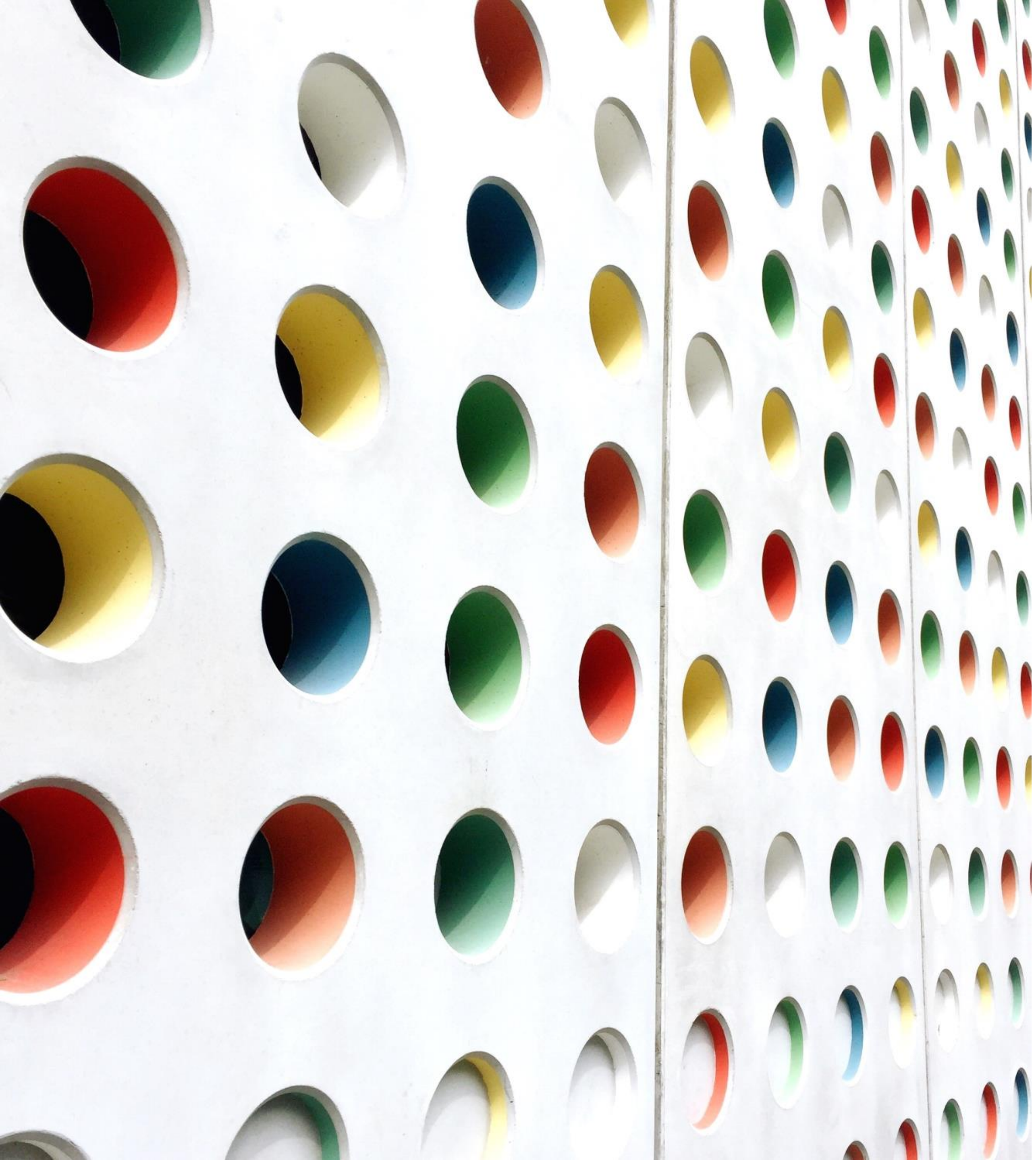


Computational Thinking

ANIP SHARMA

Module Outcomes

UNDERSTANDING COMPUTATIONAL THINKING

APPLIED COMPUTATIONAL THINKING EXERCISES

INTRODUCTION TO FLOW CHARTS & PSEUDO-CODE

FUNDAMENTAL OPERATIONS OF A COMPUTER PROGRAM

Before computers can be used to solve a problem, the problem itself and the ways in which it could be resolved must be understood.

What is Computational Thinking?

Computational thinking is an approach to solving problems using concepts and ideas from computer science

It is the step that comes before computer programming.

It is a thought process, rather than a specific body of knowledge about a device or language.

Although often associated with computers and coding, it is important to note that it can be taught without a device.

It helps students develop future skills.

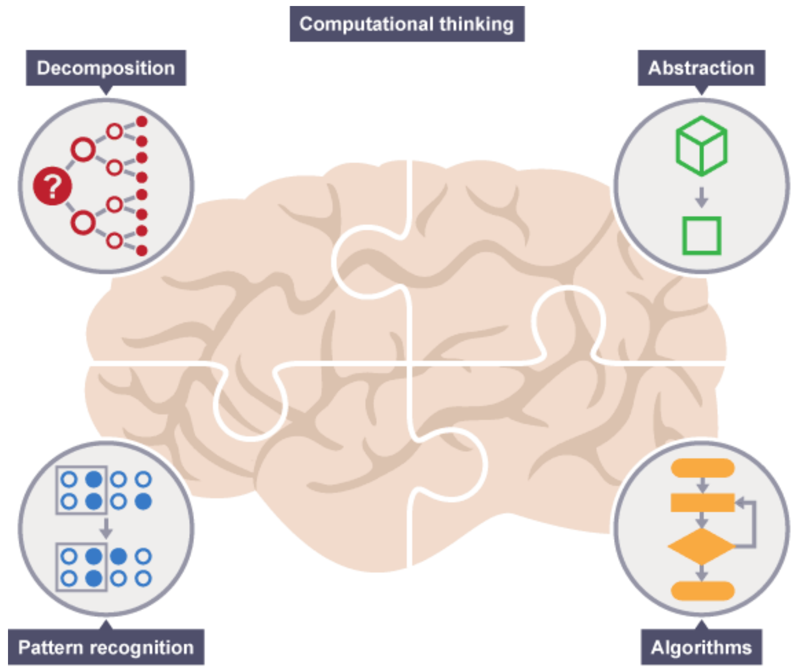
It enables students to articulate a problem and think logically.

Computer Programming tells a computer what to do and how to do it. **Computational thinking enables you to work out exactly what to tell the computer to do.**

Key Skills for Computational Thinking

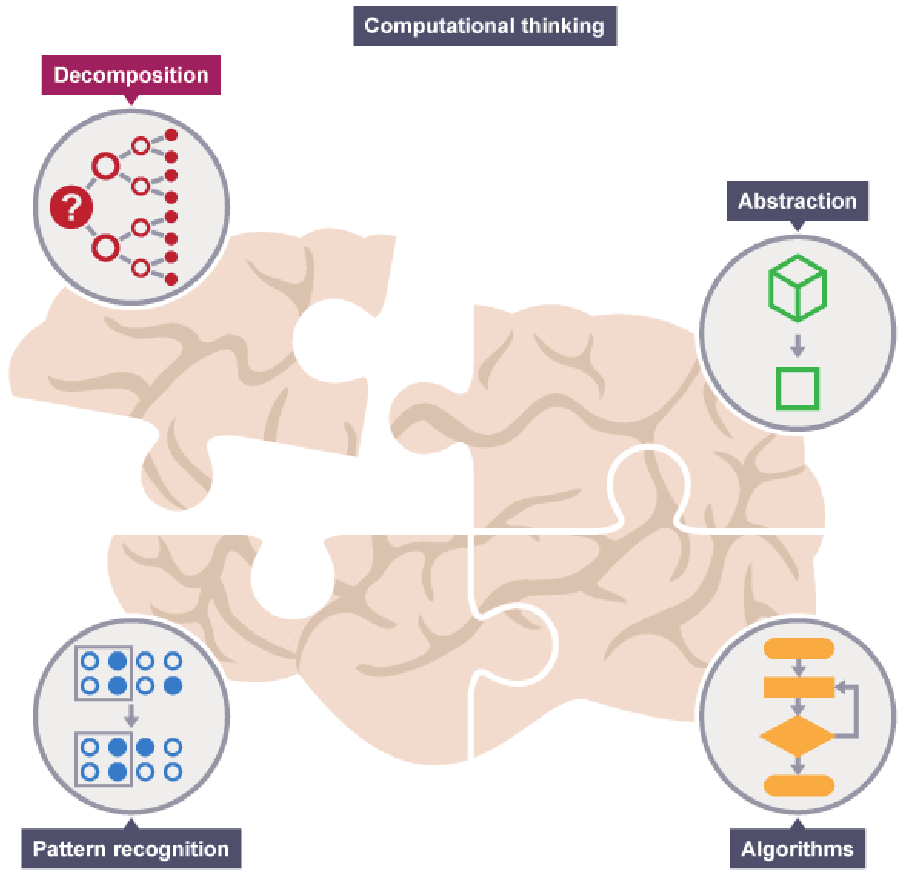
There are four key skills in computational thinking:

- Decomposition
- Pattern Recognition
- Pattern Abstraction
- Algorithm Design



Source: What is computational thinking? - Introduction to computational thinking - KS3 Computer Science Revision - BBC Bitesize

Decomposition



Decomposition

Decomposition is breaking down complex problems into smaller, more manageable chunks.

If a problem is not decomposed, it is much harder to solve.

Decomposition need students to figure out all of the steps needed to make the task happen.

Decomposition is an important life skill in the future when students and adults need to take on larger tasks.

Students will learn ways to delegate in group projects and build time management skills.

Teaching Decomposition

Students are invited into problem-solving scenarios.

Teachers share the complex, multi-step problem.

Teachers should help students in breaking down the problem.

Ideas to Try:

- ❖Teachers might describe a scenario, such as planning a birthday party, that involves multiple steps.
- ❖Students can help to break down the larger task.
- ❖Teacher can help to draw or write a visual representation of their thinking, giving students a mental map of how to solve similar problems in the future.

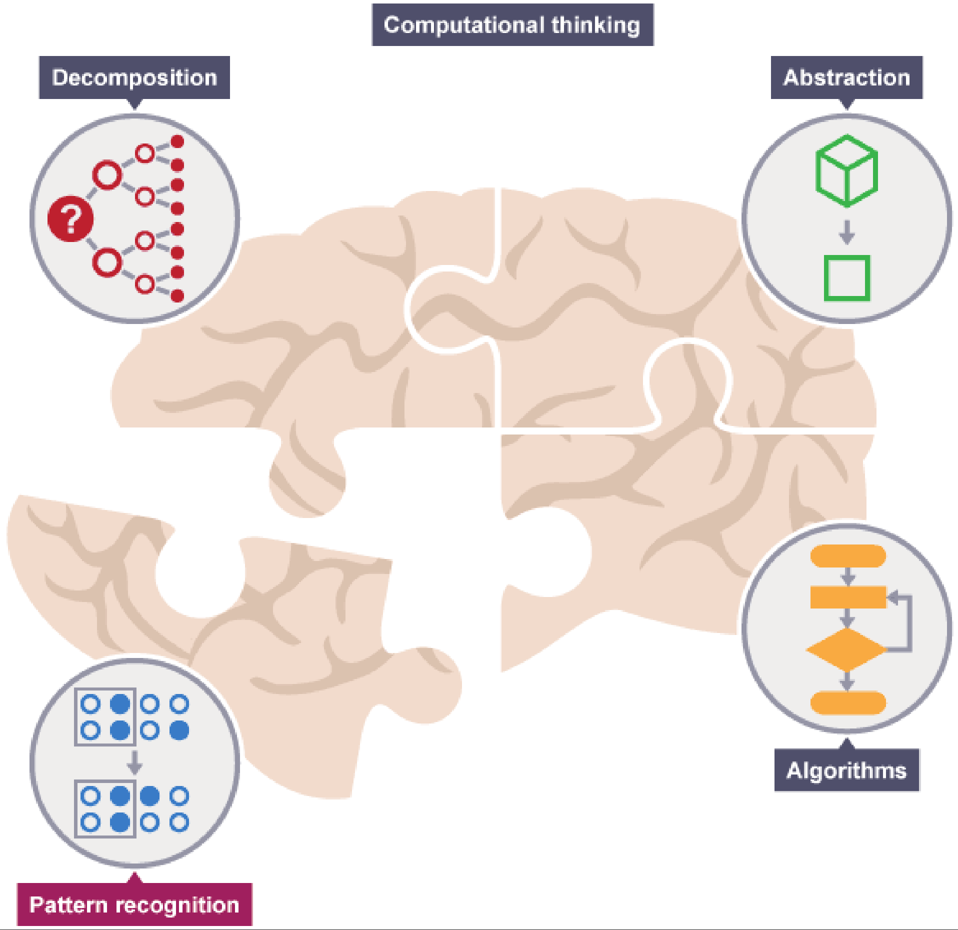
Decompose creating an app

To decompose this task, you would need to know the answer to a series of smaller problems:

- what kind of app you want to create
- what your app will look like
- who the target audience for your app is
- what your graphics will look like
- what audio you will include
- what software you will use to build your app
- how the user will navigate your app
- how you will test your app
- where you will sell your app

Once we have decomposed a complex problem, it helps to examine the small problems for similarities or ‘patterns’. **These patterns can help us to solve complex problems more efficiently.**

Pattern Recognition



Pattern Recognition

When we decompose a complex problem we often find patterns among the smaller problems we create.

The patterns are similarities or characteristics that some of the problems share.

It invites students to analyze similar objects or experiences and identify commonalities.

The more patterns we can find, the easier and quicker our overall task of problem solving will be.

By teaching students to recognize patterns, their awareness of the world around them expands.

By finding what the objects or experiences have in common, young students can begin to develop an understanding of trends and are therefore able to make predictions.

Teaching Pattern Recognition

Younger students may benefit from exploring patterns using music or colored blocks.

Older students may learn about patterns by looking at the periodic table or exploring the patterns seen in multiplication charts.

Ideas to Try:

- ❖To teach students to recognize patterns, you might begin by investigating cats.
- ❖Next, work with your students to create a collage of cats.
- ❖To extend this thinking, invite your students to draw a picture of a cat, labeling the tail, fur, and ears. Emphasize that while your class' cats might look different from one another, they are alike in their core components.

Recognizing patterns in baking

Decomposing the task of baking a cake would highlight the need for us to know the solutions to a series of smaller problems:

what kind of cake we want to bake

what ingredients we need and how much of each

how many people we want to bake the cake for

how long we need to bake the cake for

when we need to add each ingredient

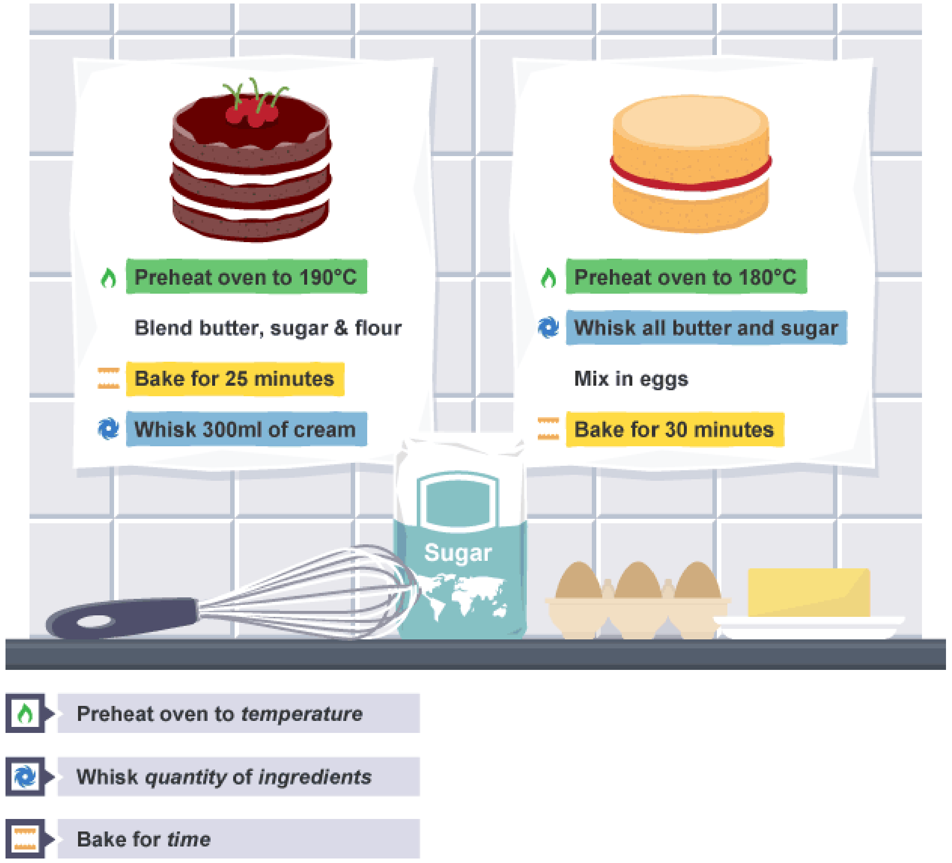
what equipment we need

Recognizing patterns in baking

Once we know how to bake one particular type of cake, we can see that baking another type of cake is not that different - because patterns exist.

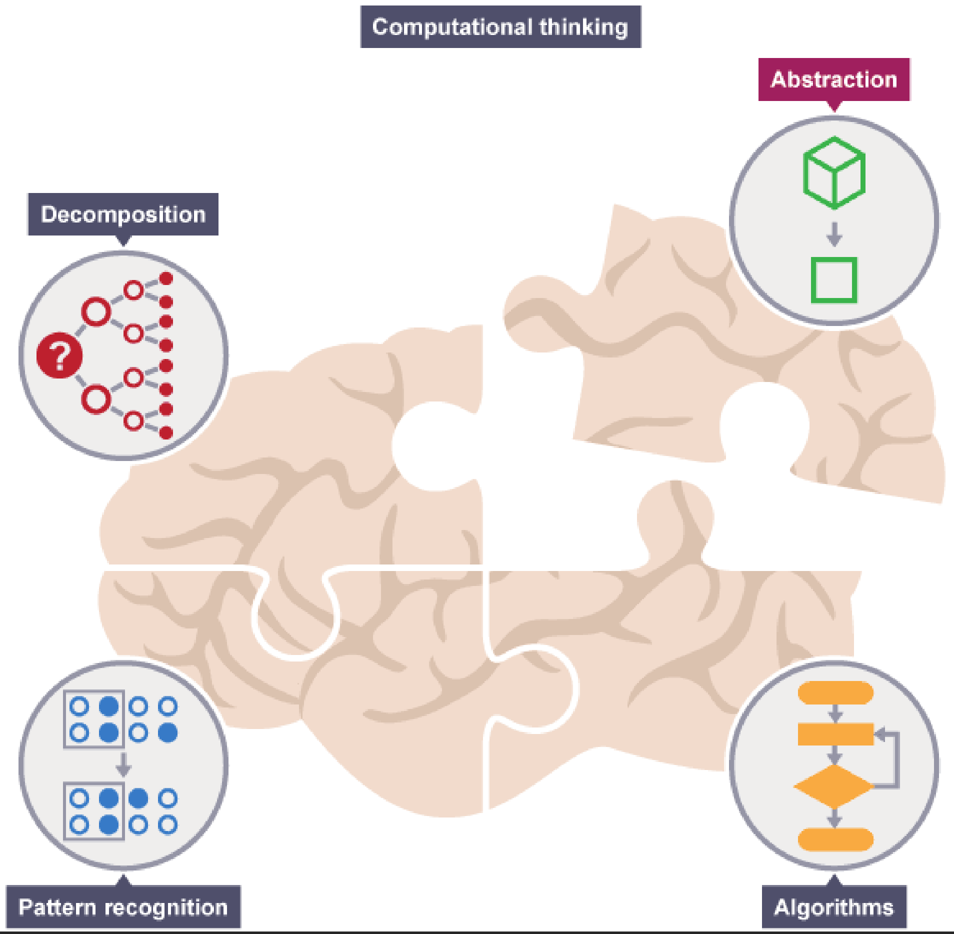
- each cake will need a precise quantity of specific ingredients
- ingredients will get added at a specific time
- each cake will bake for a specific period of time

Recognizing patterns in baking



Once we have recognized patterns in our problems, we use abstraction to gather the general characteristics and to filter out of the details we do not need in order to solve our problem.

Pattern Abstraction



Pattern Abstraction

It helps students learn to identify the details that are relevant to solving the problem and ignoring the details that aren’t.

It involves separating core information from extraneous details.

From this we create a representation (idea) of what we are trying to solve.

Abstraction allows us to create a general idea of what the problem is and how to solve it.

If we don’t abstract we may end up with the wrong solution to the problem we are trying to solve.

Teaching Pattern Abstraction

By teaching students abstraction, they are able to sort through all of the information available to identify the specific information they need.

Younger students may benefit from a building activity where a variety of extra pieces and objects are given that aren’t part of the design.

Ideas to try:

- ❖Ask your students to create a model of a cat.
- ❖Tell the students that the model needs to represent all cats.
- ❖Students would then need to abstract patterns that all cats share.

Abstracting patterns in baking

When baking a cake, there are some general characteristics between cakes. For example:

- a cake needs ingredients
- each ingredient needs a specified quantity
- a cake needs timings

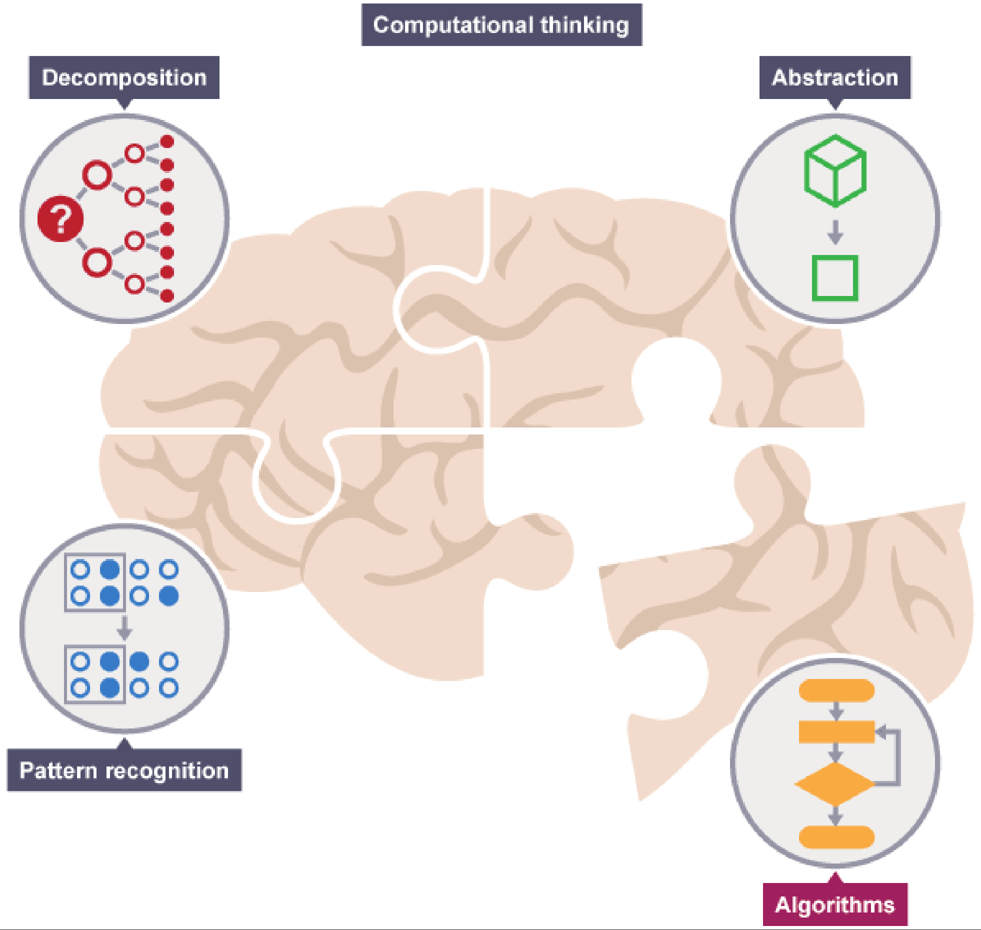
Abstracting patterns in baking

When abstracting, we remove specific details and keep the general relevant patterns.

