

Course Unit 3: Week 4

Introduction to Python

Topics

- **Basic Elements of Programming**
- Programming Structure
- Programming Environments
- Programming Languages
- Python



Objectives

1. Describe python language.
2. Discuss the different programming elements, structure and environments.
3. Differentiate the different programming languages.

What are the Elements of Programming?

Variables

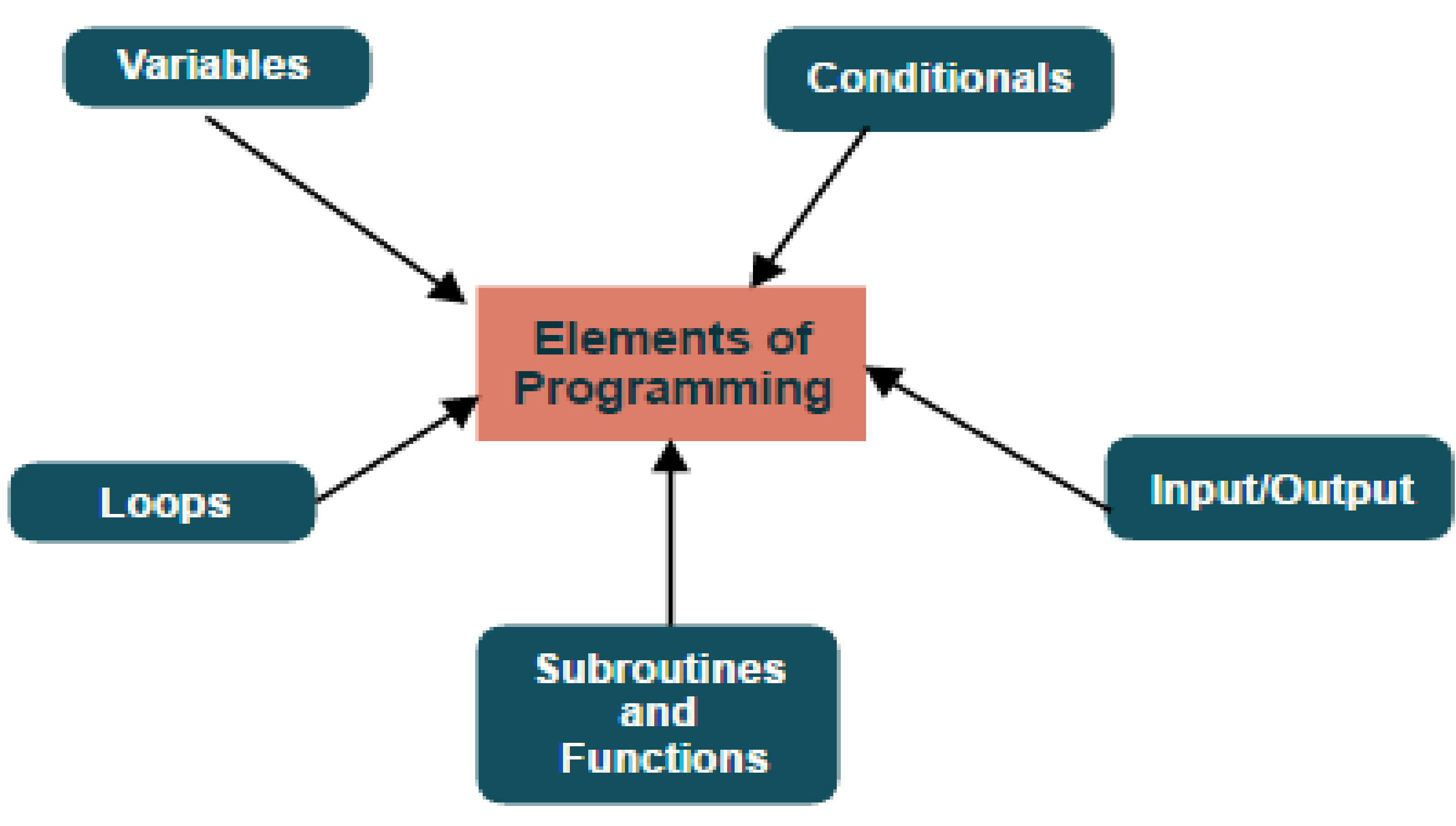
Conditionals

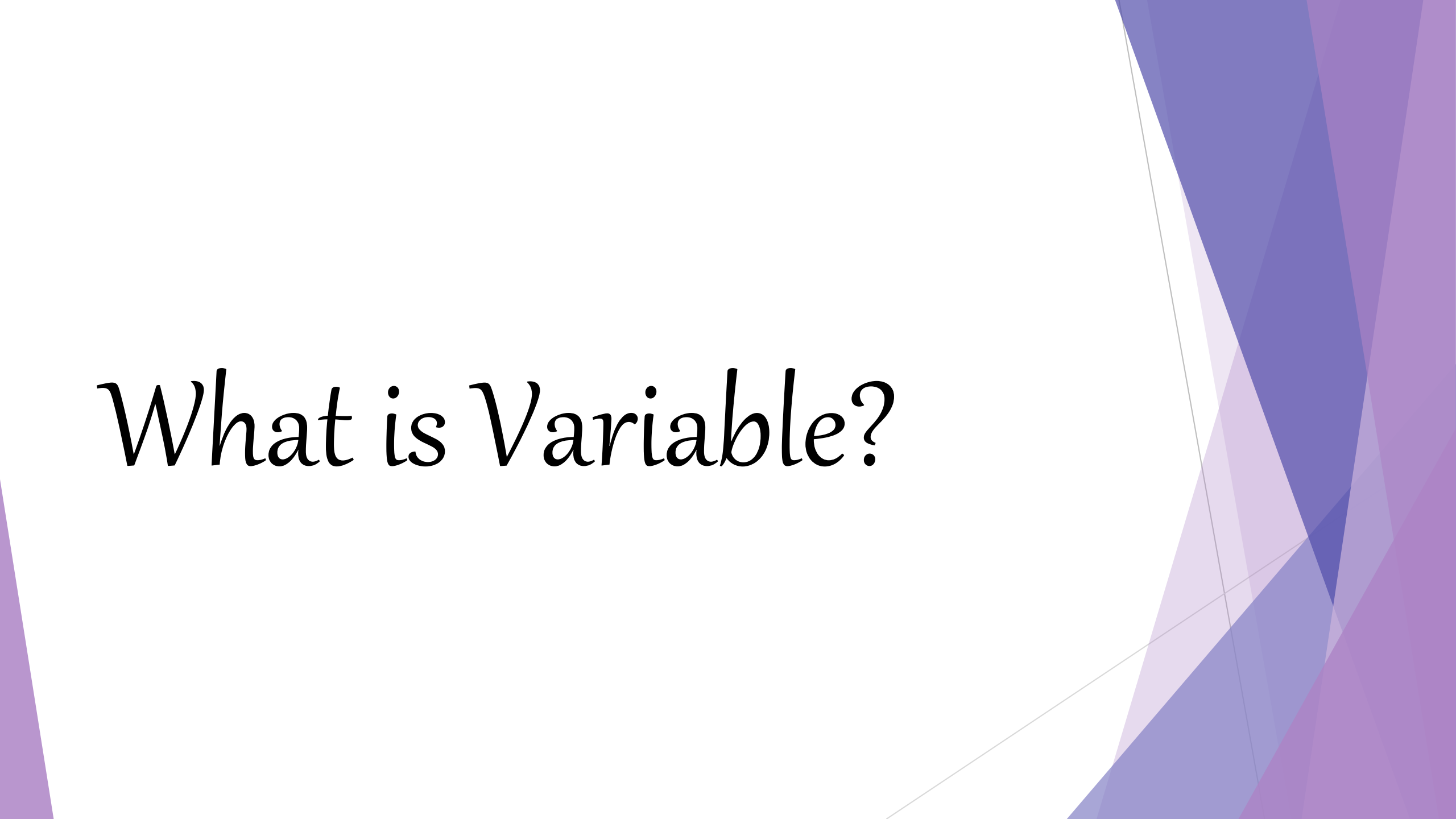
**Elements of
Programming**

Loops

Input/Output

**Subroutines
and
Functions**

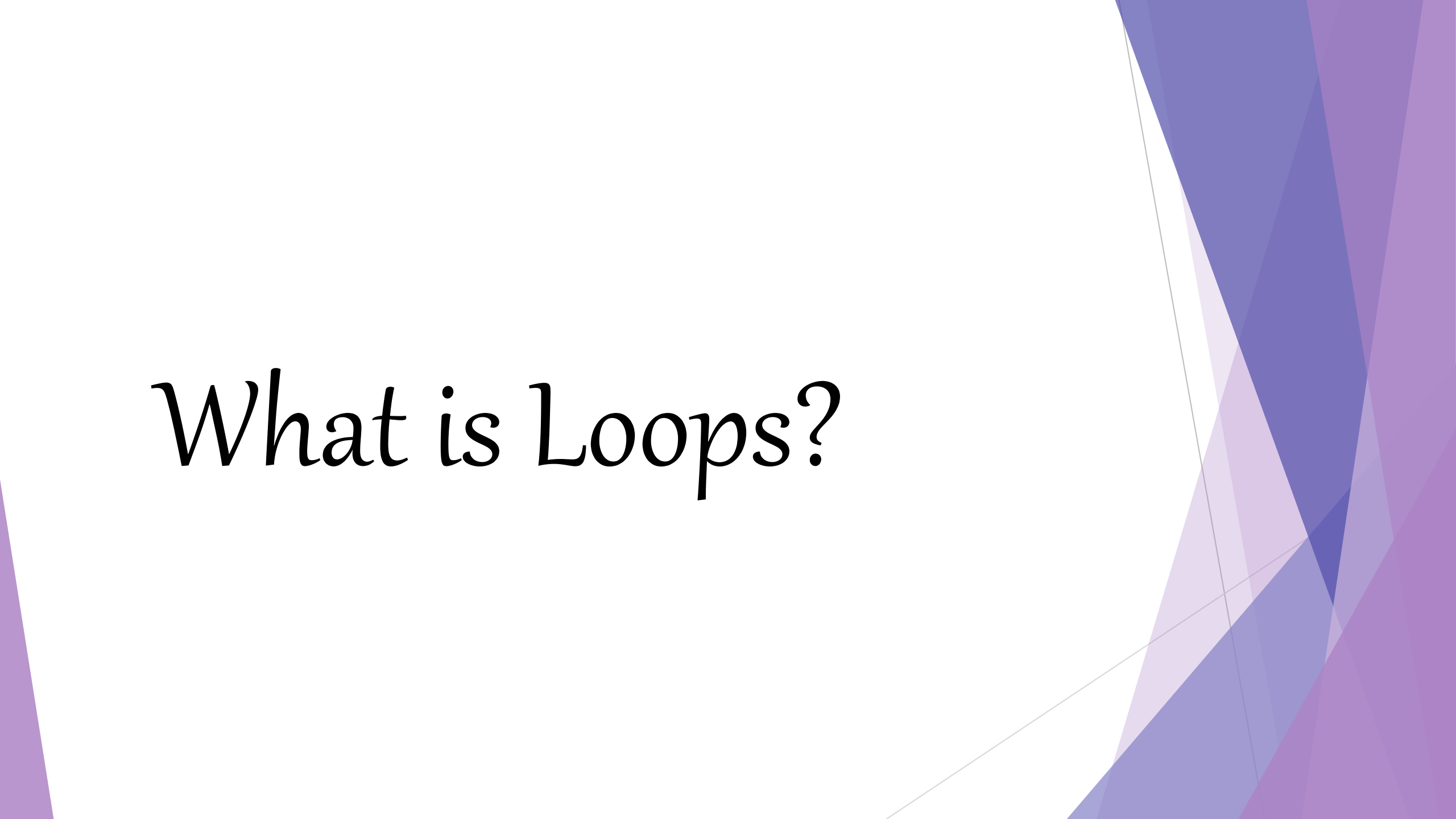


The background of the slide features an abstract design composed of overlapping, semi-transparent geometric shapes in various shades of purple and blue. These shapes, including triangles and polygons, are concentrated on the right side of the frame, creating a dynamic, layered effect. The left side of the slide is a plain white background.

What is Variable?

Variable

- tells how the data is represented which can be range from very simple value to complex one.
- The value they contain can be change depending on condition.
- Data is constant with the fixed values or variable. They can hold a very simplex value like an age of the person to something very complex like a student track record of his performance of whole year.

The background of the slide features abstract, overlapping geometric shapes in various shades of purple and blue, primarily concentrated on the right side and bottom, creating a modern, layered effect.

What is Loops?

LOOPS

- a sequence of instructions that are repeated continuously till a certain condition is not satisfied.
- Basically a loop carry out execution of a group of instruction of commands a certain number of times.
- There is also a concept of infinite loop which is also termed as endless loop is a piece of code that lack from functional exit and goes to repeat indefinitely.

