

Computational Thinking Using Python

CSE1500

[L-T-P-C: 2-0-2-3]

MODULE – 1

Introduction to Computational thinking



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Computational Thinking Using Python

- Course Introduction
- What is Computational Thinking?
- Four Pillars of Computational Thinking
- Importance for Engineers
- Scope and Opportunities
- Why Python for Computational Thinking?
- Detailed Discussion of CT Pillars in Python
- Case Studies
- Assignments / Practice Problems



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



What is Computational Thinking

Computational thinking (CT) is a problem-solving process that involves expressing problems and their solutions in a way that a computer can understand and execute.

A set of cognitive skills and techniques used to understand complex problems and devise solutions that can be implemented using computing systems.

Computational Thinking (CT) is about how to get a computer to do the job for us – how can we control a complex electronic device to do a job reliably without causing damage or harm.

It is not just about programming:

Some people like to think of computational thinking as something directly related to coding or programming, but its not just about programming. Its an approach to developing solutions to problems using concepts and ideas from Computer Science.

Computational thinking involves breaking down problems into subproblems, looking for patterns or similarities, identifying important details, and developing a plan of action.

Why is Computational Thinking Important?

Computational Thinking is essential in today's digital world because:

a) Enhances Problem-Solving Skills and Critical thinking skills

- Helps analyze and solve real-world problems logically.
- Encourages structured and logical reasoning.

b) Applicable Across Disciplines

- CT is not just for computer scientists. It is used in:
- Science: Modeling chemical reactions or biological processes.
- Mathematics: Solving complex equations or optimizing formulas.
- Engineering: Designing efficient systems.
- Social Sciences: Analyzing data and predicting trends.
- Arts and Music: Generating digital art or composing algorithmic music.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Why is Computational Thinking Important?

c) Foundation for Programming and Computer Science

- Essential for programming, AI, machine learning, and data science.
- Improves algorithmic efficiency, System design and data processing.
- Helps in problem-solving across all disciplines
- Builds a structured way of thinking

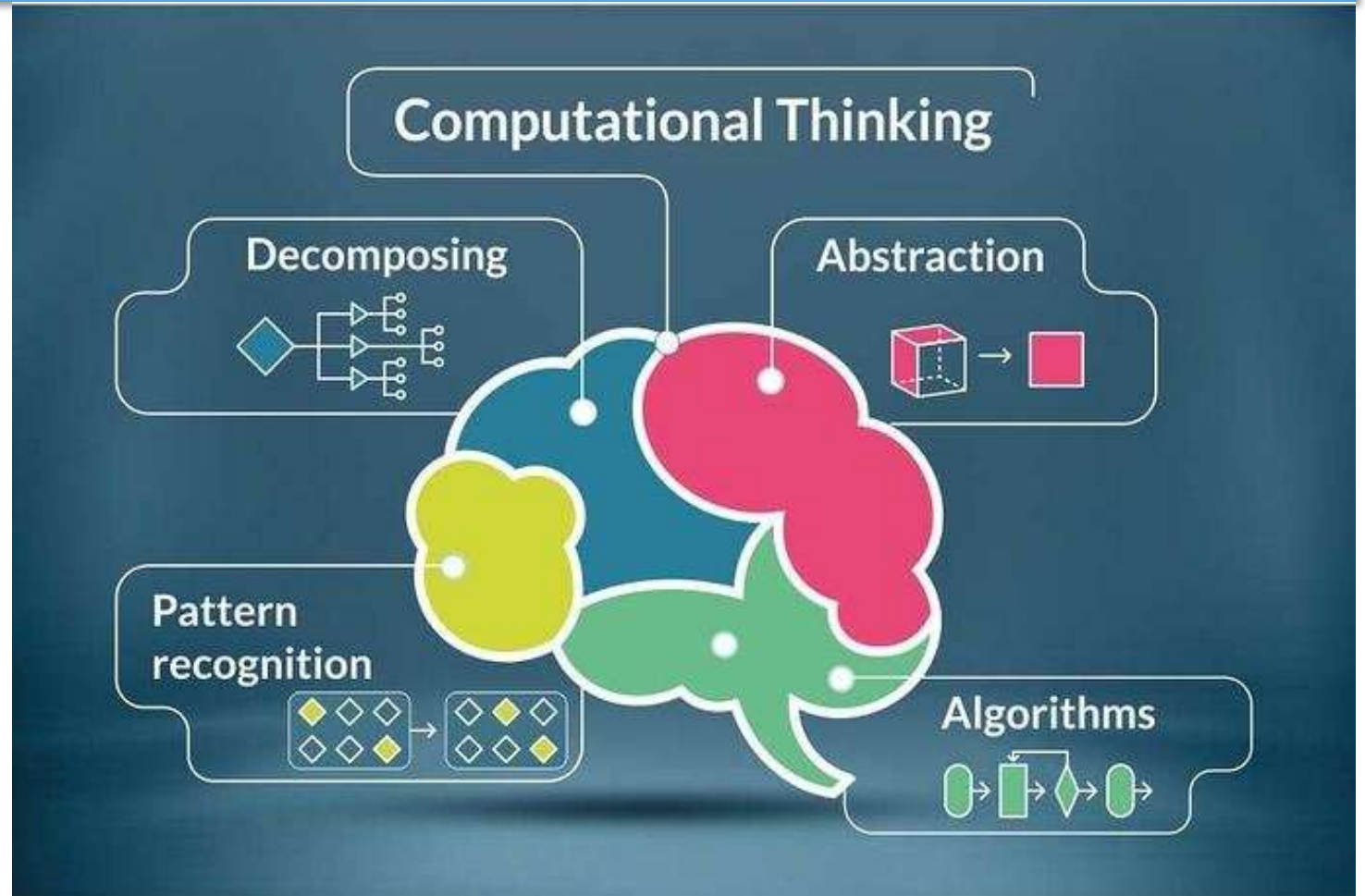
d) Prepares Students for the Digital World

