**PrintsImpulse Global**

***info@printsimpulse.com.ng***

ABOUT THIS COURSE

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

**Data science full course**

**Introduction to Complete Data Science Curriculum.**

**📅 10-Week Data Science Teaching Timetable**

**Table of Content**

**Week 1 – Python & Math Foundations**

**Goal:** Build Python fluency + statistics basics.

* **Day 1:** Python setup (Anaconda/Jupyter/Colab), variables, data types.
* **Day 2:** Loops, functions, modules.
* **Day 3:** Lists, dictionaries, tuples, sets.
* **Day 4:** Math for data – mean, median, mode, variance, standard deviation.
* **Day 5:** Probability basics (independent, conditional, Bayes rule).  
  **Exercise:** Write Python functions to compute mean/median/mode/variance.

**Week 2 – NumPy, Pandas & EDA**

**Goal:** Learn data wrangling and exploratory data analysis.

* **Day 1:** NumPy arrays, slicing, matrix operations.
* **Day 2:** Pandas DataFrames, importing CSVs, basic operations.
* **Day 3:** Handling missing data, duplicates, outliers.
* **Day 4:** Descriptive statistics, groupby, pivot tables.
* **Day 5:** Introduction to visualization (Matplotlib basics).  
  **Mini Project:** Analyze a student scores dataset (find top performers, averages, etc.).

**Week 3 – Data Visualization**

**Goal:** Master storytelling with data.

* **Day 1:** Matplotlib advanced (line/bar/scatter, formatting).
* **Day 2:** Seaborn basics (histogram, boxplot, heatmap).
* **Day 3:** Pairplots, correlation analysis.
* **Day 4:** Case study – EDA on real dataset (e.g., Titanic dataset).
* **Day 5:** Power BI/Tableau intro – dashboard creation.  
  **Mini Project:** Build a sales dashboard showing KPIs & trends.

**Week 4 – SQL for Data Analysis**

**Goal:** Query structured databases.

* **Day 1:** SELECT, WHERE, ORDER BY.
* **Day 2:** GROUP BY, HAVING, aggregate functions.
* **Day 3:** Joins (INNER, LEFT, RIGHT, FULL).
* **Day 4:** Subqueries, CTEs, Window functions.
* **Day 5:** Integrating SQL with Python (SQLite).  
  **Mini Project:** Analyze a sales DB (e.g., top 5 products, revenue by region).

**Week 5 – Statistics for Data Science**

**Goal:** Apply inferential statistics.

* **Day 1:** Hypothesis testing (null vs alternative).
* **Day 2:** T-tests, ANOVA.
* **Day 3:** Chi-square tests, correlation vs causation.
* **Day 4:** Regression basics (linear regression theory).
* **Day 5:** Hands-on regression with Python.  
  **Mini Project:** Analyze exam score data → “Does study time impact grades?”

**Week 6 – Machine Learning Foundations**

**Goal:** Understand ML concepts and first models.

* **Day 1:** What is ML? Training vs testing, supervised vs unsupervised.
* **Day 2:** Linear regression in scikit-learn.
* **Day 3:** Logistic regression for classification.
* **Day 4:** Model evaluation (accuracy, precision, recall, F1-score).
* **Day 5:** K-means clustering.  
  **Mini Project:** Predict house prices with linear regression.

**Week 7 – Intermediate ML**

**Goal:** Work with more models + feature engineering.

* **Day 1:** Decision Trees.
* **Day 2:** Random Forests.
* **Day 3:** Feature scaling, encoding categorical data.
* **Day 4:** Train/test split, cross-validation.
* **Day 5:** Model comparison exercise.  
  **Mini Project:** Customer churn prediction.

**Week 8 – Data Engineering & Automation**

**Goal:** Automate workflows + intro to big data.

* **Day 1:** What is ETL? Data pipeline concepts.
* **Day 2:** Reading/writing multiple files in Python.
* **Day 3:** Automating data cleaning with functions.
* **Day 4:** Intro to Big Data (Spark, Hadoop – concept level).
* **Day 5:** Cloud basics (Google Colab, Kaggle, optional AWS).  
  **Mini Project:** Build a Python script that merges and cleans multiple CSVs.