The background features abstract, overlapping geometric shapes in various shades of purple and blue, primarily concentrated on the right side of the frame. The shapes include triangles and polygons of different sizes and orientations, creating a dynamic, layered effect. The colors range from light lavender to deep indigo.

What is conditionals?

Conditionals

- specify the execution of the statements depending on the whether condition is satisfied or not.
- refers to an action that only fulfil when the applied condition on instructions satisfied.
- give freedom to program to act differently every time when it execute that depends on input to the instructions.

What is
INPUT/OUTPUT?

INPUT/OUTPUT

- the element of computer programming allow interaction of the program with the external entities. Example of input/output element are printing something out to the terminal screen, capturing some text that user input on the keyboard and can be include reading and writing files.

*What is Subroutines and
functions?*

Subroutines and functions

- the element of the programming allow a programmer to use snippet of code into one location which can be used over and over again.
- The primary purpose of the functions is to take arguments in numbers of values and do some calculation on them after that return a single result.
- Functions are required where you need to do complicated calculations and the result of that may or may not be used subsequently used in an expression.

Topics

- **Basic Elements of Programming**
- **Programming Structure**
- **Programming Environments**
- **Programming Languages**
- **Python**



Behind all of the software we use on a daily basis, there's a code being run with all sorts of terms and symbols.



Surprisingly, it can often be broken down into three simple *programming structures* called **sequences**, **selections**, and **loops**.



STRUCTURE

- is a data organization, management, and storage format that enables efficient access and modification.
- is a programming paradigm aimed at improving the clarity, quality, and development time of a computer program by making extensive use of the structured control flow.

There are three main categories of Structures:

- **Sequence**
- **Selection**
- **Iteration/ Loop**

Let's watch a video first

Video Source: [Computer Science Basics: Sequences, Selections, and Loops - YouTube](#)

Let's watch a video first

Video Source: [What is Sequence? | Coding for Kids | Kodable - YouTube](#)

What is Sequences?

SEQUENCES

- A sequence in programming refers to an ordered set of instructions or tasks. In common parlance, one may also use the term “algorithm” which can be defined as an ordered sequence of steps to achieve a particular task. The order is key here.

Algorithms

- Algorithms are fundamental to sequences in programming, and they describe in detail how any task is to be achieved in a step-by-step ordered manner. Algorithms can also be said to be a series of steps taken to achieve a given task.

Examples of SEQUENCES

- Morning routine
- Brushing teeth
- DIY Projects
- Cooking

A computer can only
do what it is
programmed to do.
If the steps are
programmed in the
wrong sequence, the
computer will
perform the tasks in
this sequence - even
if this is incorrect.



Complex algorithms may have hundreds, if not thousands, of steps. It is critical to make sure all steps in the algorithm are in the correct sequence before programming begins. Once programmed, trying to find an instruction in the wrong sequence can be extremely difficult.



What is Selections?

SELECTIONS

- This is where you select or choose between two or more flows. The choice is decided by asking some sort of question. The answer determines the path (or which lines of code) will be executed.

Selection is a decision or question. At some point in an algorithm there may need to be a question because the algorithm has reached a step where one or more options are available. Depending on the answer given, the algorithm will follow certain steps and ignore others.



Selection allows us to include more than one path through an algorithm. Without selection, different paths would not be included in algorithms. This means that the solutions created would not be realistic.

For example, a simple algorithm can be created to determine correct bus fares. The steps could be:

1. ask how old you are
2. if you are under 16, pay half fare
3. otherwise pay full fare

The decision comes in step 2. If you are aged less than 16, one fare is charged. Otherwise, a different fare is charged.

IF...THEN...ELSE

Selection allows several paths to be included in an algorithm. In algorithms (and in programming), selection is usually represented by the instructions IF, THEN and ELSE.

- IF represents the question
- THEN points to what to do if the answer to the question is true
- ELSE points to what to do if the answer to the question is false

Using this method, the fare-related algorithm now reads like this:

1. ask how old you are
2. IF you are under 16, THEN pay half fare
3. ELSE pay full fare

> If you try this algorithm using 15 as your age, the answer to the question at step 2 is true, so the algorithm tells you to pay half fare.

> If you try the algorithm using 16 as your age, the answer to the question at step 2 is false, so the algorithm tells us to pay full fare.

Two different paths are followed according to the answer given to a particular question.

Look at this simple six-step algorithm for comparing your dog's age with your own:

- ask how old the dog is in human years
- multiply human years by seven to find out how old the dog is in dog years
- print the answer on the screen
- ask how old you are
- if the dog's age in dog years is older than your age, say 'Your dog is older than you!'
- otherwise, say 'Your dog is not older than you.'

In pseudocode, the algorithm would look like this:

OUTPUT 'How old is your dog?'

INPUT user inputs their dog's age in human years

STORE the user's input in the human_years variable

dog_years = human_years * 7

OUTPUT 'In dog years, your dog is aged ' + dog_years

OUTPUT 'How old are you?'

