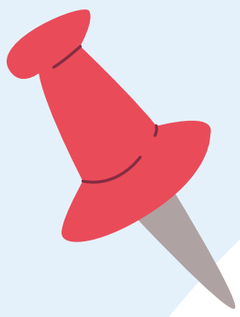
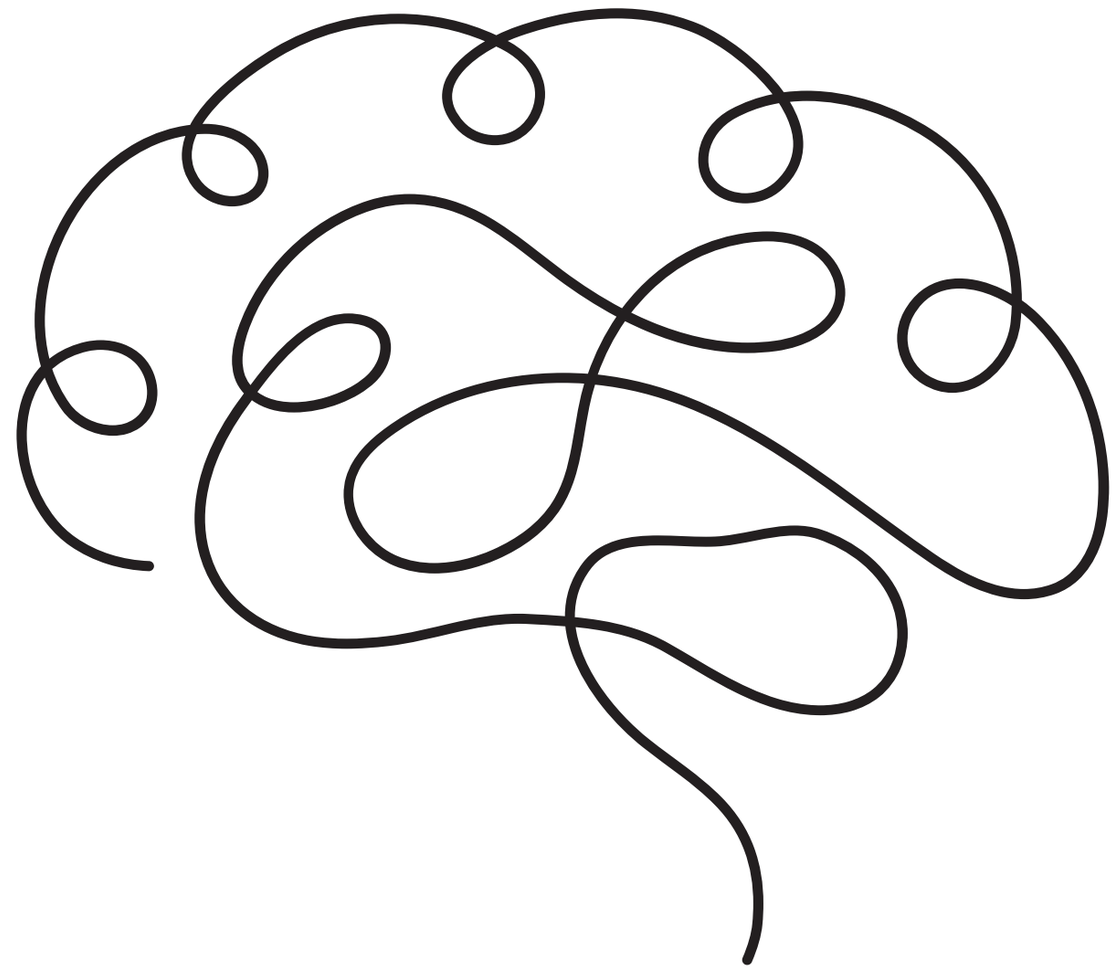
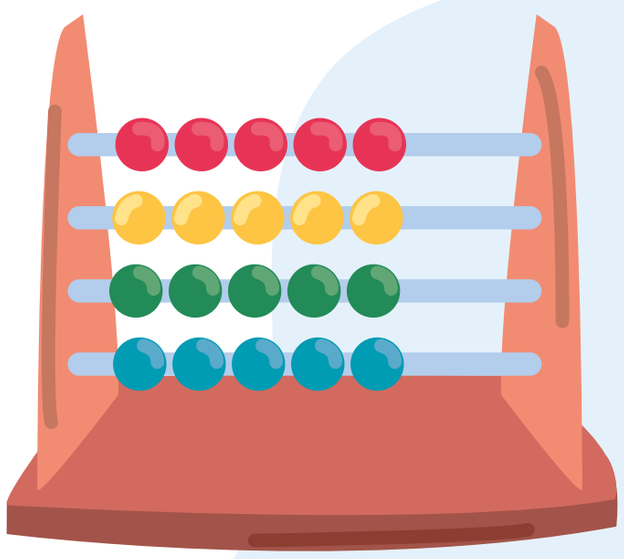