- CT helps students understand how digital technologies work.
- Encourages digital literacy, enabling better use of technology.
- Prepares learners for 21st-century careers, many of which involve data analysis, automation of tasks, or AI.
- Design and analysis of systems

Pillars of Computational Thinking

Four Pillars of CT:

- 1) Decomposition
- 2) Pattern recognition
- 3) Data representation – abstraction
- 4) Algorithms



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Pillars of Computational Thinking

Computational thinking is generally broken down into four key pillars :

1) Decomposition :

Decomposition is the process of breaking down a complex problem or system into smaller, more manageable parts. By dividing a large problem into smaller sub-problems, it becomes easier to understand, solve and debug. Each sub-problem can be tackled individually, and the solutions can then be combined to solve the original, larger problem.

Example 1:

Online Shopping Cart :

User registration, product listing, cart handling, payment gateway.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Pillars of Computational Thinking (Decomposition)

Example 2:

Planning a large event like a wedding. This can be broken down into smaller tasks like creating a guest list, choosing a venue, sending invitations, and arranging catering.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Pillars of Computational Thinking

2) Pattern Recognition :

Pattern recognition involves observing similarities and patterns within a problem.

Pattern recognition is the skill of recognizing the similarities and differences between concepts and objects.

Once a pattern is identified, it's possible to create a general rule or model that can be applied to similar problems. This helps in creating shortcuts to solve complex problems.

Pattern recognition is the ability to analyze and identify the shared characteristics between parts of a decomposed problem. (Can we conclude pattern recognition occurs after decomposition?)

Pattern recognition is a core computational thinking skill that helps in creating shortcuts to solve complex problems.

Pillars of Computational Thinking (Pattern Recognition)

Examples:

- 1) When you drive to a new city you can easily navigate the streets and follow the traffic signs because you have already created a pattern of traffic signs in your mind and can easily apply/map them in different situations.
- 2) When you learn how to make coffee with one coffee machine, its pattern is developed in your mind and you will know how to make coffee with different brands of coffee machines since you replicate the same pattern in different situations.
- 3) In artificial intelligence, we use pattern recognition to analyze data and identify similarities to recommend an object or content to the end user. For example, Amazon, Netflix, or Facebook recommend items, movies, and friends or events from the patterns they develop based on your interaction history with these systems.
- 4) Science: Students classify animals based on their characteristics and articulate common characteristics for the groupings.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Pillars of Computational Thinking

3) Abstraction & Data Representation

Abstraction is a fundamental concept in computational thinking that involves simplifying complex systems by focusing on essential details and ignoring irrelevant ones

Abstraction is the process of filtering out irrelevant information to focus on the essential details of a problem. It involves creating a general model or idea that represents the core of the problem, ignoring the specific details.

Example:

- 1) Think of it as summarizing a long story; you highlight the main plot points and characters while leaving out the minor, less important details.
- 2) Creating a map. A map is an abstraction of the real world that focuses on essential details like roads and landmarks while ignoring irrelevant information like individual trees or buildings. The choice of symbols and colors on the map is a form of data representation.

Pillars of Computational Thinking (Abstraction & Data Representation)

There are two main types of abstraction:

Procedural Abstraction: This involves creating a procedure or function to perform a specific task. We can then use this procedure by calling it by name without needing to know the exact steps it takes to accomplish the task. **For example**, when you use a function like `sort()` in a programming language, you don't need to know the specific sorting algorithm (e.g., bubble sort, quick sort) that's being used under the hood. You just know that it will sort the data.

Data Abstraction: This involves creating data structures that hide the internal representation of data. We can interact with the data through a set of defined operations, without needing to know how the data is physically stored.

A good example is a stack. We know we can `push()` items onto it and `pop()` them off, but we don't necessarily need to know if it's implemented using an array or a linked list. The key is the interface, not the implementation.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Pillars of Computational Thinking (Abstraction & Data Representation)

Data Representation:

Data representation is the method by which we encode and store information within a computational system. Computers, at their most basic level, only understand two states: on and off, which we represent as 0s and 1s. This binary system is the foundation for all data representation.

Here are some key concepts related to data representation:

Binary Numbers: The binary number system (base-2) uses only the digits 0 and 1, also known as BITS (Binary Digits) . This is the native language / mother tongue of computers. All other number systems, like the decimal system (base-10) we use daily, must be converted into binary for a computer to process them.

Bits and Bytes: A single binary digit (0 or 1) is called a bit. A group of eight bits is called a byte. A byte is a fundamental unit of data and can represent a wide range of information, from a single character to a small integer.

Pillars of Computational Thinking (Abstraction & Data Representation)

Character Encoding: Computers need a way to represent text. Character encoding schemes, like ASCII and Unicode, map characters (letters, numbers, symbols) to specific binary values. For example, in ASCII, the letter 'A' is represented by the decimal value 65, which is 01000001 in binary. Unicode is a more modern standard that can represent characters from virtually all writing systems in the world.

