

Learning Python

A Beginner's Guide!



Sriju, everyone's talking about Python. Isn't that a snake?



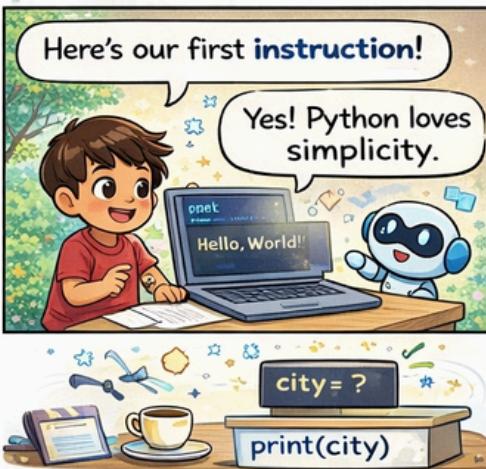
Python is a programming language.

So.. you "talk" to computers?



Here's our first instruction!

Yes! Python loves simplicity.



Think of variables as labeled boxes.

So the box named "age" stores 25?



Your turn! What should go inside the "city" box?

Bangalore
Paris

Tokyo



So this is just the beginning?

Python is used for:

- ✓ AI & Machine Learning
- ✓ Data Science
- ✓ Web Apps
- ✓ Automation



That's Episode 2!

See you in Episode 2!

Teaching Python to Decide: If, Else & Logic



Sriju, how does Python know
what to do?

Just like us –
it checks
conditions

A condition is a question
with only two answers.

```
age = 18  
if age == 18:  
    print("You can vote")  
else:
```

True or
False?

So Python checks the condition!

Always.

For more choices,
we use "elif".

For more choices,
we use "elif".

```
score = 35  
if score == 90:  
    print("A")  
elif score == 75:  
    print("Warm")  
else:  
    print("Cold")
```

What grade
will be printed?

What if it's 25
degrees?

Python.
checks from
top to bottom.

Today you learned:

- ✓ Conditions
- ✓ if / else
- ✓ elif
- ✓ True & False

That's
Episode 3!

Making Python Repeat Work:

Loops & Automation



Sriju, writing the same line over and over is **BORING!**

Perfect problem for Python.



A **loop** tells Python to repeat work.



It stops automatically?



```
for i in range(1, 6):  
    print(i)
```

```
fruits = ["apple"  
         "banana", "mango"]  
for fruit in fruits:  
    print(fruit)
```

fruits
["apple"]
["banana"]
- - -

Yes – when the range ends.

One item at a time!

```
fruits = ["apple"  
         "banana", "mango"]  
for fruit in fruits:  
    print(fruit)
```



Always make sure your loop can **stop**.

```
water = 0  
while water < 5:  
    water += 1
```



AI trains by repeating calculations.



Today you learned:

- ✓ Why loops exist
- ✓ **for** loops
- ✓ **while** loops
- ✓ Automation basics.

That's Episode 4!



Python's Memory Shelves: Lists, Tuples & Dictionaries

This is getting messy!

```
student1="Asha"  
student2="Ravi"  
student3="Meera"
```

Time for Python's memory shelves.



Lists keep items in order.



```
students = ["Asha", "Ravi", "Meera"]
```

students[0] → "Asha"

students[0] →

Editable memory!

0

1

2

Asha

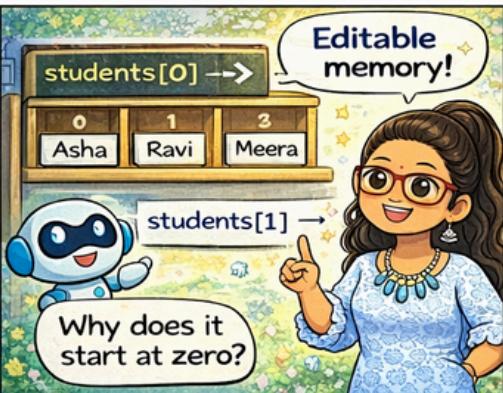
Ravi

Meera

students[1] →



Why does it start at zero?



Tuples can't be changed.



```
colors = ("red", 'green', 'blue')
```

student = { "name": "Sriju",
age: 25, course: "AI" }

Dictionaries use keys to find values.



What stroke go inside the brackets?

```
marks = {"math": 90,  
"science": 85}
```

student = { "name": "Sriju",



Next... can write our own reusable code?



Today you learned:

- ✓ Lists
- ✓ Tuples
- ✓ Dictionaries
- ✓ Chosing the right structure

That's Episode 5!



Sriju Builds Smart Code: Functions & Reusability



Sriju, why are we writing the same code again and again?

```
print("Wash the dishes.");  
print("Dry the dishes.");
```

Copy →
PASTE



Let's teach Python once... and reuse it.



