

Python语言程序设计

Python Programming

2026/27



Session 01

Tom Lotz (tom.lotz@outlook.com)

Content

Course Introduction and Requirements

01 Introduction to Python

02 Think Before Typing

03 Python and IDLE

04 Your First Program

05 Variables

06 Numbers and Basic Math

07 Practice

Course Introduction and Requirements

Course setup

- Closely following the book “Python Crash Course” by Eric Matthes (3rd Edition)
- 16 weeks
- Weeks 1–8: two 45-minute periods per week
- Weeks 9–16: four 45-minute periods per week
- Practical work in every session

Grading

- 50% course work + 50 % final project
- Course work:
 - Participation
 - Asking / answering questions
 - Paying attention
 - Small assignments over the weeks
- Final project
 - Individual student projects
 - Plan, code, and document a small program

Homework submissions

- Submit homework by email only
- Do not submit homework through WeChat
- Include your student ID in the filename
- Use the requested file type and filename format

studentID_session1.py

Dos and Don'ts

- During the course you should:
 - Pay attention
 - Mute your phone
 - Follow the rules for the respective room / building
 - Ask questions when you don't understand
 - Explain your own work
- You shouldn't:
 - Disturb others
 - Use your phone too much
 - Use AI to complete tasks

Course materials

- Download the “cheat sheets” and slides from here before every class

www.t-l.earth

Tom Lotz

Jinling Institute of Technology (Nanjing, China)
Research on Microplastics, Hydrology, and Machine Learning

Research

Teaching



Index of /teaching/files

Name	Last modified	Size	Description
Parent Directory		-	
python2025_26/	2025-09-25 09:07	-	
python2026_27/	2026-08-04 15:25	-	
python_corner_2026/	2026-05-18 05:55	-	

Apache/2.4.52 (Ubuntu) Server at t-l.earth Port 443

Course WeChat group

- For questions, info and important things



Introduction to Python

History of Python

- Created by Guido van Rossum in late 1980s, released in 1991
- Successor to ABC language → focus on readability
- Major milestones: Python 2 (2000), Python 3 (2008, now standard)
- Today: one of the most widely used programming languages

Core Attributes

- Readable syntax ('executable pseudocode')
- High-level & interpreted (no compilation step)
- Cross-platform (Windows, Mac, Linux)
- Huge standard library ('batteries included')
- Community-driven with vast open-source ecosystem

Applications of Python

- Web development (Django, Flask, FastAPI)
- Data science (NumPy, Pandas, Matplotlib)
- Machine learning & AI (scikit-learn, TensorFlow, PyTorch)
- Automation & scripting (system scripts, web scraping)
- Scientific computing (SciPy, Jupyter, hydrology, bioinformatics)
- Game development, GUI, IoT (PyGame, Tkinter, MicroPython)

Python vs Other Languages

Language	Syntax	Typing	Speed & Performance	Use Case / Environment
Python	Simple, readable	Dynamic	Slower, but flexible	Versatile: data, automation, web
Java	Verbose	Static	Fast (JVM-optimized)	Enterprise apps, Android
C / C++	Complex	Static	Very fast, low-level	System programming, embedded
JavaScript	Compact	Dynamic	Fast in browsers	Web front-end (runs in browser)

Why Python?

- Quick to learn for those with programming basics
- Great for rapid prototyping
- Bridges many domains: web, science, AI, automation
- High demand in industry & research
- Foundation for advanced study and projects

Before You Type

1. What must happen?
2. What should appear?
3. What information is needed?
4. What should Python do first?

A Simple Working Rhythm

- Understand the task
- Predict the result
- Type the instructions
- Run the program
- Compare the result
- Explain what happened

Example

- You have the numbers 3, 1, 8, 2, 7
- Can you define rules to sort them in increasing order?