General patterns	Specific details
We need to know that a cake has ingredients	We don't need to know what those ingredients are
We need to know that each ingredient has a specified quantity	We don’t need to know what that quantity is
We need to know that each cake needs a specified time to bake	We don't need to know how long the time is

After pattern abstraction, we need to be able to arrange these in a specific order to get the task done.

Algorithm


Computational thinking
Decomposition
Abstraction
Pattern recognition
Algorithms

Algorithm

An algorithm is a plan, a set of step-by-step instructions to resolve a problem.

In an algorithm, each instruction is identified and the order in which they should be carried out is planned.

Algorithms are often used as a starting point for creating a computer program.

Computers are only as good as the algorithms they are given.

An algorithm must have a starting point, a finishing point and a set of clear instructions in between.

Teaching Algorithm

An easy way to teach this concept to young learners is to give them a task and tell them to write down the steps.

To get students thinking in algorithms, invite them to design the path from their classroom to the ground by detailing a series of steps.

Invite students to think about their morning routine. What steps do they take to get ready for school each morning? How would the order impact the outcome?

Ideas to Try:

- ❖You might ask them to think about making a sandwich.
- ❖What should we do first? Second?
- ❖Conversations about sequence and order develop the foundations of algorithmic thinking.

Algorithm of baking a cake

Applied Exercise

For this activity, no instructions are provided.

This lesson gives students the opportunity to practice the four arts of computational thinking (decomposition, pattern matching, abstraction, and algorithms) in one cohesive activity.

- Getting started – 15 minutes
  - Vocabulary
  - Figuring out
- Activity – 25 minutes
- Wrap-up – 5 minutes
  - Flash chat
  - Vocab Shmocab

Computational thinking is the **step that comes before** programming. It’s **the process of breaking down a problem into simple enough steps** that even a computer would understand.

Computational Thinking in Practice

Computational thinking involves

- ❖taking that complex problem and breaking it down into a series of small, more manageable problems (**decomposition**).
- ❖Each of these smaller problems can then be looked at individually, considering how similar problems have been solved previously (**pattern recognition**).
- ❖Focus only on the important details, while ignoring irrelevant information (**abstraction**).
- ❖Simple steps or rules to solve each of the smaller problems can be designed (**algorithms**).

These simple steps or rules are used to **program** a computer to help solve the complex problem in an optimized way.

Module Outcomes

UNDERSTANDING COMPUTATIONAL THINKING

APPLIED COMPUTATIONAL THINKING EXERCISES

INTRODUCTION TO FLOW CHARTS & PSEUDO-CODE

FUNDAMENTAL OPERATIONS OF A COMPUTER PROGRAM

Algorithm of a Morning Routine

- Wake up and turn off alarm
- Get dressed
- Brush teeth
- Eat breakfast
- Go to school

Algorithm of Double Digit Multiplication

- Write the double-digit numbers on top of each other.
- Multiply the bottom one's number by the top one's number.
- Multiply the bottom one's number by the number of the top ten.
- Place a zero under your result.
- Multiply the bottom tens number by the top one's number.
- Multiply the bottom tens number by the number of the top ten.
- Add both of your results to get a final answer.

Representing an Algorithm

There are two main ways that **algorithms** can be represented –

- Pseudocode
- Flowcharts

Pseudocode

Pseudocode is a form of Structured English for writing an algorithm.

Pseudocode is **NOT** a programming language.

You do not need to worry about the detailed syntax.

Writing in pseudocode is similar to writing in a programming language.

Writing in pseudocode helps you concentrate on the logic (Process) and efficiency of your algorithm.

By learning to read and write pseudocode, we can easily communicate ideas and concepts to other programmers.

Writing a Pseudocode

A pseudocode for buying 10 chocolates

```
START
OUTPUT “I want 10 Cadbury silk chocolates”
INPUT Price of one chocolate
STORE Price of one chocolate
PROCESS 10*Price of one chocolate = Total
OUTPUT “Here are” + Total + “rupees”
STOP
```

Writing a Pseudocode

A program can be created to ask someone their name and age, & comment based on these.

Ask the person their name.

Say Hello followed by the name of the person.

Ask the person their age.

Say "You are aged to perfection" if the person is 70 years or older.

Say "You are a spring chicken!" if the person is younger than 70 years.

Writing a Pseudocode

```
OUTPUT 'What is your name?'

INPUT user inputs their name

STORE the user's input in the name variable

OUTPUT 'Hello' + name

OUTPUT 'How old are you?'

INPUT user inputs their age

STORE the user's input in the age variable

IF age >= 70 THEN
        OUTPUT 'You are aged to perfection!'
ELSE
        OUTPUT 'You are a spring chicken!'
```

Flowcharts

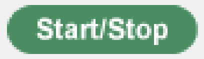
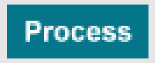
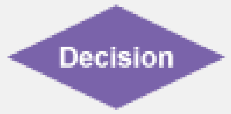
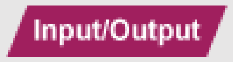
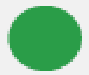
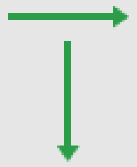
A flowchart is a graphical representation of the sequence of operations in a program.

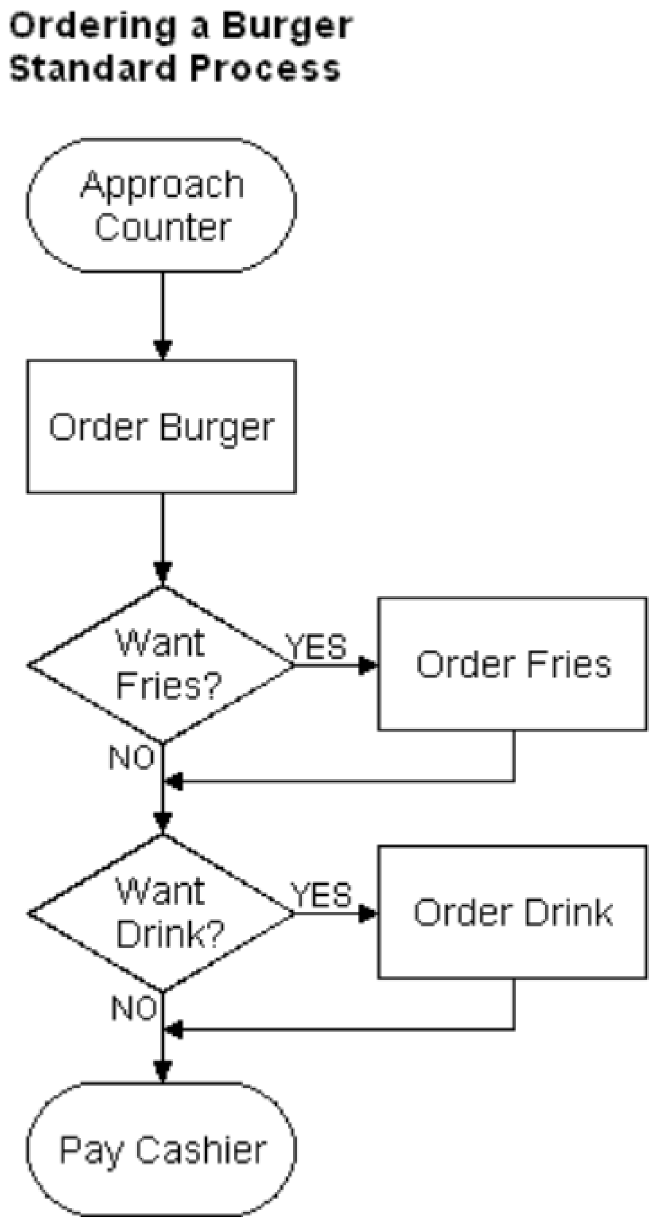
An algorithm can be represented graphically using a flowchart.

