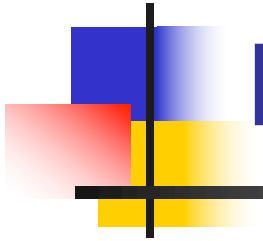
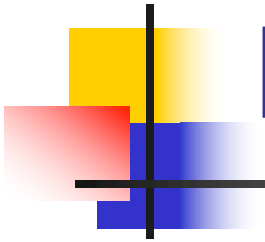


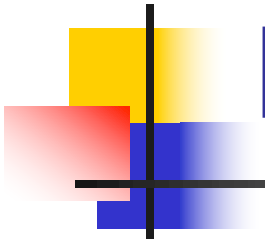
Bài 3: Các kiểu biến trong Python

TS. Trịnh Tuấn Đạt
Viện CNTT-TT, ĐHBK Hà Nội



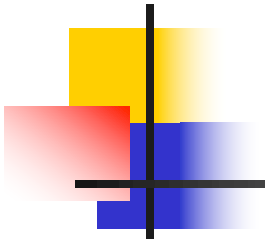
Nội dung

1. Khái niệm biến
2. Lệnh gán giá trị cho biến
3. Các kiểu dữ liệu



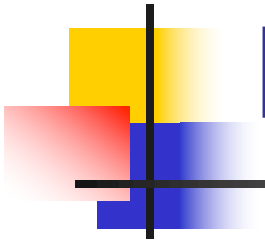
Nội dung

1. **Khái niệm biến**
2. Lệnh gán giá trị cho biến
3. Các kiểu dữ liệu



1. Khái niệm

- Biến: vùng nhớ dành riêng để lưu giá trị
- Tùy kiểu dữ liệu của biến, trình thông dịch cấp phát bộ nhớ và quyết định những gì có thể được lưu trữ trong khu nhớ dành riêng đó.
- Các kiểu dữ liệu khác nhau cho biến: số nguyên, thập phân, ký tự, …



Nội dung

1. Khái niệm biến
2. **Lệnh gán giá trị cho biến**
3. Các kiểu dữ liệu

2. Gán giá trị cho biến

- Trong Python, không cần khai báo biến một cách tường minh
- Biến được khai báo tự động khi ta gán giá trị bất kỳ cho biến với phép gán =
- Toán hạng trái là tên biến và toán hạng phải là giá trị được lưu trữ trong biến.
- Ví dụ:

```
a = 20              # Mot phep gan so nguyen
b = 100.0           # Mot so thuc
ten = "Hoang"       # Mot xau/chuoi


print (a)
print (b)
print (ten)
```

Phép đa gán (multiple assignment)

Python hỗ trợ hai kiểu đa gán

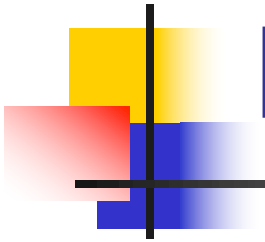
- Gán giá trị đơn cho nhiều biến

```
a = b = c = 1
```

- Hoặc gán nhiều giá trị cho nhiều biến

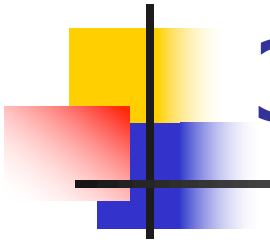
```
a,b,c = 5,10,15


print (a)
print (b)
print (c)
```



Nội dung

1. Khái niệm biến
2. Lệnh gán giá trị cho biến
3. **Các kiểu dữ liệu**



3. Các kiểu dữ liệu trong Python

- Có nhiều kiểu dữ liệu:
  - Lương nhân viên: giá trị số
  - Địa chỉ nhà: dạng xâu các ký tự chữ-số.
- Mỗi kiểu dữ liệu có các thao tác xử lý và cách thức lưu trữ riêng
- 5 kiểu dữ liệu chuẩn trong python:
  - Kiểu Number
  - Kiểu String
  - Kiểu List
  - Kiểu Tuple
  - Kiểu Dictionary