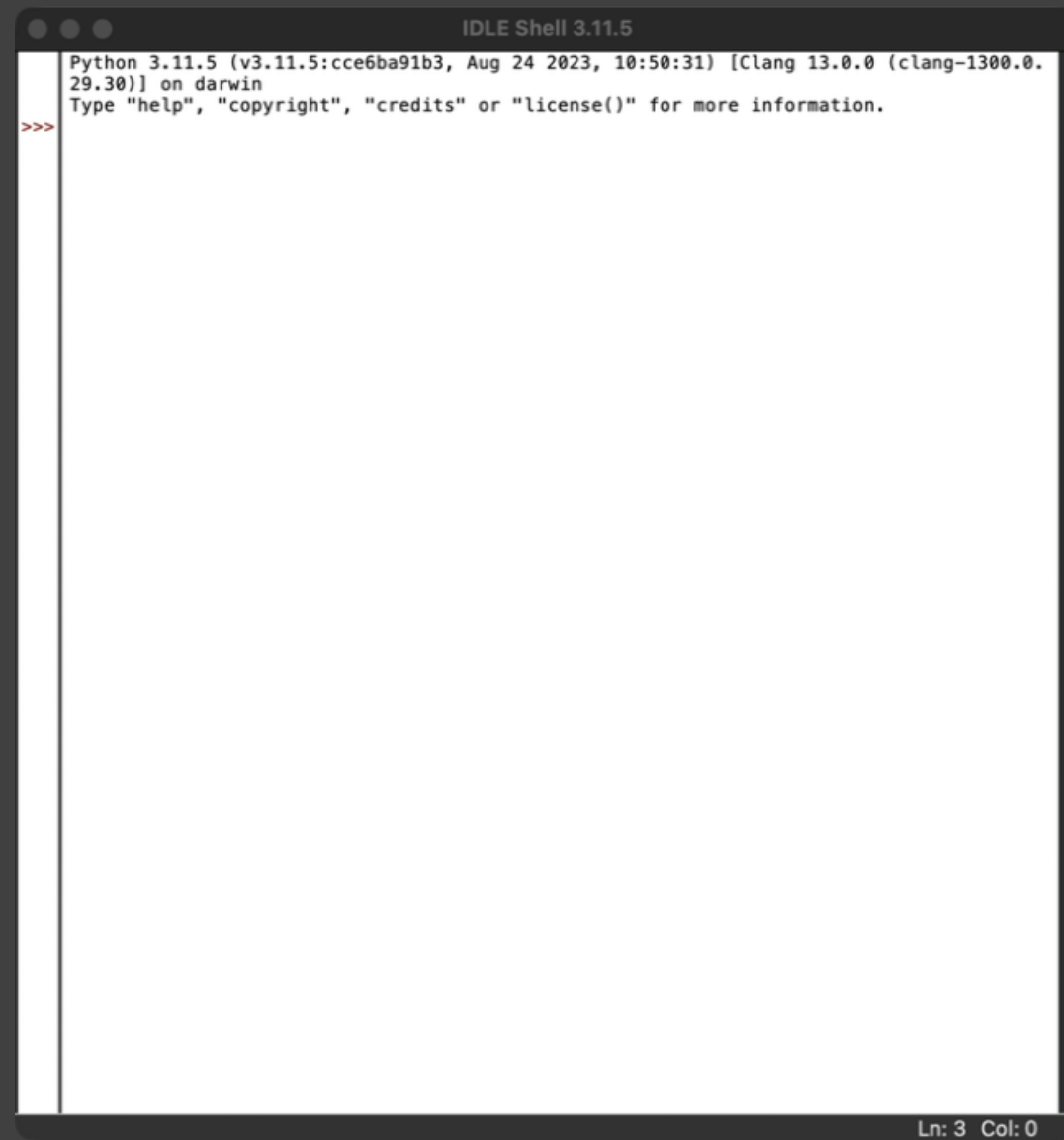
Python & IDLE

Python Code Can Be Written in Many Editors

- IDLE (built into Python)
- Any text editor
- VS Code
- Online editors and notebooks

- For this course, we use IDLE.

IDLE: Python Shell



```
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Ln: 3 Col: 0

IDLE: Editor Window

File → New File



Create and Save a Python File



```

Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1300.0.
29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
1
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
1
>>>

```

Ln: 9 Col: 0

What Is a Python Script?

- A .py file is a Python script
- Python is an interpreted language
- Code is executed line by line
- You write code in the editor
- The Python interpreter runs your script line by line



What Happens When You Run Code?

- Python reads the script from top to bottom.
- It performs one instruction after another.

If an error stops the program, Python shows where it happened.

