Python Summer Course

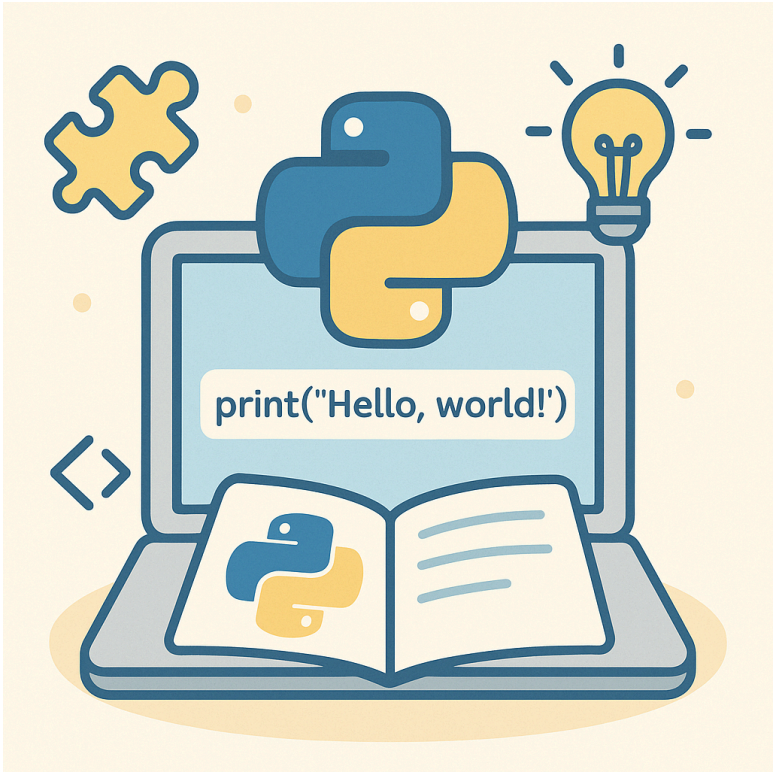
Course 1: Python Basics & Objects

Théophile Gentilhomme

July 9, 2025

Downloading Pyodide

Introduction



My first code

Python Code | Start Over | Run Code

```
1 print("Hello World!")
2 print("Welcome to the Python Summer Course!")
```

Downloading package: pyodide_http

What is programming ?

> “Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.”
> — Wikipedia

🐍 Why Learn with Python?

✅ Easy to read and write — its simple syntax is close to English

✅ Beginner-friendly — widely used in teaching and well-documented

✅ Versatile — used in data science, AI, web development, automation, and more

✅ Large community — thousands of free libraries and tutorials available

✅ Cross-platform — runs on Windows, macOS, and Linux

⚠️ Slower performance compared to compiled languages like C/C++

⚠️ Not ideal for mobile app development or real-time

🛠️ Setting Up Python: How It Works

Python is an interpreted language:

- You write source code in `.py` files
- The Python interpreter reads and executes your code line by line

To run Python code, you need:

- The **Python interpreter**
- A code editor (e.g. VS Code, Thonny, Jupyter Notebook)

❓ What do I use?

Python Code | Start Over | Run Code

```
1 import sys
2 print("I use this kernel to run this code:")
3 print(sys.executable)
```

💡 Installation Options

✅ Option 1: Install Python Locally (e.g. Download from python.org)

✅ Option 2: Use Environment management (e.g. Anaconda)

✅ Option 3: Use Python in the Browser (e.g.. Google Colab)

G Google Colab

1. Go to colab.research.google.com
2. You may have to login to your Google acount
3. Create a new Notebook
4. Start programming!

▶ notebooks in 2 mins

- Write text (Markdown) => + Text
- Write code (Python) = > + Code
- Run cells

🧮 Python as a Calculator

You can just write operation directlx into the cells

Python Code ⟳ Start Over ▷ Run Code

```
1  3 + 1
```

👉 Your turn: try it out using `+ - * / % ** //`

Variable

A variable is a name that stores a value in your program, like a labeled box that holds data.