**Week 9 – Capstone Project**

**Goal:** Apply all learned skills to a real dataset.

* **Day 1:** Select dataset (from Kaggle, World Bank, government data).
* **Day 2:** Data cleaning & preprocessing.
* **Day 3:** Exploratory analysis & visualization.
* **Day 4:** Build ML model (regression/classification).
* **Day 5:** Present results in a dashboard/report.

**Week 10 – Career & Portfolio**

**Goal:** Prepare for jobs/internships.

* **Day 1:** GitHub setup – push all projects.
* **Day 2:** LinkedIn profile optimization (Data Analyst/Scientist).
* **Day 3:** Resume building – emphasize projects & skills.
* **Day 4:** Mock interview – SQL, Python, case study questions.
* **Day 5:** Final presentation of capstone project.

**✅ Outcomes After 2.5 Months**

* **Skills:** Python, SQL, BI tools, Statistics, ML basics.
* **Portfolio:** 5+ mini-projects + 1 capstone project.
* **Career-Ready:** Job as Data Analyst, pathway to Data Scientist.

**Week 1 – Python & Math Foundations**

**Goal:** Build Python fluency + statistics basics.

* **Day 1: Python setup (Anaconda / Jupyter Notebook / Google Colab / Visual Studio Code), variables, data types.**
* **Day 2: Loops, functions, modules.**
* **Day 3: Lists, dictionaries, tuples, sets.**
* **Day 4: Math for data – mean, median, mode, variance, standard deviation.**
* **Day 5: Probability basics (independent, conditional, Bayes rule).  
  Exercise: Write Python functions to compute mean/median/mode/variance.**

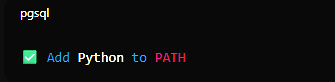
**Day 1: Python setup + variables + data types to “200%” ✅**  
This first session will focus on:

1. **Setting up Python in VS Code**
2. **Understanding Variables**
3. **Exploring Data Types …..(Python Types)**
4. **Writing and running Python code confidently**

## Step 1 – Installing and Setting Up Python in VS Code

### ✅ 1. Install Python

If you don’t already have Python installed:

* Go to [**https://python.org/downloads**](https://www.python.org/downloads/)
* Download **Python 3.x (latest version)** for your system (Windows, Mac, or Linux).
* During installation, **check the box** that says:

*This ensures your computer recognizes Python commands.*

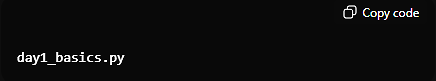
### ✅ 2. Install VS Code

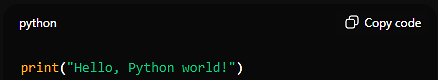
You already have VS Code, great!  
If not: download from [**https://code.visualstudio.com**](https://code.visualstudio.com)**. *(download for your operating system)***

### ✅ 3. Install the Python Extension in VS Code

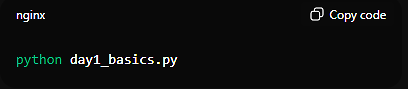
* Open VS Code.
* Press Ctrl + Shift + X (to open Extensions).
* Search **"Python"**.
* Click **Install** on the one made by Microsoft.
* ***(Note: If you want less stress of Manual Saving of your Codes while writing Python Codes, please turn on Auto Save by Clicking inside VS Code [ File – Auto Save {click to Tick it}]***

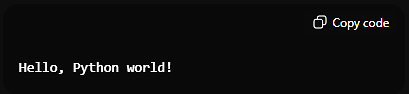
### ✅ 4. Create Your First Python File

1. Create a new folder anywhere, e.g., **C:\PythonPractice**
2. Open that folder in VS Code.
3. Create a new file named:
4. Write this simple code:



1. Save it.
2. To Run the Code:

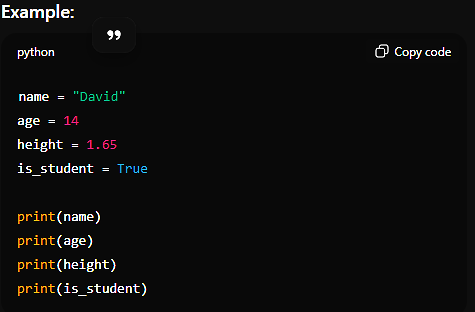
* Press **Ctrl + Shift + P**
* Type **“Python: Select Interpreter”** and choose your installed Python version.
* Then, open the terminal in VS Code using (**Ctrl + J**).
* Type:
* You should see the output as:



***🎉 Congratulations —*** *you’ve officially run your first Python program!*

**Step 2 – Variables**

A **variable** is like a container, where you can store items such as Liquids or solid objects, and when poured upside down, you get to see the content of the container, same goes to variables, you can save any of the Python Data Types in it and when printed, you see the contents.



**Value is: Data Type:** Boolean(True/False)

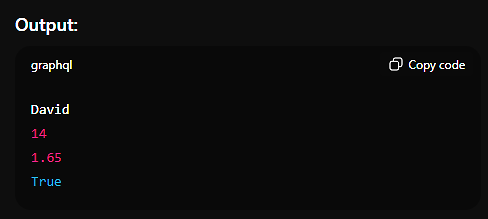
**Value is: Data Type:** Number(float)

**Value is: Type:** Number(int)

**Value is: Data Type:** str

Assignment Operator

Variable Name

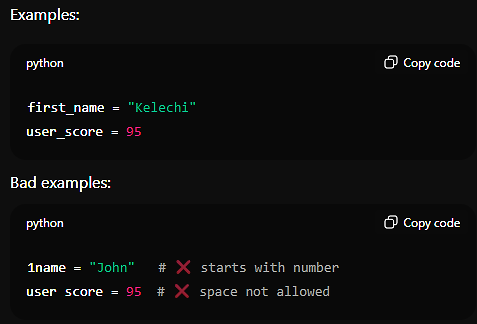


### **🔍 What’s happening here?**

* **name** = "David" → We store text (called a string) in a variable named name.
* **age** = 14 → We store a number without decimals (integer).
* **height** = 1.65 → A number **with** decimals (float) – floating point number.
* **is\_student** = True → A Boolean (True/False) value.

### **Variable Rules:**

✅ You can use letters(Stick with Alphabets a – z, **Instead of Optionally, A – Z**), numbers, and underscores ( \_ )  
❌ You **cannot** start a variable name with a number (0 – 9)  
❌ No spaces allowed  
✅ Use or stick with lowercase\_with\_underscores for readability



**🔢 Step 3 – Data Types in Python**

***Most frequently used Data Types in Python.***

Everything in Python has a **Type**.  
Here are the main ones:

| **Type** | **Example** | **Description** |
| --- | --- | --- |
| int | 500 | Whole number |
| float | 5.7 | Number with decimal |
| str | "Hello" or ‘Richard’ or “400” or ‘7&&’ | Text |
| bool | True / False | Logical value |
| list | [1, 2, 3.9, “Talk”] | Collection of values |
| tuple | (1, 2, 3) | Unchangeable list |
| dict | {"name": "John", "age": 14, "height": 173.6} | Key–value pairs |

***Example Code (try this out):***

# Integers

score = 87

age = 14

# Float

pi = 3.14159

height = 1.65

# String

greeting = "Hello there!"

# Boolean

is\_learning\_python = False

# List (a group of items)

fruits = ["apple", "banana", "cherry"]

# Tuple (like a list, but cannot be changed)

dimensions = (20, 40)

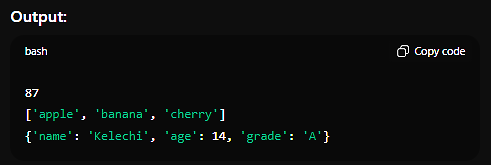
# Dictionary (store key-value info)

student = {"name": "Kelechi", "age": 14, "grade": "A"}

print(score)

print(fruits)

print(student)