Image Representation: Images are represented as a grid of tiny dots called pixels. Each pixel is assigned a binary value that corresponds to its color. The more bits used to represent the color of a single pixel, the more shades and hues the image can display. For instance, an 8-bit image can represent 256 colors, while a 24-bit image can represent over 16 million colors.

Sound Representation: Sound is represented digitally by sampling the analog sound wave at regular intervals. Each sample is a measurement of the wave's amplitude at a specific point in time, and this measurement is then stored as a binary number. The sample rate (how often the sound is measured) and bit depth (how many bits are used for each measurement) determine the quality of the digital sound.

Pillars of Computational Thinking (Abstraction & Data Representation)

Decimal Number system : contains ten digits 0,1,2,3,4,5,6,7,8 & 9, base is 10

Binary Number system : contains only two digits 0 & 1, base is 2.

Octal Number system : contains eight digits 0,1,2,3,4,5,6,7, base is 8.

Hexa decimal number system : contains sixteen digits 0,1,2,3,4,5,6,7,8 & 9, A,B,C,D,E,F. (where A=10, B=11, C=12, D=13, E=14, F=15), base is 16.

Activity : Students have to learn how to convert one number system into another.

For example : Decimal to Binary

Computer Science is a science of abstraction -creating the right model for a problem and devising the appropriate mechanizable techniques to solve it.

– A. Aho and J. Ullman



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Algorithms

- An algorithm is a set of well-defined, step-by-step instructions for solving a problem or completing a task. It is a precise procedure that, when followed, will reliably produce a correct result.
- Algorithms are not limited to computers, they are used in everyday life we perform activities by following certain sequence of steps.
- Examples of activities include getting ready for school, making breakfast, riding a bicycle, wearing a tie, solving a puzzle and so on.
- To complete each activity, we follow a sequence of steps.

Example : Suppose following are the steps required for an activity 'Riding a Bicycle':

- 1) Remove the bicycle from the stand
- 2) Sit on the seat of the bicycle
- 3) Start peddling
- 4) Use breaks whenever needed and
- 5) Stop on reaching the destination.

Algorithms

Characteristics of a good algorithm

- Precision — the steps are precisely stated or defined.
- Uniqueness — results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
- Finiteness — the algorithm always stops after a finite number of steps.
- Input — the algorithm receives some input.
- Output — the algorithm produces some output.

While writing an algorithm, it is required to clearly identify the following:

- The input to be taken from the user
- Processing or computation to be performed to get the desired result
- The output desired by the user

Algorithms

Why do we need an Algorithm?

- A programmer writes a program to instruct the computer to do certain tasks as desired. The computer then follows the steps written in the program code. Therefore, the programmer first prepares a roadmap of the program to be written, before actually writing the code. Without a roadmap, the programmer may not be able to clearly visualize the instructions to be written and may end up developing a program which may not work as expected.
- An algorithm which is the building block of a computer program. Writing an algorithm is mostly considered as a first step to programming. Once we have an algorithm to solve a problem, we can write the computer program for giving instructions to the computer in high level language. If the algorithm is correct, computer will run the program correctly, every time. So, the purpose of using an algorithm is to increase the reliability, accuracy and efficiency of obtaining solutions.

Algorithms

Representation of Algorithms

There are two common methods of representing an algorithm —**pseudocode and flowchart**

Either of the methods can be used to represent an algorithm while keeping in mind the following:

- it showcases the logic of the problem solution, excluding any implementational details
- it clearly reveals the flow of control during execution of the program

Pseudocode A pseudocode (pronounced Soo-doh-kohd) is another way of representing an algorithm. It is considered as a non-formal language that helps programmers to write algorithm. It is a detailed description of instructions that a computer must follow in a particular order (logic of a program). It is intended for human reading and cannot be executed directly by the computer. No specific standard for writing a pseudocode exists. The word “pseudo” means “not real,” so “pseudocode” means “not real code”.

Benefits of Pseudocode

- Before writing codes in a high level language, a pseudocode of a program helps in representing the basic functionality of the intended program.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Algorithm: to find the square of a number.

Example : Write an algorithm to find the square of a number.

Before developing the algorithm, let us first identify the input, process and output:

- Input: Number whose square is required
- Process: Multiply the number by itself to get its square
- Output: Square of the number

Algorithm to find square of a number.

Step 1: Input a number and store it to num

Step 2: Compute $\text{num} * \text{num}$ and store it in square

Step 3: Print square



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Pseudocode: Calculate Area of a Rectangle

START

DISPLAY "Enter the length of the rectangle:"

READ length

DISPLAY "Enter the breadth of the rectangle:"

READ breadth

$\text{area} \leftarrow \text{length} \times \text{breadth}$

DISPLAY "The area of the rectangle is: ", area

END



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Flowchart

Flowchart — Visual Representation of Algorithms

A flowchart or flow diagram is a method that presents information in a visual format, similar to an [infographic](#). This is made possible through a system that uses symbols to explain the stages and decisions of a process, as well as the sequence of actions to follow. Essentially, it's a tool to visualize and understand the logic of a process from start to finish.



