Python语言程序设计

Python Programming

2025/26

Session 01

Tom Lotz (tom.lotz@outlook.com)

Content

Course Introduction and Requirements

Course setup

- Closely following the book “Python Crash Course” by Eric Matthes (3rd Edition) – did everyone get the e-book?
- 16 weeks duration
- 32 hours of theory + 16 hours of practical experiences
- Almost all sessions will have practical elements

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

Dos and Don’ts

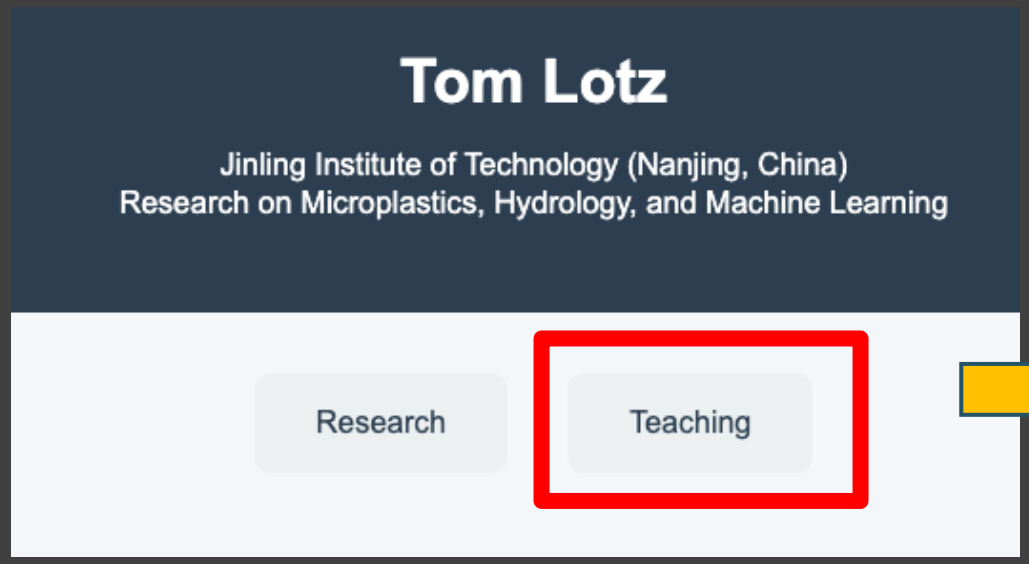
- During the course you should:
  - Pay attention
  - Mute your phone
  - Follow the rules for the respective room / building
  - Ask questions when you don’t understand

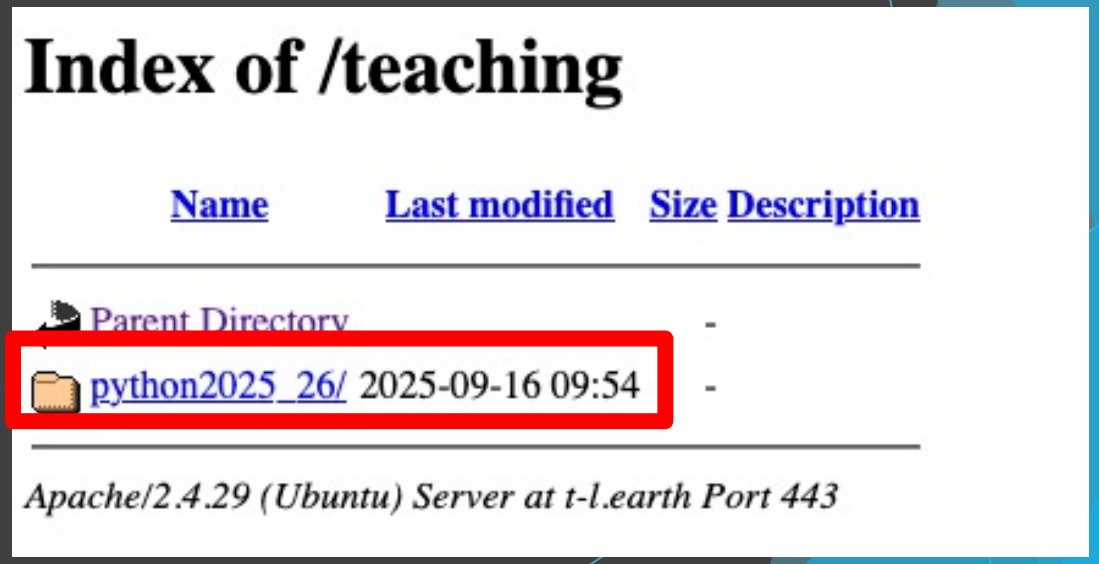
- You shouldn’t:
  - Disturb others
  - Use your phone (too much)

Course materials

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

t-l.earth





Course WeChat group

- For questions, info and important things

Group: Python 2025-26

Valid until 9/16 and will update upon joining group

Introduction to Python

Have you used Python before?