```
print('Line 1')  
print('Line 2')
```

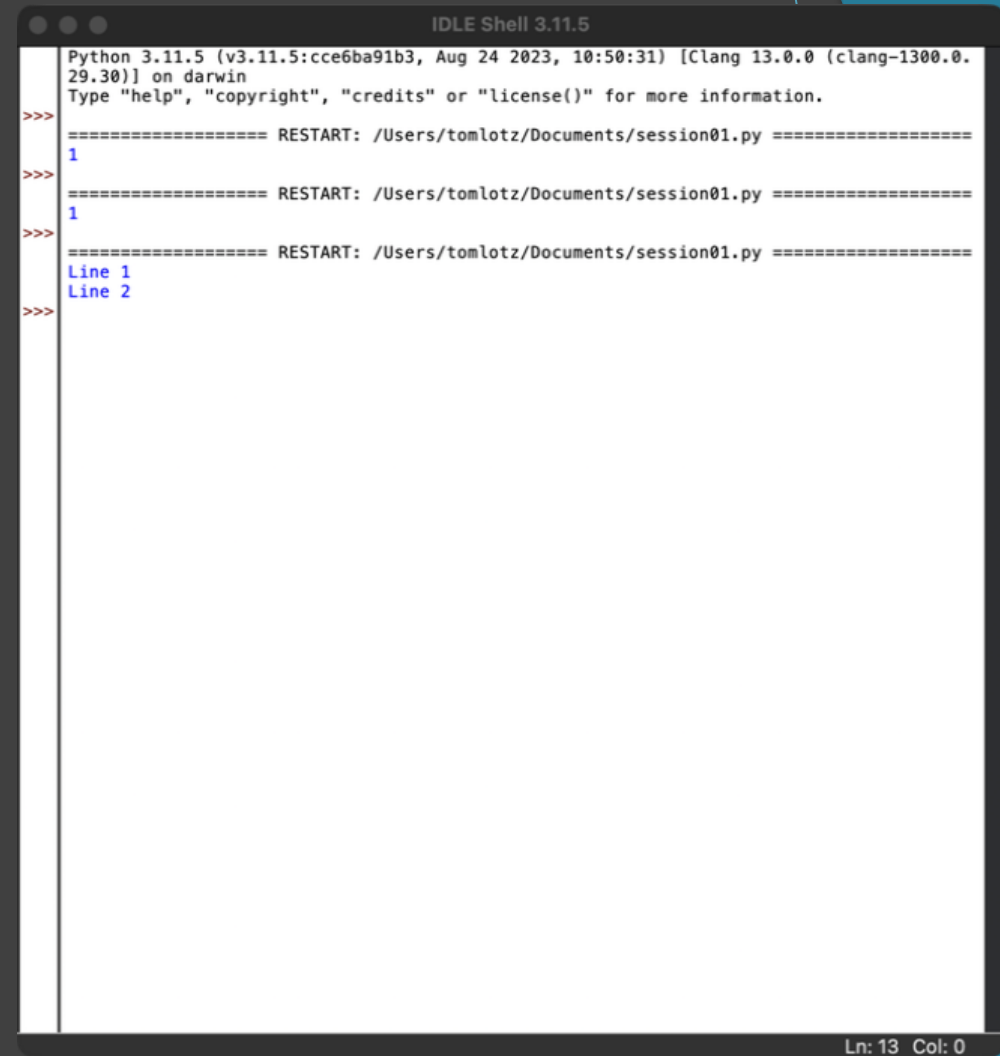
Output Appears in the Python Shell



A screenshot of a Python script file named 'session01.py' in the IDLE editor. The file path is '/Users/tomlotz/Documents/session01.py (3.11.5)'. The script contains two lines of code: `print('Line 1')` and `print('Line 2')`. The status bar at the bottom right shows 'Ln: 2 Col: 13'.

