Python Summer Course

Course 1: Python Basics & Objects

Théophile Gentilhomme

October 31, 2025

Introduction

My first code

Python Code ↻ Start Over	▷ Run Code

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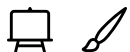
⬅ main

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...

✅ 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 systems
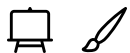
Hes·so VALAIS WALLIS

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]

```python
1  import sys # We will see what it means later
2  # By the way, this is how we write comments in Python
3
4  print("I print who is running my Python")
5  print(sys.executable)
```
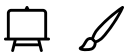
💡 **Installation Options**

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

✅ Option 2: Use Environment management (e.g. Anaconda)
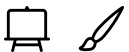
✅ Option 3: Use Python in the Browser (e.g.. Google Colab)

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!

Python Basics & Objects

notebooks in 2 mins

- Write text (Markdown) => + Text

- Write code (Python) = > + Code

- Run cells

Python as a Calculator
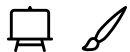
You can just write operation directlx into the cells

Python Code   ↻ Start Over   ▷ Run Code

```
1    3 + 1
```

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

`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)
```

⚠️ 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)
```

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
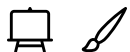
Variable: example

Python Code ⟳ Start Over   ▷ Run Code

```python
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)
8   print("Age in 5 years:", age + 5)
9   print("Height in cm:", height * 100)
10
11  # Change the values
12  name = "Bob"
13  age = age - 5
14  print("Name: ", name)
15  print("Age: ", age)
```

Types

A type in Python defines what kind of data a value is, and what you can do with it.

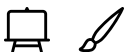Python Code   ↻ Start Over   ▷ Run Code

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

# Float
height = 1.75
print("Height:", height, "->", type(height))

# String
name = "Alice"
print("Name:", name, "->", type(name))

# Boolean
is_student = True
print("Is student:", is_student, "->", type(is_student))
```

← main

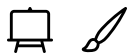Hes·so// VALAIS WALLIS

Everything is an Object in Python

- 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
# 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"))
print(type(int))
```

Python Code | ↻ Start Over | ▷ Run Code

⬅ main

Python Basics & Objects

Hes·so// VALAIS WALLIS

String Manipulation in Python
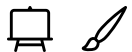
Strings are sequences of characters: you can access, combine, and transform them easily.

Python Code ⟳ Start Over   ▷ Run Code

```
1  text = "Python"
2  print(text[0])
3  print(text[-1])
4  print(text[1:4])
```
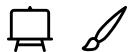
Common string methods

Python Code ⟳ Start Over   ▷ Run Code

```python
print("hello".upper())
print("HELLO".lower())
```

Common string methods

Python Code   ⟳ Start Over   ▷ Run Code

```python
1 print("Python".startswith("Py"))
```

Python Code   ⟳ Start Over   ▷ Run Code

```python
1 print("data,science".split(","))
```

Python Code   ⟳ Start Over   ▷ Run Code

```python
1 print("  clean  ".strip())
```

Common string methods

Python Code · ⟳ Start Over ▷ Run Code

```python
text = "My name is {} and I am  {}".format("Bob", 40)
print(text)
```

Python Basics & Objects

Hes·so VALAIS WALLIS

Common string methods

```python
# Chain of methods
text = " my veRy UGly?? ## text   "
clean_text = text.strip().replace("?", "").replace("#", "").replace("  ", " ").lo
print(clean_text)
```

Python Code   ↻ Start Over   ▷ Run Code

Concatenation and repetition

Python Code   ⟳ Start Over   ▷ Run Code

```python
"Py" + "thon"          # 'Python'
"ha" * 3               # 'hahaha'
```

Strings are **immutable**: you can't change them directly, but you can create new ones.

Boolean Conditions in Python

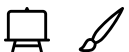Boolean conditions are used to **ask questions** in your code: they return either `True` or `False`.

Common comparison operators:

- `==` → equal

- `!=` → not equal

- `<`, `>`, `<=`, `>=` → less/greater than (or equal)

Python Code ⟳ Start Over   ▷ Run Code

```
1  x = 10
2  print(x > 5)     # True
3  print(x == 10)   # True
4  print(x != 7)    # True
```

⬅ main

Python Basics & Objects

Combine conditions using:

- `and`: both must be True

- `or`: at least one must be True

- `not`: negates the condition

Python Code ↻ Start Over   ▷ Run Code

```python
age = 20
has_id = True

print(age >= 18 and has_id)
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
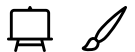
- Python uses indentation to define the blocks

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

Python Basics & Objects

Hes·so// VALAIS WALLIS

The `if` Example

Python Code ⟳ Start Over	▷ Run Code

```python
age = 18

if age >= 18:
    print("You can vote!")
else:
    print("Sorry, too young.")
```
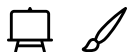
⬅ main

Python Basics & Objects

Hes·so// VALAIS WALLIS

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

```python
1  for item in sequence:
2      # do something with item
```

main

Python Basics & Objects

Example: Looping over a list

A list in Python is an ordered collection of items (like numbers, strings or any objects) that can be changed, added to, or removed; written with square brackets `[]`. (see Course 2)

Python Code ↻ Start Over   ▷ Run Code

```python
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print("I like", fruit)


# If you need the index too
for i, fruit in enumerate(fruits):
    print("Fruit number", i)
    print("I like", fruit)
```

← main

Python Basics & Objects

Hes·so VALAIS WALLIS

Example: Looping with `range()`

Python Code | ↻ Start Over | ▷ Run Code

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

Example: Looping with `break` and `continue`

Python Code | ↻ Start Over | ▷ Run Code

```python
stop = 5
for i in range(10):
    if i < 3:
        continue
    print("Number:", i)

    if  i == stop:
        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

```python
1  while condition:
2      # code to repeat
3      # Can use continue and break
```

Python Basics & Objects

Example: Counting with `while`

```python
count = 1

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

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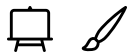
Your turn!

Write a Python program that:

1. Greets the user using their name (`input()` + `print()`)

2. Asks 3 multiple-choice questions (list of questions + list of answers)

3. Uses if statements to check answers (use object method to make it UPPER or lower case and then compare to the right answer)

4. Uses a for or while loop to ask questions one by one (and `input()` + `print()`)

5. Keeps track of the score using a variable and `if-else` statements to increment the score.

6. Prints the final score and a custom message

More references

[Python course for data analysis](#)

[The Python tutorial](#)

Solution

Show Solution | Enter code