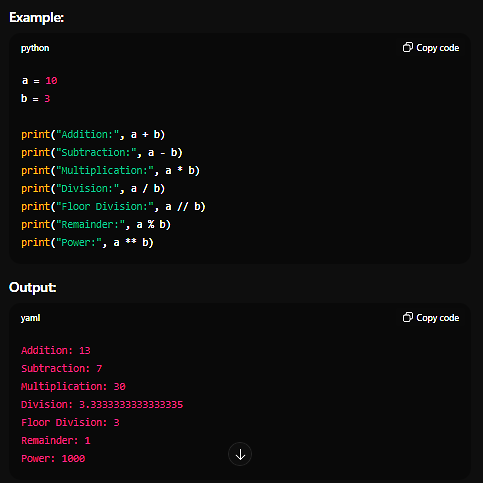
**💡 Step 4 – Type Checking**

We can always check a variable’s **Data Type** using the Python’s Built-in Function called **type() .** 

**Step 5 – Doing Math in Python**

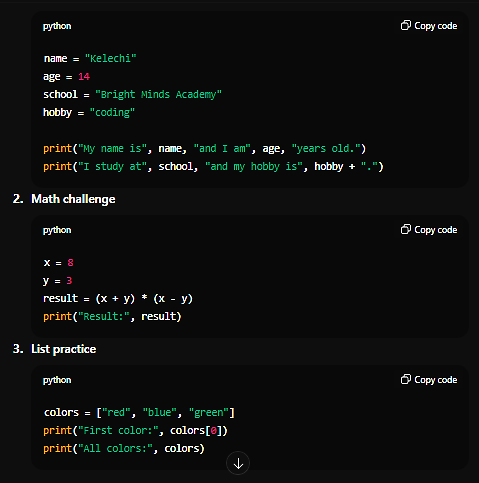
Python can act like a calculator.

| **Operator** | **Example** | **Meaning** |
| --- | --- | --- |
| + | 4 + 5 | Addition |
| - | 7 - 2 | Subtraction |
| \* | 3 \* 4 | Multiplication |
| / | 8 / 2 | Division (gives float) |
| // | 8 // 2 | Floor division (gives int) |
| % | 8 % 3 | Remainder (Modulus) |
| \*\* | 2 \*\* 4 | Exponent (Power) (2³ = 8) |



## Step 6 – Small Practice Exercises

Try these on your own in VS Code:



1. **Create a profile summary**

**🏁 Step 7 – Summary**

You’ve learned:  
✅ How to run Python in VS Code  
✅ What variables are and how to name them  
✅ Common data types in Python  
✅ How to check variable types  
✅ How to perform basic math operations

**Week 1 – Python & Math Foundations**

**Goal:** Build Python fluency + statistics basics.

* **Day 2: Loops, functions, modules.**

## Day 2: Loops, functions, modules.✅ 🎯 Goal:

By the end of today, you’ll be able to:  
✅ Repeat actions using **loops**  
✅ Write reusable code with **functions**  
✅ Organize your code into **modules**

**🏁 Step 1 – LOOPS**

A **loop** lets your program repeat actions without rewriting code.

Imagine you want to say “Hello” to 5 students.  
Without loops, you’d do:

print("Hello student 1")

print("Hello student 2")

print("Hello student 3")

print("Hello student 4")

print("Hello student 5")

With a **loop**, you can do it in just 3 lines!

### **🔹 1. The for loop**

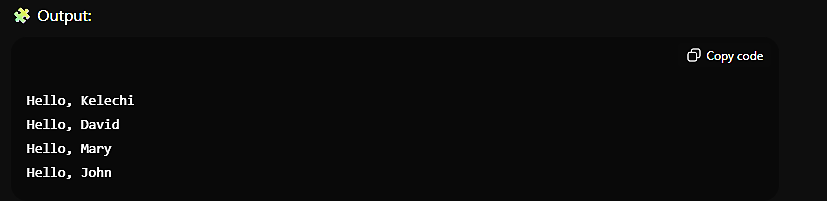
The **for** loop repeats for each item in a list, string, or range.

