

IBM PROJECT PRESENTATION

# FUNCTIONS AND DATA STRUCTURE

Here is where our presentation begins.....

PRESENTED BY:  
VAIDEHI KARWARIYA  
PURNIMA SHUKLA



# Table of contents

**01**

**FUNCTIONS  
AND DATA  
STRUCTURE**

**03**

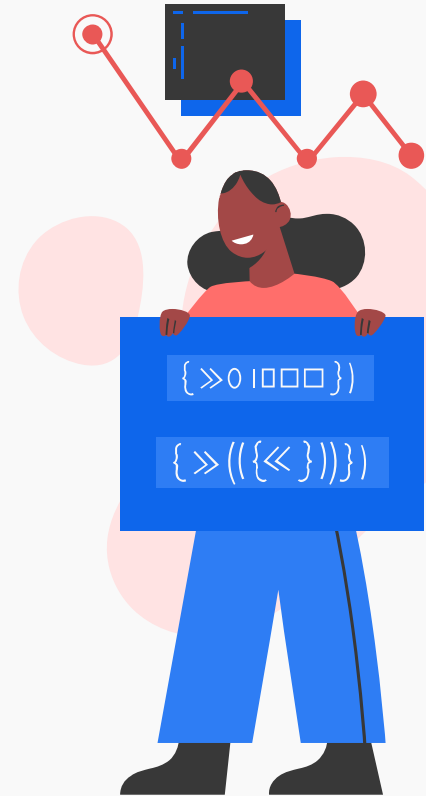
**WHAT IS  
FUNCTION**

**02**

**WHY THEY  
MATTER**

**04**

**TYPES OF  
FUNCTIONS**





# FUNCTION AND DATA STRUCTURE

**FUNCTION** is reusable block of code that performs a specific task. Think of it like a small robot, you train once. After that, you can call it anytime and it does the job exactly the same way.

## Program code:

```
Def greet(name):  
    print(greet)  
greet(Aanya)
```

**DATA STRUCTURE** these are the way to store and organize data so you can use it efficiently.



# WHY THIS MATTERS?



## Functions

1. They reduce repetition
2. They make programs easier to understand
3. They make debugging easier
4. They allow teamwork
5. They increase flexibility

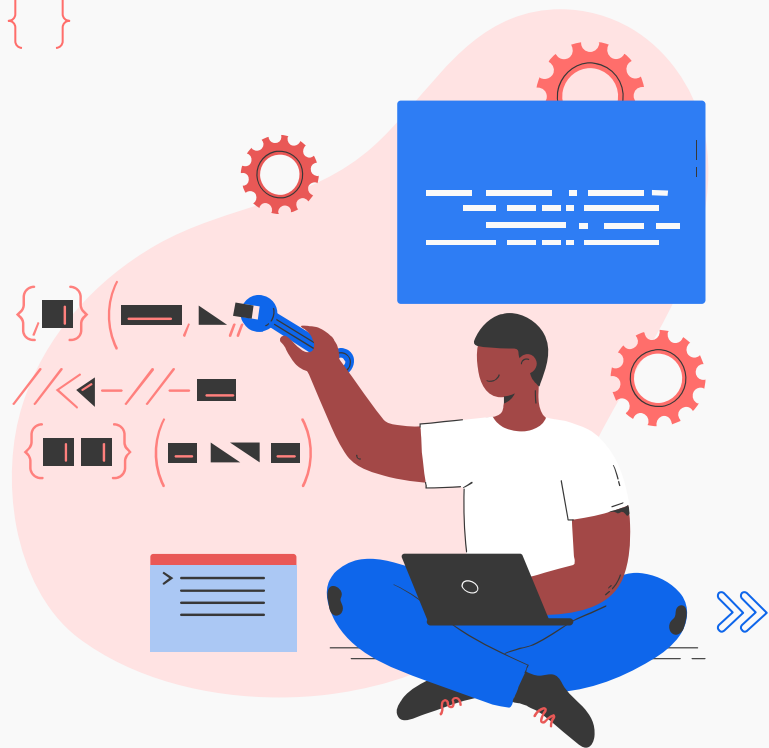
## DATA STRUCTURE

1. They make data easy to store and find
2. They handle real-world complexity
  - Graphs
  - Trees
  - Heaps
3. They improve performance
4. They are the building blocks of algorithm
5. They help solve problems properly

# 01

## HOW FUNCTIONS WORKS?

1. A function is a small block of code that performs a specific task.
2. You first define it by giving it a name and instructions.
3. It can take inputs called parameters.
4. When you call the function, you give it actual values (arguments).
5. The function creates its own workspace (scope).
6. It runs the instructions inside, top to bottom.
7. It may produce an output using **return**.
8. After returning, the function stops running.
9. You can reuse the same function many times with different inputs.
10. Functions make programs cleaner, faster, and easier to manage.





# TYPES OF FUNCTIONS



## 1. Built-in Functions

Already provided by the programming language.

➔ Example: `print()`, `len()` in Python.

## 2. User-Defined Functions

Created by the programmer to perform specific tasks.

➔ Example:

```
def add(a, b): return a + b
```



## 3. Parameterised Functions

Have inputs (parameters).

➔ Example: `def greet(name): ...`

## 4. Non-Parameterised Functions

No inputs.

➔ Example:

```
def hello(): print("Hello")
```

## 5. Lambda / Anonymous Functions

Functions without a name; used for short tasks.

➔ Example (Python):

```
square = lambda x: x*x
```



# CONCLUSION

{ }

Functions and data structures are the two most important building blocks of programming.

**Functions** break problems into smaller, reusable steps, making programs cleaner, easier to understand, and easier to maintain.

**Data structures** organize and store data in efficient ways so programs can access, modify, and process information quickly. Together, they allow programmers to solve complex problems by combining well-structured data with well-designed operations. In short, functions tell the program *how to do something*, and data structures decide *where and how the data is stored*. Both are essential for creating fast, organized, and reliable software.



# Thanks!

Do you have any questions?  
2pacraya57@gmail.com



{({({ >> })) << }



(( { >> 0 1 □ □ □ } ))

```
((: 00 - =>> } )  
{ (<1 00 1 000 >> )}  
((: 0)>">< )  
<01 001} +100 0}>  
((: 0)>">< )  
{ (<1 00 1 000 >> )}
```