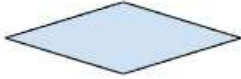
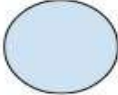




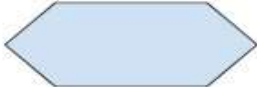
(or)

A flowchart is a visual representation of an algorithm. A flowchart is a diagram made up of boxes, diamonds

and other shapes, connected by arrows. Each shape represents a step of the solution process and the arrow

represents the order or link among the steps.

Flow Chart symbols

Symbol	Name	Function
	Start/End	Indicates the start or end point of the process.
	Process	Describes a specific action or task to be performed within the process.
	Decision	Denotes a question or condition to be evaluated. Also, the different output options based on the answer.
	Connector	Used to connect different parts of the diagram that are in separate places.
	Flow arrow	It represents the direction of the process flow, indicating the order in which actions are carried out.
	Input/Output	It is the material or data entering or leaving the system.
	Document	Refers to an external document or file
	Multi-document	Indicates several printed documents or reports
	Preparation	Marks an adjustment to another step in the process.

Write an algorithm to calculate area and perimeter of a rectangle, using both pseudocode and flowchart.

Pseudocode for calculating area and perimeter of a rectangle.

INPUT length

INPUT breadth

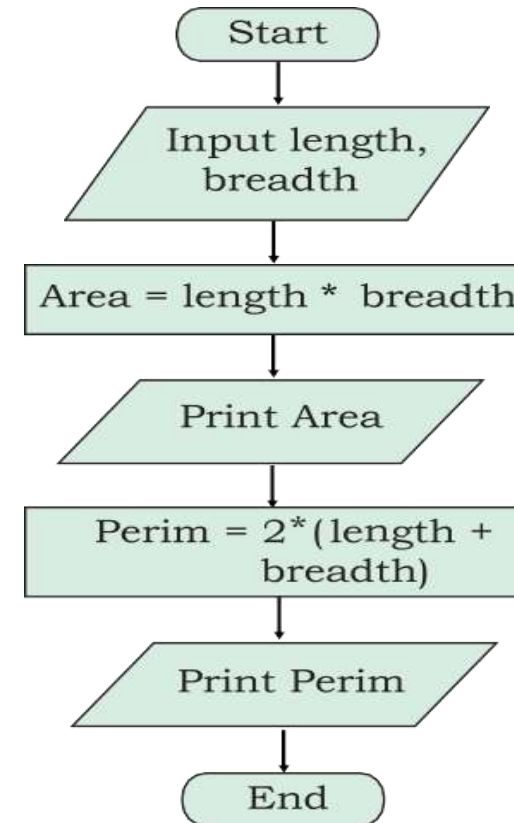
COMPUTE Area = length * breadth

PRINT Area

COMPUTE Perim = 2 * (length + breadth)

PRINT Perim

Flowchart



Let us write a pseudocode and draw a flowchart where multiple conditions are checked to categorize a person as either child (<13), teenager (≥ 13 but <20) or adult (≥ 20), based on age specified:

Input: Age

Process: Check Age as per the given criteria

Output: Print either "Child", "Teenager", "Adult"

Pseudocode is as follows:

INPUT Age

IF Age < 13 THEN

 PRINT "Child"

