

# S1: Introduction to Programming

Session - 1

# What is Programming?

- Dictionary Definition :-
  - The process of preparing an instructional program for a device.
- A simpler definition :-
  - Attempting to get a computer to complete a specific task without making mistakes



# An analogy

- You instruct your “less than intelligent” friend to build a lego set.
- He lost the instructions so he can only build based on your commands.
- You must give him exact instructions on how to build. The entire set can be ruined, if there is one piece wrongly put.
- Giving instructions to your friend is very similar to how programmers code



# Computers are dumb!

- They appear to be smart because we program them in such a way..
- A computer's main functionality comes from how we manipulate it to serve our needs.



# The Language of Code

- Computers only understand machine code.
  - A series of 1's and 0's fed and interpreted by the computer.
- Therefore, in order to talk to the computer, you must first translate your English instructions into binary.
- It would be entirely impractical to convert every programming instruction you have into binary by hand.



# Programming Languages

- Programming languages serve as a middle-man of sorts.
- They translate your instructions into machine code
- These are much easier for humans to learn and understand compared to machine code & are hence very useful for programmers..
- It is like a translator to convert one language to another.



# High vs Low Level languages

- Each language also has an attribute known as power or level.
  - This is basically how similar a language is to machine code.
- Low - Level programming languages
  - Assembly or C
- High - Level programming languages
  - Java or Python
- Lower the level -> More similar the language is to machine code
- Low level language programs are faster than their higher level analogues



# Where do we write code?

- We cannot just write code in a text document and expect the computer to translate it into machine code and carry out the required task.
- We use IDE's (Integrated Development Environment) to write code.



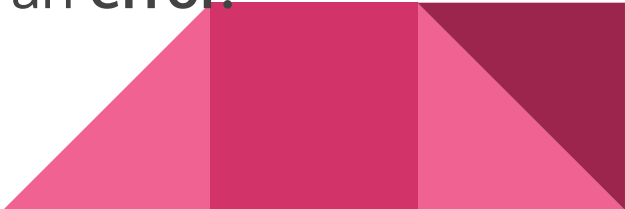


# IDE's


- An IDE is a place to write, run, and debug code and also convert it to machine code.
- IDE's are like any other program on your computer except used for the facilitation of code.
  - Ex) Visual Studio Code, Sublime Text, IntelliJ
- IDE's also have built - in error checking, which is very helpful for programmers.




# How do we write code: Learning Languages

- Learning a computer language is very similar to learning a real language.
  - All programming languages have a set of rules you must follow while writing code in that language, just like a real language.
  - This is called **syntax**. It is very similar to grammar in a real life language.
  - We need to follow the syntax strictly since computers are dumb.
  - Breaking programming rules will result into an **error**.
- 

# How do we get information from computers?

- Programmers keep track of their progress by looking at the **console**, a text interface on the computer.
  - The main use of a console is to output text from the program using code.
  - Using a **print** statement enables us to do so, i.e. instruct the console to print something.
  - Using a **print** statement, we can see the result the program written has produced.
  - The **print** statement will vary depending on the language used.
- 

# What Can Computers Do?

- Computers are dumb, but how dumb? Do they know anything intuitively?
  - The computer already knows how to do simple arithmetic.
    - Addition, Subtraction, Multiplication, Division
  - You'll be able to print the result of any math operation.
  - This may seem useless, but it comes in hand extremely often.
  - For example, we'd need to utilize this functionality in order to display the answer when you build a calculator app.
- 

# Strings

- Strings are another way of saying 'text'
  - "Hello World"
  - "A"
  - Anything enclosed by quotation marks is denoted as a string.
- Computers can concatenate strings, i.e. add strings together.
- Ex) "Hello" + " World" =====> "Hello World"



# An Important Difference

- 4 in quotation marks ("4") is treated as a **STRING**.
- 4 without quotation marks (4) is treated as an **INTEGER**.
- This might not seem as a big difference, but we need to be precise while writing code, since computers are dumb.
- If you are trying to do math with numbers in quotation marks, you will likely get an error.