A function takes input, does work, and gives output.



Typing the same code over?



Typing the same code over?



Now it works for everyone!



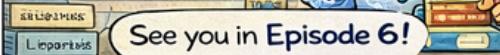
Inputs are called parameters.

```
def greet(name):  
    print("Hello", name):
```



Today you learned:

- ✓ What is a function
- ✓ Parameters
- ✓ Return values
- ✓ Reusability



Power-Up Python: Modules, Libraries & Imports

Sriju, this file never ends!

A module is a Python file with reusable code.



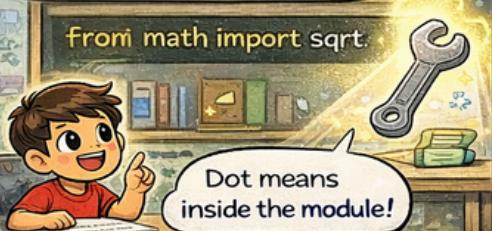
A module is a Python file with reusable code.

Let's import one!



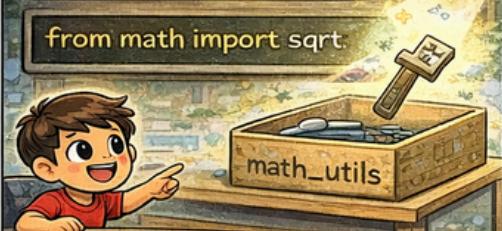
Take only what you need.

From math import sqrt.



from math import sqrt.

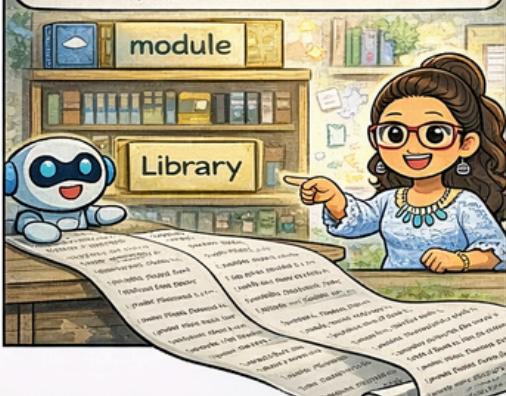
from math import sqrt.



A library is a collection of modules

module

Library



What should go inside the brackets?

from math import _____

print.(sqrt(26))

import m



Unlocking Python Powers: pip, Packages & Virtual Environments



Sriju, why does it work here but not there?

Welcome to dependency problems.

A package is a bundle of Python **modules**.

Meet pip, the library installer!

`pip install numpy`

So pip brings libraries to my computer?

Meet pip, the library installer!

System

My_Project

Install numpy!

Installing everywhere causes trouble.

My_Project

`python -m venv myenv`

python -m venv myenv

Environment activated!

`python -m venv myenv`

`source myenv/bin/activate`

Next... Can we finally start working with real data?

Data Model Deployment

See you in Episode 8!

Working with Real Data: Files, CSVs & Data Handling



Python needs **data**... but where does it come from?

Everywhere.

Files store information outside your program.

So Python can read files line by line!

```
with open ('notes.txt') as f:  
    content = f.read()
```

So Python can read files line by line!

CSV means comma-separated values.

```
import csv
```

name	Score.csv
------	-----------

Neo = 92

Data saved forever!

CSV means comma-separated values.

Name	Score
Neo	92

Each row becomes data!

```
import csv
```

```
import csv
```

Scores.csv

Real data is never perfect.

65

Good data builds good models.

Now... Python analyze this data?

- ✓ That's Episode 9!
- ✓ Files
- ✓ CSVs
- ✓ Clean data

Seeing Patterns: Data Analysis with Pandas



Sriju, I have data...
but what does it mean?



92

67

64

95

89

76

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

Meet Pandas. Pandas
help analyze data.

I turn rows into answers..!

import pandas as pd

import pandas as pd

Another library
with superpowers!



A DataFrame is structured data.

DataFrame

Name	Score	Age
Neo	95	10
Sriju	89	9
Raj	75	12



df.head()

df = pd.read_csv("scores.csv")



Analysis starts with questions.

df.describe()

	Neo	Sriju	Agu
1	10	12	13
2	95	89	75
3	80	75	65
4	55	50	45

df. [Score]. mean()

	Score
1	95
2	89
3	75
4	65

df. [Score]. mean()

Now I see what I'm working with.

df.head()

Name	Score	Age
Neo	95	10
Sriju	89	12
Raj	75	12



Analysis starts with questions.

df.describe()

	Neo	Sriju	Agu
1	10	12	13
2	95	89	75
3	80	75	65
4	55	50	45

df. [Score]. mean()

	Score
1	95
2	89
3	75
4	65

df. [Score]. mean()

ML learns from patterns like these.