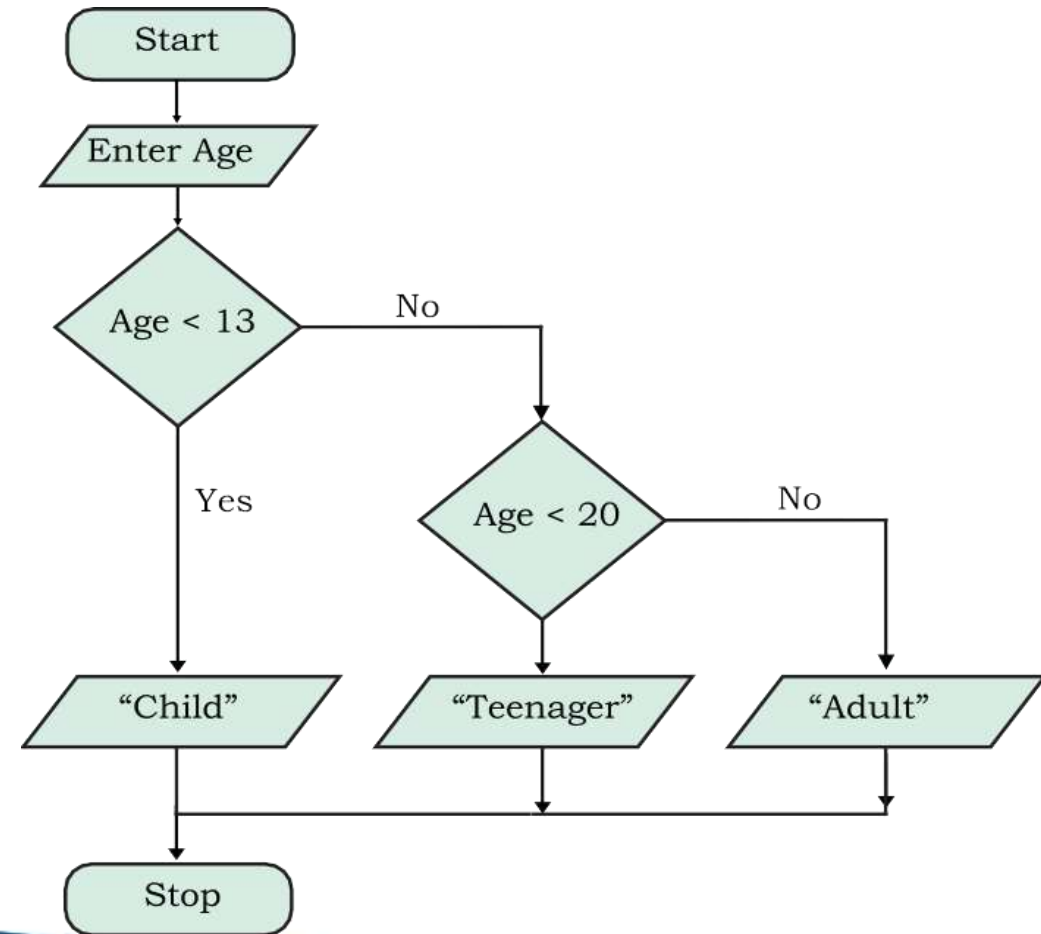
ELSE IF Age < 20 THEN

 PRINT "Teenager"

ELSE

 PRINT "Adult"

Flow Chart



Write pseudocode and draw flowchart to accept numbers till the user enters 0 and then find their average.

Pseudocode is as follows:

Step 1: SET count = 0, sum = 0

Step 2: INPUT num

Step 3: WHILE num is not equal to 0, REPEAT Steps 4 to 6

Step 4: sum = sum + num

Step 5: count = count + 1

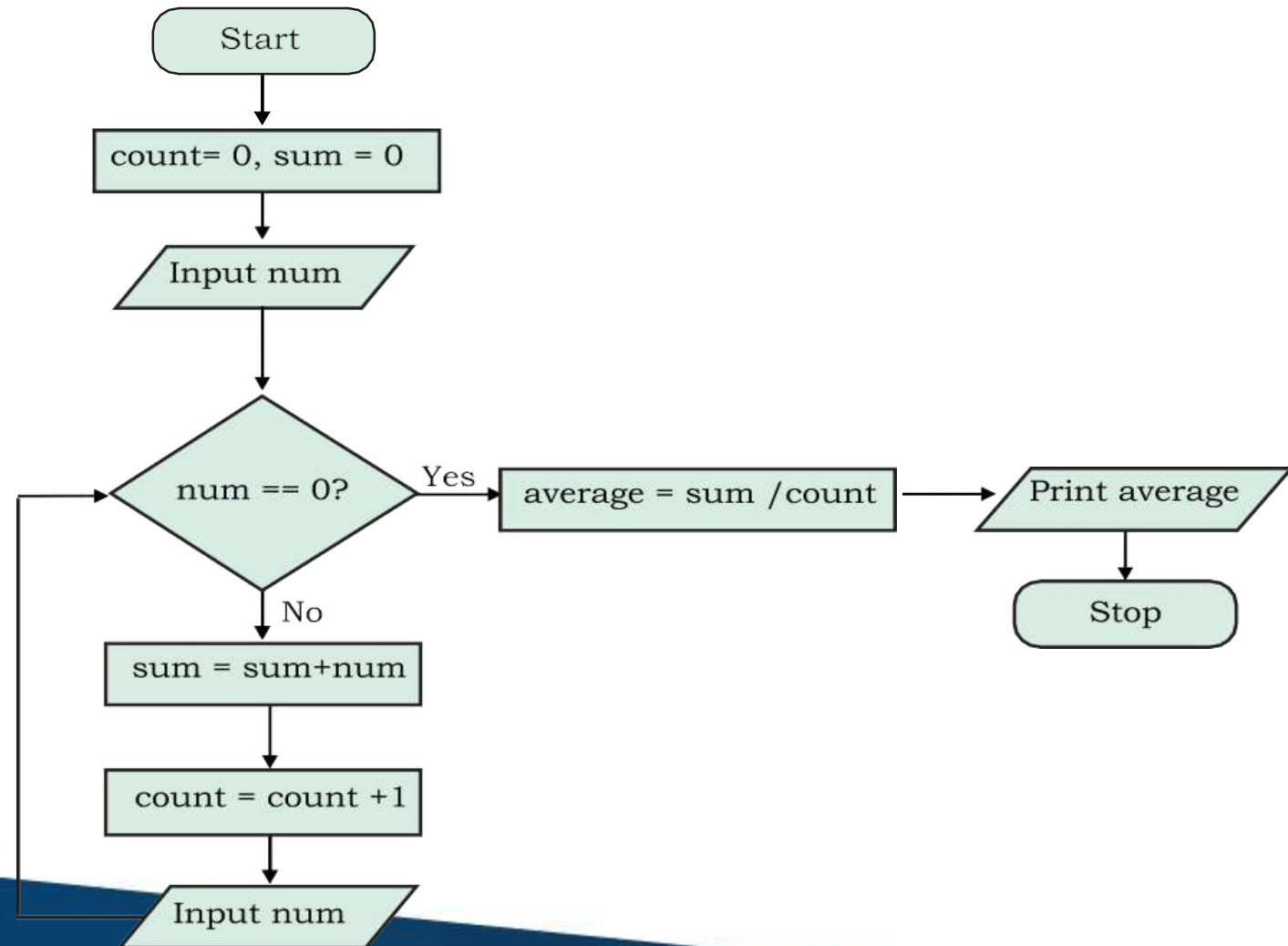
Step 6: INPUT num

Step 7: COMPUTE average = sum/count

Step 8: PRINT average

Write pseudocode and draw flowchart to accept numbers till the user enters 0 and then find their average.