# Variables

- A variable is something that can store information.
  - Can be referenced and manipulated
- Think of variables as cardboard boxes which are a means to store objects which can be changed, replaced or modified.
- Each variable has a type, a name and a piece of information stored inside of it.
  - Name is simply a name of the variable.
  - It's like writing a label on the cardboard box.



# Why are variables so useful?

- Often times you're going to want to keep track of those things such as a name or a score
  - By creating a variable, you can store this information in that variable and then **reference, add to or modify it.**
- Other important uses are :-
  - Taking input from the user
  - Making your program have variability
    - Have it change based on a certain factors





# What happens when we define a variable?

- Creates a little space in memory that stores your variable name and its contents.
- Going back to our cardboard box example, imagine if you would have your own storage facility and you make a new cardboard box labelled "name". And inside of it, you put the word "Programming".
- So, anytime you want to contents of the "name" box, you could look inside it to find the word "Programming".



# What happens when we define a variable?

- This is what a computer does, except that the storage facility is a memory in your computer, the box is a variable, and the contents of the box is whatever the variable is set equal to.
- Anytime you want to know the contents of a variable, you can simply call it and the computer will pull the information that is stored in that variable.



# Empty Variable

- You can also build a variable without putting information in it. It's like putting a blank label in the cardboard box.
- Reasons you would do this :-
  - Because you want to store information in the variable sometime later in the code.
  - Because you are going to use it to store information given to you by the user.



# Updating variables.

- For example, let's say you have an "age" variable, whose value is 18. Now, you celebrated your birthday and hence want to update your age.
- So, you simply reference the variable and assign it the new value, in this case :- **age = 19**
- This is like taking out the piece of paper from the "age" box, erasing the value written on it and updating it to a new value, and putting it back in the box.
- Keep in mind, that the variable is simply a place of memory. So, we can easily update the numbers and their place will remain constant.



# Fate of variables

- After the code has run its course, variables are deleted in memory, until you run the code again and the program dedicates space for the variable again.
- At the beginning of the code, you make new boxes in the storage facility, and at the end of the code, you destroy the boxes to make room for the new boxes next time.



# Other Ways of Manipulating Variables

- Integers, Float and Double variables can be
  - Added (Ex:-  $c = a + b$ )
  - Subtracted
  - Multiplied
  - Divided
- String variables can be
  - Added (Concatenation)



# Conditional Statements

- Having your code do the same thing each time is boring.
- Depending on certain conditions, we want our code to do different things.
- We can make this happen by the use of conditional statements.



# The If statement

- If something is True execute one set of statement
  - otherwise do nothing
- Each condition is evaluated as a Boolean
  - **True** or **False**

```
if condition is true  
    Statements evaluated If condition is True
```





