I. Lệnh print

**Lệnh print() trong Python**

**- Cú pháp cơ bản:**

print(giá\_trị\_hoặc\_biểu\_thức)

**- Ví dụ:**

print("Hello, world!") # In ra chuỗi

print(42) # In ra số

print(3 + 4) # In ra biểu thức toán học

**Một số tính năng mở rộng:**

**1. In nhiều thứ cùng lúc (cách nhau bằng dấu cách)**

print("Tổng:", 2 + 3) # Kết quả: Tổng: 5

**2. Thay đổi ký tự kết thúc (end)**

Mặc định print() kết thúc bằng xuống dòng (\n). Bạn có thể thay đổi điều này:

print("Hello", end=" ")

print("world!") # In ra: Hello world!

**3. Thay đổi ký tự phân cách (sep) giữa các giá trị**

print("10", "20", "30", sep="-") # Kết quả: 10-20-30

**4. In biến**

name = "An"

age = 20

print("Tên:", name, "Tuổi:", age)

**Mẹo nâng cao:**

**5. Sử dụng f-string (Python 3.6+)**

Bên trong {} có thể dùng biểu thức, hàm, hoặc bất cứ phép tính nào hợp lệ trong Python.

f"văn bản {biến hoặc biểu thức}"

Cho phép chèn biến trực tiếp vào chuỗi:

name = "Bình"

age = 21

print(f"Tên: {name}, Tuổi: {age}") # Dễ đọc, hiện đại

pi = 3.1415926535

print(f"Số pi là: {pi:.2f}") # Làm tròn đến 2 chữ số sau dấu phẩy

**6. In xuống dòng hoặc ký tự đặc biệt**

print("Dòng 1\nDòng 2") # \n: xuống dòng

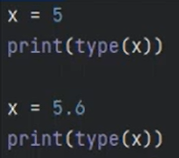
print("Cột 1\tCột 2") # \t: tab

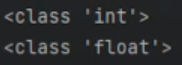
II. Biến, phép toán

1. Biến

x = 5

print(type(x)) -> kiểu của giá trị, không phải của biến





2. Phép toán

Phép cộng: print(x + y)

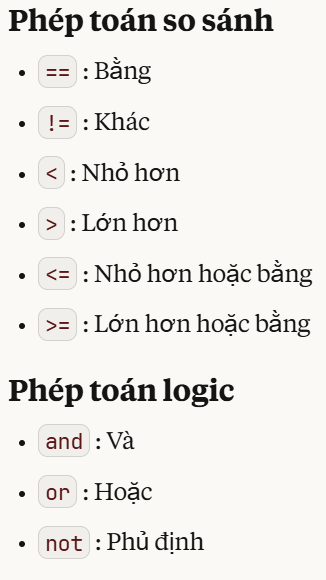
Phép trừ: print(x - y)

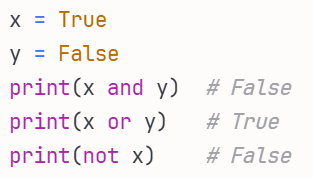
Phép chia: print(x / y) #float

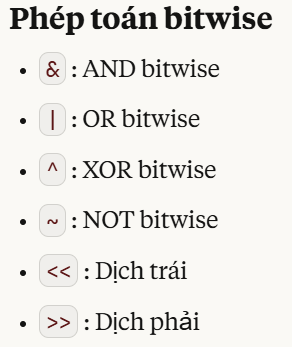
Phép chia lấy nguyên: print(x // y)

Phép chia lấy phần dư: print(x % y)

Phép luỹ thừa: print(x \*\* y) (x^y)



