Newton School of Technology

6

Functions

By Kunal Goyal

CSA101 : Problem Solving with Programming

Newton School of Technology

Join the lecture online on your dashboard

Analogy: Washing Clothes in a Washing Machine:

Steps :

1. Fill with Water
2. Wash Clothes
3. Drain Used Water
4. Fill with Water
5. Spin Clothes
6. Drain Used Water
7. Fill with Water
8. Spin Clothes
9. Drain Used Water
10. Dry Clothes

Code Snippet

Newton School of Technology

Functions

What are functions ?

- Functions let you reuse code instead of rewriting it.
- Like a recipe for a sandwich:
- Write the steps once → reuse it any no of times!
- No need to rewrite the steps every time when you make a sandwich.

Washing Machine :

ATM Machine :

Purpose of using function :

1. Code Reusability
2. Improved readability
3. Modularity
4. Abstraction
5. Helps in Testing and debugging

Overall makes life easy for you.

Defining a Function :

It’s Like Crafting a New Tool

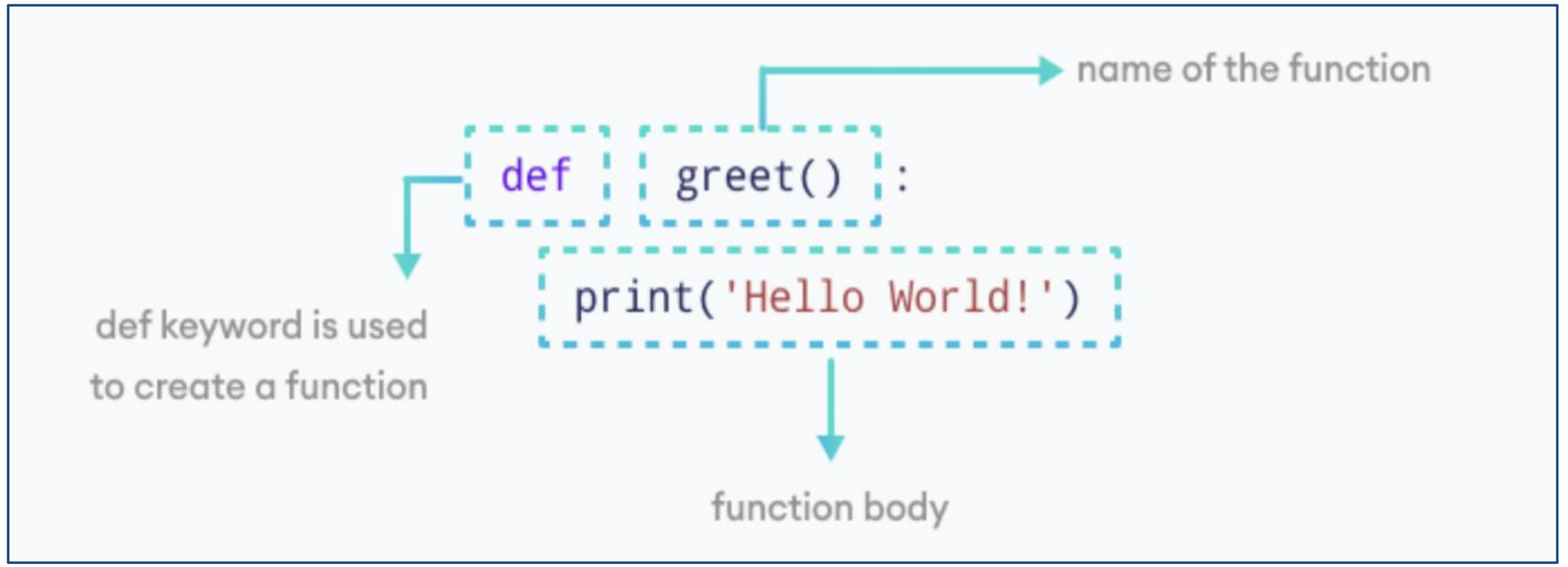
"A function’s name is its handle – make it easy to grip!"

The name of your function should describe what it does, just like a good label on a toolbox.

- **make_sandwich()** — easy to understand and use
- **calculate_total()** — tells you exactly what it’s meant for

Defining a Function (Without Parameters) :

```
def function_name():
    #body of the function
    print("Hey! I am body of function.")
```



Code Snippet

Calling a function :

Just call the name of the function…….. so simple

eg: calling greet() function.

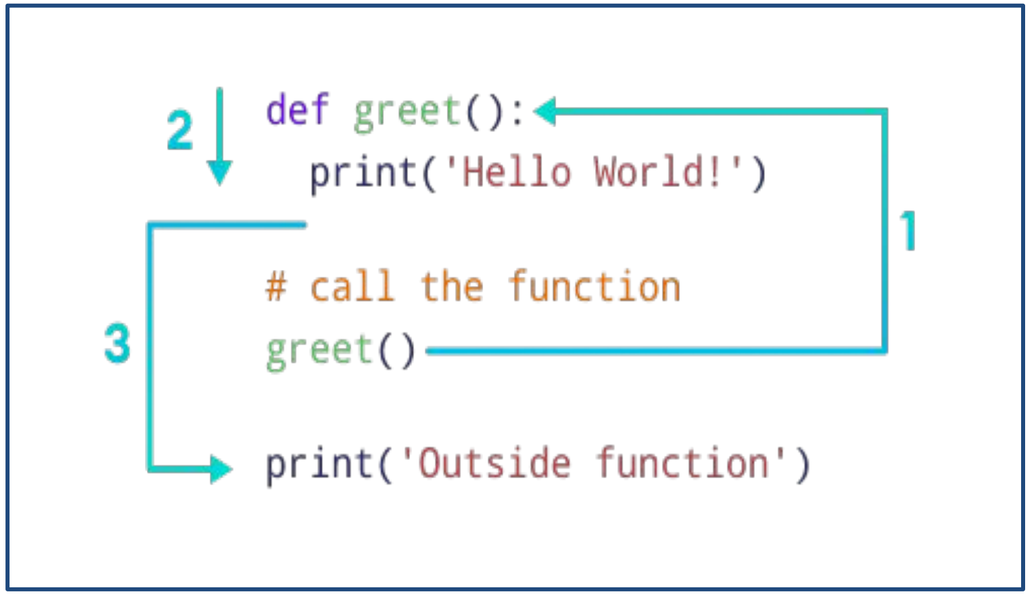
```
greet()
```

Calling a function :

What Happens When You Call It?