#### **Example 1 – Looping through a list**

students = ["Kelechi", "David", "Mary", "John"]

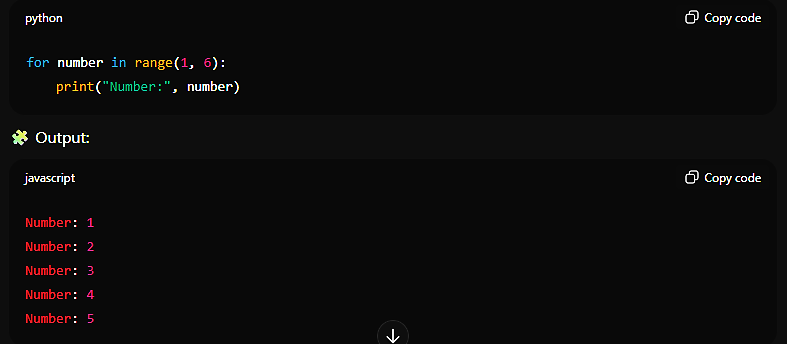
for student in students:

    print("Hello,", student)



#### **Example 2 – Using range()**

**range()**function generates a sequence of numbers.



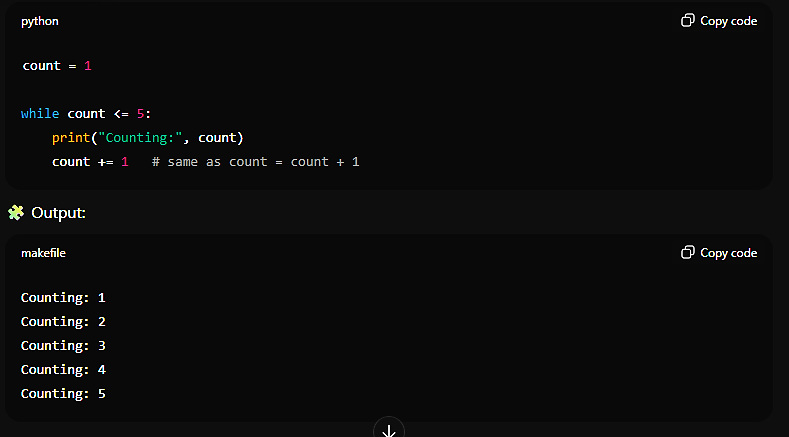
***🔎 Notes:***

range(1, 6) → starts at 1, stops before 6 (so it ends at 5).

You can also do range(6) which means 0 → 5.

### **🔹 2. The while loop**

**while** loops continue **as long as a condition is True**.



**🔎 What’s happening:**

The loop starts at count = 1

It keeps running while count <= 5 **( < less than or = equal to)**

Each time, we add 1 to count **(count = count + 1 or count += 1)**

When count becomes 6, the condition is no longer true, so the loop stops.

### **⚠️ Common Mistake: Infinite Loop**

If you forget to update your variable:

count = 1

while count <= 5:

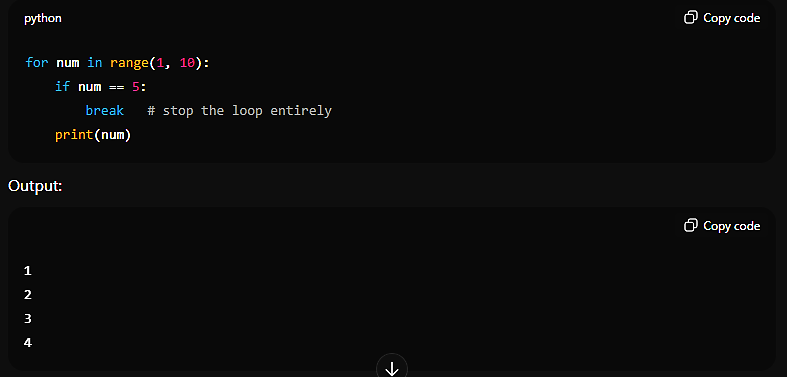
    print(count)

This will never stop **😱** because **count** never increases!  
It’ll print **1** forever until you stop the program (**Ctrl + C** in terminal).