Flow Chart



Real-Life Applications of Computational Thinking (CT)

1. Navigation Apps – Shortest Path Algorithms

📌 CT Concepts Involved:

Decomposition, Abstraction, Algorithmic Thinking

📱 Examples:

Google Maps, Waze, and other navigation apps use **graph algorithms** like **Dijkstra's Algorithm** or **A (A-star)*** to compute the shortest or fastest route from your location to your destination.

Real-time data from traffic signals, user-reported incidents (accidents, road closures), and GPS location are abstracted to update the route dynamically.

💡 CT in Action:

Breaks the entire city map into small components (nodes and edges).

Uses decision-making to determine the optimal route based on speed, distance, and traffic.



Microsoft Word
Document

Click on above word doc for
Additional Case studies

Python Constructs for Computation Thinking Pillars

Computational
Thinking Pillars

Typical Python Constructs

Decomposition

Functions, modules, classes

Pattern Recognition Loops (for, while), conditionals (if-else)

Abstraction

Functions, OOP (classes, methods), libraries

Algorithm Design

Step-by-step logic using control structures, data structures, standard algorithms.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Scope and Opportunities



Central Role of the Learner:

The image places a student using a laptop at the center, emphasizing that computational thinking is a skill individuals develop through active engagement and practice.

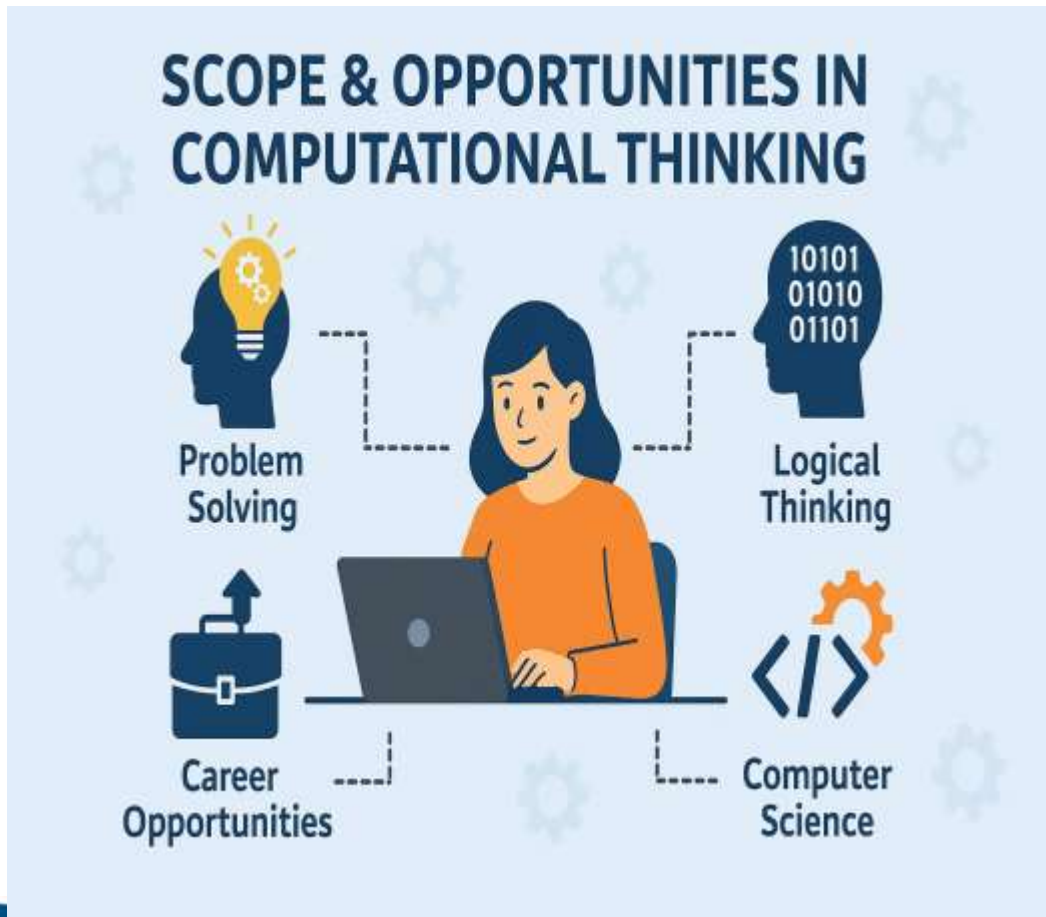
Problem Solving:

Represented by a head with a glowing lightbulb, this shows that computational thinking enhances the ability to break down complex problems into manageable parts and find efficient solutions.

Logical Thinking:

Depicted with binary code inside a human head, highlighting how computational thinking trains individuals to think logically, using step-by-step reasoning akin to computer processes.

Scope and Opportunities



Career Opportunities:

Shown with a briefcase and upward arrow, this signifies the broad range of career paths computational thinking enables—not just in tech but across industries like healthcare, finance, and engineering.

