Assignment 1

Posted on Saturday, September 28, 2024

Objective: The objective of these questions is to apply the knowledge of basic control structures such as **`if-else`** statements, **`while`** loops, **`for`** loops, and nested loops. These 4 exercises will help you understand the application of these programming constructs in solving simple real-life problems.

Late submission penalty

1. 10% is deducted for each day that the work is late. The penalty will be applied up to a maximum number of three days after and including the submission deadline day. After three days the work will be marked at zero.

2. **This assignment is individual work. All work must be done on your own. Plagiarism is serious offence. Copying code from web resources is prohibited as well. Any plagiarism case (for both the copier and the copiee) will be given a ZERO mark in this assignment.**

Instructions: Assignment I consist of FOUR questions. You have to **answer any TWO** questions or you can submit all FOUR solutions. Your final score will be based on the best two answers out of the four questions. The two highest-scoring answers will be used to calculate your final grade of assignment I.

---------- Submission Requirement ----------

1. Submit on time. Name the program of Question 1 by studentID_Q1.py in which studentID is your student ID. For example, if your student ID is 23456789D, name all the programs as **`23456789D_Q1.py`**, **`23456789D_Q2.py`** and so on.
2. Put all programs (as advised in point 1) into a folder named by your student ID (**`23456789D`**).
3. Zip the folder and submit the zip file (**`23456789D.zip`**) via the Blackboard.

4. **Warning: Any compilation error / runtime error will be led to 50% of deduction of points and partial points will be awarded depending upon the logic of what you have coded. Therefore, if you are unable to complete the whole program, try to accomplish part of the tasks and make sure it can be compiled and run successfully.**

5. **Any wrong file naming and submission will be given a ZERO mark in this assignment.** If you are using Windows, the file extension may be hidden by the operating system. Follow the steps of the link below to make sure the file extension is not hidden: https://www.howtohaven.com/system/show-file-extensions-in-windows-explorer.shtml

6. The deadline for this assignment is **23:59:00 Sunday, October 20, 2024**.

Question 1: Simulating Monthly Premium Calculations for Insurance Policies

Scenario: Imagine an insurance company that needs to calculate the monthly premiums for a set of insurance policies. Each policy has a base premium, and the final premium is adjusted based on the age of the policyholder. The goal is to simulate the calculation of the monthly premiums for a set of policyholders.

The premium for each policyholder will grow cumulatively based on an interest rate that depends on the age of the policyholder. The interest rate is higher for younger policyholders and elderlies, similar to how certain bonds work. This means that the premium will grow exponentially each month based on the monthly interest.

Problem Statement: You are required to write a program that simulates the monthly premium calculations for a set of insurance policies. The program should:

1. Initialize the base premium for each policy.
2. Use nested loops to update the premium for each policyholder based on their age.
3. Use if-else statements to determine the interest rate based on the age of the policyholder.
4. Print the premium for each policyholder at each month.
5. Use a while loop to simulate the premium calculations over a specified number of months.

Assumptions:

- There are 3 policyholders.
- The base premium for each policy is $100.
- The interest rates based on age are as follows:
  - If the policyholder is under 25, the interest rate is 10%.
  - If the policyholder is between 25 and 50, the interest rate is 5%.
  - If the policyholder is over 50, the interest rate is 7%.
- The simulation runs for 3 months.

Instructions (for reference):

1. Initialize an array representing the base premium for each policyholder.
2. Use nested loops to iterate through each policyholder.
3. Use if-else statements to apply age-based adjustments to the premiums.
4. Update the premium for each policyholder based on their age.
5. Print the premium for each policyholder at each month.
6. Use a while loop to simulate the premium calculations over 3 months.

Deliverables:

- A well-commented code that simulates the monthly premium calculations for the insurance policies.
- A brief explanation of how the code works.

Sample Output:

```
Enter age 1 between 0 and 100: 20
Enter age 2 between 0 and 100: 50
Enter age 3 between 0 and 100: 99
Month 1
Policyholder 1 Premium: 110.0
Policyholder 2 Premium: 105.0
Policyholder 3 Premium: 107.0
Month 2
Policyholder 1 Premium: 121.0
Policyholder 2 Premium: 110.25
Policyholder 3 Premium: 114.49
Month 3
Policyholder 1 Premium: 133.1
Policyholder 2 Premium: 115.76
Policyholder 3 Premium: 122.5
```

Question 2: Simulating Daily Sales and Revenue Calculation for Products

Scenario: Imagine a store that sells three products: A, B, and C. The prices of these products are $10, $20, and $30, respectively. The store records the number of units sold for each product every day over a week. The goal is to simulate the daily sales and calculate the total revenue generated from each product at the end of the week.