3	0	0	0
Never	Just a little	Often	I am an expert

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

Python Programming Environment

Python code can be written in many editors:

- Any text editor
- IDLE (built-in editor)
- VS Code
- PyCharm
- Online editors (Replit, Jupyter Notebook, etc.)
- Many, many more...

We use PyCharm in this course:

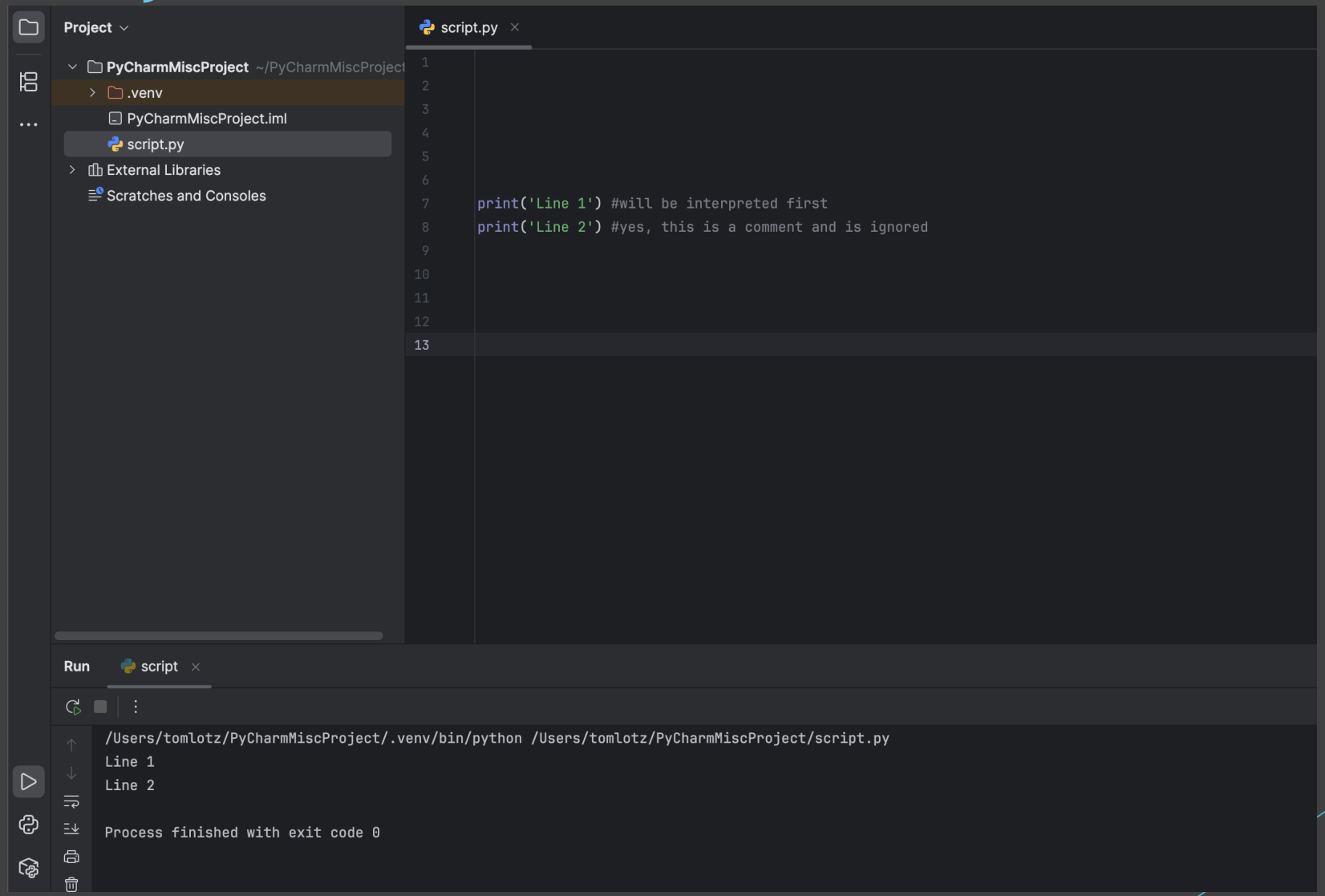
- Already installed on lab computers
- Good for both beginners and larger projects
- Integrated terminal & file manager
- Smart autocomplete and error checking

