# **🧠 Python Guide: Closures**

## **✅ What is a Closure?**

A **closure** is a function that **remembers** the variables from its **enclosing scope** — even after the outer function has finished executing.

In simple words:

The **inner function** "remembers" the values passed to the \*\*outer function".

## **🔍 Closure Example:**

def outer(msg):

def inner():

return f"Message is: {msg}"

return inner

say\_hi = outer("Vanakkam da mapla")

print(say\_hi()) # Output: Message is: Vanakkam da mapla

### **🔎 What's Happening Here?**

| **Line** | **Meaning** |
| --- | --- |
| outer("Vanakkam da mapla") | msg = "Vanakkam da mapla" is set |
| inner() | Uses msg from outer() |
| return inner | Returns the inner() function itself (not the result) |
| say\_hi() | Executes the returned function, which still "remembers" msg |

🎯 Even though outer() has finished, the inner() function **still has access** to msg.

✅ That’s a **Closure**.

## **🎯 Real-Life Analogy:**

Imagine a **custom voice recorder** app where:

recorder = create\_recorder("Gowtham says:")

recorder("I am here") # Output: Gowtham says: I am here

recorder("Let’s go") # Output: Gowtham says: Let’s go

That **"Gowtham says:"** is remembered forever — just like how msg is remembered in a closure.

## **✅ Why Use Closures?**

| **Benefit** | **Explanation** |
| --- | --- |
| **Remembers outer variables** | Even when outer function is gone |
| **Used in HOF Type 2** | Return function with context saved |
| **Build reusable logic** | Like dynamic SQL builders, custom filters, etc. |
| **Foundation for decorators** | All Python decorators rely on closures |

## **🛠 Closure Use Cases:**

1. **SQL Query Builders**
2. **Custom Greeting Generators**
3. **Event Handlers**
4. **Decorators (@login\_required)**
5. **Validation/Plugin Systems**

## **🧪 Debug Tip: See Closure with .\_\_closure\_\_**

say\_hi = outer("Vanakkam da mapla")

print(say\_hi.\_\_closure\_\_[0].cell\_contents) # Output: Vanakkam da mapla

It shows what the closure **remembers**.

## **✅ Closure Pattern Template**

def outer(value):

def inner():

# uses `value` here

return ...

return inner

## **💡 TL;DR:**

A **closure** is when a function **returns another function**,  
 and that inner function **remembers the outer function’s variables**, even after the outer is done.

### **About the Author**

**Gowtham SB** is a **Data Engineering expert, educator,** **and content creator** with a passion for **big data technologies, as well as cloud and Gen AI** . With years of experience in the field, he has worked extensively with **cloud platforms, distributed systems, and data pipelines**, helping professionals and aspiring engineers master the art of data engineering.

Beyond his technical expertise, Gowtham is a **renowned mentor and speaker**, sharing his insights through engaging content on **YouTube and LinkedIn**. He has built one of the **largest Tamil Data Engineering communities**, guiding thousands of learners to excel in their careers.

Through his deep industry knowledge and hands-on approach, Gowtham continues to **bridge the gap between learning and real-world implementation**, empowering individuals to build **scalable, high-performance data solutions**.

𝐒𝐨𝐜𝐢𝐚𝐥𝐬

🎥𝐘𝐨𝐮𝐓𝐮𝐛𝐞 - https://www.youtube.com/@dataengineeringvideos

📸𝐈𝐧𝐬𝐭𝐚𝐠𝐫𝐚𝐦 - <https://instagram.com/dataengineeringtamil>

📸𝐈𝐧𝐬𝐭𝐚𝐠𝐫𝐚𝐦 - [https://instagram.com/](https://instagram.com/dataengineeringtamil)thedatatech.in

🤝𝐂𝐨𝐧𝐧𝐞𝐜𝐭 𝐟𝐨𝐫 𝟏:𝟏 - https://topmate.io/dataengineering/

💼𝐋𝐢𝐧𝐤𝐞𝐝𝐈𝐧 - https://www.linkedin.com/in/sbgowtham/

🌐𝐖𝐞𝐛𝐬𝐢𝐭𝐞 - https://codewithgowtham.blogspot.com

💻𝐆𝐢𝐭𝐇𝐮𝐛 - http://github.com/Gowthamdataengineer

💬𝐖𝐡𝐚𝐭𝐬 𝐀𝐩𝐩 - https://lnkd.in/g5JrHw8q

📧𝐄𝐦𝐚𝐢𝐥 - atozknowledge.com@gmail.com

📱𝐀𝐥𝐥 𝐌𝐲 𝐒𝐨𝐜𝐢𝐚𝐥𝐬 - <https://lnkd.in/gf8k3aCH>