the refractive index of medium 1 [unitless]: 2.4
For these two media, the critical angle is 36.9°.
The incoming angle is larger than the critical angle.
This results in total internal reflection.
```

Example 3: Assuming the following input values, $\theta_1 = 70°$, $n_1 = 1.5$, the outputs of your flowchart and your Python program should give the following message:

```
Enter the incoming angle in degrees: 70

Enter the refractive index of medium 1 [unitless]: 1.5
For these two media, the critical angle is 73.7°.
The light ray refracts with a leaving angle of 78.2°.
The transverse distance traveled by the light ray is 54.0 cm.
```

Hint: You can print the degree symbol by including "\u00B0" in a string.

Task 6 Files:

1. `Py2_Task6_`*`username`*`.py`
2. `Py2_Ind_`*`username`*`.pdf`

Task 7 (of 7) [Individual]

Learning Objectives: Create and implement user-defined functions in Python; Construct logic operations using comparison, membership, and logical operators to generate control flow statements in Python; Utilize conditional if-elif-else statements while programming in Python; as well as previously learned objectives such as: Output data from a function to the screen in Python; Apply course code standard in development of Python scripts; Modularize and comment code in Python for readability and reusability.

Background:

At your first co-op rotation with Retro Labs, you work on a vibration absorbing system using compressed gas. Your first assignment requires the analysis of a non-ideal gas (Carbon Dioxide) under pressure subject to various temperature conditions. Your goal is to maintain the pressure in the safe zone by regulating the temperature. You decide to apply your programming skills to write a program to find the values of pressure and temperature.

From your notes, you find the *van der Waals* equation of state, written below, where P is the pressure of the gas [atm], V_m is the molar volume of the gas [L/mol], R = 0.0820573661 [L · atm/(K · mol)] is the universal gas constant, T is the temperature [K]; a [L^2 · atm/mol^2], and b [L/mol] are constants:

$$P = \frac{RT}{V_m - b} - \frac{a}{V_m^2}$$

Task:

Begin by creating a flow diagram (include it in your previously created PDF file). Then, write a Python program to solve the task. a and b are given and can be assumed to be constant for this problem. An initial value of T and the molar volume V_m are user inputs (use the `input()` function). Define a function to calculate pressure using the above equation, given the temperature and other constants. Also define another function that uses a rearranged form of the above equation, that finds a new temperature, given the pressure and other constants. These functions must be user-defined functions named pressure and temperature and <u>must</u> be saved in a separate Python module (filename must be `Py2_Task7_functions_`*`username`*`.py`).

In the main Python file, `Py2_Task7_`*`username`*`.py`, the program must first calculate Pressure given the inputs. Your program should print on screen the initial conditions (V_m, a, b and T) and the resulting pressure. If the pressure is out of the acceptable range ([30,35]atm), the program should find the smallest magnitude increment/decrement of temperature in order to have an acceptable value of pressure. Only if the pressure is outside the range, should your program print the following information: the value of the required increment/decrement of temperature to bring the pressure into range, the resulting new temperature, and the corresponding new pressure. Present your answers in a professional way.

Additionally, from your previous experience in this class, you need to be careful to make sure your code checks for possible errors (Hint: Are there any values that can make the Pressure to 'blow up'? such as 100 atm or more). In this case, print "Error:999" with a brief description of what the problem is and afterwards don't do further calculations/prints.

Example 1: Assuming the following values: a = 3.59 [$L^2 \cdot$ atm / mol^2], b = 0.0427 [L/mol], and following inputs: $T = 300$ [K], and $V_m = 1.23$ [L/mol], the outputs of your flowchart and your Python program should give the following message:

```
Input Initial Temperature in Kelvin: 300

Input molar volume in L/mol: 1.23

Initial conditions:
Molar volume = 1.23 [L/mol]
Initial temperature = 300.0 [K]
Parameter a = 3.59 [L^2*atm/(mol^2)]
Parameter b = 0.0427 [L/mol]
Initial resulting pressure = 18.3608 [atm]

Required temperature increment for in-range pressure = 168.41 [K]
New temperature = 468.41 [K]
New pressure = 30 [atm]
```

Example 2: Assuming the following values: a = 3.59 [$L^2 \cdot$ atm / mol^2], b = 0.0427 [L/mol], and following inputs: $T = 500$ [K], and $V_m = 1.3$ [L/mol], the outputs of your flowchart and your Python program should give the following message:

```
Input Initial Temperature in Kelvin: 500

Input molar volume in L/mol: 1.3

Initial conditions:
Molar volume = 1.3 [L/mol]
Initial temperature = 500.0 [K]
Parameter a = 3.59 [L^2*atm/(mol^2)]
Parameter b = 0.0427 [L/mol]
Resulting pressure = 30.5081 [atm]

The pressure is within the acceptable range
```

Example 3: Assuming the following values: a = 3.59 [$L^2 \cdot$ atm / mol^2], b = 0.0427 [L/mol], and following inputs: $T = 700$ [K], and $V_m = 1.5$ [L/mol], the outputs of your flowchart and your Python program should give the following message:

```
Input Initial Temperature in Kelvin: 700

Input molar volume in L/mol: 1.5

Initial conditions:
Molar volume = 1.5 [L/mol]
Initial temperature = 700.0 [K]
Parameter a = 3.59 [L^2*atm/(mol^2)]
Parameter b = 0.0427 [L/mol]
Initial resulting pressure = 37.8199 [atm]

Required temperature increment for in-range pressure = -50.08 [K]
New temperature = 649.92 [K]
New pressure = 35 [atm]
```

Example 4: Assuming the following values: a = 3.59 [$L^2 \cdot$ atm / mol^2], b = 0.0427 [L/mol], and following inputs: $T = 700$ [K], and $V_m = 0.5$ [L/mol], the outputs of your flowchart and your Python program should give the following message:

```
Input Initial Temperature in Kelvin: 700

Input molar volume in L/mol: 0.5

Pressure is 111.2472 [atm]

Error:999 - pressure at unacceptable levels. Danger imminent! Sound the
alarm and run away to safety.
```

Task 7 Files:
1. `Py2_Task7_`*`username`*`.py`
2. `Py2_Task7_functions_`*`username`*`.py`
3. `Py2_Ind_`*`username`*`.pdf`

Summary of files to submit for HW6 PY2 - Team
1. `Py2_Team_`*`teamnumber`*`.pdf`
2. `Py2_Task1_`*`teamnumber`*`.py`
3. `Py2_Task2_`*`teamnumber`*`.py`
4. `Py2_Task3_`*`teamnumber`*`.py`
5. `Py2_Task4_`*`teamnumber`*`.py`
6. `Py2_Task4_areas_`*`teamnumber`*`.py`
7. `Py2_Task5_`*`teamnumber`*`.py`
8. `Py2_Task5_discriminant_`*`teamnumber`*`.py`

Summary of files to submit for HW6 PY2 - Ind
1. `Py2_Ind_`*`username`*`.pdf`
2. `Py2_Task6_`*`username`*`.py`
3. `Py2_Task7_`*`username`*`.py`
4. `Py2_Task7_functions_`*`username`*`.py`