Engineering 14100
Python 2 ACT: User-Defined Functions and Conditional Statements

Recall the guidelines for activities:

1. You should work as a team; **all** team members will be held responsible for all material.
2. You should work on this Task using one computer for the entire team unless otherwise directed.
3. The time estimate given is approximately three times the amount required for an experienced user to complete the task. If you are not making progress, take action to get unstuck.
4. Do not write on the activity sheets and be sure to return them at the end of class.

Task 1 (of 5)

Objective: 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 to determine what to do next. 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:
If A = 1 and B = 0, what would be the result of 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, include a table with three columns: logic operations, hand calculation, and python calculation. Name this PDF document: `Py2_ACT_login.pdf`

Without using Python, answer the logic operations by hand and record the results in the second column.

Part B:
Perform the same logic operations in the Python 3 interpreter by using the command line prompt. Save the results in the third column of the table created in **Part A**.

Include the answers to the following questions in your previously created PDF file:

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

Part C:
You are working in a boiler 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 boilers. Develop an algorithm - in the form of a flow diagram - that makes use of a nested if-elif-else structure for the decision table shown below.

Conditional statements (Temperature in °C, Pressure in MPa)	**Warning message**
(Temperature > 1000) or (Pressure > 20)	Danger
(750 < Temperature < 1000) and (10 < Pressure < 20)	Normal
Otherwise	Warning

Include your flow diagram in your previously created PDF file: `Py2_ACT_login.pdf`

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 = 850°C and Pressure = 17MPa
    b. Temperature = 900°C and Pressure = 22MPa
    c. Temperature = 700°C and Pressure = 15MPa
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_ACT_Task1_login.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_ACT_login.pdf`
2. `Py2_ACT_Task1_login.py`

Task 2 (of 5)

Objective: 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 a boiler 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 °C and pressures in MPa.

Conditional statements	**Warning message**
(Temperature > 1000) or (Pressure > 20)	Danger
(750 < Temperature < 1000) and (10 < Pressure < 20)	Normal
Otherwise	Warning

Nested conditional statements for "Danger"	**Warning message**
Temperature > 1000	High Temperature
Otherwise	High Pressure

Nested conditional statements for "Warning"	**Warning message**
Temperature < 750	Low Temperature
Otherwise	Low Pressure

Example:

Assuming the following values, Temperature = 1050°C and Pressure = 18MPa, 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: `Py2_ACT_login.pdf`

Include your answers to the following question in the same PDF file:

1. What will be the output of your flow diagram for the following test cases:
    a. Temperature = 850°C and Pressure = 17MPa
    b. Temperature = 900°C and Pressure = 22MPa
    c. Temperature = 700°C and Pressure = 15MPa

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

Name your program `Py2_ACT_Task2_login.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:

1. `Py2_ACT_login.PDF`
2. `Py2_ACT_Task2_login.py`

Task 3 (of 5)

Objective: 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:
A new Martian rover is currently under development, and your team has been asked to write a Python code to calculate the solar cell surface area required to power the rover and the distance that the rover can drive at a given speed and time. The average speed of the rover is 1 cm/s. The solar cell efficiency equation is given below:

$$\eta = \frac{P}{E * A_c}$$

Where η is the efficiency in %, A_c is the area of the solar cell in m^2, P is the power in W and E is the solar irradiance of the planet surface in W/m^2. The power required to drive the rover is $100\ W$. The efficiency of the solar panel material is 0.25 (or 25%). The average solar irradiance at Mars is $590\ W/m^2$.

Write a Python function that computes the distance the rover can drive given a speed and time. The function should be named `distanceRover`.

Within the same .py file, write a Python function calculating the surface area of the solar panel required to produce enough power to drive the rover. The function should be named `solarPanelArea`.

Within this same .py file, write a Python program that creates the following variables:

$speed = 1\ cm/s$
$time = 2\ hours$
$P = 100\ W$
$E = 590\ W/m^2$
$\eta = 0.25$

This program should calculate the distance traveled by the rover and the solar panel surface area needed using the two functions you previously created. The program should output the distance and the solar panel surface area.

Name your program `Py2_ACT_Task3_`*`login`*`.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 (`distanceRover` and `solarPanelArea`)?
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_ACT_login.pdf`

Task 3 Files:

3. `Py2_ACT_login.pdf`
4. `Py2_ACT_Task3_`*`login`*`.py`
5. `Team_Header_Team`*`XX`*`.py`
6. `Individual_Header_`*`login`*`.py`

Task 4 (of 5)

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 circle given radius R:

```
def circArea(R):
      area = R**2
      return area
```

Save it in a module (file) called `Areas.py`

In a separate file write the following:

```
import Areas
radius = 5
A = Areas.circArea(radius)
print(A)
print(R)
```

Save this as `Py2_ACT_Task4_Main_`*`login`*`.py`

Answer the following questions:

1. What happens when you run `Py2_ACT_Task4_Main_`*`login`*`.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_ACT_`*`login`*`.pdf`

Part B

Open `Areas.py` and modify it with the following code.

```
def circArea(R):
      area = R**2
      return area

def rectArea(S1, S2):
      area = S1 * S2
      return area
```

In your main function, `PY2_ACT_Task4_Main_`*`login`*`.py`, add line(s) of code to calculate the area of a rectangle where side 1 has a length of 3 and side 2 has a length of 6.