The code **inside** the function **runs**

Every time you call it, it runs **again from the top**



Analogy - The Recipe for a Perfect Sandwich :

Newton School of Technology

Steps :

1. Get two slices of bread
2. Add peanut butter
3. Add jelly
4. Put the slices together
5. Serve

Recipe for Sandwich using Function :

```
def make_sandwich():
    print("Get two slices of bread")
    print("Add peanut butter")
    print("Add jelly")
    print("Put the slices together")
    print("Serve\n")

# # Calling the make_sandwich() function to execute its code
make_sandwich()
```

Code Snippet

Analogy - One Recipe, Four Sandwiches :

Question - 1 : Make Sandwich 4 Times

Newton School of Technology

You are asked to make a sandwich by following the same steps each time.

Write a function called make_sandwich() that prints the steps to make a sandwich.

Call this function four times to make four sandwiches.

The steps to make a sandwich are:

- **Get two slices of bread**
- **Add peanut butter**
- **Add jelly**
- **Put the slices together**
- **Serve the sandwich**

Example : You don't need to take any input.

Question - 1 : Make Sandwich 4 Times

Newton School of Technology

```
INPUT | OUTPUT | ERROR |
 1    Get two slices of bread
 2    Add peanut butter
 3    Add jelly
 4    Put the slices together
 5    Serve the sandwich
 6
 7    Get two slices of bread
 8    Add peanut butter
 9    Add jelly
10    Put the slices together
11    Serve the sandwich
12
13    Get two slices of bread
14    Add peanut butter
15    Add jelly
16    Put the slices together
17    Serve the sandwich
18
19    Get two slices of bread
20    Add peanut butter
21    Add jelly
22    Put the slices together
23    Serve the sandwich
24
```

Newton School of Technology

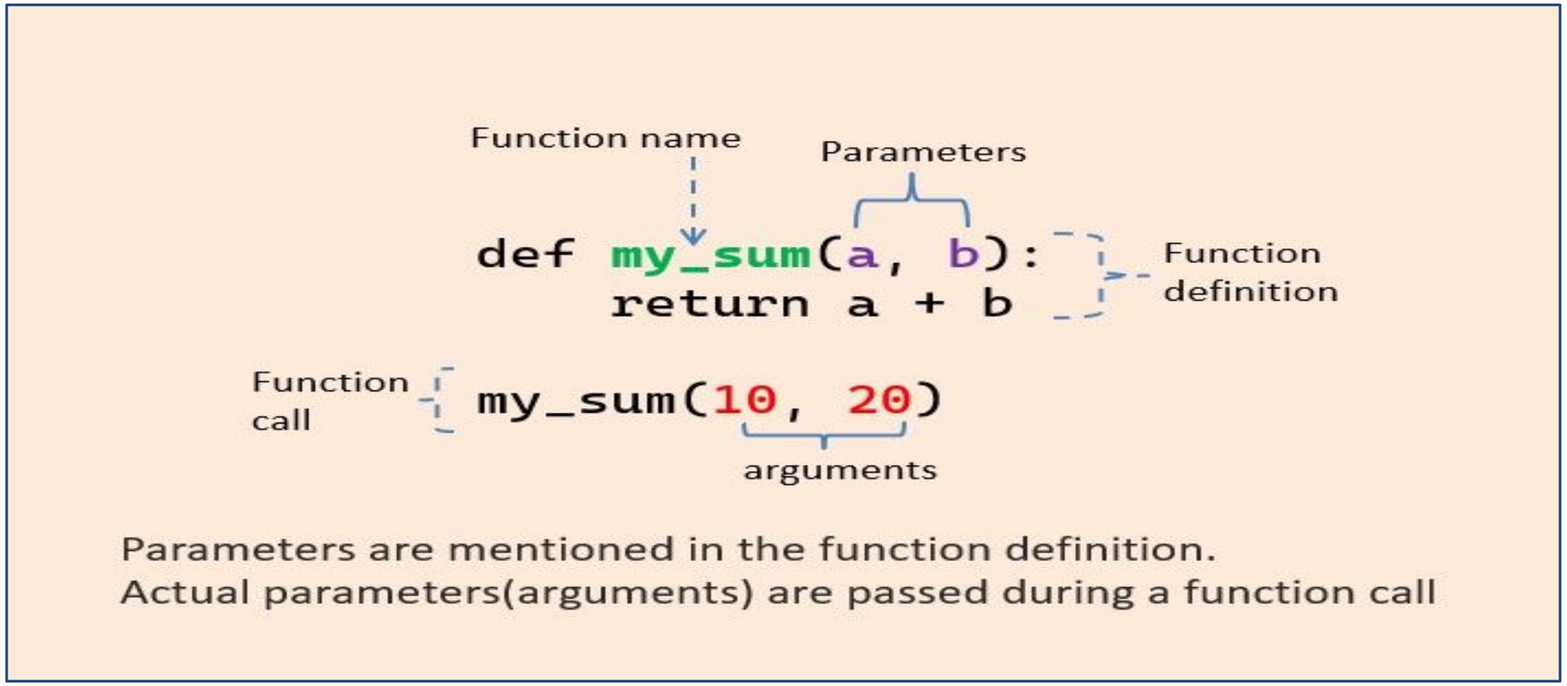
Function with Parameters and Function Arguments!