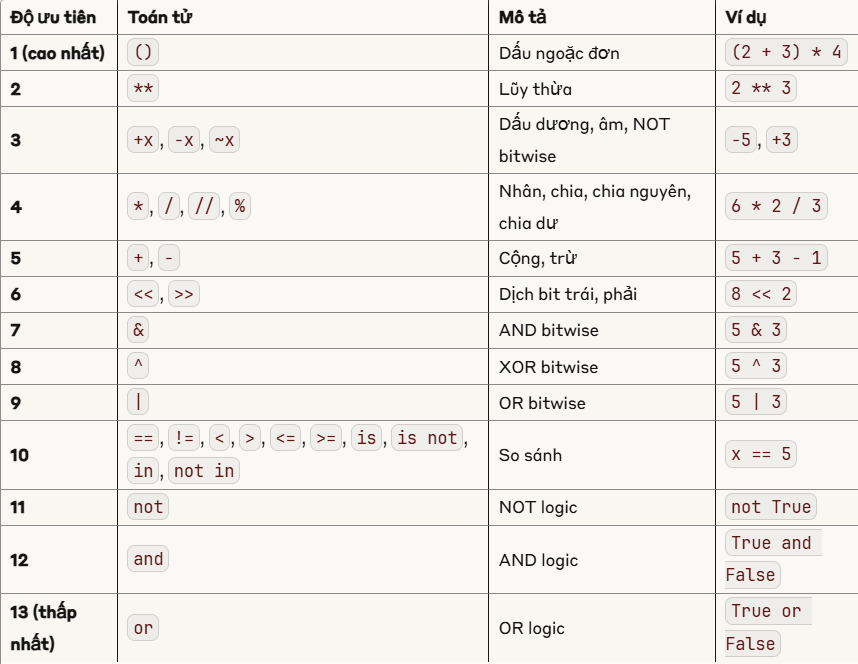








- Độ ưu tiên toán tử:



3. type

- Kiểm tra kiểu dữ liệu cơ bản:

# Số nguyên

print(type(42)) # <class 'int'>

# Số thực

print(type(3.14)) # <class 'float'>

# Chuỗi

print(type("Hello")) # <class 'str'>

# Boolean

print(type(True)) # <class 'bool'>

# List

print(type([1, 2, 3])) # <class 'list'>

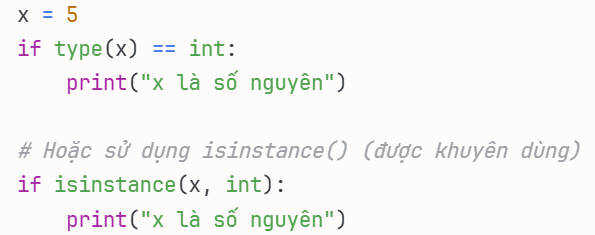
# Dictionary

print(type({"a": 1})) # <class 'dict'>

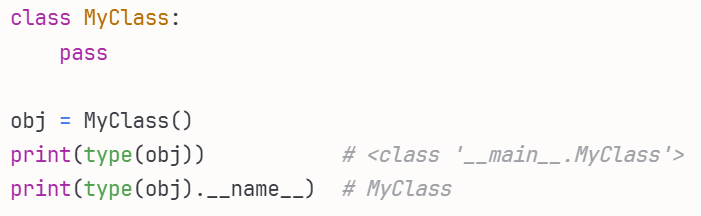
# Tuple

print(type((1, 2))) # <class 'tuple'>

- So sánh kiểu dữ liệu:

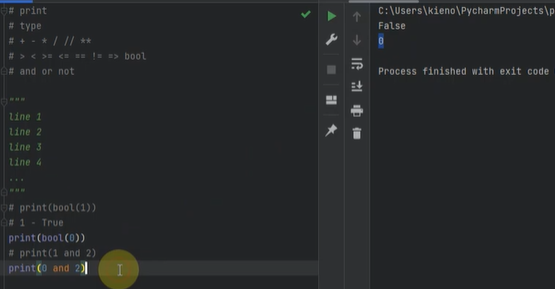


- Với class tự dịnh nghĩa:



4. and và or

- and: đúng đi tiếp



- or: sai đi tiếp

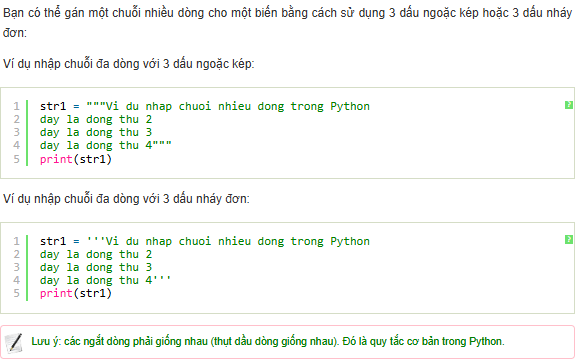
5. Chuỗi

- Gán chuỗi cho 1 biến

str1 = "Hello World!"

print(str1)

- Chuỗi đa dòng:



- Chuỗi là 1 mảng:

Các chuỗi trong Python là mảng các byte đại diện cho các ký tự unicode.

Tuy nhiên, Python không có kiểu dữ liệu ký tự, một ký tự đơn giản chỉ là một chuỗi có độ dài bằng 1.

