Python cho Khoa học dữ liệu

Bài : Comment - Unser input - Numbers - Booleans

Hà Minh Tuấn

Khoa Toán - Tin học
Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM

Ngày 28 tháng 8 năm 2025

Nội dung bài học

Python Comments

- **Comment** là chú thích trong mã, giúp lập trình viên hiểu dễ hơn.
- Python bỏ qua comment khi thực thi chương trình.
- Ba loại comment chính trong Python:
  - Single-line comments.
  - Multi-line comments.
  - Docstring comments.

Ví dụ: Single-line Comment

```
 # Đây là một comment
print("Hello, World!")
```

```
Hello, World!
```

Standalone và Inline Comment

```python
# Standalone single line comment
def greet():
    print("Hello, World!")
greet()

print("Hello, World!")  # Inline single line comment
```

```
Hello, World!
Hello, World!
```

Multi-line Comments với

```python
# Hàm này tính giai thừa của một số
# dùng cách tiếp cận lặp.
# factorial(5) = 5*4*3*2*1 = 120.
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers"
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

number = 5
print(f"The factorial of {number} is {factorial(number)}")
```

```
The factorial of 5 is 120
```

Multi-line Comments với Triple-quoted Strings

```
"""
Hàm này tính GCD của hai số
dùng thuật toán Euclid.
"""
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

result = gcd(48, 18)
print("The GCD of 48 and 18 is:", result)
```

```
The GCD of 48 and 18 is: 6
```

Docstring trong Python

- Docstring là một loại comment đặc biệt dùng để **tài liệu hóa** hàm, lớp, module.
- Được viết trong dấu triple quotes ngay sau định nghĩa.
- Có thể truy xuất qua `.__doc__` hoặc hàm `help()`.

Ví dụ: Function Docstring

```python
def greet(name):
    """
    Hàm này chào người dùng theo tên.

    Parameters:
    name (str): tên người cần chào

    Returns:
    None
    """
    print(f"Hello, {name}!")

greet("Alice")
```

```
Hello, Alice!
```

Truy xuất Docstring

```
def greet(name):
    """
    Hàm này chào người dùng theo tên.
    """
print(greet.__doc__)
```

```
Hàm này chào người dùng theo tên.
```

```
help(greet)
```

```
Help on function greet in module __main__:

greet(name)
    Hàm này chào người dùng theo tên.
```

Python User Input

- Mọi ứng dụng cần có khả năng nhận dữ liệu từ người dùng khi chạy.
- Python cung cấp 2 hàm để nhập liệu:
  - `input()` – Python 3.x
  - `raw_input()` – Python 2.x (được thay thế bằng `input()` trong Python 3)
- Dữ liệu nhập từ `input()` luôn là chuỗi (`str`).

Ví dụ: Gán giá trị trực tiếp

```
name = "Kiran"
city = "Hyderabad"

print("Hello My name is", name)
print("I am from", city)
```

```
Hello My name is Kiran
I am from Hyderabad
```

Ví dụ: Nhập dữ liệu với input()

```
name = input()
city = input()

print("Hello My name is", name)
print("I am from", city)
```

```
Ravi
Chennai
Hello My name is Ravi
I am from Chennai
```

Sử dụng input() với Prompt

```
name = input("Enter your name : ")
city = input("Enter your city : ")

print("Hello My name is", name)
print("I am from", city)
```

```
Enter your name: Praveen Rao
Enter your city: Bengaluru
Hello My name is Praveen Rao
I am from Bengaluru
```

raw_input() trong Python 2

```
name = raw_input("Enter your name - ")
city = raw_input("Enter city name - ")

print("Hello My name is", name)
print("I am from", city)
```

```
Enter your name - Ravi
Enter city name - Chennai
Hello My name is Ravi
I am from Chennai
```

Lỗi khi nhân chuỗi

```
width = input("Enter width : ")
height = input("Enter height : ")

area = width * height
print("Area of rectangle = ", area)
```

```
Enter width: 20
Enter height: 30
TypeError: can't multiply sequence by non-int of type 'str'
```