INPUT user inputs their age

STORE the user's input in the user_age variable

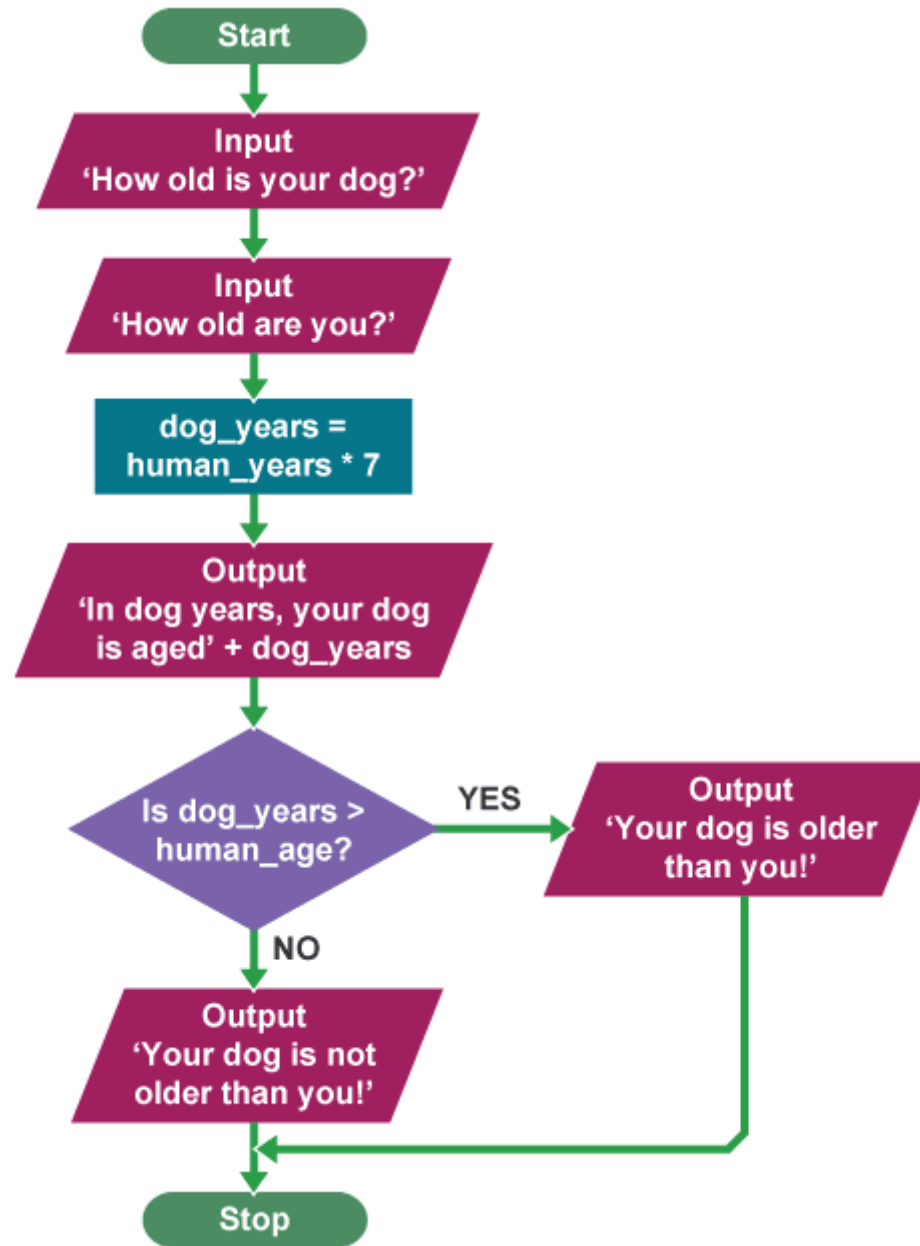
IF dog_years > user_age THEN

 OUTPUT 'Your dog is older than you!'

ELSE

 OUTPUT 'Your dog is not older than you.'

FLOWCHART



The ELSE IF instruction

The ELSE IF instruction allows there to be more than two paths through an algorithm. Any number of ELSE IF instructions can be added to an algorithm. It is used along with other instructions:

- IF represents a question
- THEN points to what to do if the answer to the question is true
- ELSE IF represents another question
- THEN points to what to do if the answer to that question is true
- ELSE IF represents another question
- THEN points to what to do if the answer to that question is true
- ELSE points to what to do if the answer to the question is false

The ELSE IF instruction

Using this method, the bus fare algorithm could be improved like this:

- ask how old you are
- IF you are under 5, THEN pay no fare
- ELSE IF you are under 16, THEN pay half fare
- ELSE IF you are an OAP, THEN pay no fare
- ELSE pay full fare

As this algorithm uses ELSE IF statements, a 14 year old would stop at step 3. However, if the algorithm was made up of three different IF statements, the same 14 year old would have to check to see if they were an OAP (step 4) even though they had already found the correct option for themselves.

What is Loops/ Iteration?

LOOPS/ITERATION

- Iteration - Also known as repetition, it allows some code (one to many lines) to be executed (or repeated) several times. The code might not be executed at all (repeat it zero times), executed a fixed number of times or executed indefinitely until some condition has been met. Also known as looping because the flowcharting shows the flow looping back to repeat the task.

Iteration allows algorithms to be simplified by stating that certain steps will repeat until told otherwise. This makes designing algorithms quicker and simpler because they don't need to include lots of unnecessary steps.

*Is the nail
all the way
in?*

NO



Pop up a new rat

Hit the rat

SCORE

Increase score



For example, a very simple algorithm for eating breakfast cereal might consist of these steps:

1. put cereal in bowl
2. add milk to cereal
3. spoon cereal and milk into mouth
4. repeat step 3 until all cereal and milk is eaten
5. rinse bowl and spoon

Suppose a person has ten top teeth. To make sure that every one of the top teeth is cleaned, the algorithm would look something like this:

1. put toothpaste on toothbrush
2. use toothbrush to clean tooth 1
3. use toothbrush to clean tooth 2
4. use toothbrush to clean tooth 3
5. use toothbrush to clean tooth 4
6. use toothbrush to clean tooth 5
7. use toothbrush to clean tooth 6
8. use toothbrush to clean tooth 7
9. use toothbrush to clean tooth 8
10. use toothbrush to clean tooth 9
11. use toothbrush to clean tooth 10
12. rinse toothbrush

Steps 2 through to 11 are essentially the same step repeated, just cleaning a different tooth every time. Iteration can be used to greatly simplify the algorithm. Look at this alternative:

1. put toothpaste on toothbrush
2. use toothbrush to clean a tooth
3. move onto next tooth
4. repeat steps 2 and 3 until all teeth are clean
5. rinse toothbrush

This algorithm is much simpler. However, there is a problem - how do we know when all teeth are clean (step 4)? A condition is needed to solve this problem.

A condition is a situation that is checked every time an iteration occurs.

Suppose a person has ten top teeth. To make sure that every one of the top teeth is cleaned, the algorithm would look something like this:

1. put toothpaste on toothbrush
2. use toothbrush to clean tooth 1
3. use toothbrush to clean tooth 2
4. use toothbrush to clean tooth 3
5. use toothbrush to clean tooth 4
6. use toothbrush to clean tooth 5
7. use toothbrush to clean tooth 6
8. use toothbrush to clean tooth 7
9. use toothbrush to clean tooth 8
10. use toothbrush to clean tooth 9
11. use toothbrush to clean tooth 10
12. rinse toothbrush

The condition, in this case, will be to check if the number of teeth cleaned equals ten. If that condition is **False** (the number of teeth cleaned is less than ten), then another iteration occurs. If the condition is **True** (the number of teeth cleaned equals ten), then no more iterations occur.

It is also important to keep a count of how many teeth have been cleaned. A **counter** is used to do this.