### **🔹 3. break and continue**

* **break** → stops the loop immediately.
* **continue** → skips the current round and moves to the next one.

***Example 1:***



### **🔍 What’s happening here?**

* If num == 5: this checks the looped numbers to see if it is equal to 5(== means equality sign)
* **Break**: Since **break** is attached to the **if statement**, it acts as a **stopper** for **Terminating** the program immediately the **if condition** is met.

***Example 2:***  


### **🔍 What’s happening here?**

* If num == 5: this checks the looped numbers to see if it is equal to 5(== means equality sign)
* **continue**: Since **continue** is attached to the **if statement**, this time, instead of **Terminating** the program, it picks the equaled number and remove it from the Loop and allows the program to continue if the **if condition** is met. Meaning, it will skip the very equaled number and continue with the loop.

**⚙️ Step 2 – FUNCTIONS**

A **function** is like a **small** **machine**:

you feed it input (**data**), it does something, and gives you output (**result**).

**🔹 1. Creating and Calling a Function**

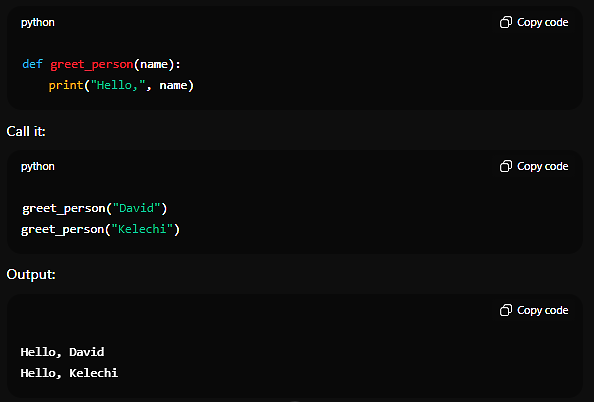
****

### **🔍 What’s happening here?**

* **def**: this is a Python built-in command we’d always use to define a function in Python.
* **greet()**: this is the name of the function, it can be anything but short and meaningful.
* **print(“Hello there!”)**: this is what would be printed to the console whenever the greet() function is called.

### **🔹 2. Functions with Parameters**

A **parameter** is a variable inside a function that takes data from the outside.

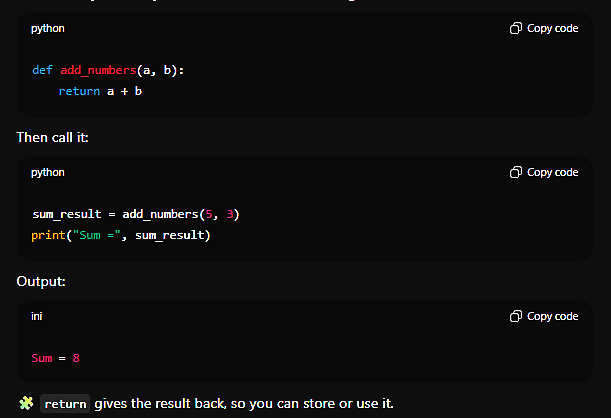
Example:

### **🔍 What’s happening here?**

* **def**: this is a Python built-in command we’d always use to define a function in Python.
* **greet\_person(name)**: this is the name of the function, and the **(name)** is the parameter for the function **greet\_name()**.. whenever the function is being called, the **(name)** will be replaced with any **Data Type** as an Argument every time it’s called.
* **print (“Hello, ”, name)**: this is the logic that prints “Hello ” with whatever the user pass to the **(name)** parameter.
* **greet\_person(“David”)**: this is how to call a function with the argument, **(“David”)** so serves as the Argument that tells the interpreter to display “Hello, David”

### **🔹 3. Functions that Return a Value**

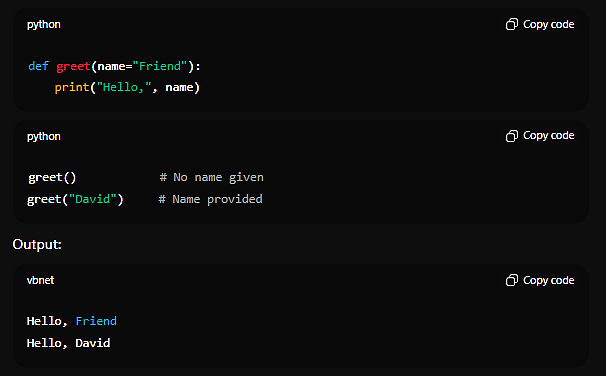
Sometimes you want your function to **calculate** and give a result.