Function Parameters and Arguments :

Newton School of Technology

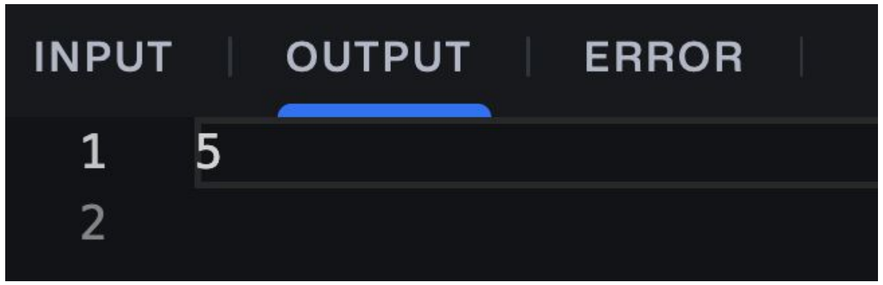
- **Parameters** are variables **defined in the function header** to receive input values. Parameters act as placeholders

- **Arguments** are the **actual values passed to the function when calling it**. arguments provide real data.



Adding Two Numbers using function :

```python
# Function Definition: my_sum with two parameters
def my_sum(a, b):
    print(a + b)

# Calling the function with arguments 2 and 3
my_sum(2, 3)
```

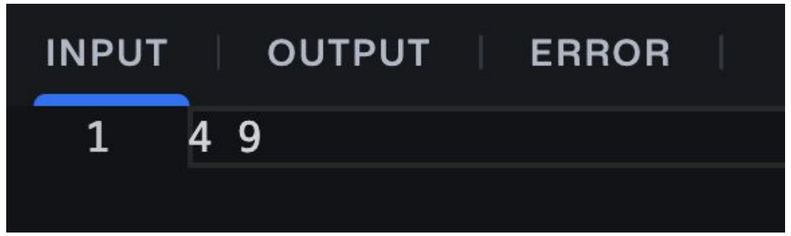


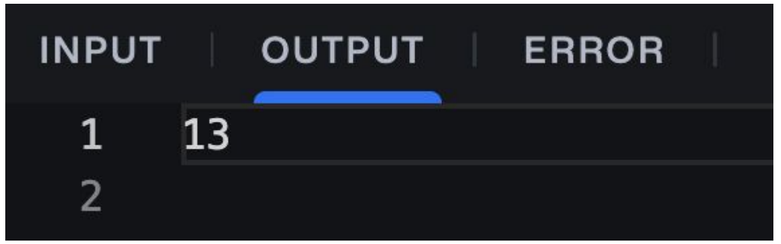
Code Snippet

Adding Two Numbers using function :

Newton School of Technology

```python
# Function Definition: my_sum with two parameters
def my_sum(a, b):
    print(a + b)

# Taking two integer inputs from the user in a single line, separated by space
# to pass them as arguments while calling the function
num_1, num_2 = map(int,input().split())

# Calling the function with arguments num_1 and num_2
my_sum(num_1, num_2)
```





Function Arguments!

Function to order custom Pizza!

Function to order custom Pizza!

Newton School of Technology

Pizza shop writes down your choices

- Size: **Large**
- Crust: **Thin**
- Topping: **Pepperoni**

Defining Function :

```
def order_pizza(size, crust, topping):
    print("Order placed: “,size,”pizza with “,crust,”crust and”,topping,“topping.")
```

You call the function like this:
👉 **order_pizza("Large", "Thin", "Pepperoni")**

Function to order custom Pizza!

Newton School of Technology

```python
# Function definition: order_pizza with 3 parameters
def order_pizza(size, crust, topping):
    print("Order placed:",size,"pizza with",crust,"crust and",topping,"topping.")

# Asking user for input values
size, crust, topping = input().split()

# Function call with user-provided inputs as arguments
order_pizza(size, crust, topping)
```

INPUT	OUTPUT	ERROR
1	large Thin Pepperoni	

INPUT	OUTPUT	ERROR
1	Order placed: large pizza with Thin crust and Pepperoni topping.	
2		

Code Snippet

Question - 2 : The Fantastic Four-Pair

Newton School of Technology

You are given 4 pairs of integers as input. For each pair, your task is to:

- **Print the sum of the two numbers.**
- **Print the difference of the two numbers (first number minus second number).**
- **sum and difference should be printed in the same line separated by the space**

Input

The input contains **four lines**.

Each line contains **two integers** separated by a space.

Output

For each pair of numbers, print two values on a single line separated by a space:

- The sum of the two numbers
- The difference of the two numbers(first number minus second number)

Each result should be printed on a new line.

Example 1 : **2 1**
4 3
5 2
1 0

Output :

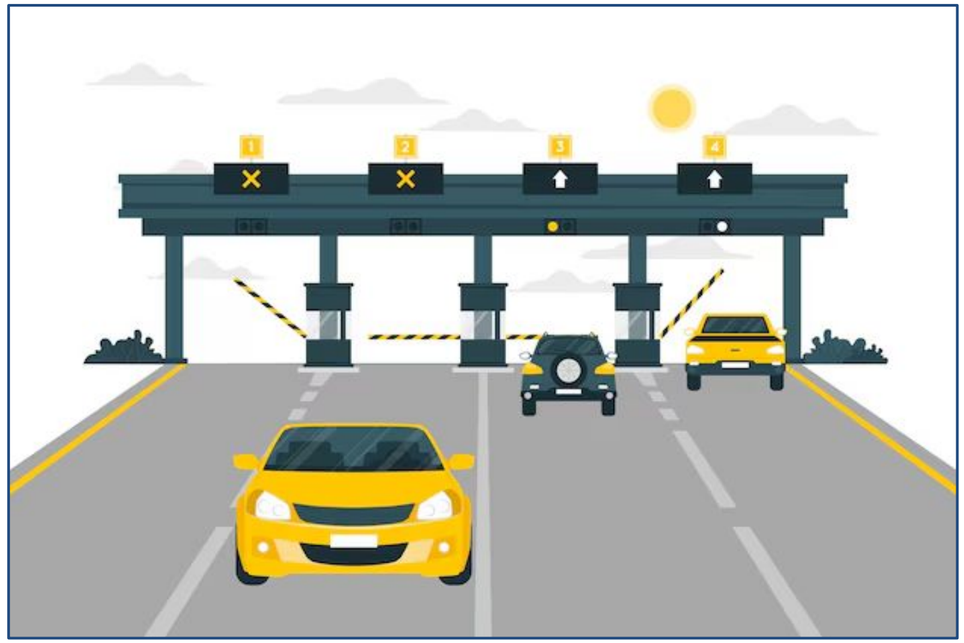
3 1
7 1
7 3
1 1

Types of Function Arguments:

1. **Positional Arguments.**
2. **Default Arguments.**
3. **Keyword Arguments.**

Positional Arguments :

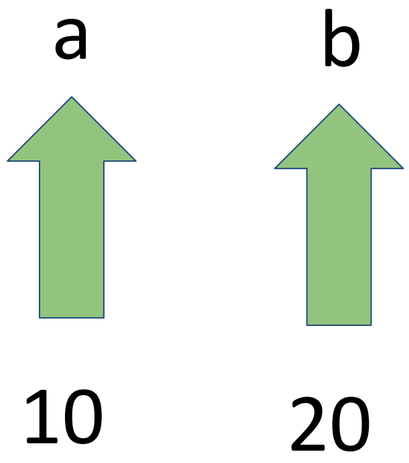
When you call a function, you provide values for its parameters in the **exact order** they are defined. These values are then assigned to the corresponding parameters in the **same order**.



Positional Arguments – Example :

Newton School of Technology

```python
# Defining a function that adds two numbers
def add_two_nums(a, b):
    c = a+ b
    print(c)

# Function call using positional arguments
# The first value (10) is assigned to parameter 'a'
# The second value (20) is assigned to parameter 'b'
# Order of arguments matters in positional arguments
add_two_nums(10, 20)
```




```
INPUT | OUTPUT | ERROR
1   30
2
```

Code Snippet

Default Arguments :

If an **argument is not provided** when the function is called, **the default value specified in the function definition is used.**

Note: Default arguments must come after any positional arguments in the function definition.

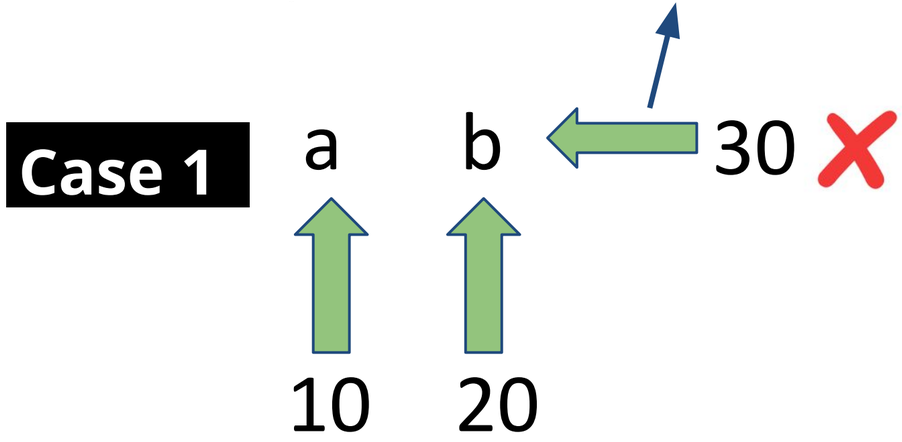
Default Arguments – Example :

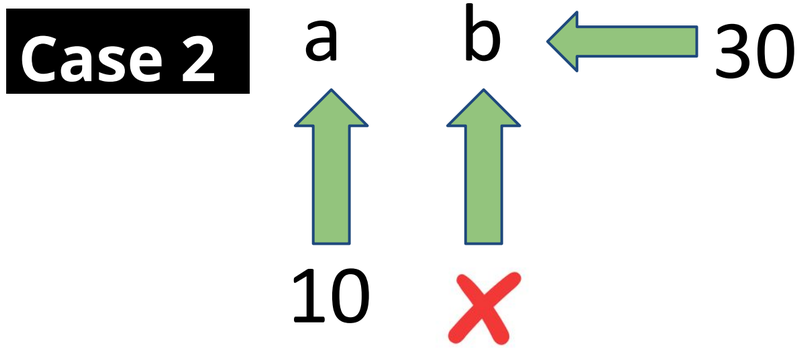
Newton School of Technology

```python
# Defining a function that adds two numbers
# Here, parameter 'a' is required,
# but parameter 'b' has a default value of 30.
# If no value is provided for 'b' during the function call,
# Python will automatically use b = 30.
def add_two_nums(a, b=30):
    c = a+ b
    print(c)

# Function call with two arguments: 10 and 20
# In this case, a = 10 and b = 20 (default value of b is NOT used)
add_two_nums(10, 20)

# Function call with only one argument: 10
# In this case, a = 10 and b takes its default value of 30
add_two_nums(10)
```

INPUT	OUTPUT	ERROR
1	30	
2	40	
3		

Only used if no value is provided in function call!