```
session01.py - /Users/tomlotz/Documents/session01.py (3.11.5)
print('Line 1')
print('Line 2')
```

Ln: 2 Col: 13



A screenshot of the IDLE Shell window for Python 3.11.5. The shell shows the initial prompt and the execution of the script. It displays the output of the first print statement, 'Line 1', and the second print statement, 'Line 2'. The status bar at the bottom right shows 'Ln: 13 Col: 0'.

```
IDLE Shell 3.11.5
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
1
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
1
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
Line 1
Line 2
>>>
```

Ln: 13 Col: 0

Your First Program

Say Hello

- The program should display one line:

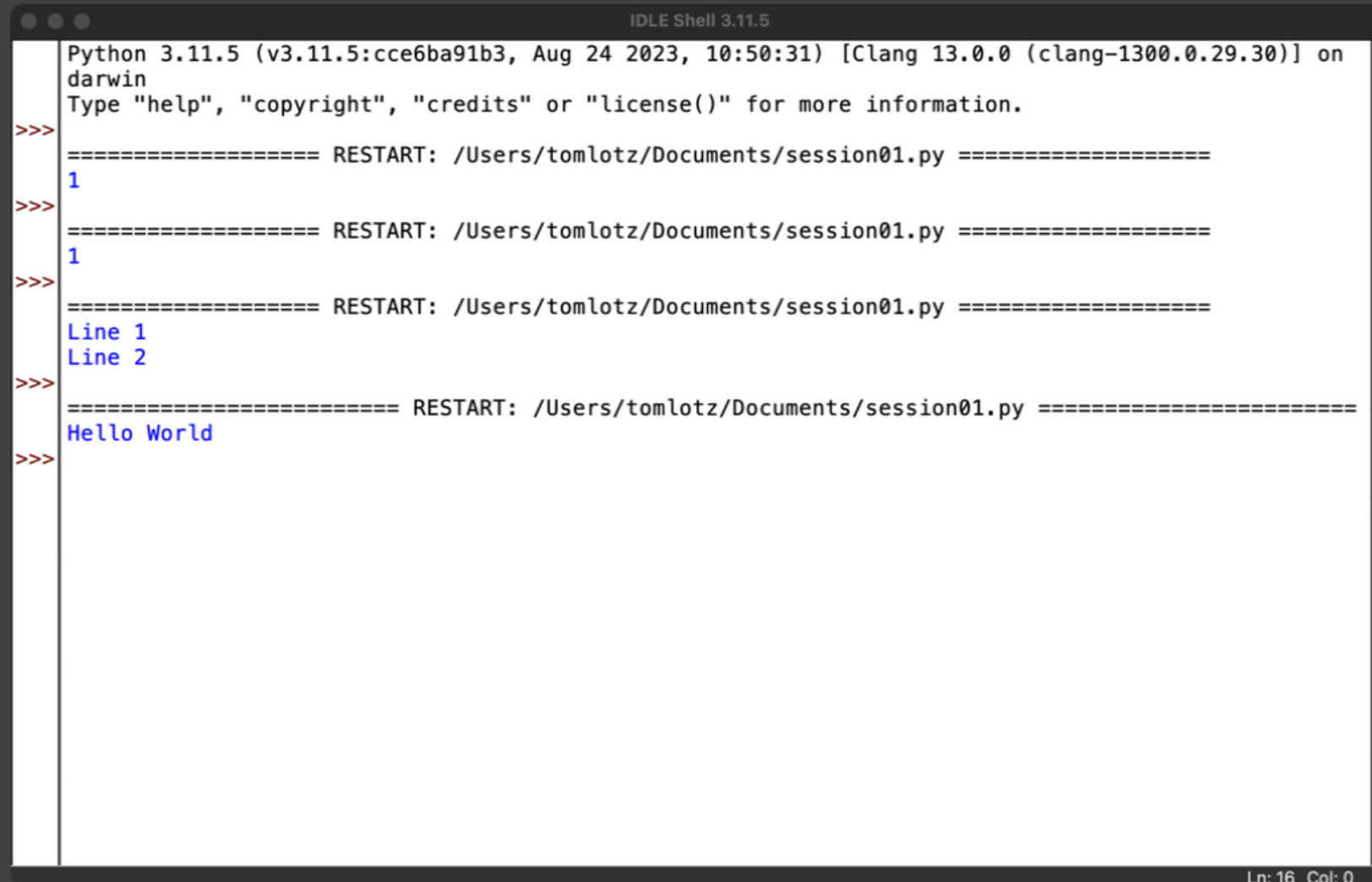
Hello World

Your First Program

A screenshot of a code editor window. The title bar at the top reads "session01.py - /Users/tomlotz/Documents/session01.py (3.11.5)". The editor area is white and contains a single line of Python code: `print('Hello World')`. The word `print` is in purple, the opening quote is in green, and the rest of the code is in black. The window has a dark grey title bar with three small circles on the left.