### **🔍 What’s happening here?**

* **def**: this is a Python built-in command we’d always use to define a function in Python.
* **add\_numbers(a, b)**: this is the name of the function, and the **(a, b)** are the given parameters for the function **add\_numbers()**.. whenever the function is being called, the **(a, b)** will be replaced with **Data Types** that Arithmetic operations could be performed on, as an Argument every time it’s called. Example is this:
  + A variable was created called sum\_result and assigned a value using the **Assignment operator ( = )**, the value was the **function** created earlier with two **Data Types (‘a’ and ‘b’)**, both are Integers and separated with commas and the result was printed as usual.

### **🔹 4. Functions with Default Parameters**

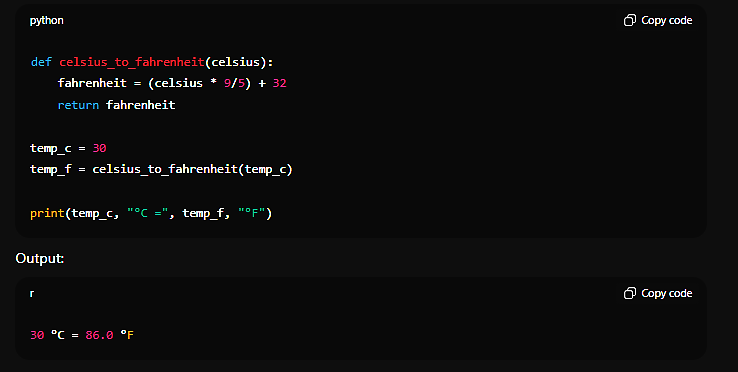


### **🔍 What’s happening here?**

* **def**: this is a Python built-in command we’d always use to define a function in Python.
* **greet(name = “Friend”)**: this is the name of the function, and the **(name = “Friend”)** is the given parameter for the function **greet()**. Whenever this function is called, it prints exactly the **Logic** but with a default attachment(value) **“Friend”**, if the **name** parameter isn’t given any **value**.. as the example above is illustrated.

**🔹 5. Function Example: Temperature Converter**

Let’s make something practical: ***(Fahrenheit vs Celsius °)***



### **🔍 What’s happening in the code**

def celsius\_to\_fahrenheit(celsius):

fahrenheit = (celsius \* 9/5) + 32

return fahrenheit

#### **🔹 def**

This keyword means **“define a function”** in Python — it’s how you tell Python you’re creating a reusable block of code.

#### **🔹 celsius\_to\_fahrenheit**

This is the **function name** — you can name it anything descriptive.  
In this case, it tells us the function converts Celsius temperature to Fahrenheit.

#### **🔹 (celsius)**

That’s the **parameter** — a placeholder variable that the function will receive when it’s called.

#### **🔹 The formula**

fahrenheit = (celsius \* 9/5) + 32

That’s the mathematical conversion from °C → °F.

#### **🔹 return fahrenheit**

The **return statement** sends the result (in this case, the Fahrenheit value) back to the caller.

### **Then, the rest of the code:**

**temp\_c** = 30

**temp\_f** = celsius\_to\_fahrenheit(**temp\_c**)

* Here, temp\_c holds 30 (degrees Celsius).
* The function celsius\_to\_fahrenheit(temp\_c) runs and gives back 86.0, which is stored in temp\_f.

### **🖨️ Finally:**

print(temp\_c, "°C =", temp\_f, "°F")

That prints:

30 °C = 86.0 °F

### **✅ Output meaning**

So the **function converts 30°C into Fahrenheit (86°F)**, then prints it out clearly.

Top of Form

Bottom of Form