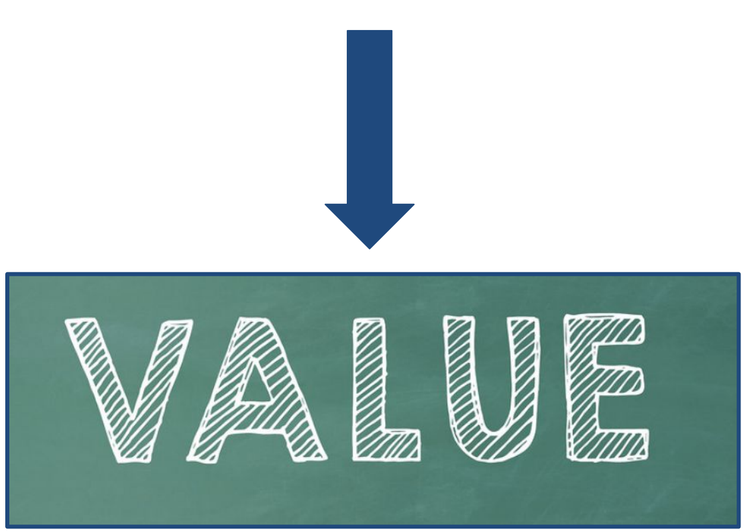
Keyword Arguments :

Newton School of Technology

Keyword arguments in Python allow you to call a function by specifying the **names of the parameters**.

This makes the function call more **explicit** and can improve **readability**.

Note: Keyword arguments must come after any positional arguments in the function definition.



Keyword Arguments – Example :

Newton School of Technology

```python
# Defining a function that adds two numbers
def add_two_nums(a, b):
    c = a+ b
    print(c)

# Function call using keyword arguments
# Here, we pass values by explicitly naming the parameters:
# a = 20 and b = 40
# The order of arguments does not matter when using keyword arguments
add_two_nums(b=40, a=20)
```

```
INPUT | OUTPUT | ERROR |
1   60
2
```

Keyword & Default Arguments - Example :

Newton School of Technology

```python
# Defining a function that adds two numbers
# Here, 'a' is a required parameter
# 'b' is an optional parameter with a default value of 20
#  If no value is provided for 'b' during the function call,
# Python will automatically use b = 20
def add_two_nums(a, b=20):
    c = a + b
    print(c)

# Function call using keyword arguments
# We explicitly assign values to parameters by name:
# a = 10, b = 10
# Since we are passing a value for b, the default value (20) is ignored
# Order of arguments does not matter when using keyword arguments
add_two_nums(b=10, a=10)
```

```
INPUT | OUTPUT | ERROR |
1   20
2
```

Return statement :

When Functions Finish Their Job… What Happens?

They don’t just talk — they **give you something back!**

Return statement :

Newton School of Technology

It **sends a result back** from a function to the place where the function was called.

```
def add(a, b):
    return a + b  # Sends the sum back


result = add(4, 5)
print(result)  # Output: 9
```

Code link

Why use `return`?

- So the function can **produce a value** you can save, print, or use later.
- Without `return`, functions only **do things** but don't give you anything back.

Spot the difference :

```python
#function to add two numbers
def add_two_nums(a, b):
    c = a+ b
    print(c)

add_two_nums(3, 7)
```

```python
#function to add two numbers
def add_two_nums(a, b):
    c = a+ b
    return c

print(add_two_nums(3, 7))
```

Code Snippet

Guess the output :

```
def add_two_nums(a, b):
    c = a+ b
    return c

d = add_two_nums(10, 20)
print(d)
```

Question - 3 : find area using function

Newton School of Technology

You are given the length and breadth of a rectangle. Your task is to complete a function named area(x, y) which takes two parameters — the length x and breadth y of the rectangle — and returns the area.

Note: This is a functional problem. You do not need to take input or print output. Just complete the function to return the required result(Area of a rectangle).

Input

The function receives two integers separated by the comms as parameters:

- length – the length of the rectangle
- breadth – the breadth of the rectangle

Note: You do not need to take any input from the user. The values will be passed directly to the function.

Custom Input
The only line containing the two positive integers separated by the space

Output

- Return a single integer — the area of the rectangle.

Question - 3 : find area using function

Newton School of Technology

Example 1 : length = 2
breadth = 3

Output : 6

Example 2 : length = 5
breadth = 4

Output : 20

Question - 4 : Find grade

Newton School of Technology

Alice appeared for her mathematics exam and performed well. After receiving her marks, she was curious to know the grade she achieved. A function named GetGrade(marks) is already defined. It takes the marks as input and returns the grade based on those marks. Your task is to write a program that:

- **Takes Alice's marks as input (as an integer).**
- **Calls the function GetGrade by passing the marks as a perameter.**
- **Prints the grade she received.**

Note: You do not need to define the GetGrade function. It is already provided. Just call it and print the returned result.

Example 1 : 87

Output : B

Example 2 : 79

Output : C

References :

Book_1 : Intro to Python by Paul Deitel		
S.No.	**Topic Name**	**Page No.**
1.	**Introduction to functions, function call and syntax, arguments and parameters, Return statement**	**119 - 124**
2.	**Default Parameters, Arguments**	**135 - 137**

Quiz Time!

Newton School of Technology

Please fill the feedback!

Thank You!

Steps :

1. Fill with Water
2. Wash Clothes
3. Drain Used Water
4. Fill with Water
5. Spin Clothes
6. Drain Used Water
7. Fill with Water
8. Spin Clothes
9. Drain Used Water
10. Dry Clothes

Code Snippet

Newton School of Technology

Coding Question

Q2. The Fantastic Four-Pair

1. Fill with Water
2. Wash Clothes
3. Drain Used Water
4. Fill with Water
5. Spin Clothes
6. Drain Used Water
7. Fill with Water
8. Spin Clothes
9. Drain Used Water
10. Dry Clothes

```
# Calling the functions in sequence for fully
automatic machine
fill_with_water()
wash_clothes()
drain_used_water()
# First rinse cycle (spinning with fresh water)
fill_with_water()
spin_clothes()
drain_used_water()
# Second rinse cycle (spinning again with fresh
water)
fill_with_water()
spin_clothes()
drain_used_water()

dry_clothes()
```

Recipe for Sandwich using Function :

```
def make_sandwich():
    print("Get two slices of bread")
    print("Add peanut butter")
    print("Add jelly")
    print("Put the slices together")
    print("Serve\n")
```

Code Snippet

Calling a function :