Python is *fun*

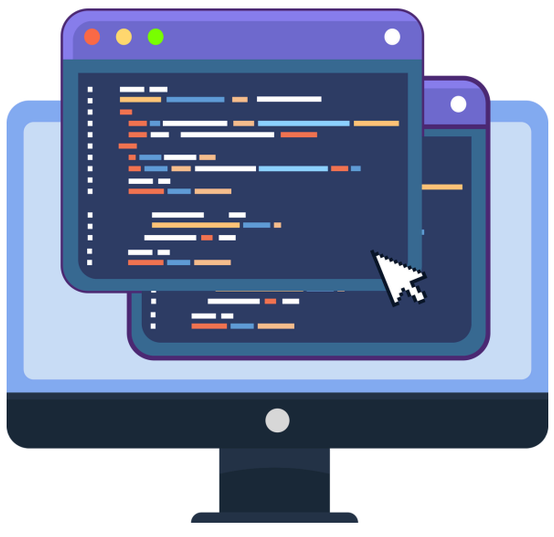


The Beginning – Why Programming?

> Computers are incredibly fast, accurate, and stupid. Humans are incredibly slow, inaccurate, and brilliant. *Together they are powerful beyond imagination.*

You're staring at a screen. Behind that screen lies the ability to control a machine that can perform billions of calculations per second, analyze mountains of data, or bring your most creative ideas to life. But it doesn’t understand plain English. It needs a translator—*you*.

That’s where programming comes in.



Why should *you* learn programming?

Programming is *problem-solving*: Whether it's calculating your monthly budget, building a mobile app, or automating a boring task, programming helps you break down complex problems into simple steps.

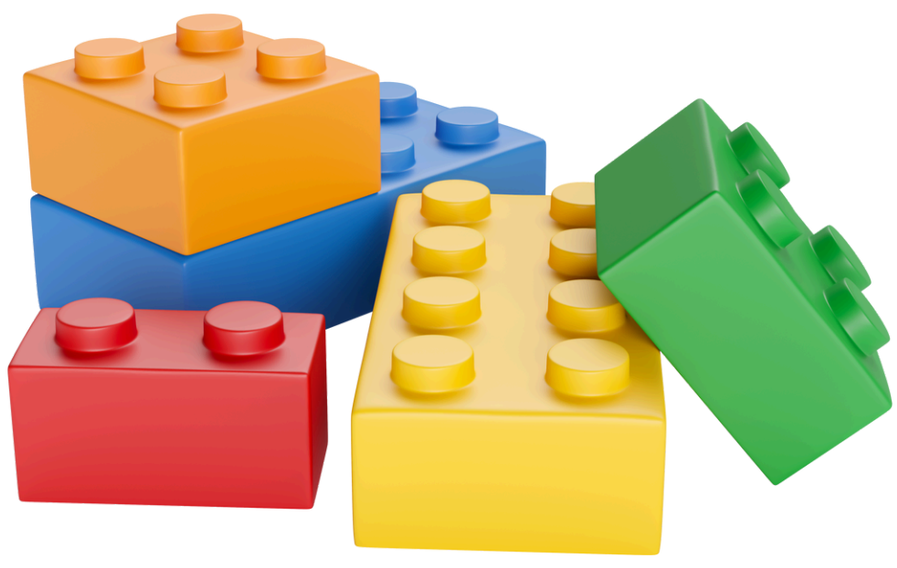
It's *everywhere*: From your phone to your fridge, your car to your calendar—code is behind it all. Learning programming opens your eyes to how the world around you works.

