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

This assignment has three sections.

I) The first section has two Team Exercises (A and B) that are intended to be done with your team without actually using Python. Work the problems together and answer the questions and present your answer to one of the instructional team. After you find the correct answers to both, you can move to the second section

II) The second section has five team tasks (Tasks 1-5) that you may work together and develop code together as long as you document who contributed.

III) The third section has two individual tasks (Tasks 6-7). You can work together to conceptually help each other but each person should work on their own code.

Section I

These are to be worked together as a team. There are no files to be uploaded to Brightspace. Rather, work the problems and come up with the answers and present to a member of the instructional team. After you get the right answers, move on to the next problem. The intent is to understand the concepts before we begin to write code ourselves.

Team Exercise A (if-elif-else)

The Python code on the right is typed in after defining x and y as shown below in the first two lines. The values shown below replace the **** in the code. Assume that all other variables are cleared before the commands are executed

```
x= ***
y = ***
if x<=50:
    z=4
    if y<30:
        z=x*y
    elif y>=100:
        z=x+y

elif y >= 60:
    if x > 80:
        z=x
    elif  y > 50:
        z=y
else:
        z=z*2

print('z=', z)
```

A) What will appear on the screen when the following Python commands executed along with the code to the right?
>> x = 40
>> y = 40

B) What will appear on the screen when the following Python commands executed along with the code to the right?
>> x = 20
>> y = 20

C) What will appear on the screen when the following Python commands executed along with the code to the right?
>> x = 55
>> y = 20

Team Exercise B (functions)

A Python program was run with the following commands. What is printed to the console window?

```
# ----------------------------------------------------------------
def myFun1(in1, in2):
   x=in1
   y=in2
   out1 = 6
   out1,out2 =myFun2(x,y)
   print(f'in1={in1:.0f}, in2={in2:.0f}, out1={out1:.0f}')
   return out1
# ----------------------------------------------------------------
def  myFun2(in2,in1):
   a=in1*3
   b=in2*2
   out1=a+b
   out2=a-b
   print(f'in1={in1:.0f}, in2={in2:.0f}, out1={out1:.0f}, out2={out2:.0f}')
   return out2,out1
# ----------------------------------------------------------------
def myFun3():
   var1=1
   var2=2
   out_put = 3
   print(f'var1={var1:.0f}, var2={var2:.0f}, out_put={out_put:.0f}')
   out_put = myFun1(var1,var2)
   print(f'var1={var1:.0f}, var2={var2:.0f}, out_put={out_put:.0f}')

#----------------------------------------------------------------
var1 = 4
var2 = 2
myFun3()
```

What variables appear in the console window after the pervious program is run and what are their values?

Section II – Team tasks

Recall the guidelines for team activities:

1. You should work as a team; **all** team members will be held responsible for all material. You may work together and contribute to one program and submit similar codes as long as the contributors to the development of the solution are documented.
2. Each student is responsible for submitting their own assignment.

Task 1 (of 7) [Team]

Learning Objectives: Construct logic operations using comparison, membership and logical operators to generate control flow statements in Python; Predict the output of a complete if-elif-else statement in Python; Utilize conditional if-elif-else statements while programming in Python;

Computer Operator: Team member carrying the most writing instruments.

Background:

Programs can make decisions based on rules provided by the user. These rules apply to the current state of values known by the program. Therefore, simple first order logic can be performed by making comparisons between values. Conditional statements, such as if-elif-else, can be used to make decisions, while combinations of rules can be used to make more complex decisions.

Part A:

Consider the following logic operations:

1. A and B
2. A or B
3. A == B
4. (A and B) == (A or B)
5. A != B
6. A > B
7. A ≥ B
8. A = B

In a PDF document, create a table with three columns: logic operations, hand calculation, and python calculation. Name this PDF document: `Py2_Team_`*`teamnumber`*`.pdf`

Assume that A = True and B = False. In many programming languages, including Python, true and false are equivalent to 1 and 0 respectively. In these comparisons you may assume this to be true. Without using Python, determine the value of each logic operation by hand and record the results in the second column.

Part B:

Using the same assumed values for A and B, perform the same logic operations in the Python 3 interpreter using the command line prompt. Save the results in the third column of the table created in

Include the answer to the following question in your previously created PDF file:

- Explain why the answers from Part A, if any, are different when computed in Python.

Part C:

You are working in an ice-cream making factory as a manufacturing engineer. Today, you are writing a program for the product quality testing equipment to run on the data gained from instruments on the ice-

cream making machines. Develop an algorithm - in the form of a flow diagram - that makes use of an if-elif-else structure for the decision table shown below.