Just call the name of the function…….. so simple

eg: calling make_sandwich function.

```
make_sandwich()
```

Newton School of Technology

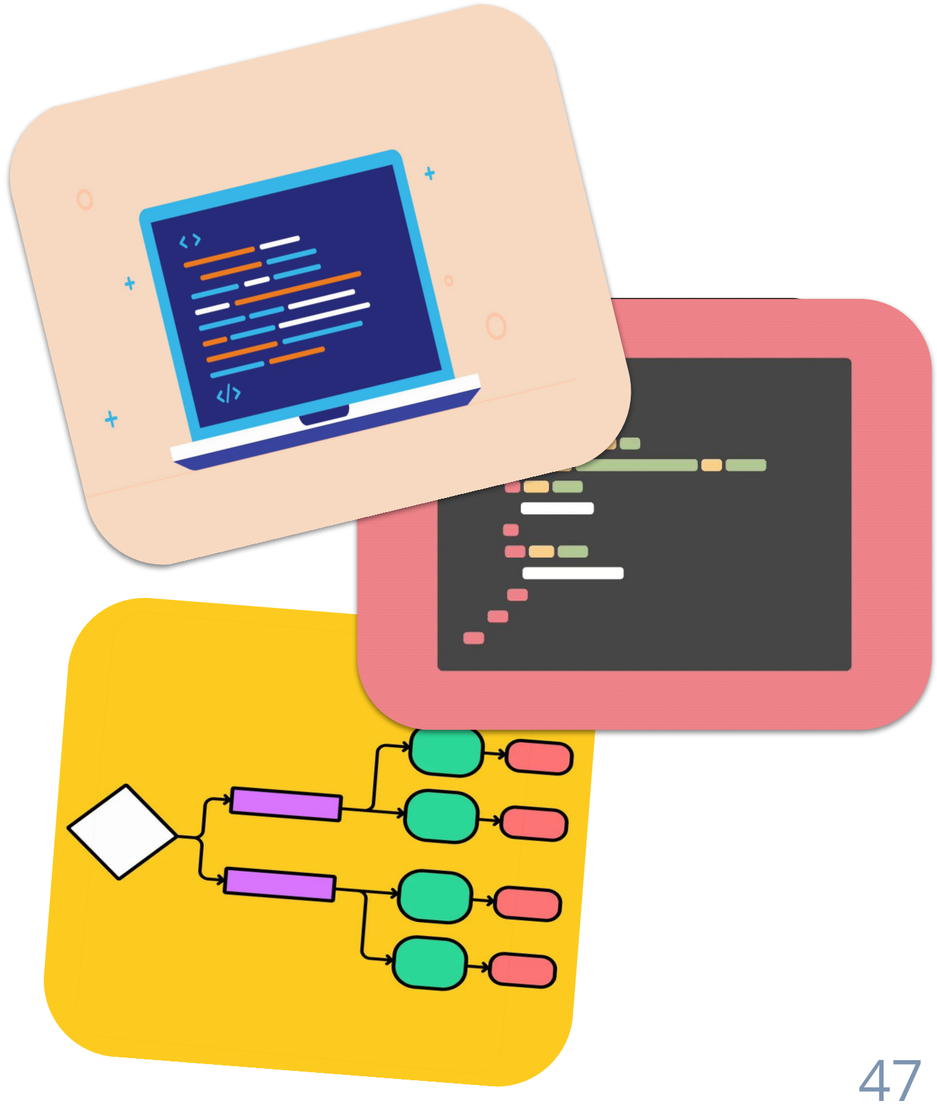
Q4. Find Area

Newton School of Technology

Q3. Find Grade

Summary

- **Functions**:
  - Definition and purpose
  - Defining, calling, arguments and return statement
  - Type of arguments



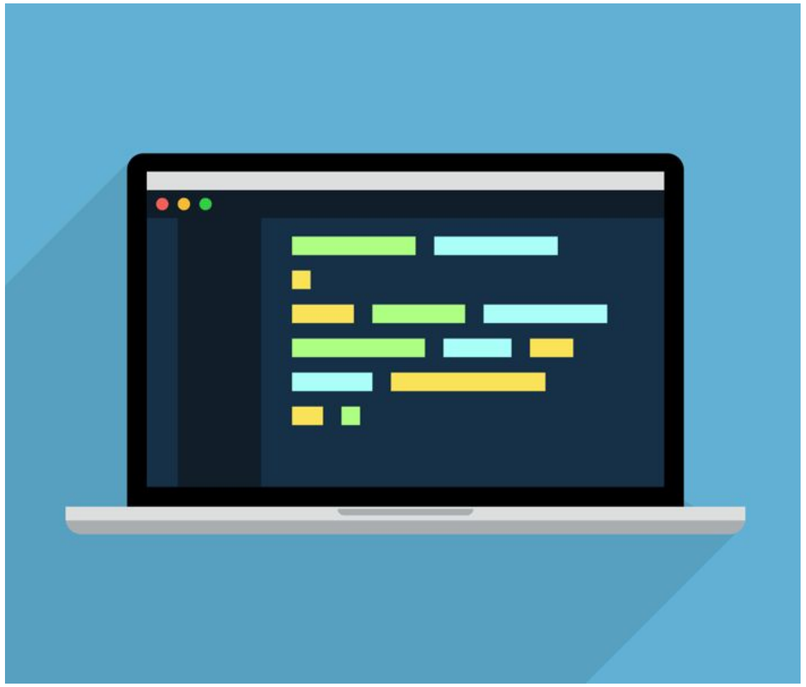
Newton School of Technology

Coding Question

Q1. Lets Make Sandwich

Quick Recap :

- if-else
- if-elif-else ladder
- Conditional statement with Logical Operators



Newton School of Technology

Scope of Variables

What will be the output?