You can download it for free:

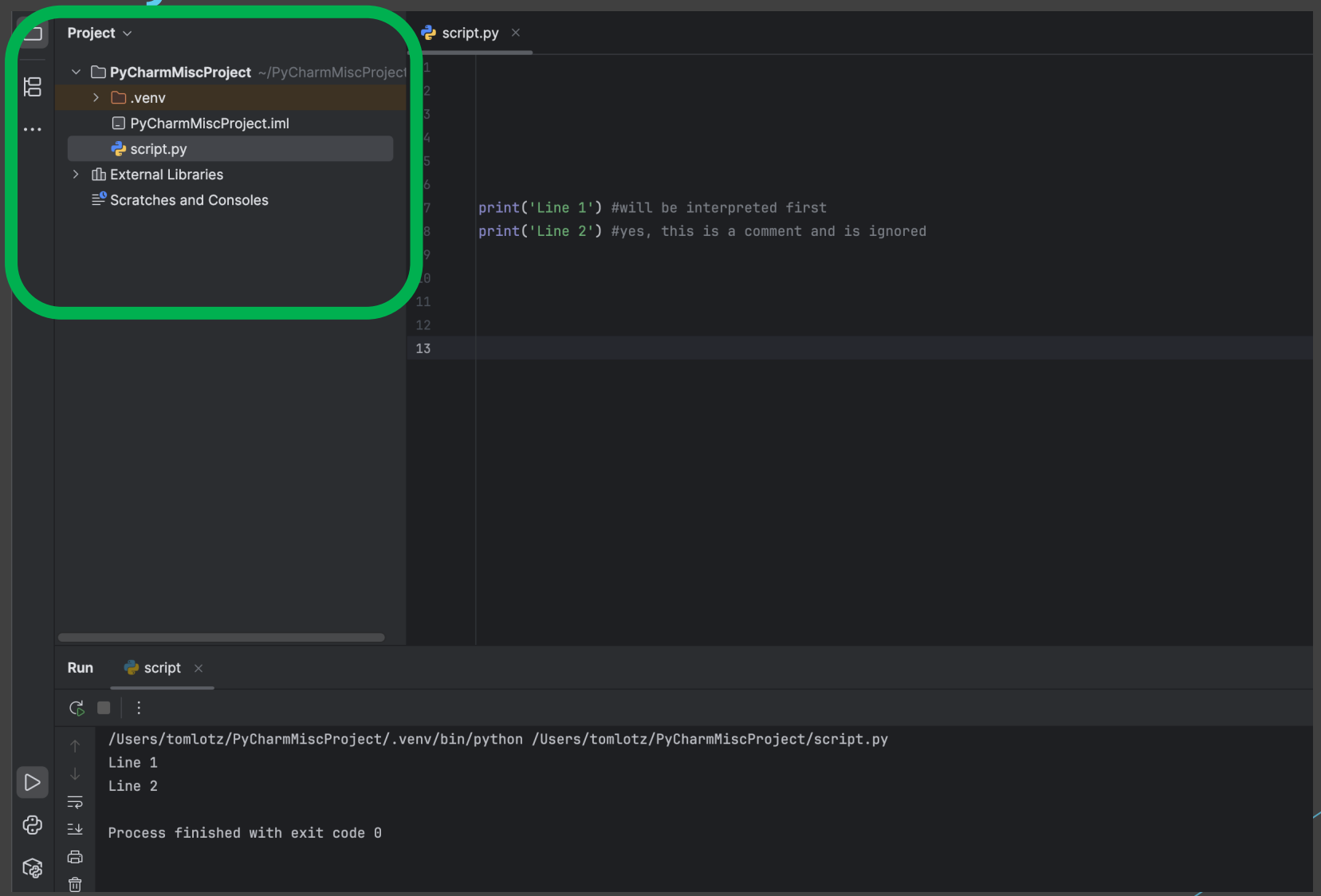
- You can install it on your own computer
- Choose the Community Edition (free)