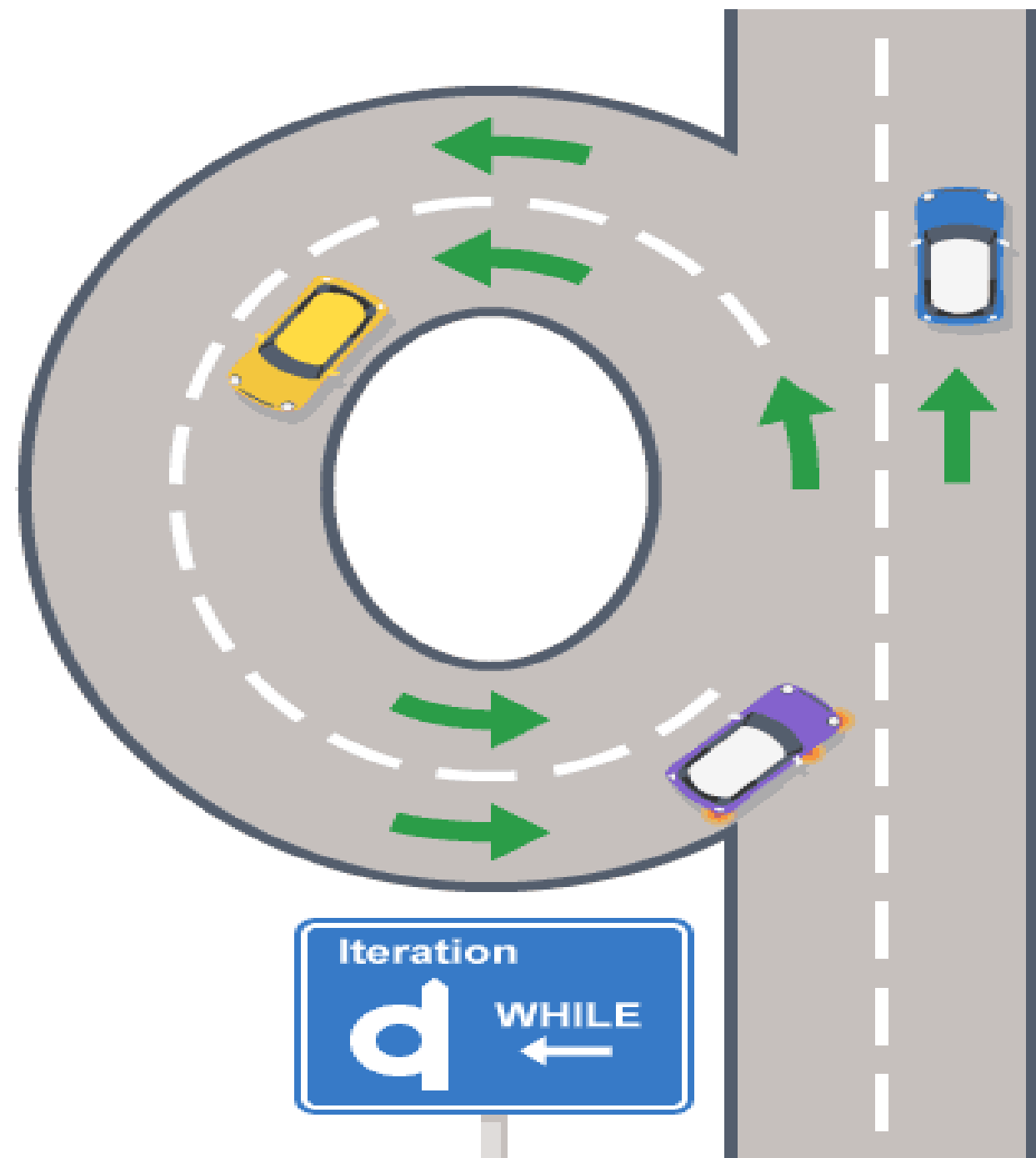
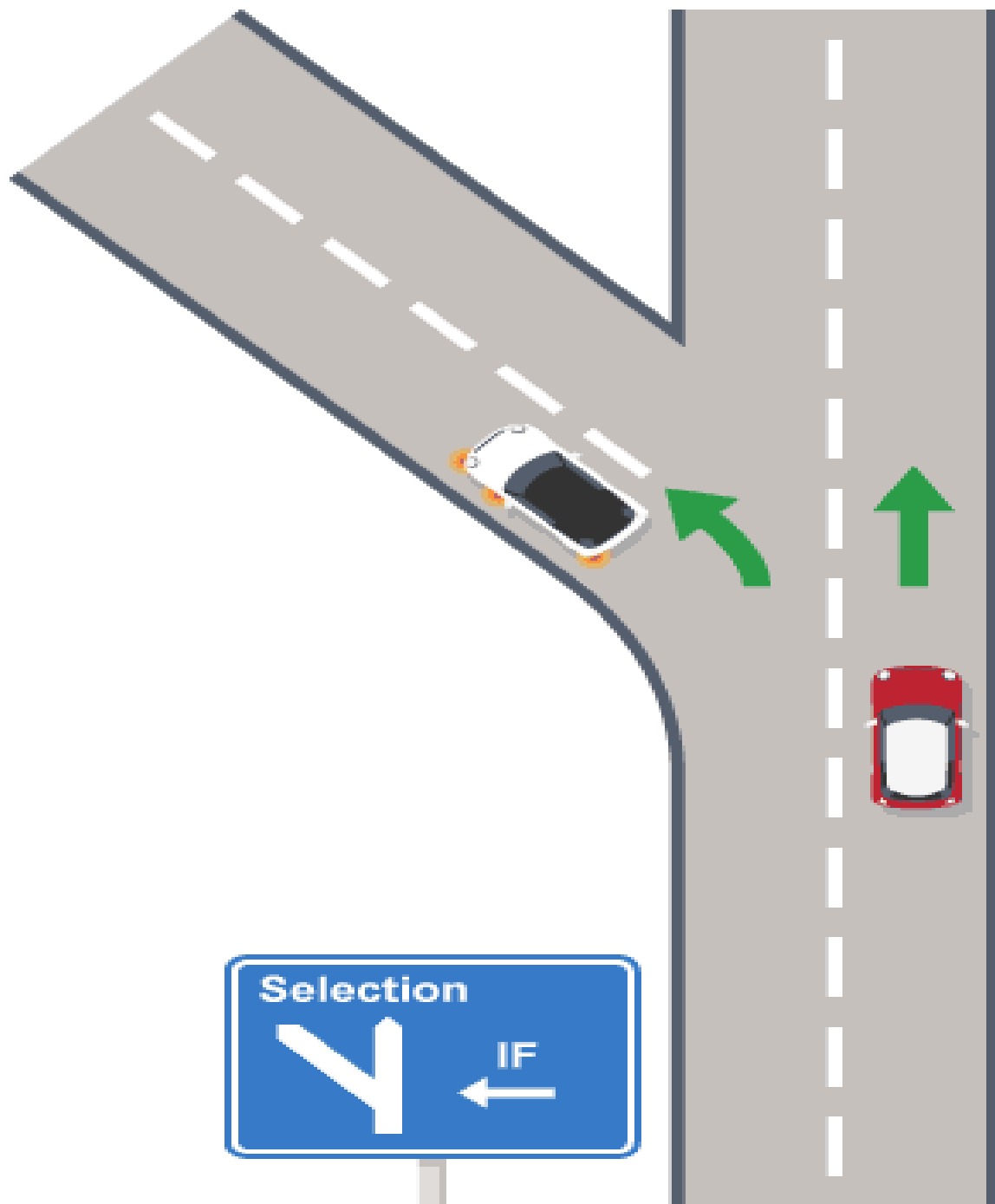
The counter can be used to see if the condition has been met. This is known as **testing the condition**. The test sees whether the condition is True or False.