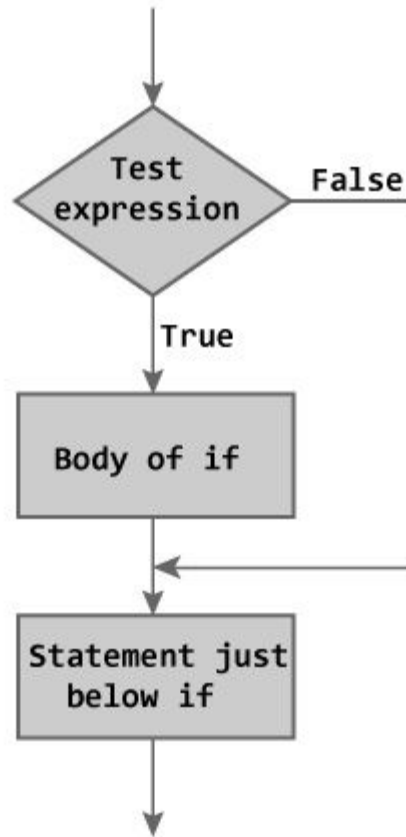
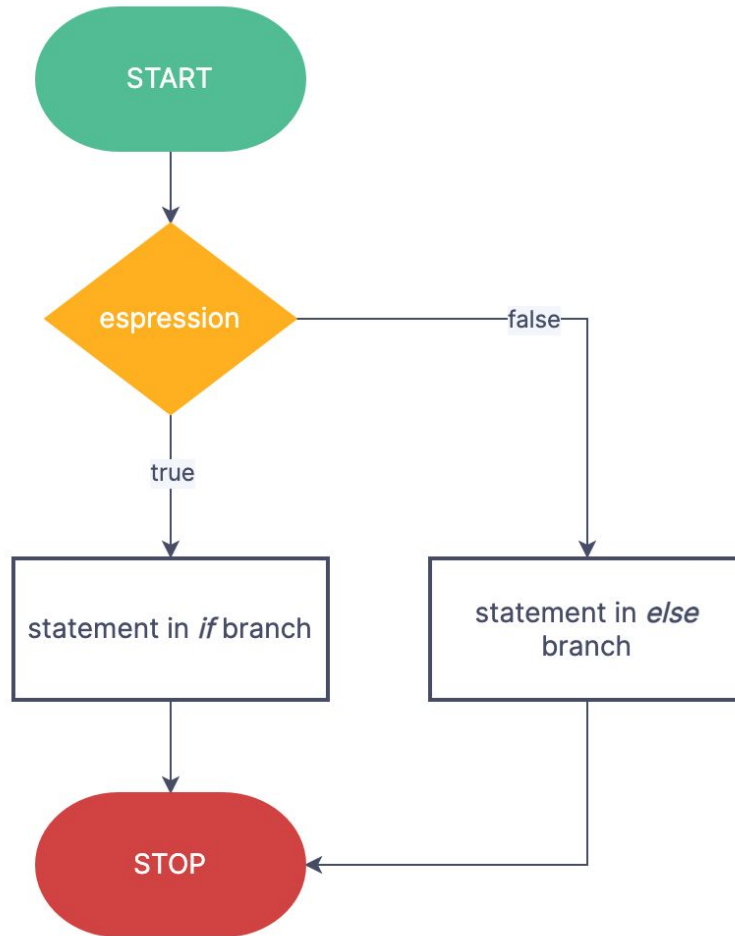


Figure: Flowchart of if Statement

# The If - Else statement

- If something is True execute one set of statement
  - otherwise do some other statements are evaluated
- Each condition is evaluated as a Boolean
  - **True** or **False**