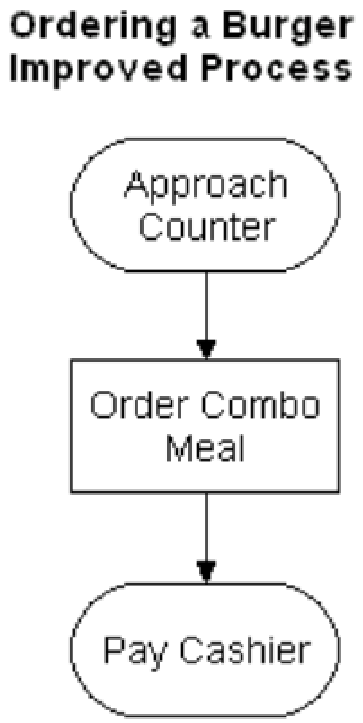
Flowcharts normally use standard symbols to represent the different instructions.

A flowchart illustrates the steps in a process.

By visualizing the process, a flowchart can quickly help identify bottlenecks or inefficiencies where the process can be streamlined or improved.

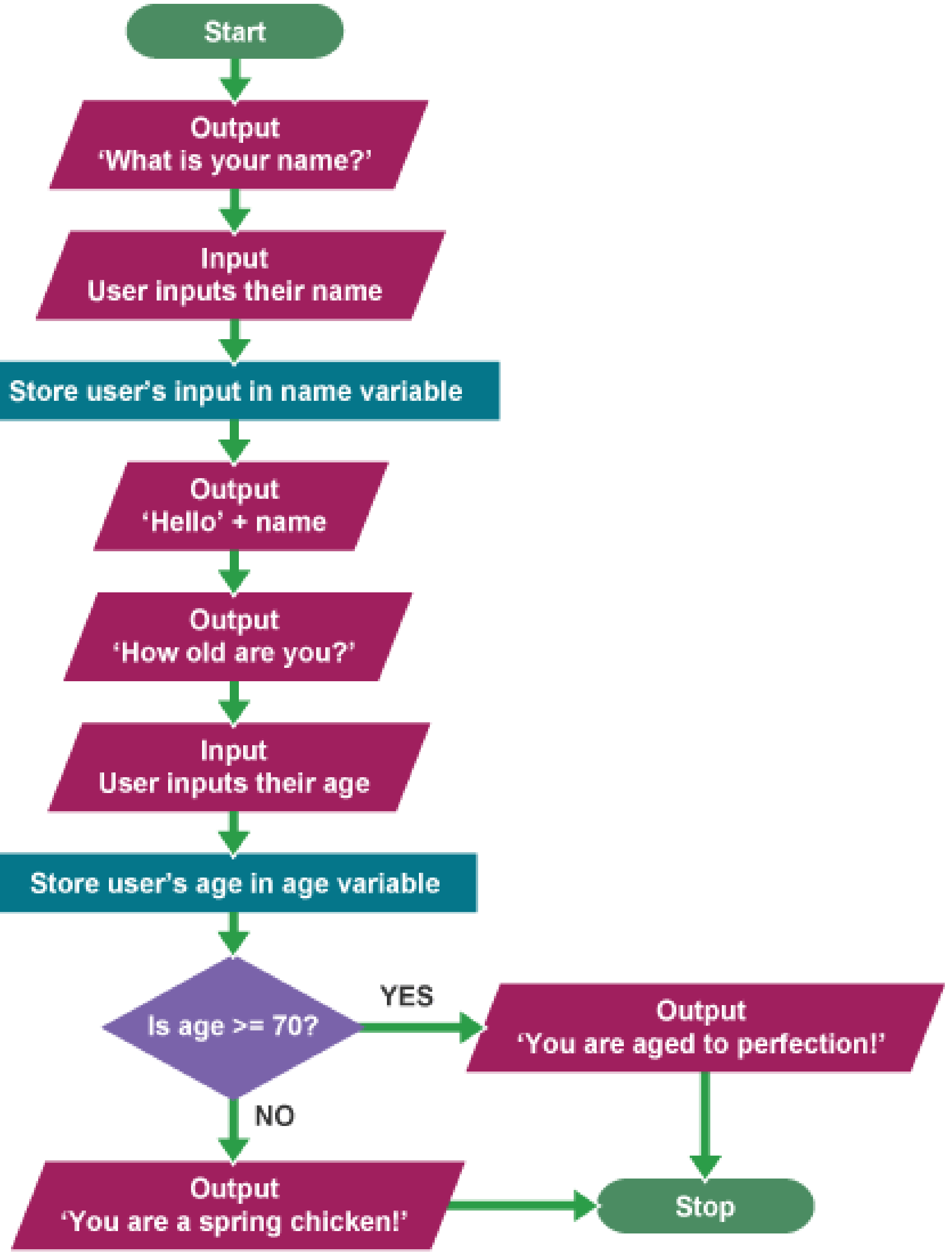
Name	Symbol	Usage
Start or Stop		The beginning and end points in the sequence.
Process		An instruction or a command.
Decision		A decision, either yes or no.
Input or Output		An input is data received by a computer. An output is a signal or data sent from a computer.
Connector		A jump from one point in the sequence to another.
Direction of flow		Connects the symbols. The arrow shows the direction of flow of instructions.


Ordering a Burger
Standard Process
Approach Counter
Order Burger
Want Fries?
YES
Order Fries
NO
Want Drink?
YES
Order Drink
NO
Pay Cashier


Ordering a Burger
Improved Process
Approach Counter
Order Combo Meal
Pay Cashier

Flowcharts

Let’s create a Flowchart for a program to ask someone their name and age, & comment based on these.


Start
Output
'What is your name?'
Input
User inputs their name
Store user's input in name variable
Output
'Hello' + name
Output
'How old are you?'
Input
User inputs their age
Store user's age in age variable
Is age >= 70?
YES
Output
'You are aged to perfection!'
NO
Output
'You are a spring chicken!'
Stop

Fundamental Operations of a Computer Program

Each computer program performs a few fundamental operations.

- Input - This is the act of feeding in the data and instruction to the computer.
- Processing - The task of performing calculations and comparisons are known as processing.
- Output – This is presenting the information to the user in suitable form.
- Decision making – Conditional statements to be executed in different cases.
- Loops – This repeats a sequence of instructions until a specific condition is met.

Input

This is the act of feeding in the data and instruction to the computer.

It is to send data and/or instruction to the computer in the required format.

Information and programs are entered into the computer through Input devices.

The input device also retrieves information off memory storage.

Processing

The task of performing calculations and comparisons are known as processing.

The unit in Computer System that is responsible for processing is ALU (Arithmetic and Logical Unit).

All calculations & comparisons are made in the ALU.

The data and instructions stored in the primary storage are transferred to it as when required.

After completion of processing the final results are send to storage units from ALU.

Output

Output is receiving processed information from processing unit and present it to the user in the suitable form.