If a condition and a counter are included, the algorithm would look like this:

1. set number_of_teeth_cleaned to 0
2. put toothpaste on toothbrush
3. use toothbrush to clean a tooth
4. increase number_of_teeth_cleaned by 1
5. if number_of_teeth_cleaned < 10 then go back to step 3
6. rinse toothbrush

The counter is called number_of_teeth_cleaned, and the condition is 'if number_of_teeth_cleaned < 10'.

This is very similar to selection as two routes are created. However, with iteration there is a loop back into the algorithm, rather than creating and keeping two separate routes as would occur with selection.




Most adults have 32 teeth in total. To amend the original algorithm to clean 32 teeth:

1. set number_of_teeth_cleaned to 0
2. put toothpaste on toothbrush
3. use toothbrush to clean a tooth
4. increase number_of_teeth_cleaned by 1
5. if number_of_teeth_cleaned < 32 then go back to step 3
6. rinse toothbrush

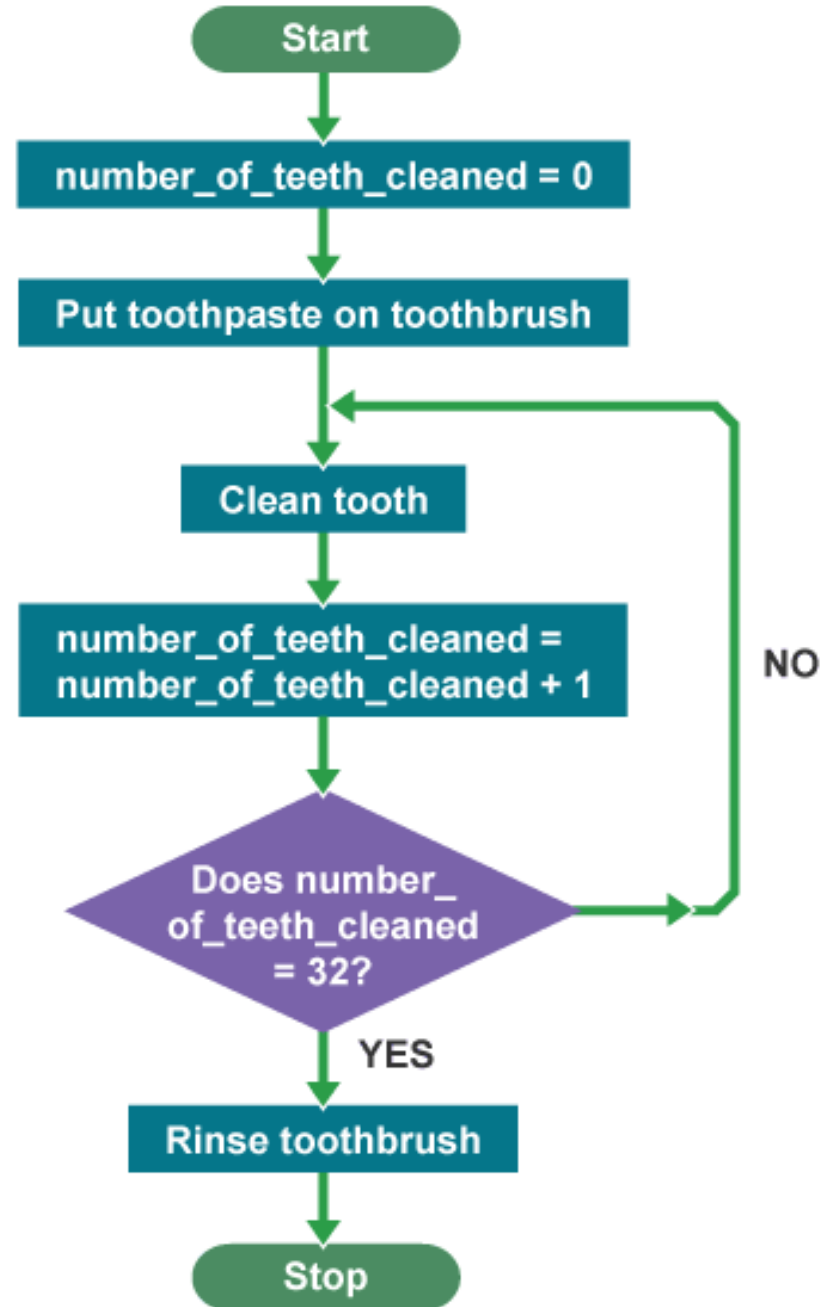


In pseudocode, the algorithm would look like this:

```
number_of_teeth_cleaned = 0
put toothpaste on toothbrush
REPEAT
    procedure clean_tooth
        number_of_teeth_cleaned = number_of_teeth_cleaned + 1
UNTIL number_of_teeth_cleaned = 32
rinse toothbrush
```



FLOWCHART



There are mainly two types of loops:

1. **Entry Controlled loops:** In this type of loops the test condition is tested before entering the loop body. **For Loop** and **While Loop** are entry controlled loops.

2. **Exit Controlled Loops:** In this type of loops the test condition is tested or evaluated at the end of loop body. Therefore, the loop body will execute atleast once, irrespective of whether the test condition is true or false. **do - while loop** is exit controlled loop.

Loops

Entry Controlled

Exit Controlled

for

while

do-while

```
for( initialization ; condition; updation)
{
}

```

```
while( condition )
{
}

```

```
do
{
}while( condition )

```

for Loop

A for loop is a repetition control structure which allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line.