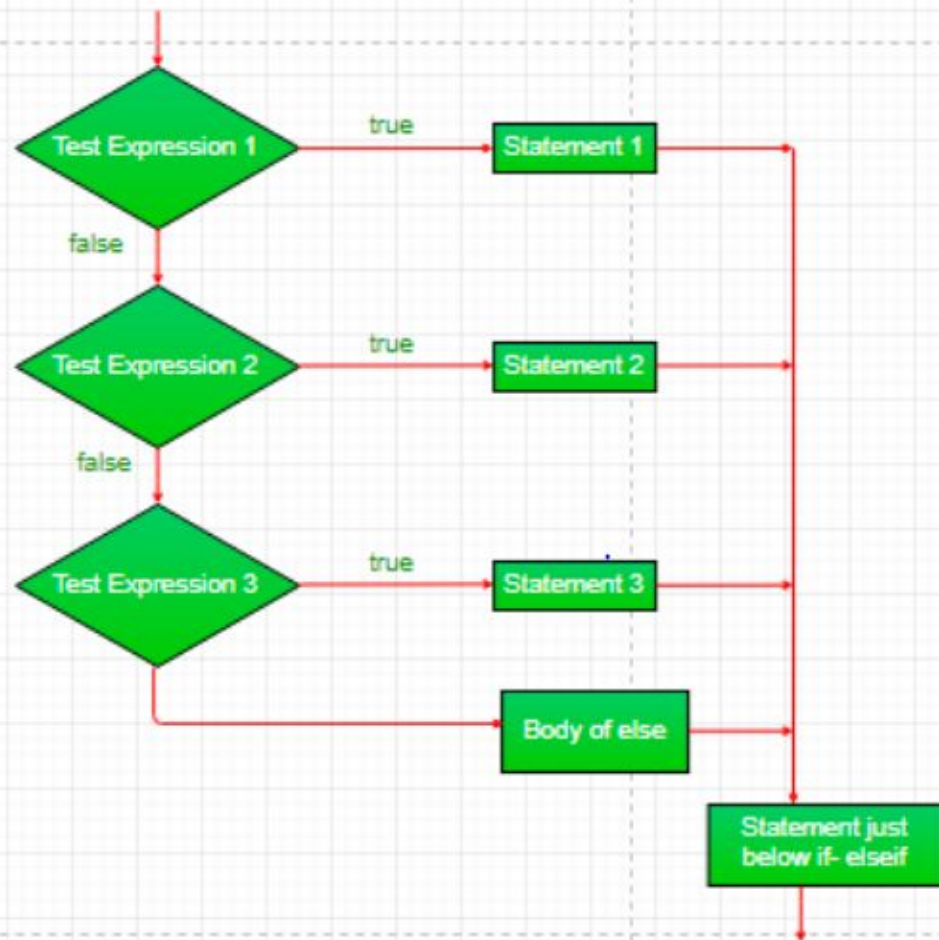
```
if condition is true
    Statements evaluated If condition is True
else
    Statements evaluated If condition is False
```



# The Else if ladder

- If something is True execute one set of statement
  - otherwise if something else is true, execute another set of statements.
  - otherwise execute another set of statements.

```
if condition1 is true
    Evaluated If condition1 is True
else if condition2 is true
    Evaluated If condition1 is False and condition2 is True
else
    Evaluated If condition1 and condition2, both are False
```



# Example

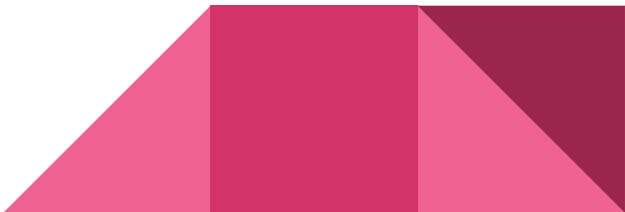
Q) Find whether the number in variable "num" is positive or negative? (Consider 0 as positive)

```
if(num >= 0)
    print("Number is positive")
else
    print("Number is negative")
```



# Calculator!

Q) Let us design a calculator, which provides the functionality.

- i) Add two numbers
  - ii) Subtract two numbers
  - iii) Multiply two numbers
  - iv) Divide two numbers (Print an error message if the second number = 0)
    - The first number is stored in the variable "a"
    - The second number is stored in the variable "b"
    - The operator is stored in the variable "ch"
    - Store the answer in the variable "ans"
- 

```
if (ch equals '+')
    ans = a + b
else if(ch equals '-')
    ans = a - b
else if(ch equals '*')
    ans = a * b
else if(ch equals '/')
{
    if (b equals 0)
        print("Division by 0!")
    else
        ans = a / b
}
```



# Usefulness of Conditional Statements

- Adds variability to programming
  - Program runs differently based on user input
- If a user does something, we want to be able to adapt accordingly.
- Without any conditional statements, the program would run the same way every time.



# Think Like A Coder by Ted-Ed

- A nice animated series which introduces programming, with a set of fun algorithmic puzzles.

Link :- [Think Like A Coder](#)





Thank you!