It lets you remember values (like numbers, text, results of calculations)

You can use or change the value later

Python creates the variable when you assign it a value

Python Code | Start Over | Run Code

```
1  # Assign values to variables
2  name = "Alice"
3  age = 30
4  height = 1.65
5
6  # Use variables in expressions
7  print("Name:", name)
```

Types

Python Code | Start Over | Run Code

```python
# Integer
age = 25
print("Age:", age, "->", type(age))  # <class 'int'>

# Float
height = 1.75
print("Height:", height, "->", type(height))  # <class 'float'>

# String
name = "Alice"
print("Name:", name, "->", type(name))  # <class 'str'>

# Boolean
is_student = True
print("Is student:", is_student, "->", type(is_student))  # <class 'bool'>

# List
grades = [88, 92, 79]
print("Grades:", grades, "->", type(grades))  # <class 'list'>
```

🧱 Everything is an Object in Python

In Python, **everything**, i.e. numbers, strings, functions, etc., is an object.

Objects have data (attributes) and behaviors (methods)

You can call methods with dot syntax: object.method()

Even simple things like numbers and strings are full objects

🧱 Everything is an Object in Python

An object is an instanciation of a `class` (we will see what a `class` is later).

Python Code | Start Over | Run Code

```
# String is an object with methods
text = "hello"
print(text.upper())        # 'HELLO'
print(text.replace("l", "x"))   # 'hexxo'

# Method (function) of an object is also an object!
func = text.upper
print(type(func))

# Even types are objects
print(type(42))
print(type("hi"))
```

🟢 Boolean Conditions in Python

Boolean conditions are used to **ask questions** in your code: they return either `True` or `False`.

Common comparison operators:

- `==` → equal
- `!=` → not equal
- `<`, `>`, `<=`, `>=` → less/greater than (or equal)

Example: Number check

Python Code | Start Over | Run Code

```
1  x = 10
2  print(x > 5)      # True
```

Combine conditions using:

- `and`: both must be True
- `or`: at least one must be True
- `not`: negates the condition

Example: Combining conditions

Python Code | Start Over | Run Code

```
1  age = 20
2  has_id = True
3
4  print(age >= 18 and has_id)
5  print(age < 18 or not has_id)
```

🔀 The `if` Statement

The `if` statement allows your program to **make decisions** based on conditions.

- Runs code only when the condition is `True`
- Can include `elif` (else-if) and `else` branches
- Python uses indentation to define the blocks

✅ Syntax

```
if condition:
    # do something
elif other_condition:
    # do something else
else:
    # fallback
```

🔀 The `if` Example

Python Code | Start Over | Run Code

```
1  age = 18
2
3  if age >= 18:
4      print("You can vote!")
5  else:
6      print("Sorry, too young.")
```

🔁 The `for` Loop

The `for` loop lets you **repeat a block of code** for each item in a sequence.

- Commonly used to loop over lists, strings, or ranges
- Automatically stops when the sequence ends

✅ Syntax

```
for item in sequence:
    # do something with item
```

Example: Looping over a list

Python Code | Start Over | Run Code

```
1  fruits = ["apple", "banana", "cherry"]
2
3  for fruit in fruits:
4      print("I like", fruit)
```

Example: Looping with `range()`

Python Code | Start Over | Run Code

```
1 for i in range(1, 4):
2     print("Number:", i)
```

Example: Looping with `break` and `continue`

Python Code | Start Over | Run Code

```
1  stop = 5
2  for i in range(10):
3      if i < 3:
4          continue
5      print("Number:", i)
6
7      if  i == stop:
8          break
```

🔄 The `while` Loop

A `while` loop repeats a block of code **as long as a condition is `True`**.

- Good for loops where you **don't know in advance how many times to repeat**
- The condition is checked **before** each loop

Syntax

```
while condition:
    # code to repeat
    # Can use continue and break
```

Example: Counting with `while`

Python Code | Start Over | Run Code

```python
count = 1

while count <= 3:
    print("Count is:", count)
    count += 1

while True:
  count -= 1
  if count < -3:
    break
```

 `print()` and `input()` in Python

These are basic but powerful tools for **interacting with the user**.

`print()`: Display output

`input()`: Get user input

Used to **ask the user for information**. Always returns a **string**.

Example

```
1 user_name = input("What is your name? ")
2 print("Nice to meet you,", user_name)
```

⚠️ Reminder:

If you need a number from `input()`, use `int()` or `float()`:

```
1 age = int(input("Enter your age: "))
2 print("In 5 years, you'll be", age + 5)
```

Your turn!

📋 Instructions

Write a Python program that:

1. Greets the user using their name (input() + print())
2. Asks 3 multiple-choice questions
3. Uses if statements to check answers
4. Uses a for or while loop to ask questions one by one
5. Keeps track of the score using a variable
6. Prints the final score and a custom message (e.g., “Well done!” or “Try again”)