Syntax:

```
for (initialization expr; test expr; update expr)
{
    // body of the loop
    // statements we want to execute
}
```

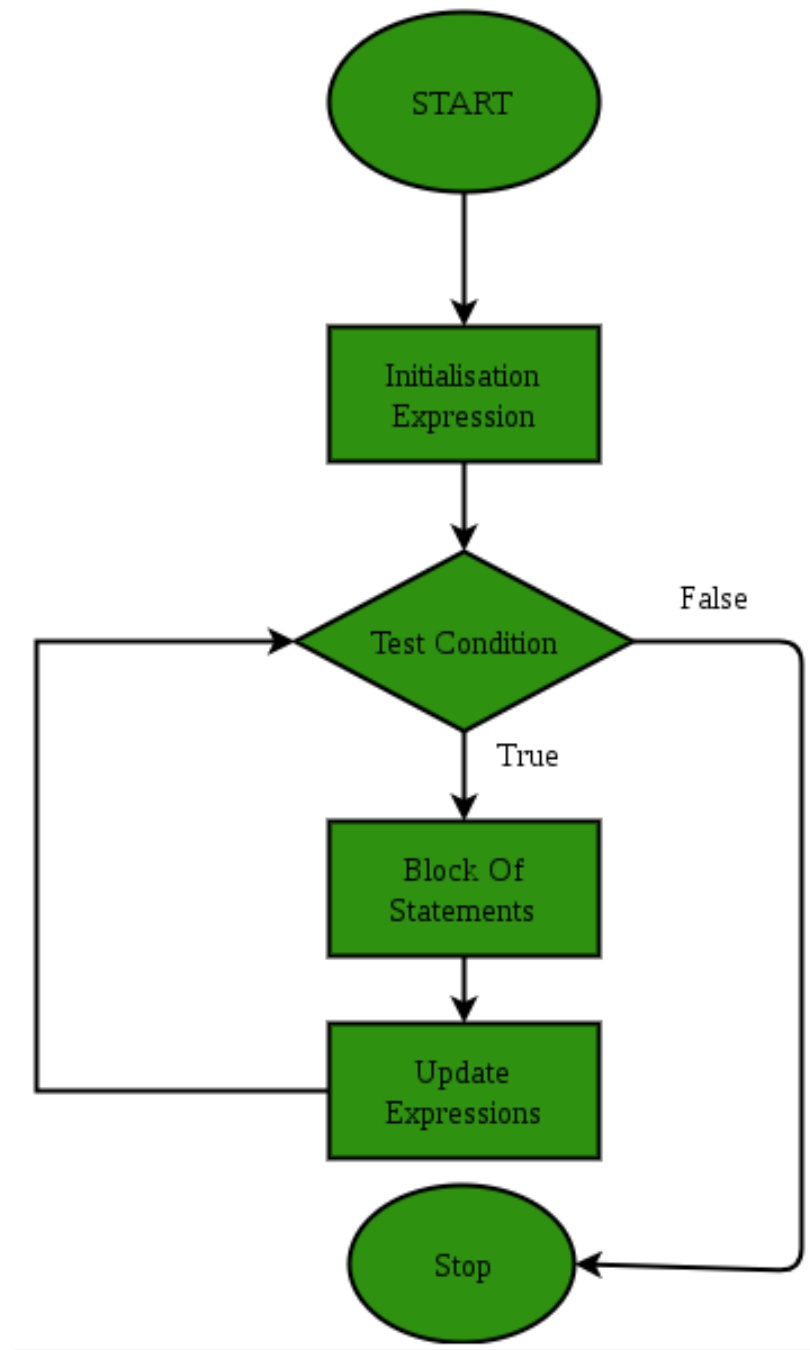
for Loop

Initialization Expression: In this expression we have to initialize the loop counter to some value. for example: `int i=1;`

Test Expression: In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of loop and go to update expression otherwise we will exit from the for loop. For example: `i <= 10;`

Update Expression: After executing loop body this expression increments/decrements the loop variable by some value. for example: `i++;`