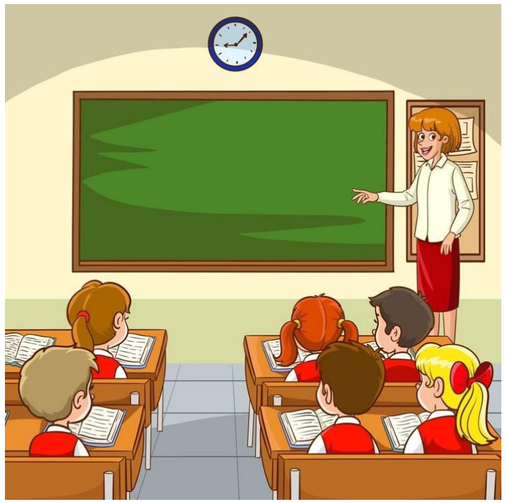
```
def add_and_subtract_two_nums(a, b):
    c = a + b
    return c
    d = b - a
    return d

print(add_and_subtract_two_nums(10, 20))
```

Once the return statement is executed in a function it exits and rest of the code is not executed.

Scope : School analogy

Local Scope: Classroom rules

Global Scope: Principal's rules



Scope refers to the region of the program where a variable is recognized. If a variable is out of scope, it cannot be accessed or used.

Local scope :

Newton School of Technology

Variables defined inside a function or a block are said to have a local scope.

These variables can only be accessed within the function or block where they are defined.

Scope : State VS Central Schemes

Local Scope: MH State Govt Schemes

Global Scope: Central Govt Schemes

Scope refers to the region of the program where a variable is recognized. If a variable is out of scope, it cannot be accessed or used.

Local scope :

Variables defined inside a function or a block are said to have a local scope.

These variables can only be accessed within the function or block where they are defined.

Global scope

Variables defined at the top level of a script or module, outside any function or block, are said to have a global scope.

These variables can be accessed anywhere in the module.

Newton School of Technology

Global Keyword

Local scope: Example

Newton School of Technology

```
def cafeteria():
    lunch_special = "Pizza"  # Local variable
    print("Today's special:", lunch_special)

cafeteria()  # ✅ Works (output: "Today's special: Pizza")
print(lunch_special)  # ❌ CRASH! NameError: 'lunch_special' not defined
```

Global Scope: Example

```
tv_channel = "Cartoon Network"  # 📺 Global variable (living room TV)

# Any function can READ it directly
def dad_watching():
    print("Dad is watching", tv_channel)  # ✅ Just use it!

dad_watching()  # Output: "Dad is watching Cartoon Network"
```

Resolving scope: Example

Local scope/variable will be given priority over global variable/scope.

```
# Global variable declaration
x = "I am a global variable"

def my_function():
    # Local variable declaration with the same name as the global variable
    x = "I am a local variable"
    print(x)  # Output: I am a local variable

# Calling the function to demonstrate local variable usage
my_function()

# Accessing the global variable outside the function
print(x)  # Output: I am a global variable
```

Resolving scope: Error

```
num = 10  # Global variable

def check_scope():
    num2 = 10 + num  # Gives error as num used before declaration
    num = 5  # Local variable num
    print(num2)

check_scope()
```

Resolving Scope: Fix for the problem

Newton School of Technology

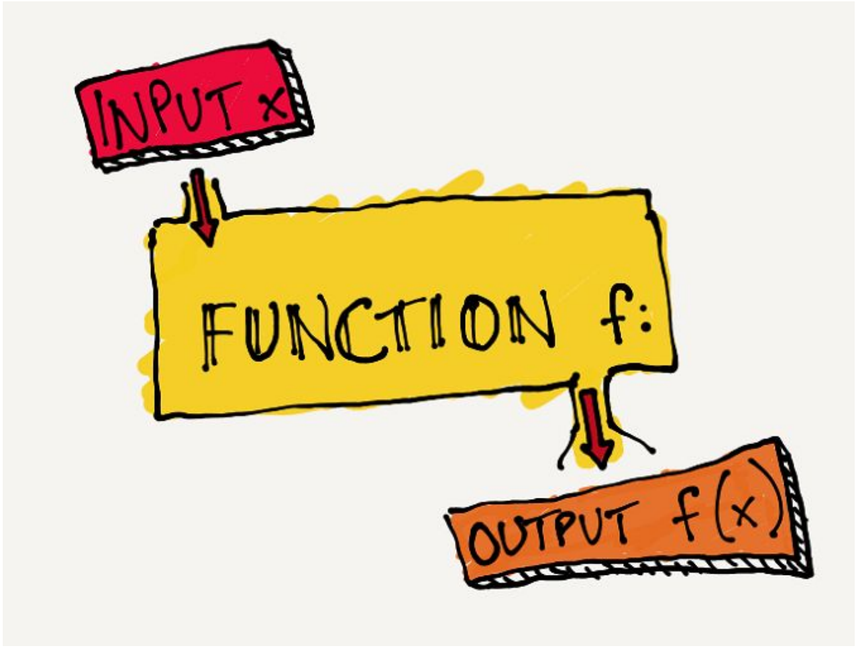
Ideally you should avoid variable with same names in global and local scope.