Problem Statement: You are required to write a program that simulates the daily sales of products and calculates the total revenue generated. The program should:

1. Initialize the prices of the products.
2. Use nested loops to update the number of units sold for each product each day.
3. Use if-else statements to determine the number of units sold based on the day of the week.
4. Print the sales for each product at the end of each day.
5. Use a while loop to simulate the daily sales over a week.
6. Calculate the total revenue generated from each product at the end of the week.
7. Print the total revenue for each product.

Assumptions:

- There are 7 days in a week.
- The prices of products A, B, and C are 10, 20, and $30, respectively.
- 5 units of each product are sold on even days.
- 3 units of each product are sold on odd days.

Instructions (for reference):

1. Initialize the prices of the products.
2. Use nested loops to iterate through each day and each product.
3. Use if-else statements to determine the number of units sold based on the day of the week.
4. Update the number of units sold for each product each day.
5. Print the sales for each product at the end of each day.
6. Use a while loop to simulate the daily sales over a week.
7. Calculate the total revenue generated from each product at the end of the week.
8. Print the total revenue for each product.

Deliverables:

- A well-commented code that simulates the daily sales and calculates the total revenue for the products.
- A brief explanation of how the code works.

Sample Output:

```
Enter number of days between 1 - 7: 7
Day 1
Product A Sales: 3
Product B Sales: 3
Product C Sales: 3

Day 2
Product A Sales: 5
Product B Sales: 5
Product C Sales: 5

Day 3
Product A Sales: 3
Product B Sales: 3
Product C Sales: 3

Day 4
Product A Sales: 5
Product B Sales: 5
Product C Sales: 5

Day 5
Product A Sales: 3
Product B Sales: 3
Product C Sales: 3

Day 6
Product A Sales: 5
Product B Sales: 5
Product C Sales: 5

Day 7
Product A Sales: 3
Product B Sales: 3
Product C Sales: 3

Total Revenue at the end of the week:
Product A Total Revenue: 270
Product B Total Revenue: 540
Product C Total Revenue: 810
```

Question 3: Simulating the Daily Temperature Changes in a Room

Scenario: Imagine a room where the temperature changes throughout the day. The room is divided into a 3x3 grid, where each cell represents a section of the room. The temperature in the room starts at a uniform value, but it changes based on the time of day. The goal is to simulate how the temperature changes in the room over a 24-hour period.

Problem Statement: You are required to write a program that simulates the temperature changes in the room. The program should:

1. Initialize the temperature of the room.
2. Use nested loops to update the temperature of each section of the room at each hour.
3. Use if-else statements to apply the temperature changes based on the time of day.
4. Print the temperature distribution in the room at each hour.
5. Use a while loop to simulate the temperature changes over a 24-hour period.

Assumptions:

- The room is a 3x3 grid.
- The initial temperature of the room is 20℃.
- The temperature increases by 1℃ every hour from 6 AM to 6 PM.
- The temperature decreases by 1℃ every hour from 6 PM to 6 AM.
- The simulation runs for 24 hours.

Instructions (for reference):

1. Initialize a 3x3 grid representing the temperature of the room.
2. Use nested loops to iterate through each section of the room.
3. Use if-else statements to apply the temperature changes based on the time of day.
4. Update the temperature of each section in the grid.
5. Print the temperature distribution in the room at each hour.
6. Use a while loop to simulate the temperature changes over 24 hours.

Deliverables:

- A well-commented code that simulates the temperature changes in the room.
- A brief explanation of how the code works.

Sample Output:

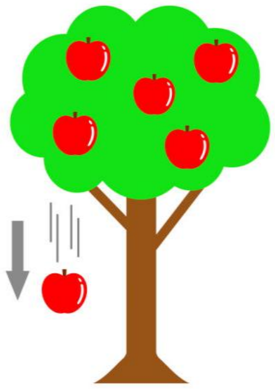
```
Enter the Room Temperature: 30
Enter the Number of Hours between
0-24: 3

Hour 0
29.0 29.0 29.0
29.0 29.0 29.0
29.0 29.0 29.0

Hour 1
28.0 28.0 28.0
28.0 28.0 28.0
28.0 28.0 28.0

Hour 2
27.0 27.0 27.0
27.0 27.0 27.0
27.0 27.0 27.0
```

Question 04: Simulating the Motion of a Particle in a Gravitational Field

Scenario: Imagine a particle is released from a certain height above the ground and is subject to gravitational acceleration. The particle falls freely under gravity until it hits the ground. For simplicity, assume that air resistance is negligible, and the only force acting on the particle is gravity.

