Softech®
Gather to thrive together

Aptech™
COMPUTER EDUCATION
Unleash your potential

BÀI 2

Biến và Kiểu dữ liệu

Tóm Tắt Nội Dung

Trong bài học này chúng ta sẽ đi tìm hiểu lần lượt các nội dung

1. Cách khai báo BIẾN trong Python
2. Nguyên tắc đặt tên BIẾN
3. Phạm vi sử dụng biến (Scope)
4. Các kiểu dữ liệu của BIẾN
5. Dữ liệu kiểu số và Các toán tử số học trong Python

2.1 Cách khai báo BIẾN trong Python

Định nghĩa: Biến là gì ?

Biến (variable) là một khái niệm cơ bản trong lập trình, nó là tên của một khu vực trong bộ nhớ của máy tính, được sử dụng để lưu trữ và đại diện cho các giá trị dữ liệu như số, chuỗi, đối tượng, và nhiều loại dữ liệu khác

Khi ta tạo một biến trong Python, ta đang cung cấp một tên duy nhất cho biến đó và gán cho nó một giá trị cụ thể. Biến có thể được sử dụng để tham chiếu đến giá trị của nó trong các phép toán, điều kiện, hoặc các phần khác của chương trình.

Biến có thể chứa dữ liệu kiểu số (int), chuỗi (str), logic (bool)…

2.1 Cách khai báo BIẾN trong Python

Cú pháp khai báo biến

tên_biến = giá_trị

- **tên_biến** là tên mà ta muốn đặt cho biến
- **giá_trị** là giá trị mà ta muốn gán cho biến

Khi khai báo biến trong Python ta không cần phải khai báo kiểu dữ liệu ta chỉ cần gán một giá trị cho tên biến, Python sẽ tự động suy ra kiểu dữ liệu tương cho biến đó.

Sử dụng hàm **type()** để kiểm tra kiểu dữ liệu hiện tại của biến

2.1 Cách khai báo BIẾN trong Python

Ví dụ về khai báo biến

Ví dụ để định nghĩa biến **name** với giá trị **Python** ta viết như sau:

```
name  = ‘Python’ # name là kiểu chuỗi str
```

Ví dụ để định nghĩa biến **age** với giá trị **35** ta viết như sau:

```
age  = 35 # age là kiểu số int
```

Kiểm tra kiểu dữ liệu hiện tại của biến

```
print(type(name)) # Kết quả in ra Terminal: <type 'str'>
```

2.2 Nguyên tắc đặt tên biến trong Python

1. Tên biến chỉ chứa các ký tự a-z, A-Z, 0-9 và dấu gạch dưới _
2. Phải bắt đầu bằng một ký tự hoặc dấu gạch dưới _
3. Tên biến không bắt đầu bằng một con số
4. Tên biến phân biệt chữ hoa chữ thường (age, Age , AGE là 3 biến khác nhau)
5. Tên biến không chứa các từ khóa của Python

Langue_name = ‘Python’

Từ khóa trong Python

Keywords

```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

Soft Keywords

```
match      case       _
```

2.2 Nguyên tắc đặt tên biến trong Python

Một số cách đặt **hợp lệ**

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

Một số cách đặt **không hợp lệ**

```
2myvar = "John"
my-var = "John"
my var = "John"
```

Đặt tên kiểu **Snake Case**

```
my_variable_name  = "John"
```

Đặt tên kiểu **Pascal Case**

```
MyVariableName  = "John"
```

Đặt tên kiểu **Camel Case**

```
myVariableName  = "John"
```

2.2 Nguyên tắc đặt tên biến trong Python

Gán nhiều giá trị cho nhiều biến

```
x, y, z  =  "Orange", "Banana", "Cherry"
print(x) # x = Orange
print(y) # y = Banana
print(z) # z = Cherry
```

Hoán đổi giá trị của biến gán đồng thời

```
x, y   =  1, 2
print(x, y) # (1, 2)
x, y = y, x  # Hoán đổi giá trị x và y cho nhau
print(x, y) # (2, 1)
```

2.2 Nguyên tắc đặt tên biến trong Python

Gán một giá trị cho nhiều biến

```
x, y, z  =  "Orange“
print(x) # x = Orange
print(y) # y = Orange
print(z) # z = Orange
```

Xuất giá trị của biến (Output variables)

```
x =  “Hello“
y =  “Python“
print(x, y)
print(x + y)
```

2.3 Phạm vi sử dụng biến

```
x =  "awesome “
def myfunc():
   print("Python is " + x)

myfunc()
```

Biến khai báo bên ngoài khối, sử dụng được bên trong khối

```
x =  "awesome"
def myfunc():
    x =  "fantastic"
    print("Python is " + x)

myfunc()

print("Python is " + x)
```

Bạn tạo một biến trong khối, thì biến đó chỉ dùng được bên trong khối hay gọi là **local variable.**
Dù nó trùng tên với ngoài khối

2.3 Phạm vi sử dụng biến

Biến toàn cục (Global variable) với từ khóa global