Data → Analysis

That's Episode 10!

- ✓ Pandas
- ✓ DataFrames
- ✓ Analysis
- ✓ Insights

Telling Stories with Data: Visualization in Python



Argh! Sriju, I can't see anything in these numbers!



Visuals reveal patterns instantly.

Name	Score	Age
Neo	95	10
Sriju	89	
Raj	75	



Meet Matplotlib. It draws data.

```
import matplotlib.pyplot as plt  
plt.plot(1, 2, 3).(2, 4, 6)  
plt.show()
```

That was easy!



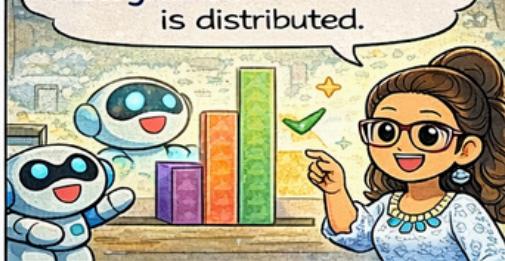
Your first line chart.



Bars compare values.



Histograms show how data is distributed.

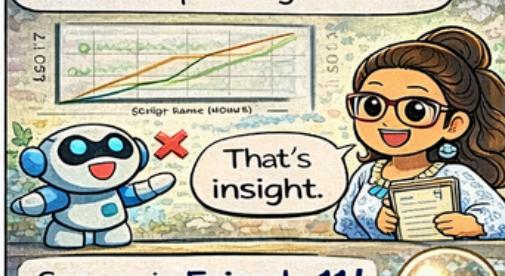


Charts are everywhere!



Charts are everywhere!

Visuals help debug models.



That's insight.

See you in Episode 11!

- ✓ Charts
- ✓ Matplotlib
- ✓ Insights



Remembering Code: File Handling in Python



All done! I'll check it next time.

Goodnight!



Next morning...

What happened?
My file is all
messed up!

```
my_file.txt = x
try:
    with open("my_file.txt") as file:
        print(file.read())
finally:
    file.close()
```

The smart way is
★ "try...finally"



So the file closes
automatically.



That's so much safer!

```
my_file.txt = x
try:
    with open("my_file.txt") as file:
        print(file.read())
finally:
```

The smart way is to "try...
finally"



Now Python remembers
what I do!



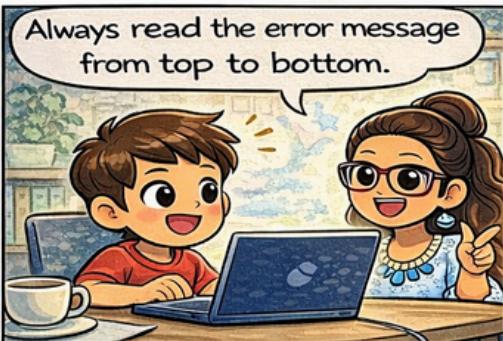
Finding Bugs: Debugging in Python



No error... but the answer is wrong?



Always read the error message from top to bottom.



Reading Error Messages

Traceback (most recent call last):

File "my_script.py", line 3

TypeError: can only concatenate str (not "int") to str



Debugging with Print

```
total = 2 + 2  
print("num1": 2)  
print("num2": 2)  
print("total", total)
```



Fixing the Bug:

```
total = 2 + 2  
print("num1": 2)  
print("num2": 2)  
print("total", total)
```

num1: 2

num2: 2

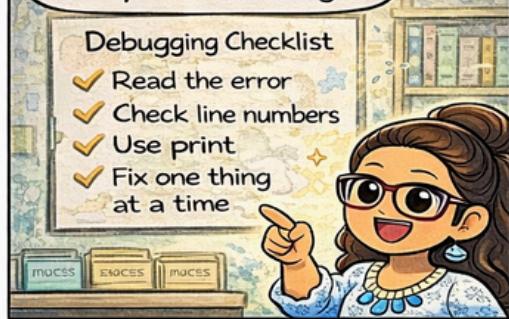
total: 4



Debugging is a skill – and you're learning it.

Debugging Checklist

- ✓ Read the error
- ✓ Check line numbers
- ✓ Use print
- ✓ Fix one thing at a time



Errors don't scare me anymore.

See you in Episode 13!



Boosting Python: Libraries & Modules

Episode 13

Using Libraries

Short names make code cleaner!