It amplifies your *creativity:* Have an idea for a game, a tool, a chatbot, or an art project? Programming is the canvas. You bring the imagination, code brings it to life.

It’s a *career* skill: Even if you don’t want to be a “programmer,” knowing the basics gives you a huge edge in fields like data science, business, design, biology, journalism—you name it.

You *don’t need to be a genius* to start: You don’t need to know math like Einstein. You don’t need to be a computer nerd.
You just need curiosity, and a willingness to try.

Think of programming as building something from blocks.
You start small: maybe print a message on the screen.
Then you *connect* the blocks—*variables*, *conditions*, *loops*.
Soon, you’re building full systems that talk to the internet, read files, or even play games.



sb
CHALLENGE YOURSELF

Why Python*?*

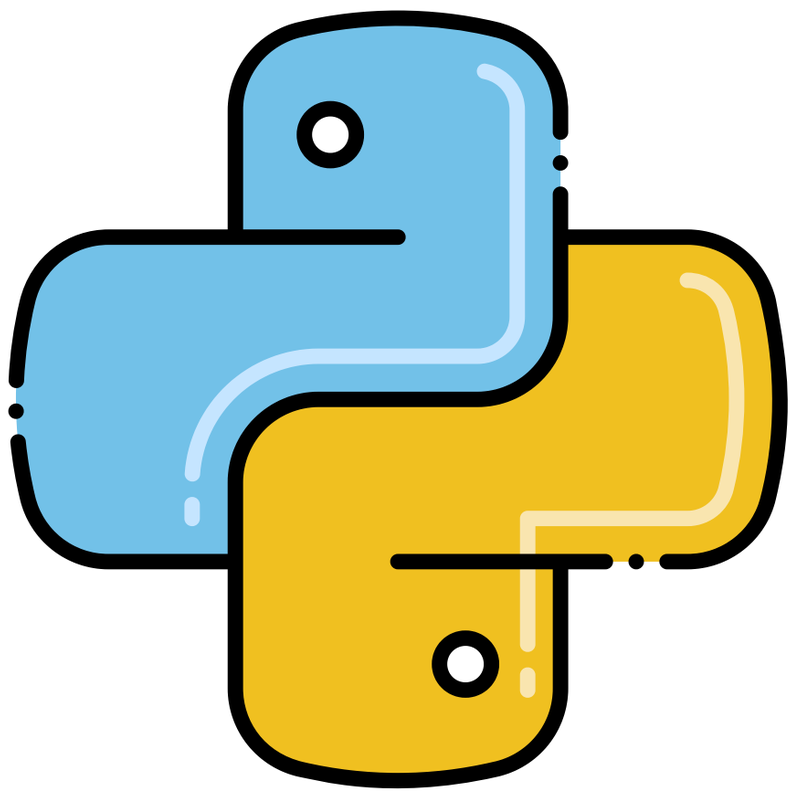
Python is beginner-friendly – the syntax reads like English. You *focus* on solving problems, not struggling with code structure.

Popular and widely used –From web apps to AI models, Python is everywhere. It’s trusted by startups and tech giants alike.

Huge community & resources – Stuck on something? A million others have been there. Tutorials, forums, and answers are just a search away.

Versatile – Want to build a website, analyze data, or make a game? Python can do it all.

Future-proof –Python powers tools at Google, Netflix, and NASA. Learning it today *opens* doors to tomorrow’s tech.



Origin of Python

I was looking for a hobby project during Christmas

– Guido van Rossum, Creator of Python

In the late 1980s, a Dutch programmer named Guido van Rossum decided to create a new programming language during his holiday break. He wanted something simple, elegant, and fun—a language that didn’t get in the way of thinking.