Dấu ngoặc vuông [] có thể được sử dụng để truy cập các phần tử của chuỗi. Ký tự đầu tiên có chỉ số là 0.

str1 = "HELLO"

print(str1[0])

print(str1[1])

print(str1[2])

print(str1[3])

print(str1[4])

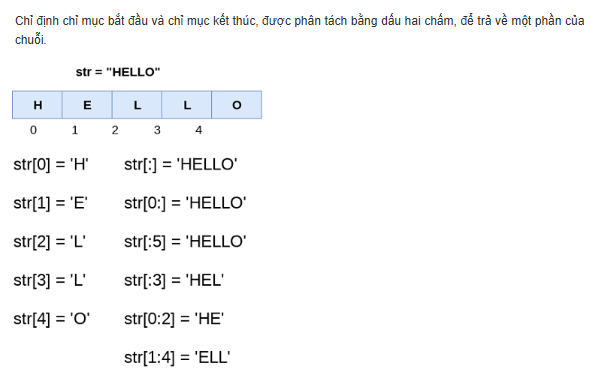
H

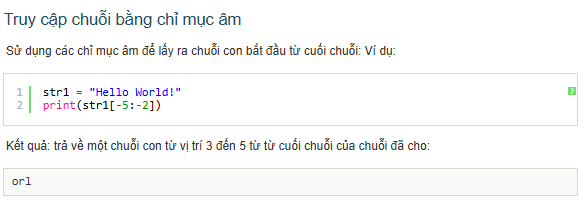
E

L

L

O



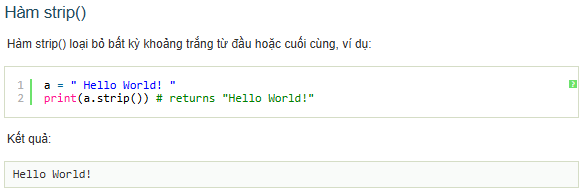


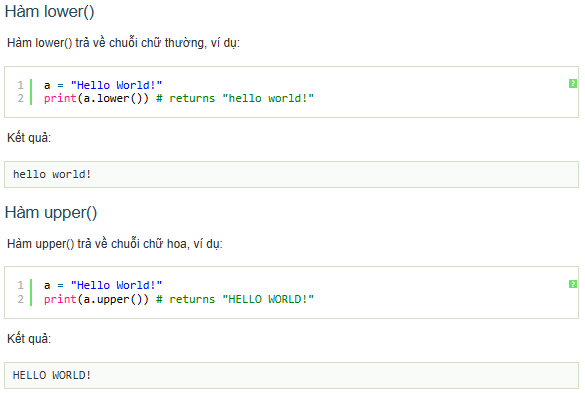
- Chiều dài chuỗi trong python: hàm len()

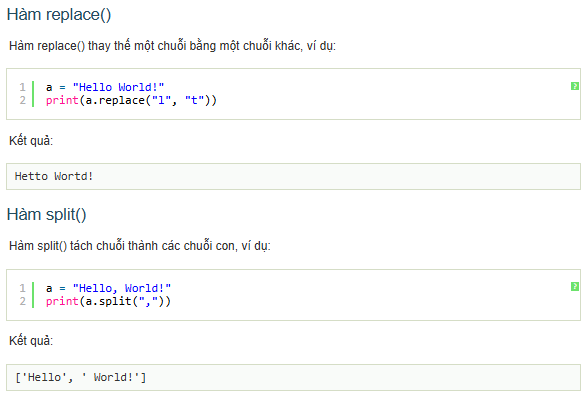
a = "Hello World!"

print(len(a))

- Các hàm cơ bản xử lý String trong Python

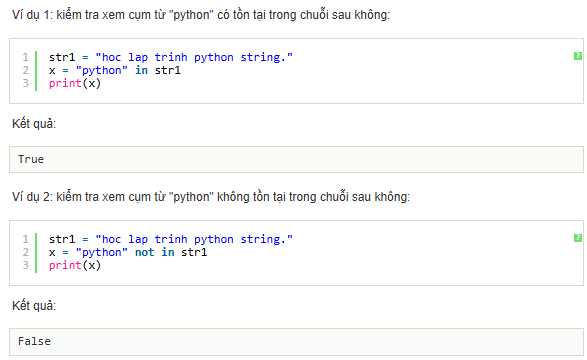






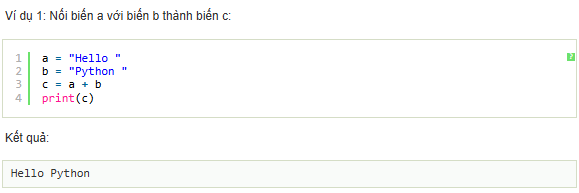
- Kiểm tra sự tồn tại của chuỗi trong chuỗi:

Để kiểm tra xem một cụm từ hoặc ký tự nào đó có trong một chuỗi hay không, chúng ta có thể sử dụng các từ khóa **in** hoặc **not in**



- Nối chuỗi trong Python:

Để nối hoặc kết hợp hai chuỗi với nhau bạn có thể sử dụng toán tử +.



