Start coding with PYTHON in 5 minutes! 🐍

🐍 Getting Started with Python Programming

Welcome to your very first Python lesson! In this article, you'll learn **how to set up Python**, write your **first lines of code**, and understand some key programming concepts — all explained clearly for beginners.

By the end, you'll know how to:

- Download and install Python
- Set up a coding environment (IDE)
- Write your first program
- Use the `print()` function
- Add comments to your code

Let's get started!

🧠 What Is Python?

Python is a **programming language** — a way to give instructions to a computer. You write commands in plain English-like text, and Python turns those commands into actions.

For example, you can use Python to:

- Build apps and games
- Analyze data
- Automate tasks
- Create websites

But before we can write Python code, we need to set up our tools.

💻 Step 1: Installing the Python Interpreter

A **Python interpreter** is the software that reads and runs your Python code. Without it, your computer wouldn't understand what your code means.

How to Install Python

1. Go to python.org.
2. Click on the **Downloads** tab and choose the latest version for your operating system.
3. Open the file you downloaded.
4. **If you're using Windows**, make sure you check the box that says:
   ✅ **Add Python.exe to PATH**
   This allows you to run Python from anywhere on your computer.
5. Click **Install** and wait for it to finish.

When the installation is complete, you'll see a message saying **"Setup was successful."**
Now you have Python installed!

🧩 Step 2: Installing an IDE (Integrated Development Environment)

An **IDE** is like a digital notebook for writing, running, and testing your code.
Think of it as your **coding workspace**.

There are many IDEs available, but the two most popular for Python are:

1. **PyCharm** – beginner-friendly and easy to use.
2. **VS Code** – a lightweight and powerful editor (you'll need to install the Python extension).

How to Install PyCharm

1. Go to **jetbrains.com/pycharm**.
2. Click the green **Download** button.
3. Choose the **Community Edition** (it's free).
4. Open the downloaded file and follow these steps:
   - Click **Next**.
   - Choose a destination folder (or leave the default).
   - Check the box for a **desktop shortcut** if you want one.
   - Click **Install**.

When the setup is complete, check **"Run PyCharm"** and click **Finish**.

🗂️ Step 3: Creating Your First Python Project

When PyCharm opens, it will ask you to create a new project.

1. Click **New Project**.
2. Choose a project name (e.g., "MyFirstProgram").
3. Select the latest Python version from the options.
4. Click **Create**.

Now you'll see your project window on the left side.
Right-click the project folder and select:
File → New → Python File
Name your file `main.py`.

> 🔍 *All Python files end with the* `.py` *extension.*

🖨️ Step 4: Writing Your First Program

Let's start by printing something to the console — the place where your program's output appears.

In your `main.py` file, type:

```
print("I like pizza")
```

Now, click the **green arrow** ▶️ at the top to run your program.

Output:

```
I like pizza
```

Congratulations! You just wrote and ran your first Python program.

📄 Step 5: Printing Multiple Lines

You can print as many lines as you want by adding more `print()` statements:

```
print("I like pizza")
print("It's really good")
```

Output:

```
I like pizza
It's really good
```

Each `print()` statement writes one line of text to the console.

💬 Step 6: Adding Comments

Sometimes you'll want to add **notes** to your code to explain what's happening.
These notes are called **comments**.

In Python, comments start with a # symbol.
The computer ignores anything after this symbol.

Example:

```
# This is my first Python program
print("I like pizza")
print("It's really good")
```

Output:

```
I like pizza
It's really good
```

The comment doesn't appear in the output. It's only for you (and anyone else reading your code).

🧱 Step 7: Review and Key Ideas

Let's review what you learned:

CONCEPT	EXPLANATION	EXAMPLE
Interpreter	Runs your Python code	Installed from python.org
IDE	A program for writing and running code	PyCharm, VS Code

CONCEPT	EXPLANATION	EXAMPLE
print()	Displays text in the console	`print("Hello")`
Comment	Notes ignored by Python	`# This is a comment`
.py file	The file type for Python code	`main.py`

🧠 Challenge Practice

Try these mini exercises to test your skills:

1. Print your name and favorite hobby. `print("My name is Alex") print("I love playing basketball")`
2. Add a comment explaining what the code does. `# This program prints my name and favorite hobby`
3. Change the text to describe your favorite movie or food.

🚀 What's Next?

In the next lesson, you'll learn about **variables** — a way to store and reuse information in your programs.

For now, enjoy what you've accomplished — you've taken your first big step into programming!

Learn Python VARIABLES in 10 minutes!