- Go to: https://www.jetbrains.com/pycharm/download

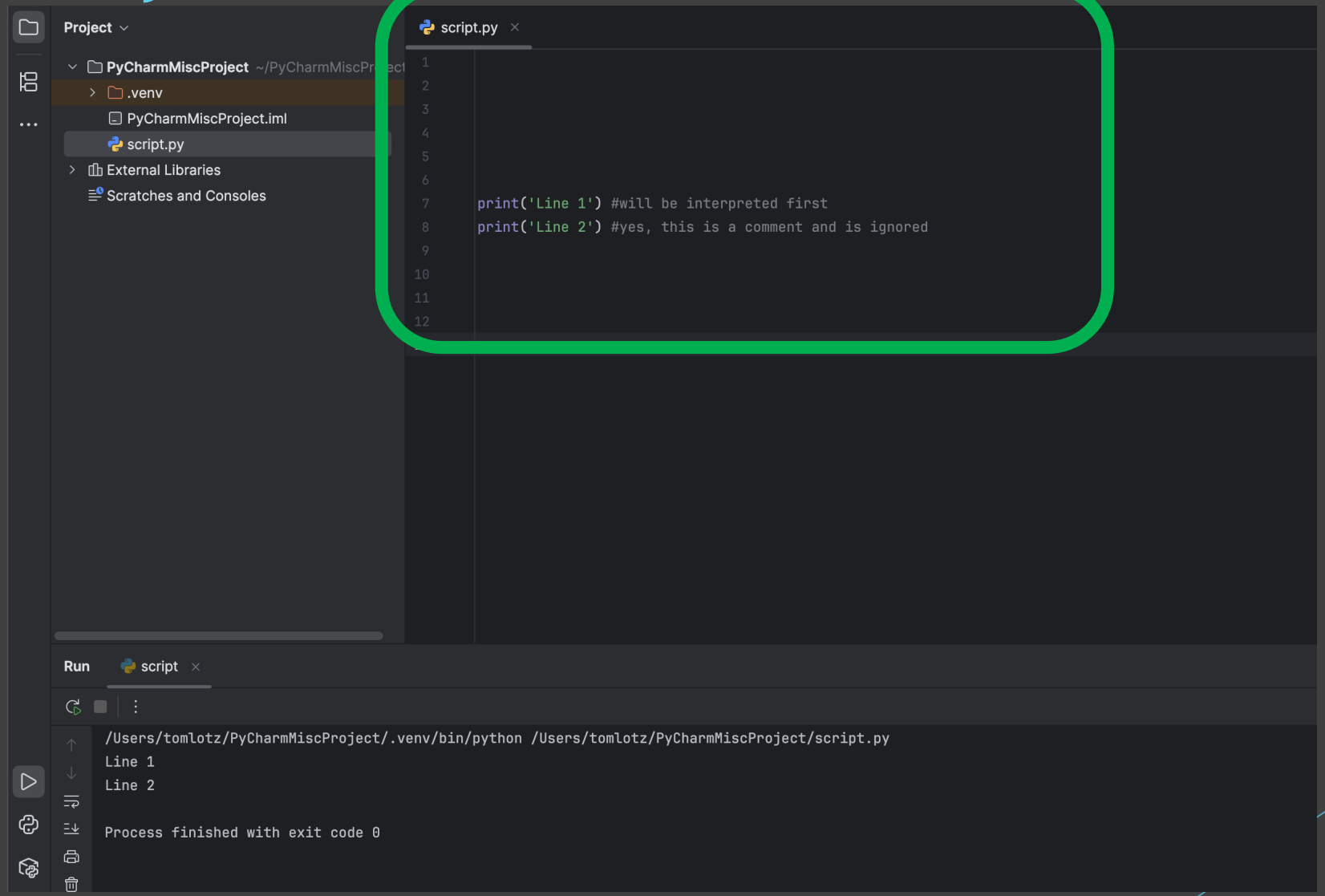
Basic PyCharm Interface



Basic PyCharm Interface



Basic PyCharm Interface



Basic PyCharm Interface

Project

- PyCharmMiscProject ~/PyCharmMiscProject
  - .venv
  - PyCharmMiscProject.iml
  - script.py
- External Libraries
- Scratches and Consoles

script.py

```python
print('Line 1') #will be interpreted first
print('Line 2') #yes, this is a comment and is ignored
```