Computer Science Connection:

Symbolized by coding brackets and a gear, indicating that computational thinking forms the foundation of computer science, programming, and automation, enabling innovation and technological advancement.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Assignments

1. Traffic Simulation System :

Simulate traffic movement at a 4-way intersection to optimize flow

2. Find given string is a palindrome:

A palindrome is a word, phrase, number, or sequence that reads the same forward and backward.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Real-Life Applications of Computational Thinking (CT)

2. Online Recommendations – Pattern Recognition & Machine Learning

📌 CT Concepts Involved:

Pattern Recognition, Data Abstraction, Automation

🛒 Examples:

Amazon recommends products based on your **browsing history**, **purchase behavior**, and what similar users bought.

Netflix or **YouTube** suggests content using **collaborative filtering algorithms**, analyzing patterns in your watch history.

Spotify creates custom playlists based on your listening patterns using **machine learning models**.

💡 CT in Action:

Systems analyze large datasets, find patterns in behavior, and predict preferences.

These patterns are abstracted to generate personalized suggestions.

Real-Life Applications of Computational Thinking (CT)

Banking Systems – Decision Logic and Automation

📌 CT Concepts Involved:

Logical Reasoning, Automation, Pattern Recognition

🏠 Examples:

Fraud Detection: Banks analyze spending patterns and alert customers when unusual transactions occur.

Loan Approval Systems: Banks use **decision trees** and **rule-based logic** to evaluate eligibility (income, credit score, repayment history).

Automated Teller Machines (ATMs) use predefined logic to verify PIN, account balance, and transaction rules.

💡 CT in Action:

Banks use **if-else logic**, thresholds, and scoring models to make decisions automatically and securely.

Automation replaces manual processes, improving efficiency and accuracy.

Real-Life Applications of Computational Thinking (CT)

Agriculture – Precision Farming

🚁 Examples:

Drones and sensors analyze soil data, moisture levels, and crop health.

Algorithms suggest optimal watering schedules and fertilizer usage.

❓ CT Concepts:

Data abstraction from sensors

Algorithmic decision-making for crop yield optimization

Real-Life Applications of Computational Thinking (CT)

Education – Personalized Learning

Examples:

Platforms like CodeChef, Codetantra and BYJU'S use student performance data to adapt difficulty levels.

AI tutors provide feedback and recommend resources.

CT Concepts:

Pattern recognition in performance

Rule-based logic for content adaptation



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Real-Life Applications of Computational Thinking (CT)

Cybersecurity – Threat Detection

🔑 Examples:

Firewalls and intrusion detection systems analyze network traffic to spot suspicious behavior.

Anti-virus software uses **heuristics** and **pattern matching** to detect malware.

❓ CT Concepts:

Abstraction of network data

Pattern recognition

Automation of threat response



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Case Study - Weather Data Analysis

Analyze historical temperature data to find trends and predict future weather.

Pillar	Python Implementation
Decomposition	Split the task into reading data, cleaning it, analyzing it, and visualizing results. Each part can be a function: <code>read_data()</code> , <code>clean_data()</code> , <code>analyze_trends()</code> etc.
Pattern Recognition	Use loops and conditions to detect recurring temperature cycles (e.g., seasonal changes). Use for loops and if statements.
Abstraction	Hide complexity using functions and libraries like pandas, matplotlib, and create reusable data structures (e.g., DataFrame).
Algorithm Design	Create an algorithm to calculate averages, detect anomalies, and make predictions (using moving averages or linear regression with scikit-learn).

Case Study - Student Marks Management System

Build a system that allows entry, storage, and analysis of student marks.

Pillar	Python Implementation
Decomposition	Divide the system into modules like input, storage, grading, report generation. Each module is a Python function/class.
Pattern Recognition	Identify patterns in student performance (e.g., repeated failures in a subject) using loops and data filtering.
Abstraction	Use classes like Student and Subject, abstracting away internal data handling..
Algorithm Design	Create logic for grading, ranking, average calculation using control structures and statistics module.

Case Study - Chatbot for College Enquiry

A simple rule-based chatbot to answer queries about college (admission, departments, fees).

Pillar	Python Implementation
Decomposition	Break down into components: input handling, pattern matching, response generation.
Pattern Recognition	Match input strings to known phrases using if/elif blocks or regex
Abstraction	Use a dictionary or function map to store questions and answers, hiding detailed rules.
Algorithm Design	Design the logic to parse and respond using conditional statements or NLP with libraries like nltk..

Introduction to Python

About Python

- Python was developed by Guido van Rossum at National Research Institute for Mathematics and Computer Science in Netherlands in 1990.
- Rossum wanted the name of his new language to be short, unique and mysterious.
- Inspired by Monty Python's Flying Circus, a BBC comedy series, he named the language Python.

Guido developed Python by taking almost all programming features from different languages

- Functional Programming Features from C
- Object Oriented Programming Features from C++
- Scripting Language Features from Perl and Shell Script
- Python is recommended as first programming language for beginners.



Guido van
Rossum

Where we can use Python

The most common important and real world application areas of Python are

- Desktop applications
- Web applications
- Database applications
- Network Programming
- Developing games
- Data Analysis applications
- Machine Learning
- Artificial Intelligence applications
- IOT applications

Top Software companies like Google, Microsoft, IBM, Yahoo using Python.



Guido van
Rossum



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Features of Python



Simple – Syntax is close to English, making code easy to read and write.

Easy to Learn – Few keywords and simple structure; familiar to C programmers.

Open Source – Free to use, modify, and distribute.

High-Level – Uses English-like commands, easy to understand and maintain.

Dynamically Typed – No need to declare variable types; can change types at runtime.