Conditional statements (Temperature in °F, Hydration in %)	**Warning message**
(Temperature > 14) or (Hydration > 60)	Danger
(10 < Temperature < 14) and (55 < Hydration < 60)	Normal
Otherwise	Warning

Include your flow diagram in your previously created PDF file.

Include the answers to the following questions in the same PDF file:

1. What will be the output of your flow diagram for the following test cases?
    a. Temperature = 12.2°F and Hydration = 55.5%
    b. Temperature = 32°F and Hydration = 57.5%
    c. Temperature = 0°F and Hydration = 50%
2. Why is it important to be able to utilize logical, conditional or comparison operations in programs? Or are they completely useless? Justify your response.

Part D:

1. Once you have checked your algorithm, convert it into an appropriate Python program.
2. Name your program `Py2_Task1_`*`teamnumber`*`.py`
3. Verify its operation with the same test cases as in Part C
4. Record the results in the same PDF file

Your headers should be included in all programming files to indicate their authorship and date of last update. Thus, they are important aspects of coding. Practice including headers by using them in all coding documents (.py) in the rest of this assignment.

Task 1 Files:

1. `Py2_Team_`*`teamnumber`*`.pdf`
2. `Py2_Task1_`*`teamnumber`*`.py`

Task 2 (of 7) [Team]

Learning Objectives: Construct logic operations using comparison, membership and logical operators to generate control flow statements in Python; Predict the output of a complete if-elif-else statement in Python; Utilize conditional if-elif-else statements while programming in Python;
Computer Operator: Whoever has the longest hair, to the nearest inch.

Background: You are still a manufacturing engineer at an ice-cream making factory, and you have been asked to turn your flow diagram from **Task 1** into a program.

Part A: We want to extend the algorithm from **Task 1** and make more sophisticated decisions by using nested if-elif-else structures. Develop an algorithm, in the form of a flow diagram, that makes use of a nested if-elif-else structure for the decision tables shown below. Recall, the table represents temperatures in °F and hydration in %.

Conditional statements	**Warning message**
(Temperature > 14) or (hydration > 60)	Danger
(10 < Temperature < 14) and (55 < hydration < 60)	Normal
Otherwise	Warning

Nested conditional statements for "Danger"	**Warning message**
Temperature > 14	High Temperature
Otherwise	High hydration

Nested conditional statements for "Warning"	**Warning message**
Temperature < 10	Low Temperature
Otherwise	Low hydration

Example:
Assuming the following values, Temperature = 18°F and hydration = 58%, the outputs of your flow diagram should give the following 2 lines message:

```
Danger
High Temperature
```

Include your flow diagram in the previously created PDF file.

1. Include your answers to the following question in the same PDF (`Py2_Team_`*`teamnumber`*`.pdf`) file:
1. What will be the output of your flow diagram for the following test cases?
    a. Temperature = 12.2°F and Hydration = 65.5%
    b. Temperature = 32°F and Hydration = 57.5%
    c. Temperature = 0°F and Hydration = 50%

Part B: Once you have checked your algorithm, convert it into an appropriate Python program.

Name your program `Py2_Task2_`*`teamnumber`*`.py`

Verify its operation with the same test cases as in Part A, and record the results in the same PDF file as in **Part A.**

Task 2 Files:

2. `Py2_Team_`*`teamnumber`*`.pdf`
3. `Py2_Task2_`*`teamnumber`*`.py`

Task 3 (of 7) [Team]

Learning Objectives: Explain the advantages of user-defined functions in Python; Create and implement user-defined functions in Python; explain the similarities and differences between a main function and a called function.
Computer Operator: Person with the heaviest backpack

Background:
Python 3 is a powerful programming language for performing computations and database management. It contains several built-in functions commonly used for computations and mathematical operations. In addition, it offers the possibility to the user to define his/her own functions. This makes it easy for the programmer to generalize processes so they can be reused. In addition, it helps to modularize code, making it easier to read and troubleshoot.

Part A:
You are developing a new electric bike for a prominent automobile company, and your team has been tasked to write a Python code to calculate the blade sweep area required to power the bike and the distance that the bike can drive at a given speed and time. The wind power generated can be calculated using the equation given below:

$$\mathrm{P} = \frac{1}{2}\rho \mathrm{A} \mathrm{v}^3 \mathrm{C_p}$$

Where P is the power in W, ρ is the Air density in kg/m^3, A is the area of the blade sweep in m^2, v is the wind velocity in m/s, and $\mathrm{C_p}$is the unitless Power co-efficient. The average speed of the bike is 700 cm/s. The power required to drive the bike is $500\ W$. It is assumed that the wind turbine is located offshore and situated at sea-level, so, the air density is 1.23 kg/m^3. The wind velocity is 6 m/s. The $\mathrm{C_p}$ value is unique to every turbine and is a function of wind speed that the turbine is operating in; it is 0.4 in this case.