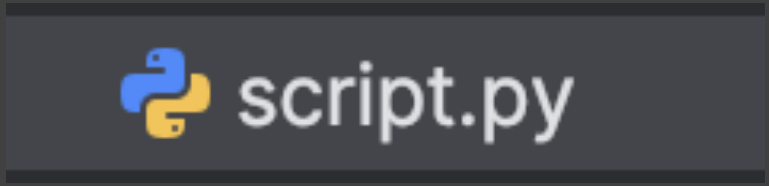
Run script

```
/Users/tomlotz/PyCharmMiscProject/.venv/bin/python /Users/tomlotz/PyCharmMiscProject/script.py
Line 1
Line 2

Process finished with exit code 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?

- The interpreter reads your script
- Finds keywords, function names, variables, etc.
- Executes code from top to bottom

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

What Happens When You Run Code?

- Output appears in the terminal or console

```
/Users/tomlotz/PyCharmMiscProject/.venv/bin/python /Users/tomlotz/PyCharmMiscProject/script.py
Line 1
Line 2

Process finished with exit code 0
```

Hello World

Creating a Project in PyCharm

- Open PyCharm
- Select "New Project"
- Choose a location (e.g. Desktop/python_work)
- Select interpreter: existing Python installation

Creating Your First File

- Right-click project folder > New > Python File
- Name it hello_world.py
- File must end in .py

Writing Your First Program

```
10
11
12
13  print("Hello, world!")
14
15
16
```

- This line prints a message to the screen
- print() is a built-in Python function
- The text is a string inside quotes

Running Your First Program

- Click "Run" or right-click file > "Run 'hello_world'"
- Or use shortcut: **Shift + F10**
- Output will appear in the lower terminal window


```
/Users/tomlotz/PyCharmMiscProject/.venv/bin/python /Users/tomlotz/PyCharmMiscProject/script.py
Hello world!

Process finished with exit code 0
```

What Does Exit Code 0 Mean?

- At the bottom you might see:

```
Process finished with exit code 0
```

- That means:
  - Python ran your code without any errors ✅
  - Your script may or may not have printed anything

- It’s not an error! It just says "the program ended normally"

Exit Code 1

```
Process finished with exit code 1
```

- Python shows a traceback when something goes wrong
- It includes:
  - File name and line number
  - Type of error (e.g. SyntaxError, NameError, IndentationError)

```
/Users/tomlotz/PyCharmMiscProject/.venv/bin/python /Users/tomlotz/PyCharmMiscProject/script.py
Traceback (most recent call last):
  File "/Users/tomlotz/PyCharmMiscProject/script.py", line 9, in <module>
    Print('Hello world!') #yes, this is a comment and is ignored
    ^^^^^
NameError: name 'Print' is not defined. Did you mean: 'print'?

Process finished with exit code 1
```

To join, go to: ahaslides.com/ZOV79

AhaSlides

lello World program run successfully?

Join at:
ahaslides.com/
ZOV79

0 0/50

Some quick info about Python standards

```python


print("Hello World!")
Print('Hello world!')

# This is a single line comment

'''
This is a multi-line string, often used as a comment.
Everything here will be ignored.
'''
```

First look at variables

Using a Variable

- A variable stores a value (like a label)
- Variable declaration is very easy in Python – just assign a value
- You can reuse and change it
- 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 anytime
- Python will always use the latest value

```python

message = "Hello, Python!"
print(message)
message = "Hello, Crash Course!"
print(message)


```

Careful!

- This flexibility of Python can cause problems

```python

message = "Hello, Python!"
print(message)
message = "Hello, Crash Course!"
print(message)
message = 29.5
print(message)

```

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 keywords (like print, for, etc.)
- Use lowercase and descriptive names: user_name, not u

To join, go to: ahaslides.com/ZOV79

AhaSlides

hese is a valid variable name?

Join at:
ahaslides.com/
ZOV79

	0	0	0
e	1stname	user name	_tempVar

2 0 0/50

Strings

Strings and String Methods

```python
name = "john doe"

name_title = name.title()
print(name_title)

print(name.title())
print(name.upper())
print(name.lower())
```

Strings and String Methods

```
name = "john doe"

name_title = name.title()
print(name_title)

print(name.title())
print(name.upper())
print(name.lower())
```

Strings and String Methods

```python
name = "john doe"

name_title = name.title()
print(name_title)

print(name.title())
print(name.upper())
print(name.lower())
```

```
John Doe
John Doe
JOHN DOE
john doe
```

f-Strings (Formatted Strings)

- f-strings let you insert variables inside strings
- Very useful for dynamic messages