He called it Python.

No, not after the snake 🐍—but after his favorite British comedy group: *Monty Python’s Flying Circus*.

Python wasn’t built to impress.
It was built to make coding *easier*.

Setting Up – *Google Colab* Platform

So you're ready to write your first line of Python code—but where do you do that? You could install Python on your computer… but there's an easier, faster way to start: *Google Colab*.

💡 What is Google Colab?
Google Colab (short for Collaboratory) is a free, cloud-based coding environment. It lets you write and run Python code right in your web browser—no installation, no setup, no *stress*.

It's like a Google Doc, but for code.
And yes, it even *autosaves* to your Google Drive!

Why Colab?

Zero setup – Just open a browser and start coding.

Beginner-friendly – Simple, clean interface with helpful tips.

Free access to power – Run Python on Google's servers (even with GPUs!).

Shareable notebooks – like sharing a Doc or Slide.

🧱 What's a Notebook?
A notebook is your *coding journal*. It's made up of cells, where:

Code cells run Python code.

Text cells hold notes, explanations, etc. (written in *Markdown*).

Use code and text to make your work more readable and organized.

Building Blocks – Data Types & Variables

Before writing complex programs, we need to learn the small Lego blocks that make up every Python program:
data types and variables.

Think of data types as the *kinds of information* your program can handle—like numbers, words, or True/False values.

And variables? They're just names you give to those pieces of data so you can use them later.

🧱 What Are Data Types?

Python has several basic types of data. Let's meet the main ones:

Data Type	Example	Description
`int`	`10`, `-5`, `0`	Whole numbers
`float`	`3.14`, `-2.0`	Decimal numbers
`str`	`"Hello"`	Text (called *strings*)
`bool`	`True`, `False`	Boolean values (yes/no logic)

🏷️ What Are Variables?

A variable is like a labeled box where you store data.
You give it a name, assign a value, and you can use or change it later.

Strings – Working with *Text* in Python

So far, we've worked with numbers and variables. But what about words, names, or entire sentences? That’s where strings come in!

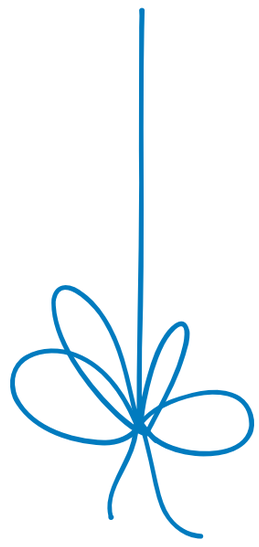
A string in Python is simply text enclosed in quotes.

You can use single ('), double ("), or even triple quotes (''' """) for multi-line text.

You can create a string by assigning text to a variable:

```
name = "Alice"
greeting = 'Hello, world!'
multiline_text = """This is
a multiline string."""
```

🔍 String Operations

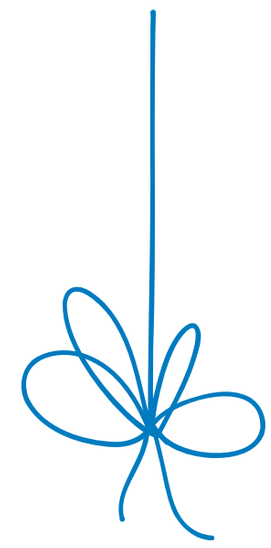


1️⃣ Combining Strings (Concatenation)

Use + to join strings together:

```
first = "Hello"
second = "Python"
result = first + " " + second
print(result)  # Output: Hello Python
```

🔍 String Operations



2️⃣ Repeating Strings

Multiply a string to repeat it

```
laugh = "Ha! "
print(laugh * 3)  # Output: Ha! Ha! Ha!
```

3️⃣ Accessing Characters

Strings are like arrays of characters—you can access them using an index (0 for the first letter):

```
word = "Python"
print(word[0])   # Output: P
print(word[-1])  # Output: n (last character)
```