FLOWCHART

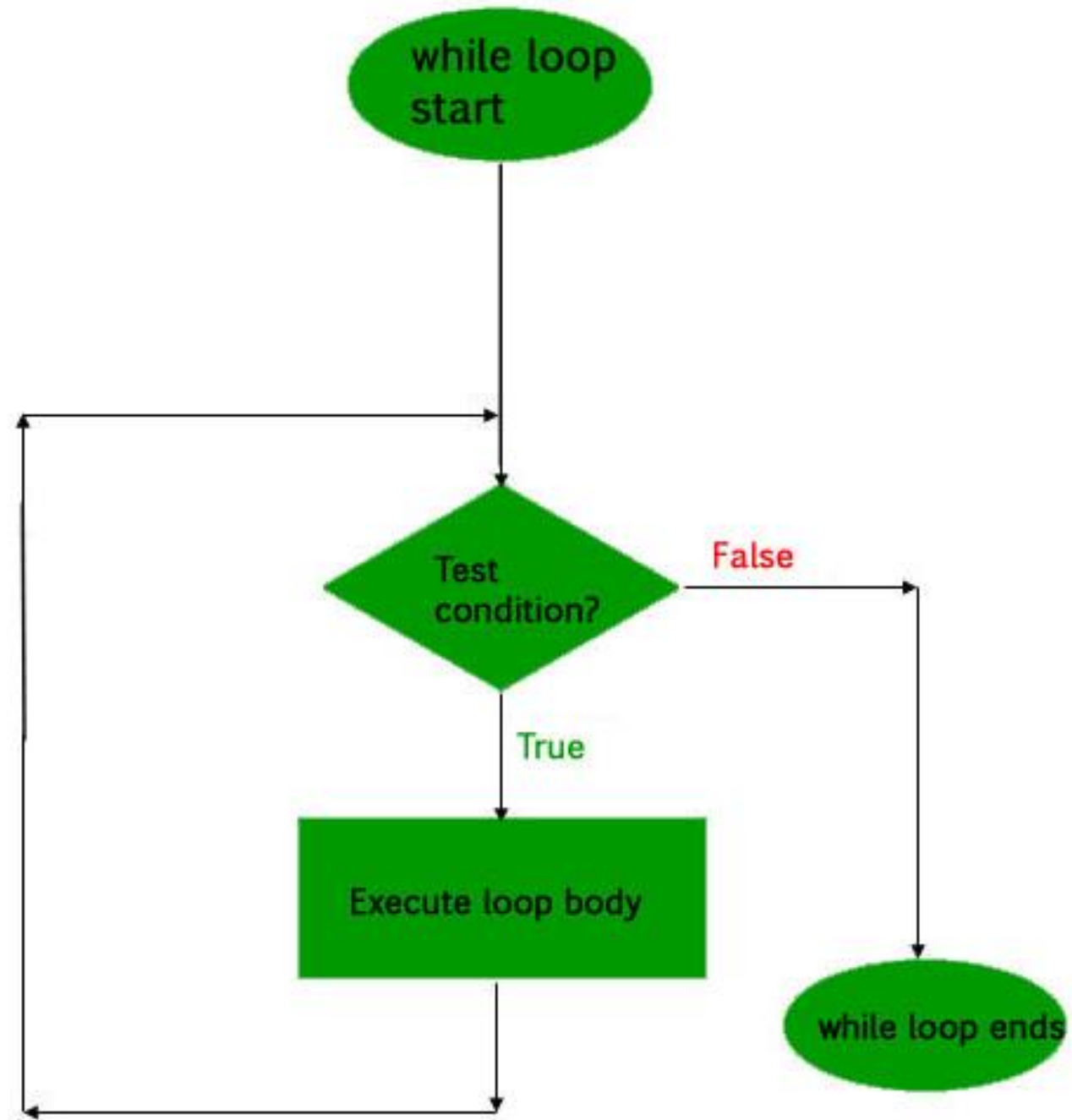


while Loop

while loops are used in situations where we do not know the exact number of iterations of loop beforehand. The loop execution is terminated on the basis of test condition.

```
initialization expression;  
while (test_expression)  
{  
    // statements  
  
    update_expression;  
}
```


FLOWCHART



do - while Loop

In do while loops also the loop execution is terminated on the basis of test condition. The main difference between do while loop and while loop is in do while loop the condition is tested at the end of loop body, i.e do while loop is exit controlled whereas the other two loops are entry controlled loops.

Note: In do while loop the loop body will execute at least once irrespective of test condition.

do - while Loop

Syntax:

```
initialization expression;  
do  
{  
    // statements  
  
    update_expression;  
} while (test_expression);
```

FLOWCHART

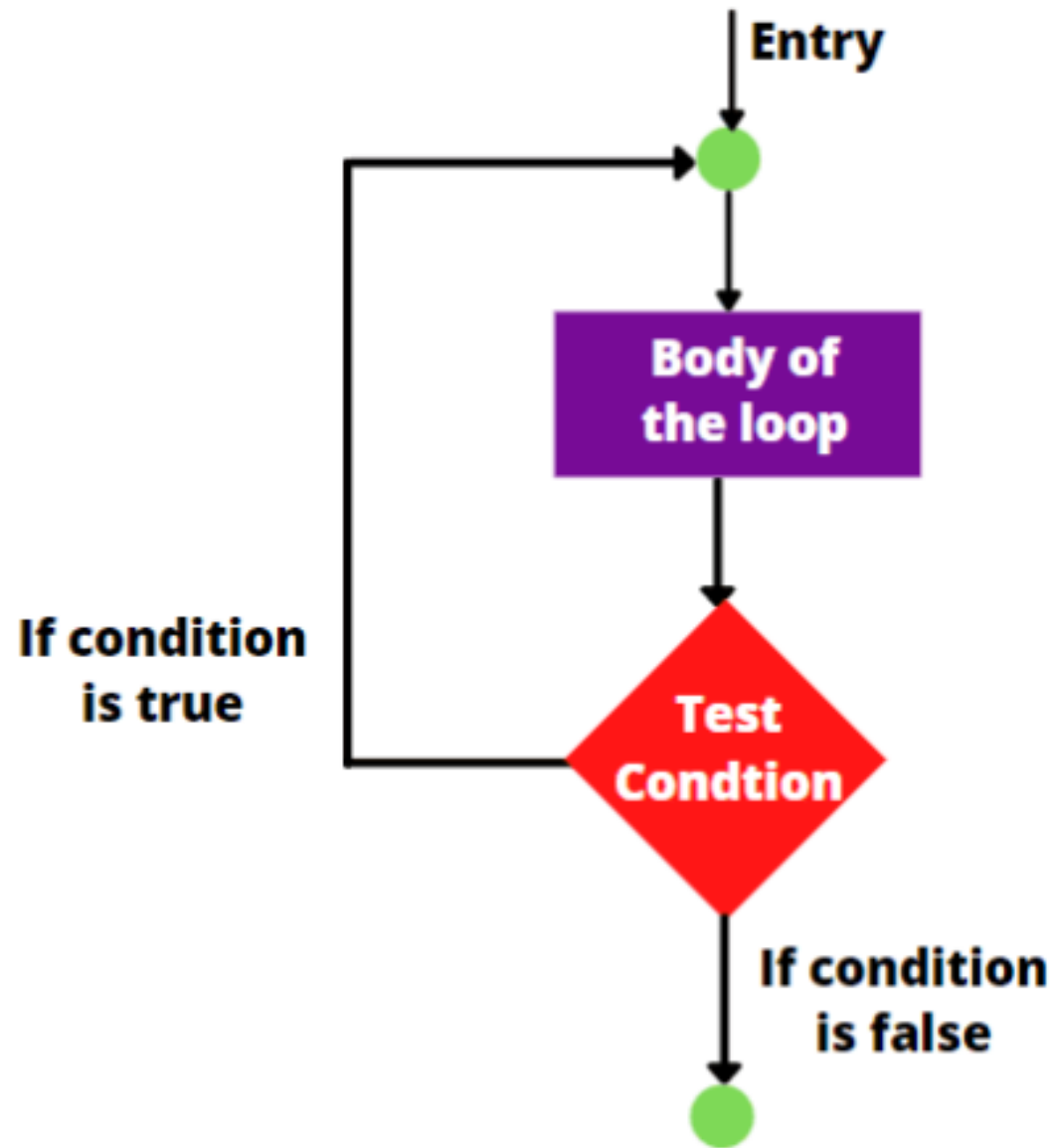


Fig: do while loop flowchart

Computers rely heavily on algorithms to do useful work. Computers can neither function with intuitive knowledge, nor do they have their own intelligence to make decisions. They depend on humans to provide them with the “thought pattern” or the “method” of doing work.



Any questions???

REFERENCES:

- 5 Basic Elements of Programming. (2021).
https://www.assignmenthelp.net/assignment_help/elements-of-programming
- Busbee and Braunschweig. (n.d.). Structured Programming.
<https://press.rebus.community/programmingfundamentals/chapter/structured-programming/>
- Computer Programming Tutorial. (n.d.).
https://www.tutorialspoint.com/computer_programming/index.htm
- Costa, C. (2022). Top Programming Languages and Their Uses.
<https://www.kdnuggets.com/2021/05/top-programming-languages.html>
- Python Introduction. (n.d.).
https://www.w3schools.com/python/python_intro.asp
- Simmons, L. (2023). 2023 Guide to the Top 12 Coding Languages.
<https://www.computerscience.org/resources/computer-programming-languages/>