```
session01.py - /Users/tomlotz/Documents/session01.py (3.11.5)  
print('Hello World')
```

Check the Output



```
IDLE Shell 3.11.5
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
1
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
1
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
Line 1
Line 2
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
Hello World
>>>
```

Variables

What Changes? What Stays?

- A program displays a message.
- The instruction stays the same.
- The message can change.
- What information should the program remember?

Using a Variable

- A variable stores a value under a name.
- Python can use that value later.
- `message` is the variable name.

```
5  
6 message = "Hello, Python!"  
7 print(message)  
8
```

Using a Variable

- You can assign a new value to a variable at any time
- Python will always use the latest value

```
5  
6     message = "Hello, Python!"  
7     print(message)  
8     message = "Hello, Crash Course!"  
9     print(message)  
10  
11
```

Variable Naming Rules

- Use letters, numbers, and underscores: `greeting_1`
- Must start with a letter or underscore (not a number)
- No spaces allowed
- Cannot use Python reserved words (like `print`, `for`, etc.)
- Use lowercase and descriptive names: `user_name`, not `u`

Try: Store a Number

- Store the number 20 in a variable called age.
- Then display the value.

Check: Store a Number

A screenshot of a Python script editor window. The window has a dark title bar with three small circles on the left and the text '*session01.py - /Users/tomlotz/Documents/session01.py (3.11.5)*'. The main area is white and contains two lines of code: 'age = 20' and 'print(age)'. The 'print' function is highlighted in purple.

```
*session01.py - /Users/tomlotz/Documents/session01.py (3.11.5)*  
age = 20  
print(age)
```

Numbers and Basic Math

Numbers Do Not Need Quotation Marks



```
*session01.py - /Users/tomlotz/Documents/session01.py (3.11.5)*  
age = 20  
print(age)
```

Working with Integers

- Use +, -, *, / for basic math
- Python follows order of operations

```
print(2 + 3)
print(3 - 2)
print(2 * 3)
print(3 / 2)

print(2 + 3 * 4) # 14
print((2 + 3) * 4) # 20
```

Predict: What Will Python Display?



```
*session01.py - /Users/tomlotz/Documents/session01.py (3.11.5)*  
  
print(3 + 4)
```

Check the Result

```
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1300.0.29.30)] on
darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
1
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
1
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
Line 1
Line 2
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
Hello World
>>>
===== RESTART: /Users/tomlotz/Documents/session01.py =====
7
>>>
```

Ln: 19 Col: 0

Try: Add Two Stored Numbers

- Store 4 in `first_number`.
- Store 3 in `second_number`.
- Create `total` by adding them.
- Display `total`.

Check: Add Two Stored Numbers

```
session01.py - /Users/tomlotz/Documents/session01.py (3.11.5)

first_number = 4
second_number = 3

total = first_number + second_number

print(total)
```

Try Again: Change One Number

- Change only one stored number.
- Predict the new total.
- Run the program.
- Compare the result with your prediction.

Practice

Practice 1: Print and Predict

1. Predict the output of `print(8)`.
2. Type and run it in IDLE.
3. Change 8 to another number.
4. Explain what `print()` does.

Practice 2: Variables

1. Store your age in a variable called age.
2. Predict the output of `print(age)`.
3. Run the program.
4. Change the value and run it again.

Practice 3: Basic Math

1. Store two numbers in variables.
2. Add them and store the result in total.
3. Predict the result.
4. Display total and compare.

Submit the results

- Send your completed practice file to me by email.