4️⃣ Slicing Strings

Get part of a string using [:]

```
text = "Programming"
print(text[0:4])  # Output: Prog
print(text[:5])   # Output: Progr (from start)
print(text[4:])   # Output: ramming (to end)
```

🔍 String Operations

Python provides many *built-in* string functions to modify and format text

Method	Description	Example Output
`upper()`	Converts to UPPERCASE	`"hello".upper()` → `"HELLO"`
`lower()`	Converts to lowercase	`"HELLO".lower()` → `"hello"`
`title()`	Capitalizes Each Word	`"python rocks".title()` → `"Python Rocks"`
`strip()`	Removes spaces at the edges	`" hello ".strip()` → `"hello"`
`replace(a, b)`	Replaces `a` with `b`	`"cat".replace("c", "b")` → `"bat"`
`split()`	Splits into a list	`"a,b,c".split(",")` → `['a', 'b', 'c']`

🎭 Formatting Strings (f-strings)

Instead of concatenation, you can use f-strings (f"{}") to insert values into a string

```
name = "Alice"
age = 25
print(f"My name is {name} and I am {age} years old.")
# Output: My name is Alice and I am 25 years old.
```

💡 Challenge Yourself!

Try solving these in Google Colab:

1. Create a variable word = "Python" and print only "Pyt".
2. Convert " this is messy TEXT! " into "This Is Messy Text!".

sb
CHALLENGE YOURSELF

Lists – *Storing* Multiple Values in Python

So far, we've worked with single values like numbers and strings. But what if you need to store *multiple values* together?

That's where lists come in! 📝

A list in Python is like a container that holds multiple items, all in one place. Store numbers, strings, or even other lists inside it.

📌 Creating a List
Lists are created using *square brackets* [] and items are separated by commas.

```
# List of numbers
numbers = [10, 20, 30, 40, 50]

# List of strings
fruits = ["apple", "banana", "cherry"]

# Mixed data types
random_list = [5, "hello", True, 3.14]
```

✅ Lists can store different data types!
✅ Lists maintain order! (First item stays first, last stays last.)

Lists – *Storing* Multiple Values in Python

📍 Accessing Items in a List

Each item in a list has an index (starting from 0).

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0])   # Output: apple
print(fruits[2])   # Output: cherry
```

✂ Slicing a List

You can extract a portion of a list using slicing ([start:end]).

```
numbers = [10, 20, 30, 40, 50]

print(numbers[1:4])   # Output: [20, 30, 40]
print(numbers[:3])    # Output: [10, 20, 30]  (from start)
print(numbers[2:])    # Output: [30, 40, 50]  (to end)
```

🔄 Changing and Updating Lists

Lists are mutable, meaning you can change their values.

```
fruits = ["apple", "banana", "cherry"]
fruits[1] = "mango"   # Changing "banana" to "mango"
print(fruits)         # Output: ['apple', 'mango', 'cherry']
```

sb
CHALLENGE YOURSELF

Lists – *Storing* Multiple Values in Python

🛠️ List Methods (*Built-in* Tools)

Python provides powerful methods to modify lists.

Method	Description	Example Output
`append(x)`	Adds `x` to the end of the list	`fruits.append("orange")` → `['apple', 'banana', 'cherry', 'orange']`
`insert(i, x)`	Inserts `x` at index `i`	`fruits.insert(1, "grape")` → `['apple', 'grape', 'banana', 'cherry']`
`remove(x)`	Removes first occurrence of `x`	`fruits.remove("banana")` → `['apple', 'cherry']`
`pop(i)`	Removes and returns item at `i`	`fruits.pop(1)` → `['apple', 'cherry']`
`sort()`	Sorts the list in ascending order	`[3, 1, 2].sort()` → `[1, 2, 3]`
`reverse()`	Reverses the order of the list	`[1, 2, 3].reverse()` → `[3, 2, 1]`

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")  # Adds "orange"
fruits.sort()            # Sorts the list
print(fruits)            # Output: ['apple', 'banana', 'cherry', 'orange']
```

🔍 Checking if an Item Exists

Use the in keyword to check if an item is in a list.