Ép kiểu sang số nguyên

```
width = int(input("Enter width : "))
height = int(input("Enter height : "))

area = width * height
print("Area of rectangle = ", area)
```

```
Enter width: 20
Enter height: 30
Area of rectangle = 600
```

Ép kiểu sang số thực

```
amount = float(input("Enter Amount : "))
rate = float(input("Enter rate of interest : "))

interest = amount * rate / 100
print("Amount:", amount, "Interest:", interest)
```

```
Enter Amount: 12500
Enter rate of interest: 6.5
Amount: 12500.0 Interest: 812.5
```

Python print() Function

- `print()` là hàm tích hợp dùng nhiều nhất trong Python.
- Có thể in nhiều biến, biểu thức trong cùng một lệnh.
- Tham số hữu ích:
  - `sep` – ký tự phân tách giữa các giá trị.
  - `end` – ký tự kết thúc, mặc định là `"\n"`.

Ví dụ với print()

```
a = "Hello World"
b = 100
c = 25.50
d = 5+6j

print("Message:", a)
print(b, c, b-c)
print(pow(100, 0.5), pow(c, 2))
```

```
Message: Hello World
100 25.5 74.5
10.0 650.25
```

Sử dụng sep trong print()

```
city = "Hyderabad"
state = "Telangana"
country = "India"
print(city, state, country, sep=',')
```

```
Hyderabad,Telangana,India
```

Sử dụng end trong print()

```
city = "Hyderabad"
state = "Telangana"

print("City:", city, end=" ")
print("State:", state)
```

```
City: Hyderabad State: Telangana
```

Python Numbers

- Python hỗ trợ nhiều kiểu số học:
  - `int` – số nguyên
  - `float` – số thực dấu phẩy động
  - `complex` – số phức
  - `bool` – giá trị logic, là kiểu con của `int` (`True=1`, `False=0`)
- Số có thể biểu diễn theo hệ nhị phân, bát phân, thập lục phân.

Ví dụ: Integer Numbers

```
a = 10
b = 20
c = a + b

print("a:", a, "type:", type(a))
print("c:", c, "type:", type(c))
```

```
a: 10 type: <class 'int'>
c: 30 type: <class 'int'>
```

Chuyển đổi kiểu về int

```
a = int(10.5)
b = int("100")

print(a, type(a))
print(b, type(b))
```

```
10 <class 'int'>
100 <class 'int'>
```

Số nhị phân trong Python

```
a = 0b101
print("a:", a, "type:", type(a))

b = int("0b101011", 2)
print("b:", b, "type:", type(b))
```

```
a: 5 type: <class 'int'>
b: 43 type: <class 'int'>
```

Hàm bin()

```
a = 43
b = bin(a)
print("Integer:", a, "Binary equivalent:", b)
```

```
Integer: 43 Binary equivalent: 0b101011
```

Số bát phân (Octal)

```
a = 00107
print(a, type(a))

b = int('20', 8)
print(b, type(b))
```

```
71 <class 'int'>
16 <class 'int'>
```

Hàm oct()

```
a = 71
print(oct(a), type(oct(a)))
```

```
0o107 <class 'str'>
```

Số thập lục phân (Hexadecimal)

```
a = 0XA2
print(a, type(a))

b = int("0X1e", 16)
print(b, type(b))
```

```
162 <class 'int'>
30 <class 'int'>
```

Hàm hex()

```
num_string = "A1"
number = int(num_string, 16)
print("Hexadecimal:", num_string, "Integer:", number)

print(hex(161), type(hex(161)))
```

```
Hexadecimal: A1 Integer: 161
0xa1 <class 'str'>
```

Hỗn hợp các hệ số

```
a = 10        # decimal
b = 0b10      # binary
c = 0O10      # octal
d = 0XA       # hexadecimal
e = a+b+c+d

print("addition:", e)
```

```
addition: 30
```

Floating Point Numbers

- Số thực có phần nguyên và phần thập phân.
- Có thể viết dưới dạng ký hiệu khoa học: `1.23E3`, `9.9E-5`.
- Được biểu diễn bởi lớp `float`.