Problem Statement: You are required to write a program that simulates the motion of the particle.

1. Initialize the height from which the particle is released.
2. Use a loop to update the position of the particle at each time step.
3. Use an if-else statement to check if the particle has hit the ground.
4. Print the height of the particle at each time step until it hits the ground.



Assumptions:

- Gravitational acceleration (g) is 9.8 m/s^2.
- The time step (Δt) for the simulation is 0.1 seconds and *t* is the current time.
- The initial height (h) from which the particle is released is 100 meters.
- Assume velocity and mass are ZERO (for ease of programming)

Instructions (for reference):

1. Initialize the height of the particle to 100 meters.
2. Use a while loop to simulate the motion of the particle until it hits the ground.
3. Inside the while loop, use an if-else statement to check if the particle has hit the ground.
4. Update the height of the particle using the equation of motion: $h = h_{initial} - 1/2 * g * (t)^2$.
5. Print the height of the particle at each time step.
6. Use a for loop to iterate through the time steps.

Deliverables:

- A well-commented code that simulates the motion of the particle.
- A brief explanation of how the code works.

Sample Output 1:

```
Enter the time step in seconds: 0.1
Enter the height in meters: 1

Time: 0.0 s, Height: 1.00 m
Time: 0.1 s, Height: 0.95 m
Time: 0.2 s, Height: 0.80 m
Time: 0.3 s, Height: 0.56 m
Time: 0.4 s, Height: 0.22 m
Time: 0.5 s, Height: 0.00 m

The particle has hit the ground.
```

Sample Output 2:

```
Enter the time step in seconds: 0.2
Enter the height in meters: 2

Time: 0.0 s, Height: 2.00 m
Time: 0.2 s, Height: 1.80 m
Time: 0.4 s, Height: 1.22 m
Time: 0.6 s, Height: 0.24 m
Time: 0.8 s, Height: 0.00 m

The particle has hit the ground.
```

Sample Output 3:

```
Enter the time step in seconds: 0.1
Enter the height in meters: 100

Time: 0.0 s, Height: 100.0 m
Time: 0.1 s, Height: 99.95 m
Time: 0.2 s, Height: 99.80 m
Time: 0.3 s, Height: 99.56 m
Time: 0.4 s, Height: 99.22 m
Time: 0.5 s, Height: 98.78 m
Time: 0.6 s, Height: 98.24 m
Time: 0.7 s, Height: 97.60 m
Time: 0.8 s, Height: 96.86 m
Time: 0.9 s, Height: 96.03 m
Time: 1.0 s, Height: 95.10 m
Time: 1.1 s, Height: 94.07 m
Time: 1.2 s, Height: 92.94 m
Time: 1.3 s, Height: 91.72 m
Time: 1.4 s, Height: 90.40 m
Time: 1.5 s, Height: 88.97 m
Time: 1.6 s, Height: 87.46 m
Time: 1.7 s, Height: 85.84 m
Time: 1.8 s, Height: 84.12 m
Time: 1.9 s, Height: 82.31 m
Time: 2.0 s, Height: 80.40 m
Time: 2.1 s, Height: 78.39 m
Time: 2.2 s, Height: 76.28 m
Time: 2.3 s, Height: 74.08 m
Time: 2.4 s, Height: 71.78 m
Time: 2.5 s, Height: 69.37 m
Time: 2.6 s, Height: 66.88 m
Time: 2.7 s, Height: 64.28 m
Time: 2.8 s, Height: 61.58 m
Time: 2.9 s, Height: 58.79 m
Time: 3.0 s, Height: 55.90 m
Time: 3.1 s, Height: 52.91 m
Time: 3.2 s, Height: 49.82 m
Time: 3.3 s, Height: 46.64 m
Time: 3.4 s, Height: 43.36 m
Time: 3.5 s, Height: 39.97 m
Time: 3.6 s, Height: 36.50 m
Time: 3.7 s, Height: 32.92 m
Time: 3.8 s, Height: 29.24 m
Time: 3.9 s, Height: 25.47 m
Time: 4.0 s, Height: 21.60 m
Time: 4.1 s, Height: 17.63 m
Time: 4.2 s, Height: 13.56 m
Time: 4.3 s, Height: 9.40 m
Time: 4.4 s, Height: 5.14 m
Time: 4.5 s, Height: 0.77 m
Time: 4.6 s, Height: 0.00 m

The particle has hit the ground.
```

***** The End *****