Write a Python function that calculates the area of the blade sweep of the wind turbine required to produce enough power to drive the bike, given the values for required power, air density, velocity of the wind, and power co-efficient takes as its arguments. The function should be named `windTurbineArea`.

Within the same .py file, write another python function that uses speed and a time as input, and computes the distance the bike can drive given those values. The function should be named `distanceBike`.

Finally, write a main program within the same .py file, that uses the functions that you have created to output both the distance and the solar panel surface area required. Define and assign the following values to the variables at the beginning of this main program,

$speed = 700\ cm/s$
$time = 2\ hours$
$P = 500\ W$
$\rho = 1.23\ kg/m^3$
$v = 6\ m/s$
$C_p = 0.4$

Name your program `Py2_Task3_`*`teamnumber`*`.py`

Part B:

Answer the following questions.

1. What Python function can you use to output variables to the screen?
2. What syntax differences exist between the two Python functions (`distanceBike` and `windTurbineArea`)?
3. What syntax differences exist in your Python program when calling these two functions?
4. What are the differences between a main program and a called function in Python? What are the advantages of user-defined functions in Python?

Include the answers from **Part B** in your previously created PDF document: `Py2_Team_`*`teamnumber`*`.pdf`

Task 3 Files:

1. `Py2_Team_`*`teamnumber`*`.pdf`
2. `Py2_Task3_`*`teamnumber`*`.py`

Task 4 (of 7) [Team]

Learning Objectives: Create and implement user-defined functions in Python; Describe the difference between local and global variables specifically with functions in Python.
Computer Operator: The person who ate the biggest breakfast

Background:
As an engineer, you may be working on modeling physical systems consisting of common shapes. You may need to analyze your design with basic shapes of the system or look for shapes in an image file. Computers can help make this process less tedious by automating the computation.

Part A:
Write the following code to define the area of a sphere given radius R:

```
from math import pi

def sphereArea(R):
     area = 4 * pi*R**2
     return area
```

Save it in a module (file) called `Py2_Task4_areas_`*`teamnumber`*`.py`

In a separate file stored in the **same** directory as `Areas.py`, write the following:

```
import Py2_Task4_areas_teamnumber
radius = 5
A = Py2_Task4_areas_teamnumber.sphereArea(radius)
print(A)
print(R)
```

Save this as `Py2_Task4_`*`teamnumber`*`.py`

Answer the following questions:

1. What happens when you run `Py2_Task4_`*`teamnumber`*`.py`?
2. What happens when you change line 5 of the Python file from `print(R)` to `print(radius)`?
3. What happens when you change line 4 of the Python file from `print(A)` to `print(area)`?
4. What happens when you add a line in your function such as `global area` just after the definition line and then change line 4 of the Python file from `print(A)` to `print(area)`?
5. What does it tell you about the difference between local and global variables?

Include the answers from **Part A** in the previously created PDF: `Py2_Team_`*`teamnumber`*`.pdf`

Part B:
Open Py2_Task4_areas_*teamnumber*.py and modify it with the following code.

```
from math import pi

def sphereArea(R):
     area = 4 * pi*R**2
     return area

def triangleArea(S1, S2):
     area = (1/2) * S1 * S2
     return area
```

In your main function, Py2_Task4_*teamnumber*.py, add line(s) of code to calculate the area of a triangle where the base has a length of 3 and height has a length of 6.

Include the answer to following question in your previously created PDF file:

1. Are S1 and S2 global or local variables?

Task 4 Files:

1. Py2_Task4_*teamnumber*.py
2. Py2_Team_*teamnumber*.pdf
3. Py2_Task4_areas_*teamnumber*.py

Task 5 (of 7) [Team]

Learning Objectives: Create and implement user-defined functions in Python; Utilize conditional if-elif-else statements while programming in Python.
Computer Operator: Whoever has the best (school-appropriate) one-liner joke.

Background:
As an engineer, you may need to solve cubic equations which are widely used in Elliptic Curve Cryptography. An example with which you may be familiar is Curve interpolation (cubic splines), a very important tool in solid modelling and graphics applications (otherwise known as CAD). This can be done by hand but is more easily done by a computer.