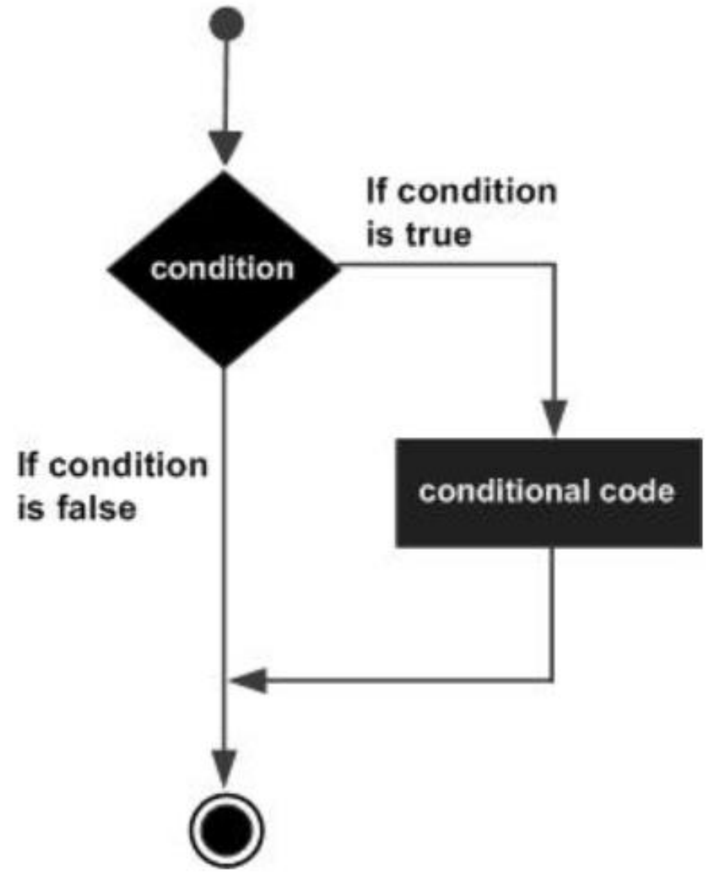
The devices that can output information from a computer are known as output unit devices

Decision Making

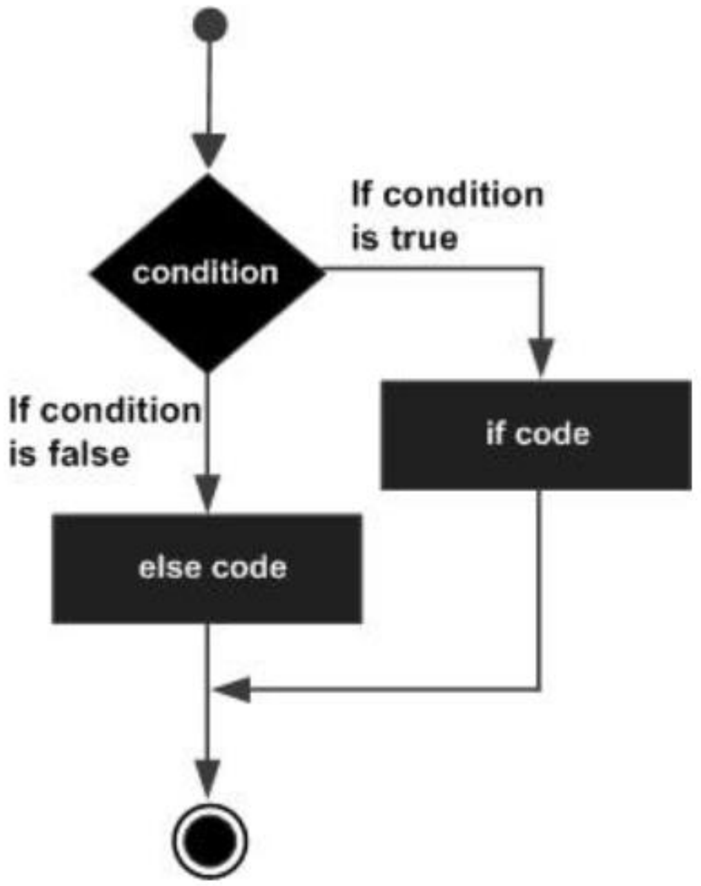
The programmer specifies one or more conditions to be evaluated or tested by the program

- A statement or statements to be executed if the condition is determined to be true,
- and optionally, other statements to be executed if the condition is determined to be false.