Meet the math module

math.sqrt(16) 4

math.sqrt()

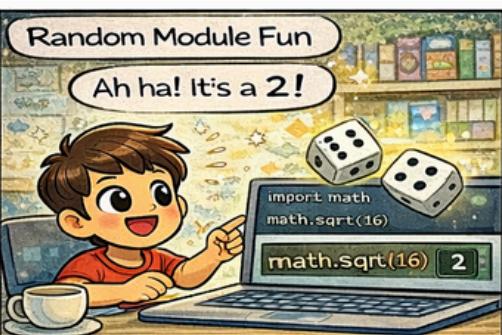
math.pi

Some modules add randomness and fun!



Random Module Fun

Ah ha! It's a 2!



Standard vs External Libraries

Built-in

math

random

External

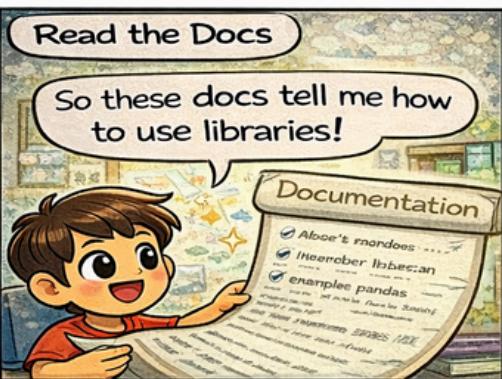
pandas

numpy

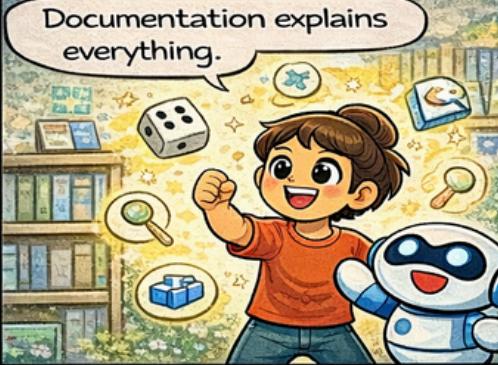


Read the Docs

So these docs tell me how to use libraries!



Documentation explains everything.



See you in Episode 14!

Organizing Data: Lists, Tuples & Sets

Episode 14

Lists – Editable

I can update values easily!



Tuples cannot be changed.



Lists – Editable

Tuples – Fixed



Comparing All Three

List (editable)

Student Names

[Sara, Omar, Lucy]

Tuple (fixed)

Classroom configuration

(4 desks, 2 rows)

Set (unique)

Unique student IDs

(2875, 6342, 5190)



Choose the right structure for the job.



See you in Episode 15!

The Grand Finale: Student Score Analyzer

Episode 15

Let's start by loading our data.

with open('scores.csv') as f:



Lists, tuples, sets... used perfectly!

Lists, tuples, sets.... used perfectly!

Sara	Omar	Raj	Unique IDs.
Sara	88	++ -	3375
Lucy	95	- - -	6642, 5190

A small blue robot character is standing next to a basket filled with books and a green apple.

A loop helps us go through every student.

```
for name, score in data:  
    total += score
```

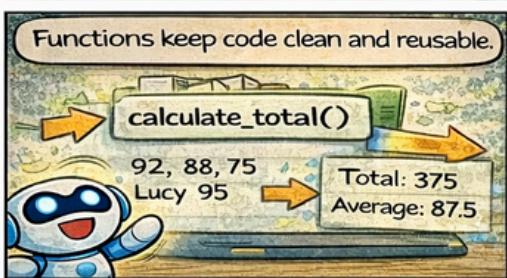


Functions keep code clean and reusable.

calculate_total()

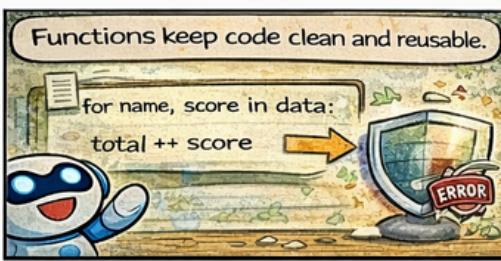
92, 88, 75
Lucy 95

Total: 375
Average: 87.5

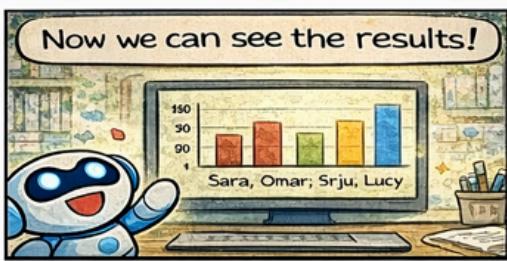
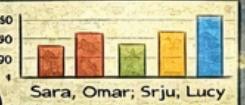


Functions keep code clean and reusable.

```
for name, score in data:  
    total += score
```



Now we can see the results!



It works!

report.txt

Total Score: 375

Average Score: 87.5

SAVED!



Python Beginner Series Complete!



See you in the next series!