```python
first_name = "john"
last_name = "doe"

full_name = f"{first_name} {last_name}"

print(f"Hello, {full_name.title()}!")
```

```
Hello, John Doe!
```

Tabs and Newlines

- \n = new line
- \t = tab (indent)
- Used to format multi-line output

```
print("Languages:\n\tPython\n\tC\n\tJavaScript")
```

```
Languages:
	Python
	C
	JavaScript
```

Stripping Whitespace

- rstrip() = remove space on the right
- lstrip() = on the left
- strip() = both sides

```
name = " python "
print(name.rstrip())
print(name.lstrip())
print(name.strip())
```

```
 python
python
python
```

Removing Prefixes

- Use removeprefix() to clean up strings (same for removesufix())
- Original value is not changed unless reassigned

```
url = "https://example.com"
print(url.removeprefix("https://"))
```

```
example.com
```

Syntax Errors with Strings

```
message = 'One of Python's strengths is...'
# SyntaxError: unterminated string
```

Syntax Errors with Strings

```
message = 'One of Python's strengths is...'
# SyntaxError: unterminated string
```

```
message = "One of Python's strengths is..."

message2 = 'One of Python\'s strengths is...'
```

Numbers

Python Number Types

- **Integers** (whole numbers)
  - Examples: -2, 0, 42
- **Floats** (numbers with decimals)
  - Examples: 3.14, 0.0, -2.5
- Python automatically chooses the type

```
print(type(5)) # <class 'int'>
print(type(2.0)) # <class 'float'>
```

- You don’t need to declare types in advance

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
```

Division and Mixed Operations

- `/` always returns a float, even for integers
- `//` performs integer division (truncates decimal)
- Mixing int and float in any operation → result is a float

```
print(4 / 2) # 2.0
print(5 // 2) # 2 (integer division)
print(5 / 2) # 2.5
print(1 + 2.0) # 3.0
```

Exponents and Floats

- ** is the exponent operator

```
print(2 ** 3) # 8
print(3 ** 2) # 9
```

More about numbers

- Underscore _ in integers will be ignored

```
universe_age = 14_000_000_000
print(universe_age)
```

- Multi-assignment is possible

```
x, y, z = 0.1, 0, 0
print(x) # 0.1
```

- Python doesn't have real constants

```
MAX_CONNECTIONS = 5000
```

Exercises

Exercises 1

- **2-1**: Assign a message to a variable, then print it.
  (Create a variable like `message = "Hello!"` *and use* `print()` *to show it)*

- **2-2**: Reassign the message variable and print again.
  (Change the value of your `message` *variable to something else and print again)*

- **2-3**: Personal message to someone using a variable.
  (Use a variable like `name = "Tom"` *and print a line like* `Hello Tom, how are you?`*)*

Exercises 2

- **2-4**: Print a name in lowercase, uppercase, and title case.
  (Try using lower(), upper(), and title() methods on a name string)

- **2-5**: Print a quote by a famous person.
  (Example: Albert Einstein once said, "A person who never made a mistake never tried anything new.")

- **2-6**: Store the person's name in a variable, and use it in your quote.
  (Use variables like famous_person and message to build the quote with an f-string)

Exercises 3

- **2-7**: Use \n and \t in a name, and use strip() methods.
  (Try adding whitespace and cleaning it with strip(), lstrip(), rstrip())

- **2-8**: Remove .txt using removesuffix() from a filename.
  (Create a variable like filename = 'notes.txt' and remove the suffix)

- **2-9**: Write 4 operations (add, sub, mult, div) that result in 8.
  (Use print() to show each result, like 5+3, 16/2, etc.)

Exercises 4

- **2-10**: Store your favorite number in a variable, and print a message with it.
  (Use a variable and f-string to display the number in a sentence)

- **2-11**: Add comments to your previous programs.
  (Write at least one comment in each file using # to describe what the code does)

- **2-12**: Run `import this` and skim through the Zen of Python.
  (Try it in the Python terminal or at the top of a script, and read the output)

Session feedback

The speed of the session was too...

Slow

Fast

1

5

To join, go to: **ahaslides.com/ZOV79**

AhaSlides

Which part was most confusing or unclear today?

Get Feedback

Group

0 0/50

Do you plan to install Python and PyCharm on your own laptop?

Yes	No	Not sure yet
0	0	0