Include the answers to following questions in your previously created PDF file:

1. Are S1 and S2 global or local variables?
2. What is the advantage or disadvantage of the method of calling variables used in the function `rectArea` compared with the method used in the function `circArea`?

Task 4 Files:

1) `Py2_ACT_Task4_Main_`*`login`*`.py`
2) `Py2_ACT_`*`login`*`.pdf`
3) `Areas.py`

Task 5 (of 5)

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 will often solve quadratic equations to find the intercepts of a parabola. 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 second order quadratic equation:

$$ax^2 + bx + c = 0$$

The equation of the discriminant:

$$dis = b^2 - 4ac$$

Your function should take three inputs (`a, b, c`) and return the discriminant (`dis`). Save this in a module called `Py2_ACT_Task5_Discriminant_`*`login`*`.py`

Part B:
In your main function (named `Py2_ACT_Task5_Main_`*`login`*`.py`), write a piece of code that determines the number of real roots (values of x that solve the equation) of a second order quadratic equation. If the discriminant is less than zero, there are no real roots. If it equals zero, there is one real root. If it is greater than zero, there are two real roots. For example, the equation $x^2 + 5x + 6 = 0$ can be simplified to (x+2)(x+3)=0 with a discriminant of 1. Therefore, it has two real roots. So an example of the program output is given below:

Example:

```
The inputs are:
a=1, b=5, c=6

No real roots: False
One real root: False
Two real roots: True
```

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_ACT_Task5_Main_`*`login`*`.py`
2) `Py2_ACT_Task5_Discriminant_`*`login`*`.py`

Bonus Activity Submissions

Instructions: Complete and submit **ALL** Task materials associated with this Activity (see 'Submit Files' below). You are allowed to combine the work you and your team completed during the Activity with materials you individually (or as a team) complete outside of class. The Bonus Activity Submission will not be graded and returned to you like a typical assignment. Instead, it will be reviewed, and the bonus point awarded, for its completeness, i.e., for completing ALL the Tasks associated with the Activity. Submitting an incomplete Bonus Activity (something less than all of the Tasks) will be considered an act of **Academic Dishonesty** for which the penalty will be forfeiture of the opportunity to earn future Bonus Activity Submission points.

There are two options for completing the materials for the Bonus Activity Submission:

As an Individual: Combine the work you and your team completed in class with materials you have individually completed outside of class. When submitting an individual Bonus Activity Submission you will append your electric signature (i.e., your typed name) at the top of the file that represents your individual work. Your electronic signature indicates that this is your individual work and you have not collaborated with other individuals (other than the teaching team) to obtain the final materials being submitted – working with other individuals/groups (e.g., discussing ideas and concepts, helping find errors, talking about potential solutions to errors) is permissible up to the point where the work represents a coloration (i.e., working with another person or group to achieve an answer). Any work previously completed by your team should include each team member's electronic signature. The significance of an electronic signature by an individual for team work is stated below.

As a Team or Ad Hoc Group: Combine the work you and your team completed in class with materials your team (or ad hoc group) completed outside of class **(For the Bonus Activity Submission ONLY**: you are allowed to work with any other members of the class to complete the assignment). However, you should exercise care when appending your electronic signature to ensure you are in full compliance). When submitting a Bonus Activity that has been worked on as a team (or ad hoc group) each person will append his/her electronic signature (i.e., his/her typed name) at the top of the file that represents the collaborative work. The electronic signature of each individual implies he/she: was an active participant in the preparation of the materials; and has a general understanding of **ALL** the materials being submitted. Even for work submitted as a team, each individual who wishes to receive credit must submit the team's file (with all appropriate signatures) to their own individual assignment drop box.

Submit Files: Submit *all* files electronically via Blackboard to the appropriate box on time.

1. `Py2_ACT_`*`login`*`.pdf`
2. `Py2_ACT_Task1_`*`login`*`.py`
3. `Py2_ACT_Task2_`*`login`*`.py`
4. `Py2_ACT_Task3_`*`login`*`.py`
5. `Team_Header_Team`*`XX`*`.py`
6. `Individual_Header_`*`login`*`.py`
7. `Py2_ACT_Task4_Main_`*`login`*`.py`
8. `Areas.py`
9. `Py2_ACT_Task5_Main_`*`login`*`.py`
10. `Py2_ACT_Task5_Discriminant_`*`login`*`.py`