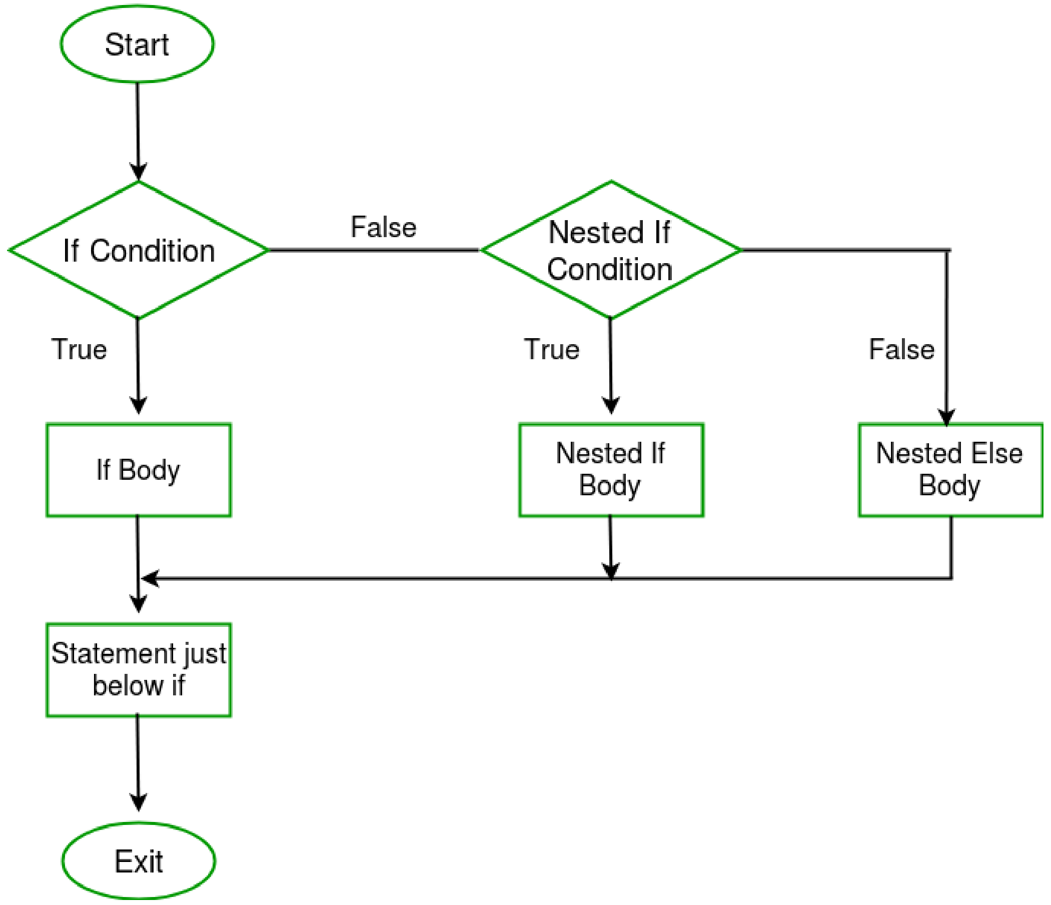
If statement



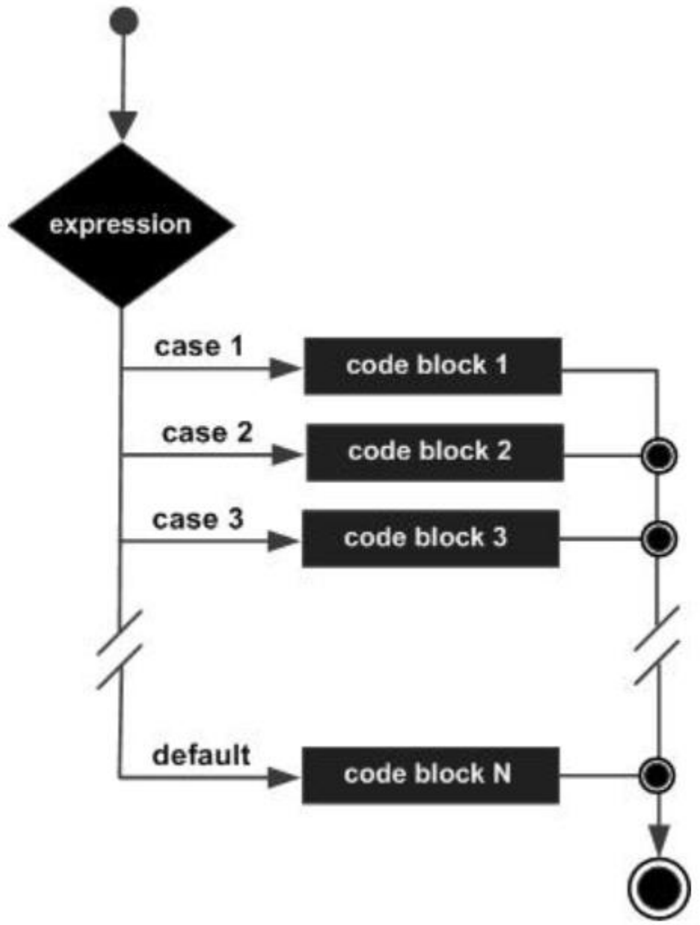
If-else statement



Nested If-Else statement



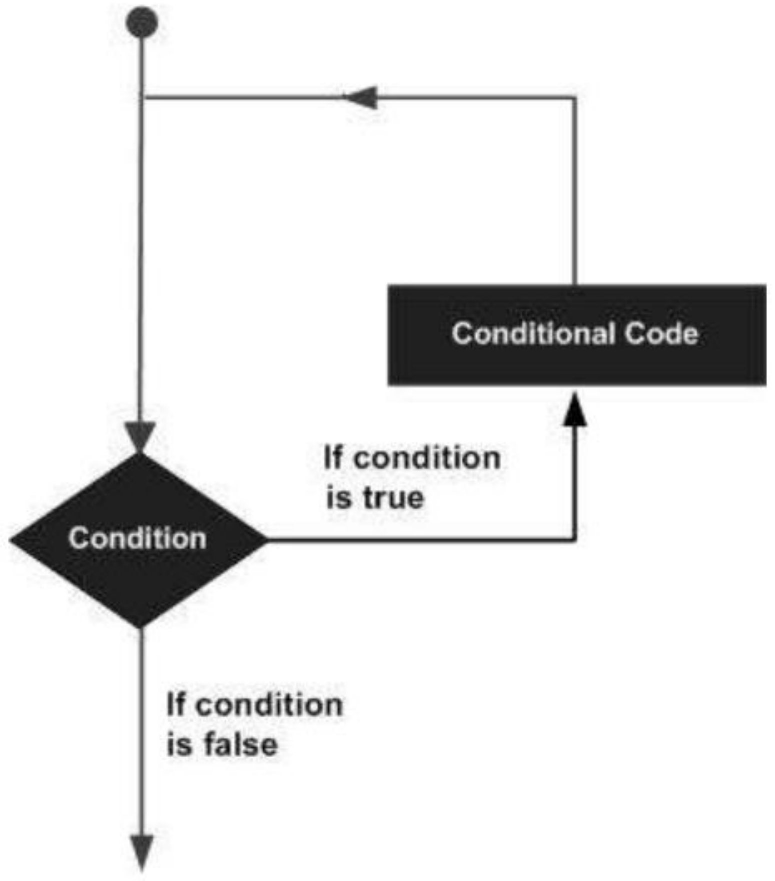
Switch statement



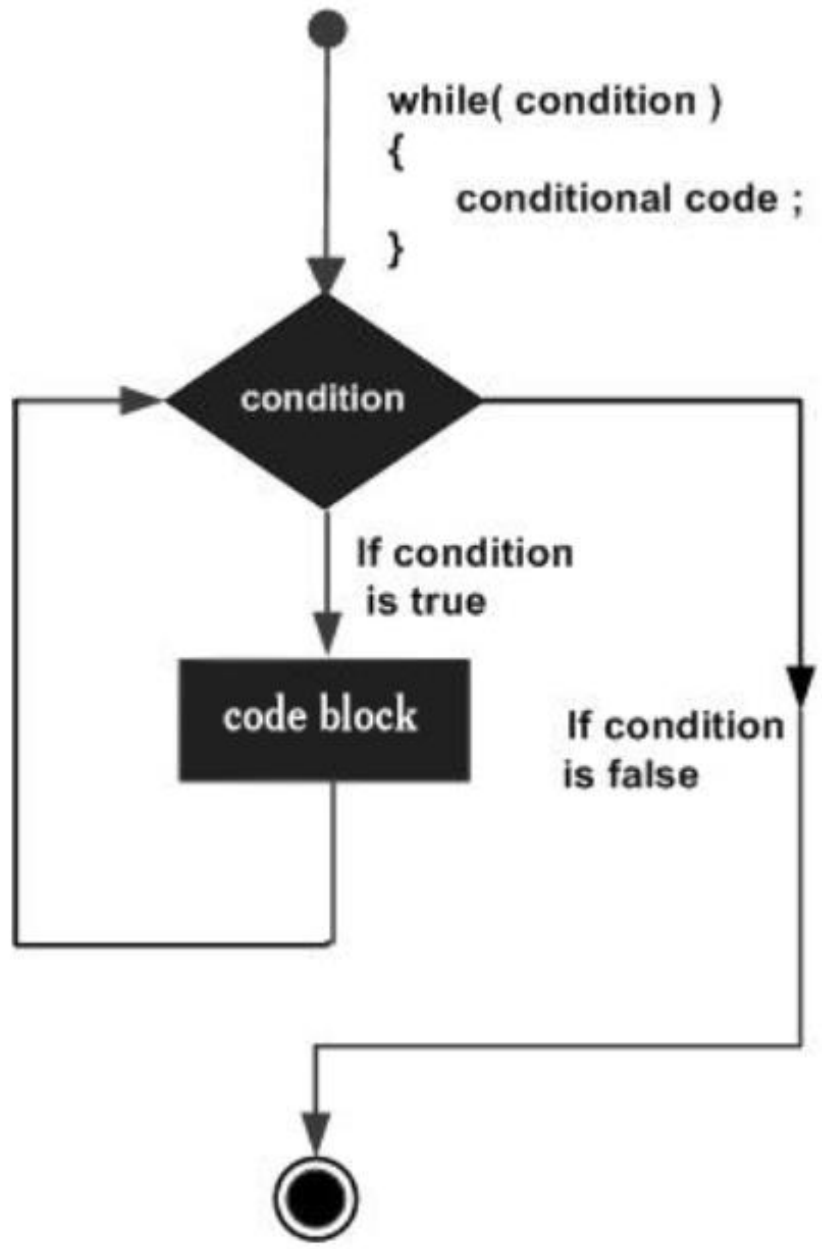
Loops

Almost all the programming languages provide a concept called loop.