- Định dạng chuỗi trong Python:

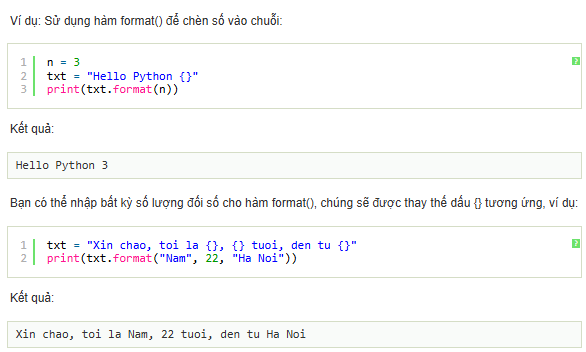
Trong Python, chúng ta không thể kết hợp các chuỗi và số như thế này:

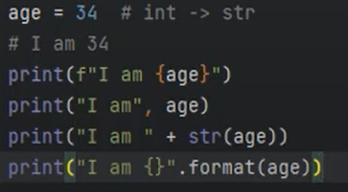
n = 3

txt = "Hello Python " + 3

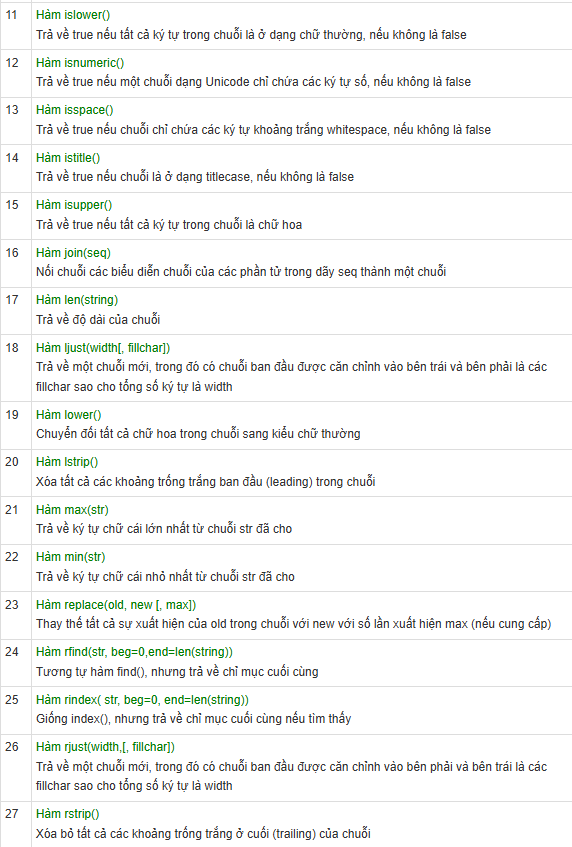
print(txt)

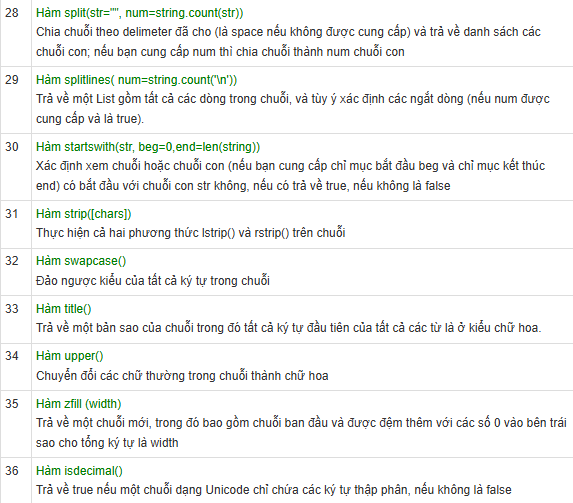
Chúng ta có thể kết hợp chuỗi và số bằng cách sử dụng hàm **format()**





­­­­ 





6. Hàm input



**Lưu ý quan trọng:** input() luôn trả về kiểu **string**.