Ví dụ float

```
a = 10.33
b = 2.66
c = a/b
print("c:", c, "type:", type(c))
```

```
c: 3.8834586466165413 type <class 'float'>
```

Hàm float()

```
a = float(0b10)
b = float(0O10)
c = float(0xA)
print(a, b, c, sep=",")
```

```
2.0,8.0,10.0
```

Infinity và NaN

```
a = 1.0E400
print(a, type(a))

b = float("Infinity")
print(b, type(b))

c = float("Nan")
print(c, type(c))
```

```
inf <class 'float'>
inf <class 'float'>
nan <class 'float'>
```

Complex Numbers

- Số phức có dạng `x + yj`.
- Dùng trong nhiều ứng dụng toán học và kỹ thuật.
- Tạo bằng cách:
  - Dùng biểu diễn trực tiếp: `5+6j`
  - Hàm `complex(x,y)`

Ví dụ Complex Numbers

```
a = complex(5.3, 6)
b = complex(1.01E-2, 2.2E3)
print("a:", a, "type:", type(a))
print("b:", b, "type:", type(b))
```

```
a: (5.3+6j) type: <class 'complex'>
b: (0.0101+2200j) type: <class 'complex'>
```

Thuộc tính và phương thức của số phức

```
a = 5+6j
print("Real part:", a.real)
print("Imag part:", a.imag)
print("Conjugate:", a.conjugate())
```

```
Real part: 5.0
Imag part: 6.0
Conjugate: (5-6j)
```

Các hàm toán học trong Python

- Hàm tích hợp sẵn: `abs()`, `max()`, `min()`, `pow()`, `round()`, `sum()`.
- Mô-đun `math` cung cấp:
  - Các hàm số học: `ceil()`, `floor()`, `gcd()`, `factorial()`...
  - Hàm mũ, logarit: `exp()`, `log()`, `sqrt()`...
  - Hàm lượng giác: `sin()`, `cos()`, `tan()`...
  - Hằng số toán học: `pi`, `e`, `tau`, `inf`, `nan`.
- Mô-đun `random`: tạo số ngẫu nhiên.

Kiểu dữ liệu Boolean trong Python

- `bool` là một kiểu con của `int`.
- Chỉ có hai giá trị: `True` và `False`.
- `True` tương ứng với 1, `False` tương ứng với 0.
- Có thể chuyển đổi sang các kiểu số: `int()`, `float()`, `complex()`.

Ví dụ chuyển đổi bool sang số

```
a = int(True)
print("bool to int:", a)

a = float(False)
print("bool to float:", a)

a = complex(True)
print("bool to complex:", a)
```

```
bool to int: 1
bool to float: 0.0
bool to complex: (1+0j)
```

Biểu thức Boolean

- Biểu thức Boolean là biểu thức trả về `True` hoặc `False`.
- Thường liên quan đến toán tử so sánh (==, >, <...).
- Hàm `bool(x)` dùng để kiểm tra giá trị logic của một đối tượng.
  - Trả về `False` cho: `False`, `None`, số 0, chuỗi/tuple/list/dict rỗng.
  - Ngược lại trả về `True`.

Ví dụ hàm bool()

```
# Check true / false
print(bool(True))
print(bool(False))

# Check 0 và 1
print(bool(0.0))
print(bool(1.0))

# So sánh
a, b = 5, 10
print(bool(a == b))

# Kiểm tra None
print(bool(None))
```

Ví dụ hàm bool()

```
# Chuỗi, tuple, dict rỗng
print(bool(()))
print(bool({}))

# Chuỗi không rỗng
print(bool("Tutorialspoint"))
```

```
True
False
False
True
False
False
False
False
True
```