## **🔥 Python Logging – Complete Guide for Beginners**

### **📌 1. What is Logging?**

Logging is a way to **track the flow of a program** and record events like info, warnings, or errors.  
 Unlike print(), logging can write messages to files, differentiate between message levels, and be turned off without modifying the code.

### **❓ Why Not Just Use print()?**

| **Feature** | **print()** | **logging** |
| --- | --- | --- |
| Shows info | ✅ Yes | ✅ Yes |
| Shows warnings/errors | ❌ No | ✅ Yes (warning, error, etc.) |
| Logs to file | ❌ No | ✅ Yes |
| Timestamps | ❌ Manual | ✅ Yes |
| Production ready | ❌ No | ✅ Yes |

### **✅ 2. print() Version – Basic Debugging**

def divide(a, b):

print(f"Dividing {a} by {b}")

try:

result = a / b

print(f"Result: {result}")

return result

except ZeroDivisionError:

print("Error: Division by zero")

return None

# Test

divide(10, 2)

divide(10, 0)

❗ Simple, but not scalable or safe for production.

### **✅ 3. logging Version – Production Grade**

import logging

# Basic logging config

logging.basicConfig(

level=logging.DEBUG, # capture all levels

format='%(asctime)s - %(levelname)s - %(message)s',

filename='app.log', # save to file

filemode='w' # overwrite file each time

)

def divide(a, b):

logging.info(f"Dividing {a} by {b}")

try:

result = a / b

logging.debug(f"Result: {result}")

return result

except ZeroDivisionError:

logging.error("Tried to divide by zero!")

return None

# Test

divide(10, 2)

divide(10, 0)

### **🔨 4. Mini Project: Invoice Calculator with Logging**

#### **📄 Project: Calculate total invoice value and log steps**

import logging

# Configure logger

logging.basicConfig(

level=logging.INFO,

format='%(asctime)s - %(levelname)s - %(message)s',

handlers=[

logging.FileHandler("invoice.log"),

logging.StreamHandler() # logs to console also

]

)

def calculate\_invoice(items):

logging.info("Invoice calculation started")

total = 0

for name, price in items:

if price < 0:

logging.warning(f"Negative price found for item {name}")

logging.debug(f"Adding {name}: ₹{price}")

total += max(price, 0) # ignore negative price

logging.info(f"Invoice calculation completed. Total: ₹{total}")

return total

# Sample items (item name, price)

items = [

("Mouse", 500),

("Keyboard", 1000),

("Monitor", -1500), # Wrong entry

("Laptop", 55000)

]

final\_amount = calculate\_invoice(items)

print(f"Final Invoice Amount: ₹{final\_amount}")

#### **✅ Output in Terminal + invoice.log file**

2025-05-17 02:20:01,123 - INFO - Invoice calculation started

2025-05-17 02:20:01,124 - WARNING - Negative price found for item Monitor

2025-05-17 02:20:01,125 - INFO - Invoice calculation completed. Total: ₹56500

## **🎯 5. Interview: How to Explain Logging (with Story)**

### **💼 Interview Story**

"In my last project, we were building an invoice processing module. Initially, I used print() to debug issues, like incorrect totals. But as the code grew, debugging became harder — especially in production, where we couldn’t see terminal outputs.

I replaced print() with Python’s logging module. I set up log levels like info, debug, and error, and configured logs to write to both the console and a file.

This helped our team track what went wrong (like incorrect prices or zero divisions), and we could debug issues from log files even after the job finished.

Since then, I always prefer logging over print for any real-world applications."

### **💡 Bonus: When Asked "Why Logging Over Print?"**

Say:

"Print is okay for simple scripts. But logging gives me more control — I can turn it off without code changes, route logs to files, add timestamps, and set severity levels. It’s much safer and more scalable."

## **✅ Summary**

| **Topic** | **Covered** |
| --- | --- |
| print() vs logging | ✅ Yes |
| Code examples | ✅ Yes |
| Mini Project | ✅ Yes |
| Interview explanation | ✅ Yes |

### **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>