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

print("banana" in fruits)  # Output: True
print("grape" in fruits)   # Output: False
```

Lists – *Storing* Multiple Values in Python

📊 List Length

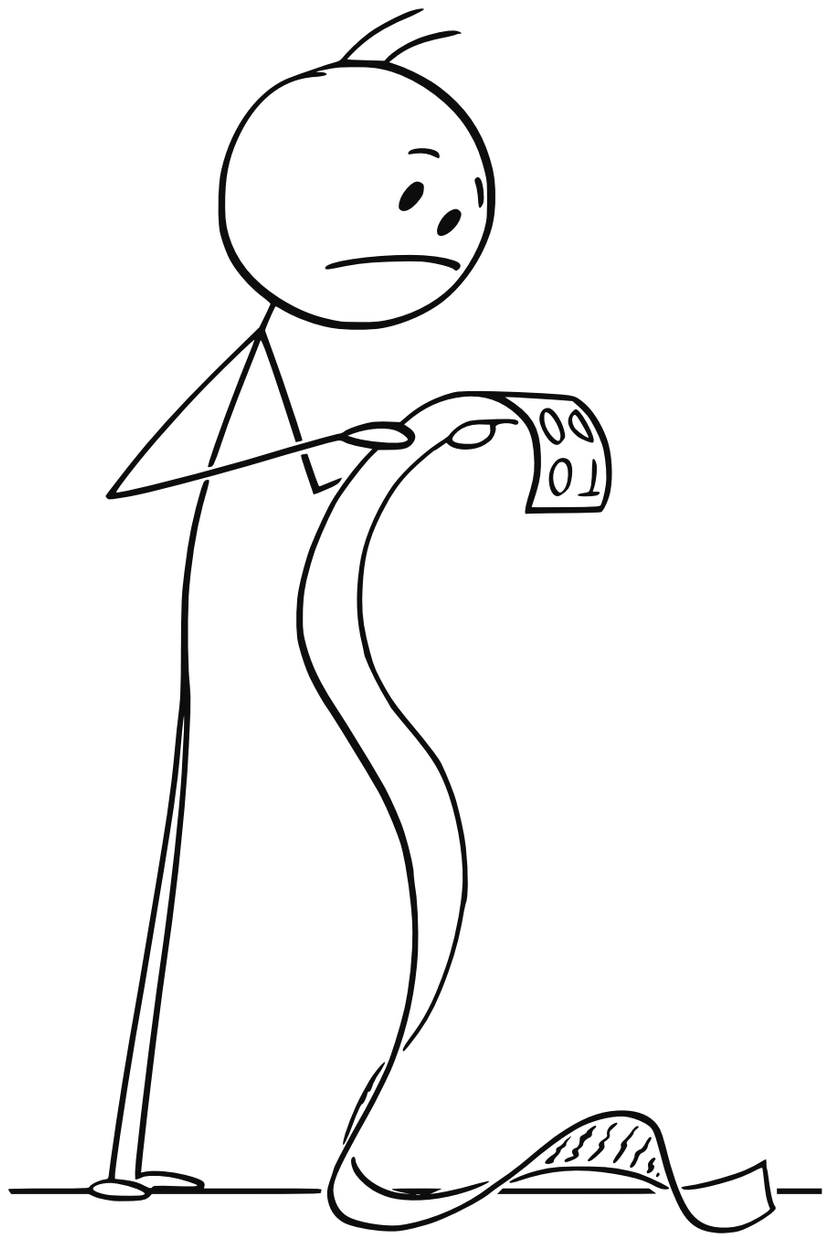
Find out how many items are in a list using len().

```
numbers = [10, 20, 30, 40]
print(len(numbers))  # Output: 4
```

💡 Challenge Yourself!

Try solving these in Google Colab:

1️⃣ Create a list of your 5 favorite movies and print the third one.
2️⃣ Add a new movie to the end of the list.
3️⃣ Remove the first and last item from the list.
4️⃣ Sort the list in alphabetical order and print it.



sb
CHALLENGE YOURSELF