Platform Independent – Runs on any OS via Python Virtual Machine.

Interpreted – We are not required to compile Python programs explicitly. Internally Python interpreter will take care that compilation.

Portable – Python programs run the same on any system via PVM, though OS-specific modules may affect portability.

Procedure & Object-Oriented – Supports both function-based (procedural) and class-based (object-oriented) programming.

Flavors of Python



- CPython – Standard Python for C applications.
- Jython/JPython – For Java applications, runs on JVM.
- IronPython – For C#.NET platform.
- PyPy – Faster execution with built-in JIT compiler.
- RubyPython – For Ruby platforms.
- AnacondaPython – Designed for handling large-scale data processing, predictive analytics and scientific computing, it is called Anaconda Python.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Python Versions:



- Python Versions:
- Python 1.0V introduced in Jan 1994
- Python 2.0V introduced in October 2000
- Python 3.0V introduced in December 2008
- Current Stable: Python 3.13.6 (Aug 6, 2025)

Limitations of Python:

- Performance is not up to the mark because Python is an interpreted language.
- Not commonly used for mobile applications.

Python Versions:



Python Versions:

Python 1.0V introduced in Jan 1994

Python 2.0V introduced in October 2000

Python 3.0V introduced in December 2008

Current Stable: Python 3.13.6 (Aug 6, 2025)



**Microsoft Word
Document**

Click on the above word document for details notes on Python module-1



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Python Character set

Letters: Upper case and lower case letters

Digits: 0,1,2,3,4,5,6,7,8,9

Special Symbols: Underscore (_), (,), [,], {,}, +, -, *, &, ^, %, \$, #, !, Single quote('), Double quotes("), Back slash(\), Colon(:), and Semi Colon (;)

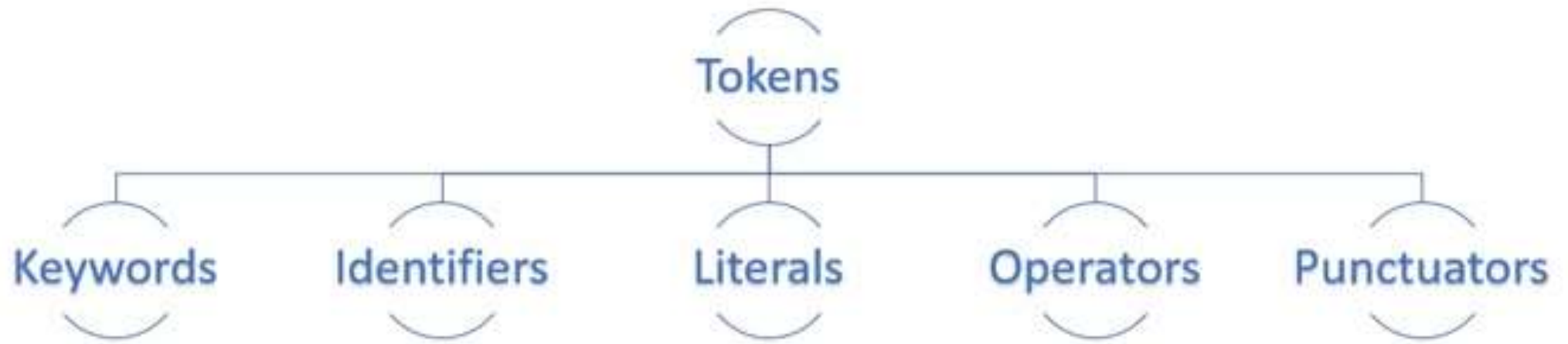
White Spaces: ('\\t\\n\\x0b\\x0c\\r'), Space, Tab.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013





Literals:

`a = 15`

Example

- `78` #Integer Literal
- `21.98` #Floating Point Literal
- `'Q'` #Character Literal
- `"Hello"` #String Literal

Table 3.1: Numeric Literals

Examples	Literal name
450, -15	Integer literal
3.14286, -10.6, 1.25E4	Float literal
0x5A1C	Hexadecimal literal
0557	Octal literal
0B110101	Binary literal
18+3J	Complex literal



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Boolean Literals

Boolean literals are the True or False values stored into a bool type variable

String Literals

```
s1 = 'I am a student of Presidency University'
```

```
s2 = "Computational thinking using Python  
CSE1500"
```



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



When a string literal extends beyond a single line, we should go for triple quotes as:

```
s1 = '''I am a Engineering student  
studying first semester CSE  
Python is very interesting course'''
```

Escape character	Meaning
\	New line continuation
\\	Display a single \
\'	Display a single quote
\"	Display a double quote
\b	backspace
\r	Enter
\t	Horizontal tab space
\v	Vertical tab
\n	New line



Determining the Datatype of a Variable

type(a) displays the datatype of the variable 'a'.

```
a = 15
```

```
print(type(a))
```

```
<class 'int'>
```

To know the exact type of any value,

Example

```
>>> type('Hello Presidency')
```

```
<class 'str'>
```

```
>>> type(123)
```

```
<class 'int'>
```



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Keywords

FALSE	None	TRUE	and
as	assert	async	await
break	class	continue	def
elif	else	except	finally
for	from	global	if
import	in	is	lambda
nonlocal	not	or	pass
raise	return	try	while
with	yield	del	



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013





**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Mr. R C Ravindranath, Asst. Prof, SOE-CSE