🐍 Introduction to Variables in Python

In this chapter, we will learn about **variables** and the **four main data types** in Python. These are:

1. **Strings**
2. **Integers**
3. **Floats**
4. **Booleans**

🌟 What is a Variable?

A **variable** is like a **container** that holds information. Think of it like a labeled box. Whatever you put in the box can be used later. Each variable should have a **unique name**, so you know what it stores.

🧠 Think About It:

If you have a box labeled `first_name`, and you put the word `"Alex"` in it, anytime you use `first_name`, Python will treat it like it's the word `"Alex"`.

📝 Creating a Variable

In Python, to **assign** a value to a variable, we use the **equals sign** `=`.

Example:

```
first_name = "Alex"
```

In this case, `first_name` is the variable, and `"Alex"` is the value stored inside it.

🔤 1. Strings

A **string** is a series of characters. It can include **letters**, **numbers**, **symbols**, and **spaces**.

Strings are written between **quotes** — either **double quotes** (`"`) or **single quotes** (`'`). Most people use double quotes.

Example:

```
email = "alex123@example.com"
favorite_food = "Pizza"
```

✅ Printing Strings

You can show your string using the `print()` function.

```
print(first_name)
```

⚠️ Be Careful!

```
print("first_name")  # This will print: first_name (not the value!)
```

If you use **quotes around the variable name**, Python will think you mean the actual word `"first_name"` — not the value inside the variable.

🧠 F-Strings: Combining Text and Variables

An **f-string** lets you mix text and variables easily.

```
print(f"Hello {first_name}")
```

This prints something like:

```
Hello Alex
```

The `f` stands for **format**. Anything inside `{ }` will be replaced with the value of the variable.

Example:

```
food = "Pizza"
print(f"Hello {first_name}, you like {food}")
```

Output:

```
Hello Alex, you like Pizza
```

🔢 2. Integers

An **integer** (often shortened to **int**) is a **whole number**. It has **no decimal point**.

Examples:

```
age = 15
students_in_class = 30
items_bought = 3
```

Printing Integers:

```
print(f"You are {age} years old")
print(f"Your class has {students_in_class} students")
print(f"You are buying {items_bought} items")
```

❌ Don't Put Integers in Quotes

```
age = "15"  # This is actually a string!
```

If you put a number in quotes, it becomes a string, not an integer.

🔣 3. Floats

A **float** is a number that includes a **decimal point**. It can represent more precise values.

Examples:

```
price = 10.99
gpa = 3.2
distance = 5.5
```

Printing Floats:

```
print(f"The price is ${price}")
print(f"Your GPA is {gpa}")
print(f"You ran {distance} km")
```

✅ 4. Booleans

A **Boolean** is a special data type that only has **two possible values**:

- `True`
- `False`

Booleans are used to **answer yes or no questions**, like:

- Are you a student?
- Is the item for sale?
- Is someone online?

Example:

```
is_student = True
is_online = False
```

Printing Booleans:

```
print(f"Are you a student? {is_student}")
print(f"Are you online? {is_online}")
```

But usually, Booleans are used in **conditions**, not just printed directly.

❓ Using Booleans in an If Statement

Booleans are often used with **if** statements to control what your program does.

Example:

```
is_student = True

if is_student:
    print("You are a student")
else:
    print("You are not a student")
```

More Boolean Examples:

```
# Example 1: For Sale
for_sale = True

if for_sale:
    print("That item is for sale")
else:
    print("That item is not available")

# Example 2: Online Status
is_online = True

if is_online:
    print("You are online")
else:
    print("You are offline")
```

📦 Summary: Variable Types

TYPE	NAME	EXAMPLE	WHAT IT REPRESENTS
`str`	String	`"Hello"`	Text or characters
`int`	Integer	`15`	Whole numbers
`float`	Float	`3.14`	Numbers with decimals
`bool`	Boolean	`True` or `False`	Yes or No / On or Off / True or False

📝 Assignment

Create **four variables**:

1. A **string** (your name, favorite movie, etc.)
2. An **integer** (your age, number of pets, etc.)
3. A **float** (price of something, GPA, etc.)
4. A **Boolean** (are you hungry, is it raining, etc.)

Example:

```
name = "Jordan"
age = 14
gpa = 3.8
is_hungry = True

print(f"Hello {name}")
print(f"You are {age} years old")
print(f"Your GPA is {gpa}")
print(f"Are you hungry? {is_hungry}")
```

Try it yourself and experiment with different values!

🎉 Congratulations!

You've just learned how to:

- Create and use variables in Python
- Understand the four basic data types
- Use `print()` and `f-strings`
- Work with simple conditions using Booleans