```
def myfunc():
    global x
    x =  "fantastic"

myfunc()

print("Python is " + x)
```

Nếu dùng từ khóa **global** trong khối, thì biến đó dùng được ngoài khối

```
x = "awesome"
def myfunc():
    global x
    x =  "fantastic"

myfunc()

print("Python is " + x)
```

Nếu dùng từ khóa **global** trong khối, và ngoài khối cũng có biến trùng tên thì nó làm thay đổi giá trị biến đã khai báo ở trước

2.4 Các Kiểu dữ liệu trong Python

Ví dụ	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	Range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = None	NoneType

2.4 Các Kiểu dữ liệu trong Python

Ép kiểu dữ liệu	Data Type
x = str("Hello World")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("apple", "banana", "cherry"))	list
x = tuple(("apple", "banana", "cherry"))	tuple
x = range(6)	Range
x = dict(name="John", age=36)	dict
x = set(("apple", "banana", "cherry"))	set
x = frozenset(("apple", "banana", "cherry"))	frozenset
x = bool(5)	bool

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.1 Số trong Python - Python Numbers

Có 3 loại số trong Python: int, float, complex

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

```
print(type(x))
```

Dùng hàm type() để kiểm tra kiểu dữ liệu của biến

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.1 Số trong Python - Python Numbers

int

Hay integer, bao gồm số, số nguyên âm, số nguyên dương, không bao gồm số thập phân, không giới hạn độ dài

```
x = 1
y = 43545454656
z =  -8547584
```

float

là một số, dương hoặc âm, chứa một hoặc nhiều số thập phân.

```
x = 1.1
y = 1.02
z =  -8,78
```

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

Phép tính	Ý nghĩa	Ví dụ
+	Cộng	x + y
-	Trừ	x - y
*	Nhân	x * y
/	Chia	x / y
%	Chia lấy phần dư	x % y
**	Lũy thừa	x ** y
//	Chia lấy phần nguyên	x // y

Trong Python, phép tính được thực hiện theo thứ tự ưu tiên sau đây:

1. Các phép tính trong dấu ngoặc đầu tiên được thực hiện trước.
2. Các phép nhân và chia được thực hiện trước các phép cộng và trừ.
3. Các phép cộng và trừ được thực hiện từ trái sang phải.

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

Thứ tự ưu tiên được mô tả trong bảng bên dưới, bắt đầu với ưu tiên cao nhất ở trên cùng:

Operator	Description
()	Trong ngoặc
**	Lũy thừa
+x -x ~x	Cộng một ngôi, trừ một ngôi và theo chiều bit KHÔNG
* / // %	Phép nhân, phép chia, phép chia sàn và lấy dư (mod)
+ -	Phép cộng, trừ
<< >>	Dịch chuyển trái và phải theo từng bit

Xem tiếp trang sau

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

Thứ tự ưu tiên các phép tính trong Python

Operator	Description
^	Bitwise XOR
|	Bitwise OR
== != > >= < <= is is not in not in	Toán tử so sánh: bằng, không bằng, lớn hơn, lớn hơn hoặc bằng, nhỏ hơn, nhỏ hơn hoặc bằng
not	Logical NOT
and	AND
or	OR

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

Một số ví dụ về toán tử số học

```
x = 5
y = 3
print(x + y) # Phép cộng
print(x - y) # Phép trừ
print(x * y) # Phép nhân
print(x / y) # Phép chia
print(x % y) # Chia Mod
print(x // y) # Chia sàn
```

Biểu thức nhiều phép tính

```
print(5 + 4 - 7 + 3)
```

Tình từ trái qua phải
5 + 4 = 9
9 - 7 = 2
2 + 3 = 5 là kết quả

2.5 Dữ liệu kiểu Số và toán tử số học trong Python

2.5.2 Toán tử số học trong Python (Arithmetic Operators)

```
result = 2 + 3 * 4 - 6 / 2 ** 2
```

Trong ví dụ này, chúng ta có một biểu thức gồm các toán tử số học: cộng (+), nhân (*), chia (/), và lũy thừa (**). Ta sẽ giải thích từng bước của phép tính theo thứ tự ưu tiên

- 2 ** 2: Thực hiện phép lũy thừa. 2 ** 2 tương đương với 2^2 = 4
- 3 * 4: Thực hiện phép nhân. Kết quả của phép nhân là 12
- 6 / 4: Thực hiện phép chia. Kết quả của phép chia là 1.5
- 2 + 12: Thực hiện phép cộng. Kết quả của phép cộng là 14
- 14 - 1.5: Thực hiện phép trừ.
- Kết quả của phép trừ là 12.5

Tổng kết lại bài 2

1. Biết cách khai báo BIẾN trong Python
2. Nắm được nguyên tắc đặt tên BIẾN
3. Nắm được phạm vi sử dụng biến (Scope)
4. Nắm được các kiểu dữ liệu của BIẾN, ép kiểu dữ liệu
5. Nắm được cách thực hiện các phép tính trong toán tử số học