Week 01: Thinking Like a Developer – Algorithms and Problem Solving

Authors: Refat Othman and Diaeddin Rimawi

Objective

- Understand what an algorithm is and how it is used in problem solving.
- Learn to think in structured steps like a developer.
- Develop the ability to use pseudocode for logic representation.
- Reinforce key programming constructs: sequential execution, conditionals, and loops.

Lecture I: Theory

Section 1: What is an Algorithm?

An **algorithm** is a step-by-step set of operations to be performed. Algorithms are fundamental to computer science and serve as the blueprint for computer programs.

The term algorithm comes from the name of the Persian mathematician **Mohammed ibn-Musa al-Khwarizmi**, highlighting the historical roots of modern computing in mathematical problem-solving.

Example – Making a Cup of Tea

- Boil water
- Place teabag in cup
- Pour hot water into the cup
- Wait 3 minutes
- Remove teabag
- Add sugar or milk
- Serve

Key Characteristics of Algorithms:

- **Finite**: Must complete after a certain number of steps.
- **Well-defined**: Each step is clear and unambiguous.
- **Input/output**: Takes input and produces an output.

- **Effective**: Achieves the desired result efficiently.

Example – Real-life Airport Pickup Algorithms: Demonstrates that multiple algorithms can solve the same problem with different trade-offs (time, cost, complexity).

Class Activity: Think of different ways to prepare a sandwich. Each student writes 5 steps for their version. Discuss variations.

Section 2: Pseudocode Basics

Pseudocode is a tool to describe logic using natural language-like syntax. It is used to focus on logic without worrying about the programming language.

Common Keywords and Their Usage:

- Start / End: Optional keywords that define the beginning and end of the pseudocode.
- Set: Used to assign an initial value to a variable. E.g., Set total to 0
- Input: Receives data from the user. E.g., Input number
- Output / Print: Displays data to the user. E.g., Print result
- If, Else If, Else: Used for conditional logic. E.g., If score > 50
- While: Used for pre-condition loops. E.g., While count < 10
- For: Used for definite iteration. E.g., For i from 1 to 10
- Repeat / Until: Used for post-condition loops.
- Increment / Decrement: Modifies values by increasing or decreasing.
- Break / Continue: Used to exit or skip loop iterations (optional in high-level pseudocode)

Why Pseudocode?

- Clarifies thought process
- Language independent
- Easier for team discussions
- Reduces bugs before implementation

Note: Pseudocode does not follow strict syntax but must be readable and logically consistent.

Section 3: Sequential Problem Solving

Sequential Execution means instructions are carried out one after another, in order.

Example – Greeting Sequence:

- Print “Hello”
- Print “My name is Ali”
- Print “Welcome to programming”

Example – Making a Sandwich

- Get two slices of bread
- Spread butter on both slices
- Add cheese and tomato
- Put slices together
- Serve

Example – Booking a Flight

- Choose destination
- Select dates
- Search flights
- Select flight
- Enter passenger info
- Pay
- Receive confirmation

Mathematical Examples – Sequential Thinking:

- **Sum of Two Numbers:**
    - Ask user to enter the first number
    - Input first number
    - Ask user to enter the second number
    - Input second number

    - Add the two numbers
    - Print the result
- **Area of a Square:**
    - Ask user to enter the length of one side
    - Input the length of one side
    - Multiply the side by itself
    - Print the area
- **Convert Celsius to Fahrenheit:**
    - Ask user to enter the temperature in Celsius
    - Input temperature in Celsius
    - Multiply by 9, divide by 5, add 32
    - Print the result
- **Perimeter of a Rectangle:**
    - Ask user to enter the length
    - Input length
    - Ask user to enter the width
    - Input width
    - Multiply length and width by 2 and sum
    - Print the perimeter

Exercise: Write a step-by-step pseudocode for preparing your school bag in the morning.

Section 4: Conditional Statements (If-Else)

Conditional logic allows the program to make decisions based on data.

Example – Grade Check

- Ask user to enter the grade
- Input grade

- If grade >= 60 → print "Pass"
- Else → print "Fail"

Example – Weather Decision

- Ask user to enter the temperature
- Input temperature
- If temperature > 30 → print “Wear light clothes”
- Else if temperature < 10 → print “Wear a jacket”
- Else → print “Normal clothes”

Example – Login Validation

- Ask user to enter username and password
- Input username and password
- If both match stored data → print “Login successful”
- Else → print “Access denied”

Exercise: Write pseudocode that checks if a number is divisible by 5 and 3.

Section 5: Loops (While and For)

Loops allow repeating steps until a condition is met.

While Loop Example – Count to 5

- Set counter to 1
- While counter <= 5
    - Print counter
    - Increment counter

For Loop Example – Print First 10 Even Numbers

- For number from 2 to 20 in steps of 2
    - Print number

Example – Sum of First N Numbers

- Ask user to enter a number, n
- Input n
- Set sum to 0
- For i from 1 to n
    - Add i to sum
- Print sum

Example – Countdown

- Set number to 10
- While number > 0
    - Print number
    - Decrement number

Exercise: Write a loop that prints every third number from 3 to 30.

Lecture II: Practice and Exercises

These exercises reinforce what was learned in the theory section. Each can be implemented later in Python.

1. **Write an algorithm that finds the area of a triangle, knowing that the area is calculated using the formula:**
   Area = (1/2) × base × height. Ask the user to input the base and the height of the triangle, then calculate and print the area.

2. **Write an algorithm that functions as a simple calculator. The program should ask the user to input two numbers and an arithmetic operator (+, -, *, /, or %). Based on the operator, perform the corresponding operation and print the result.**

3. **Write an algorithm to find the maximum of three numbers. Ask the user to enter three different numbers, then compare them and print the largest value.**

4. **Write an algorithm to find the maximum number in a list of integers. First, ask the user how many numbers they want to enter. Then, input the numbers one by one and determine the largest among them.**

5. **Write an algorithm that counts the number of passes and fails in a group of students. Keep asking the user to enter student grades until the user enters -1 to stop. A passing grade is considered 60 or higher. Print the number of passing and failing students, and the failure percentage.**

6. **Write an algorithm to compute the average of an unknown number of grades. Ask the user to input grades continuously until -1 is entered. Then, calculate and print the average of all entered grades (excluding -1).**

7. **Write an algorithm to compute the factorial of a number. Ask the user to input a positive integer n, then calculate and print the factorial value (n! = n × (n-1) × ... × 1).**

8. **Write an algorithm to check if a give number is prime or not.**

Submit your solution for the following questions to the technical assistant before Friday:

- Write pseudocode for checking if a year is a leap year.
- Write a loop that prints the multiplication of a given number.

Wrap-Up Discussion

- Why step-by-step thinking is the foundation of coding.
- How logic translation makes programming easier.
- How pseudocode helps avoid syntax distractions.

Next Lecture Preview: Writing your first Python script