Part A:
Write a function `discriminant` that computes the discriminant of a third order cubic equation:

$$ax^3 + bx^2 + cx + d = 0$$

The equation to computer the discriminant:

$$dis = b^2c^2 - 4ac^3 - 4b^3d - 27a^2d^2 + 18abcd$$

Your function should take three inputs (`a, b, c, d`) and return the discriminant (`dis`). Save this in a module called `Py2_Task5_discriminant_`*`teamnumber`*`.py`

Part B:
In your main function (named `Py2_Task5_`*`teamnumber`*`.py`), write a piece of code that determines the number of real roots (real values of x that solve the equation) of a third order cubic equation. A cubic equation with real coefficients always has *at least* one real root. If the discriminant is positive, the cubic's roots are all real and distinct. If the discriminant is zero, the cubic has all real roots with at least two of the roots being the same. If the discriminant is negative, one root is real and the other roots occur in a conjugate pair of complex roots. For example, the equation $x^3 - 6x^2 + 11x - 6 = 0$ can be simplified to (x-1)(x-2)(x-3) = 0 with a discriminant of 4. Therefore, it has three real and distinct roots. So, an example of the program output is given below:

Example:

```
The inputs are:
a=1, b=-6, c=11, d=-6

Real and distinct three roots: True
Real with at least two equal roots: False
Conjugate pair of complex roots: False
```

Hint: use the function you created in **Part A** to compute the discriminant. Then, use the if-elif-else structure to determine the number of roots and to print the output.

Task 5 Files:
1. `Py2_Task5_`*`teamnumber`*`.py`
2. `Py2_Task5_discriminant_`*`teamnumber`*`.py`

Individual Tasks

Section III

Guidelines for Tasks 6–7:
These two tasks are to be completed individually. 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.

Task 6 (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:
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.

Looking back to your PHYS 241 notes, you recall that 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:

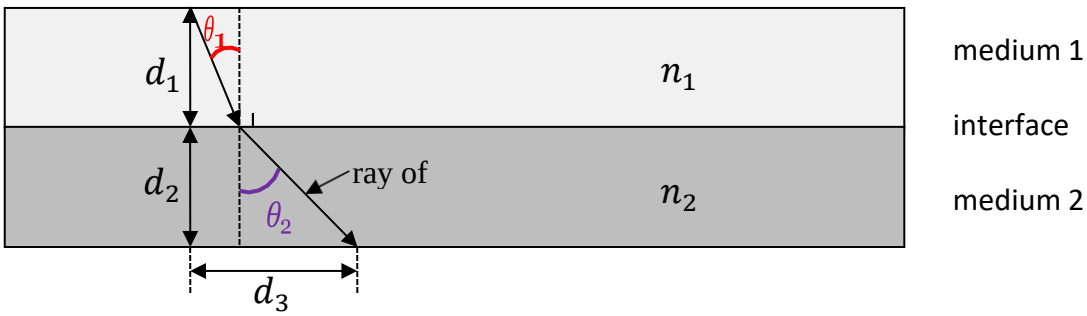

Figure 1. *Diagram of optical media.*

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.

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. All outputs must be formatted so that only one decimal is shown.

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 two different user-defined functions: one for calculating the angle θ_2 and another to calculate the distance d_3.

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 $n_2 = 1.3$, $d_1 = 3.8$ cm and $d_2 = 9.1$ cm. Initialize these values in the initialization part (in the beginning) of your script.

Example: Assuming the following values: $\theta_1 = 45°$, $n_1 = 1.5$
The outputs of your flowchart and your Python program should give the following message:

```
Input incoming angle [degrees]: 45.0
Input refractive index medium 1 [unitless]: 1.5
There is refraction with a leaving angle of 54.7
degrees. The ending distance for the light ray is
16.6 cm.
For these two media, the critical angle is 60.1 degrees.
```

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 under pressure subject to various temperature conditions. Your goal is to maintain the pressure in the interval [1.1, 1.2] atm 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 must 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 ([1.8, 2.2]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 Pressure 'blow up'?). In this case, print "Error" with a brief description of what the problem is and afterwards don't do further calculations/prints.

Example: Assuming the following values: a = 6.49 [$L^2 \cdot atm / mol^2$], and b = 0.0562 [L/mol].

The outputs of your flowchart and your Python program should give the following messages (note that the bold messages should only appear if a correction is necessary):

```
Input Initial Temperature in Kelvin: 300
Input molar volume in L/mol: 1.5

Initial conditions:
Molar volume = 1.5 [L/mol]
Initial temperature = 300.0 [K]
Parameter a = 6.49 [L^2*atm/(mol^2)]
Parameter b = 0.0562 [L/mol]
Resulting pressure = 14.1658 [atm]
Required temperature increment for in-range pressure = -228.13 [K]
New temperature = 71.87 [K]
New pressure = 1.2 [atm]
```

Task 7 Files:

1. Py2_Task7_*username*.py
2. Py2_Task7_functions_*username*.py
3. Py2_Ind_*username*.pdf

Submit Files:

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
9. Py2_Task6_*username*.py
10. Py2_Task7_*username*.py
11. Py2_Task7_functions_*username*.py
12. Py2_Ind_*username*.pdf