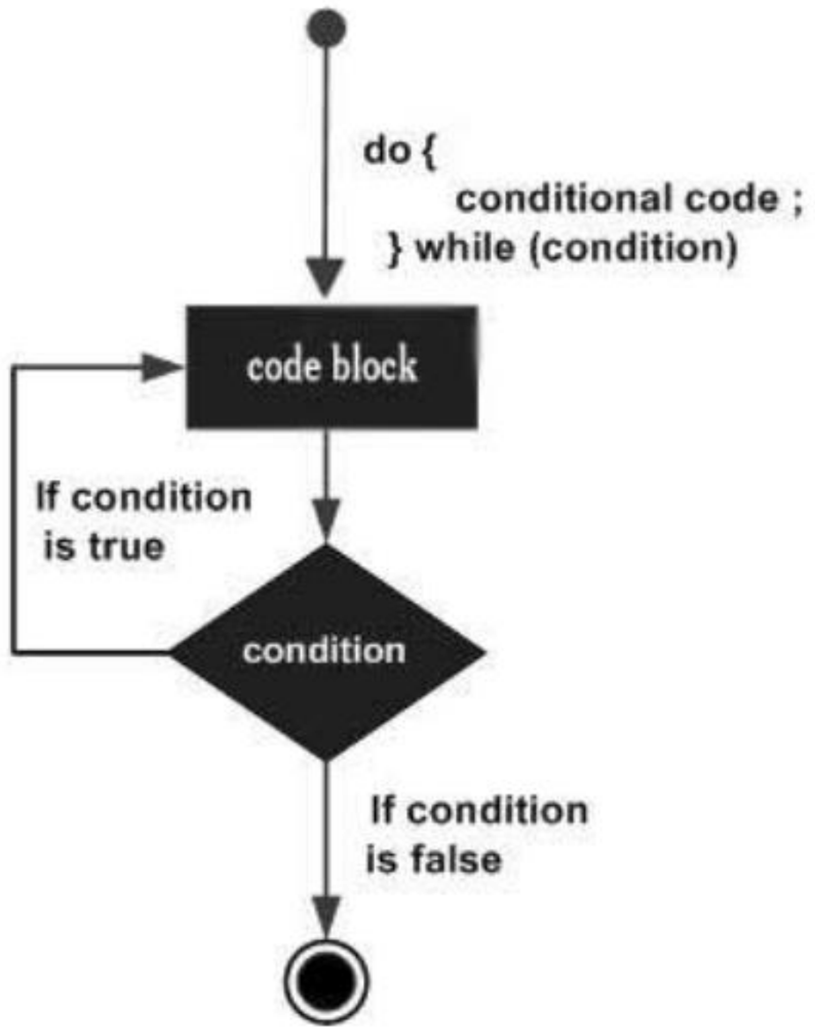
Loops help in executing one or more statements up to a desired number of times or until a desired condition is met.



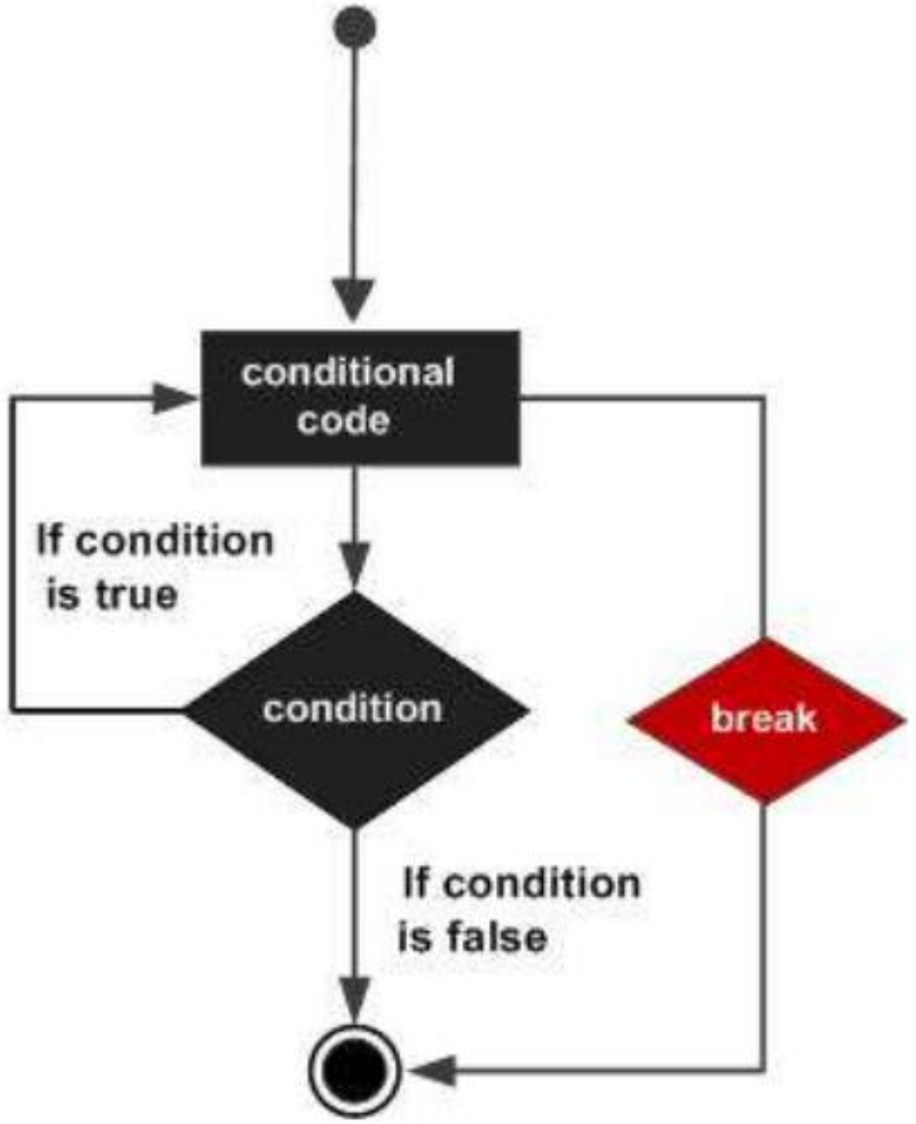
While loop


while(condition)
{
conditional code ;
}
condition
If condition is true
code block
If condition is false

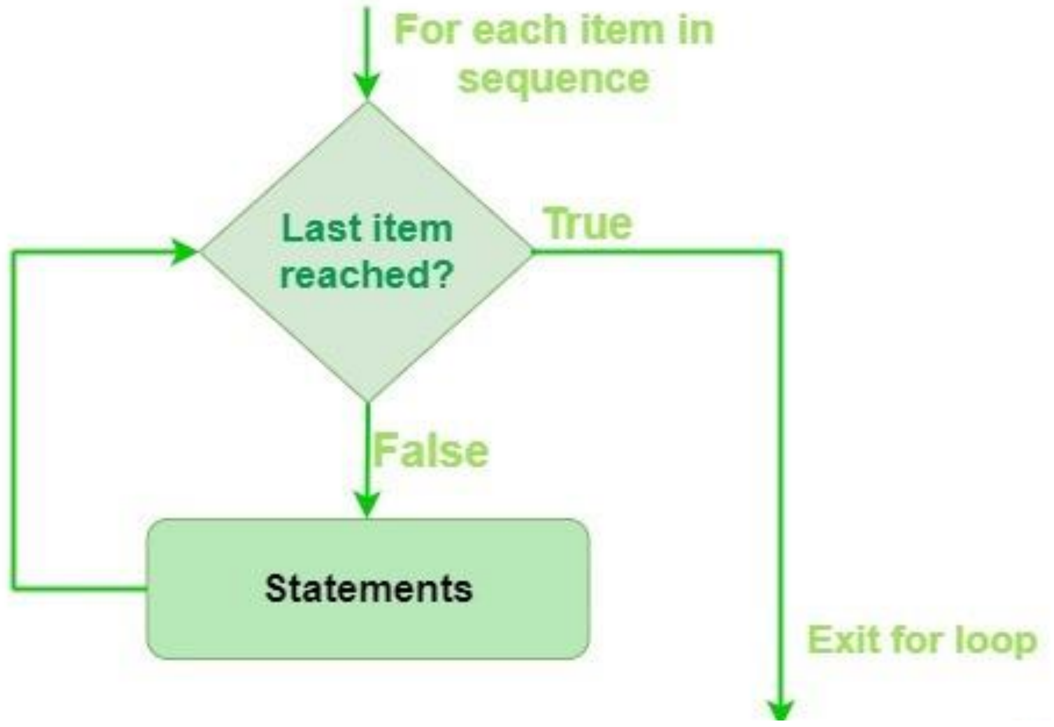
Do-While loop



Break statement



For loop


For each item in sequence
Last item reached?
True
False
Statements
Exit for loop

Thank you!
☺