

ChatGPT

yesterday python and pip are lost from my system
what are them called packages,dependencies or what?

Good question — this is about **terminology**, and the distinction matters.

What are Python and pip *called*?

They are **not** “packages” or “dependencies” in the usual sense.

1. Python

- **Python is a programming language and a runtime/interpreter**
- More precisely:
 - A **language** (Python)
 - An **interpreter/runtime** (e.g., CPython)
- It is considered **system software / a runtime environment**

So you would call it:

- *“Python interpreter”*
- *“Python runtime”*
- *“Python installation”*

2. pip

- **pip is a package manager**
- It is a **tool** used to install Python packages (dependencies)

You would call it:

- *“pip (Python package manager)”*
- *“dependency manager for Python”*

Why they are not called packages or dependencies

Term	Why it does / doesn't fit
Package	✗ Python installs packages; it is not one

Term	Why it does / doesn't fit
Dependency	✗ Your app depends on Python, but Python itself is not a dependency inside Python
Library	✗ Python includes libraries, but is not a library
Runtime / Interpreter	✓ Correct for Python
Package manager	✓ Correct for pip

Correct way to describe what happened

Instead of saying:

“Python and pip are lost”

More accurate phrasing:

- “The **Python runtime/interpreter** was removed from my system”
- “The **pip package manager** is missing”
- “My **Python installation is broken or uninstalled**”

Common reasons this happens

(If you want to debug later)

- OS update removed it
- PATH environment variable was reset
- Python was uninstalled accidentally
- Multiple Python versions conflicted

If you want, tell me:

- Your **OS** (Windows / macOS / Linux)
- What command fails (`python` , `python3` , `pip` , etc.)

I can help you **identify exactly what broke and how to fix it.**

what are packages and dependencies what are many more terms describe in easy words what are another terms will come in I want to learn basic core concepts all\

Great goal 👍

I'll explain this **from the ground up**, in **easy words**, and give you a **mental map** of the core concepts you'll keep seeing in Python (and programming in general).

The Big Picture (one sentence)

A program runs on a **runtime**, uses **libraries**, which are made of **packages** and **modules**, installed by a **package manager**, and those packages may have **dependencies**.

Now let's unpack every word slowly.

1 Program / Application

What it is:

A program is just **instructions for the computer**.

Example:

```
python  
  
print("Hello")
```

You write programs → the computer runs them.

2 Programming Language

What it is:

A language is the **rules and words** you use to write instructions.

Examples:

- Python
- JavaScript
- C++

Python is **easy for humans**, strict for computers.

3 Interpreter / Runtime

What it is:

Something that **reads your code and executes it**.

- Python code cannot run by itself
- It needs the **Python interpreter**

Think:

Python code = recipe
Interpreter = chef

📌 Also called:

- Runtime
 - Execution environment
 - Python installation
-

4 Library

What it is:

A **collection of ready-made code** you can reuse.

Instead of writing everything yourself, you reuse libraries.

Example:

- Math functions
- File handling
- Networking

📌 In Python:

```
python
```

```
import math
```

5 Module


What it is:

A **single file** containing Python code.

Example:

```
lua
```

```
math.py ← module
```

 Rule of thumb:

One `.py` file = one module

6 Package


What it is:

A **folder of modules**.

Example:

```
markdown
```

```
requests/  
  __init__.py  
  api.py  
  models.py
```

 Rule of thumb:

Many modules grouped together = package


7 Dependency

What it is:

Something **your program needs** to work.

Example:

- Your program uses `requests`
- `requests` is a **dependency**

 Plain English:

“My program depends on this”

Dependencies can be:

- Libraries
- Packages
- Other software

8 Package Manager

What it is:

A tool that **downloads, installs, upgrades, and removes packages**.

In Python:

- `pip`

Example:

```
bash
```

```
pip install requests
```

Think:

Package manager = app store for code

9 Standard Library

What it is:

Libraries that come **with Python already installed**.

Examples:

- `math`
- `os`
- `sys`
- `datetime`

📌 No need to install them.

10 Third-Party Package

What it is:

Packages written by **other people**, not included with Python.

Examples:

- `requests`
- `numpy`
- `pandas`

Installed using pip.

1 1 Environment

What it is:

The **place where Python runs**.

Examples:

- Your system Python
- A virtual environment
- A container

Think:

Environment = workspace for Python


1 2 Virtual Environment (very important)

What it is:

An **isolated Python environment**.

Why?

- Different projects need different package versions
- Prevents conflicts

 Think:

Separate boxes for separate projects

1 3 Version

What it is:

The **specific release** of something.

Examples:

- Python 3.10 vs 3.12
- requests 2.31.0

Versions matter a lot.

1 4 Path / PATH

What it is:

A system setting that tells your OS **where to find programs**.

If Python “disappears”, often:

- It’s installed
 - But PATH is broken
-

1 5 Build / Install

Build: prepare software

Install: put it on your system

In Python:

- Most packages don't need building
 - pip installs them directly
-

1 6 Script

What it is:

A small program meant to **run once or automate tasks**.

Example:

```
bash
```

```
python backup.py
```

1 7 Framework

What it is:

A **big collection of tools** that helps build apps faster.

Examples:

- Django (web)
- Flask
- FastAPI

 Think:

Library = tool

Framework = toolbox + rules

1 8 API

What it is:

A **way to interact** with code or services.

Example:

```
python
```

```
requests.get(url)
```

📌 API = how you talk to something

19 Configuration

What it is:

Settings that control behavior without changing code.

Examples:

- `.env` files
 - config files
-

20 Dependency Tree

What it is:

Dependencies that depend on **other dependencies**.

Example:

```
markdown
```

```
your app
```

```
└─ requests
```

```
    └─ urllib3
```

Quick Mental Map (memorize this)

```
SCSS
```

```
Your Code
```

```
↓
```

```
Python Interpreter (runtime)
```

```
↓
```

```
Libraries
```

```
└─ Modules
```

└─ Packages



Dependencies



Installed by pip



Inside an Environment

If you want next

I can:

- Explain **virtual environments step-by-step**
- Show **what breaks when Python disappears**
- Teach **how to read error messages**
- Give you a **beginner roadmap**

Just tell me 👍