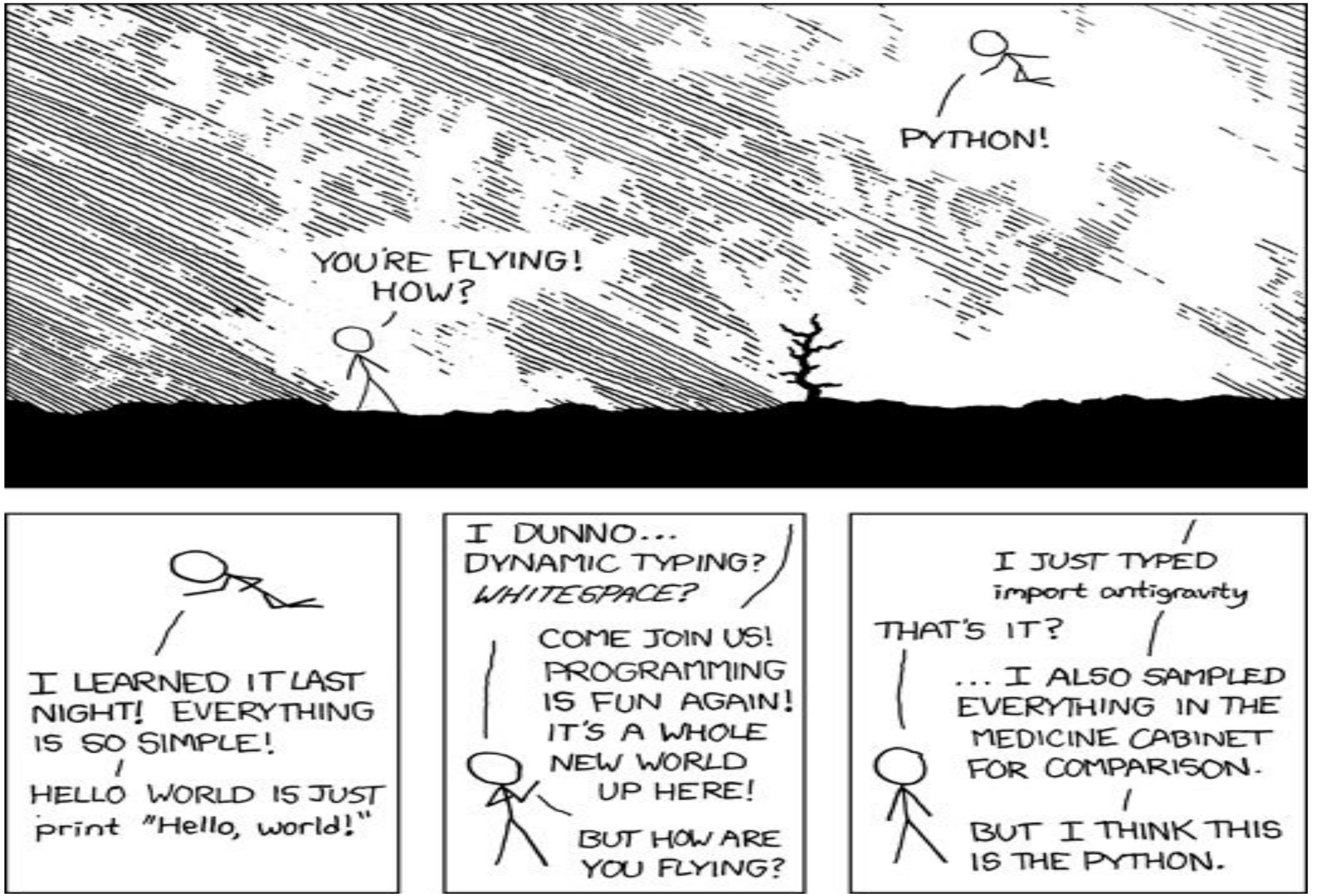
```
num = 10  # Global variable

def check_scope():
    global num  # Declare num as global
    num2 = 10 + num  # Accessing global variable num
    num = 5  # Modifying global variable num
    print(num2)  # Output: 20

check_scope()
print(num)  # Output: 5
```

Washing Machine :

Washing Machine :

Washing Machine :




PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

What are arguments ?

Newton School of Technology

When you call a function, the **values** you pass into it are called **arguments**.

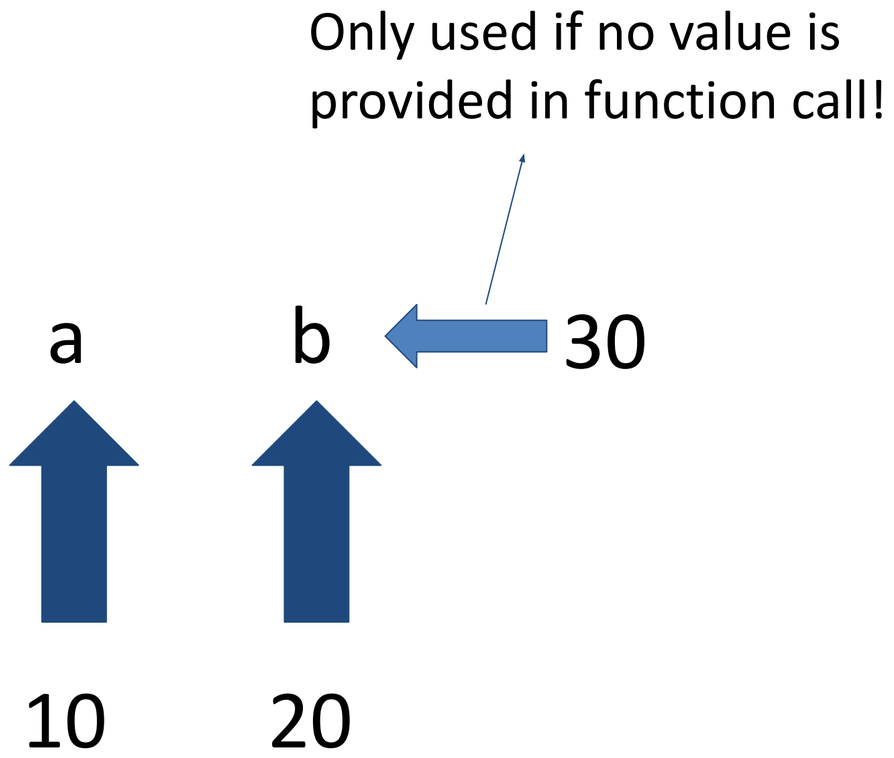
They’re like the **specific choices** you give when ordering a pizza.

Default Arguments: Example

```python
#Function to add two numbers
def add_two_nums(a, b=30):
    c = a + b
    print(c)

add_two_nums(10, 20)
add_two_nums(10)
```

[Code_link]

Output:
30
40



Arguments

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

```
add(5, 3)
```

Code link

Arguments are the actual values you send into a function, so it can do something useful.