3.1. Kiểu dữ liệu Number (DL số)

- Kiểu dữ liệu Number: lưu các giá trị số
- VD tạo các đối tượng Number thông qua phép gán

```
var1 = 1
var2 = 10
var3 = 15
```

- Để xóa các đối tượng Number vừa tạo, dùng lệnh del

```
var1 = 1
var2 = 10
var3 = 15

# del var1
# del var1, var2, var3

print(var1)
print(var2)
print(var3)
```

3.1. Kiểu dữ liệu Number

- Python có 3 kiểu dữ liệu Number:
  - **Kiểu int**: kiểu số nguyên có dấu (số có thể là số âm hoặc số dương)
  - **Kiểu float**: số thực với dấu thập phân. Kiểu float cũng có thể được viết ở dạng số mũ của 10 với E hoặc e như (2.5e2 = 2.5 x 10^2 = 250).
  - **Kiểu số phức**: là trong dạng a + bj, với a và b là số thực và J (hoặc j) biểu diễn căn bậc hai của -1. Phần thực là a và phần ảo là b

3.1. Kiểu dữ liệu Number

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-.9	9.322e-36j
0o20	32.3e+18	.876j
-0490	-90.	-.6545+0J
-0x260	-32.54e100	3e+26J
0x69	70.2E-12	4.53e-7j

Chuyển đổi kiểu dữ liệu số

- **int(x):** chuyển đổi số x thành số thuần nguyên
- **float(x):** chuyển đổi số x thành số thực
- **complex(x):** chuyển đổi số x thành số phức với phần thực là x và phần ảo là 0
- **complex(x, y):** chuyển đổi số x và y thành số phức với phần thực là x và phần ảo là y
- Python chuyển đổi kiểu dữ liệu tự động. Ví dụ:

```
f1 = 5.4
f2 = 5
f3 = f1 + f2
```

3.2. Kiểu dữ liệu String

- String: là một chuỗi liên tiếp các ký tự đặt trong dấu nháy đơn, nháy kép hoặc trích dẫn tam. Ví dụ:

```
var1 = 'Hello \nWorld!'
var2 = "Py'thon Programming"
var3 = '''Line1
Line2
Line3'''
var4 = """Line4
 Line5
 Line6"""
print(var1)
print(var2)
print(var3)
print(var4)
```

```
Hello
World!
Py'thon Programming
Line1
Line2
Line3
Line4
 Line5
 Line6
```

3.2. Kiểu dữ liệu String

- Thao tác cơ bản: lấy xâu con, nhân xâu, ghép xâu

```
str = 'Hello World!'

print (str)          # In ra toàn bộ xâu
print (str[0])       # Lấy ký tự đầu tiên của xâu và in ra
print (str[2:5])     # Lấy xâu con từ ký tự thứ 3 tới ký tự thứ 5 và in ra
print (str[2:])      # Lấy xâu con từ ký tự thứ 3 tới hết và in ra
print (str * 2)      # Nối một xâu với chính nó rồi in ra
print (str + "TEST") # Nối một xâu với một xâu khác và in ra
```

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

3.3. Kiểu dữ liệu List (danh sách)

- List
  - kiểu dữ liệu tổng hợp
  - chứa nhiều item (mục) ngăn nhau bằng dấu phẩy (,) và bao trong dấu ngoặc vuông []
  - Các item có thể có kiểu dữ liệu khác nhau

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print (list)           # Prints complete list
print (list[0])        # Prints first element of the list
print (list[1:3])      # Prints elements starting from 2nd till 3rd
print (list[2:])       # Prints elements starting from 3rd element
print (tinylist * 2)   # Prints list two times
print (list + tinylist) # Prints concatenated lists
```

```
['abcd', 786, 2.23, 'john', 70.2]
abcd
[786, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

3.4. Kiểu dữ liệu Tuple

- Tuple: tương tự như List, bao gồm các chuỗi giá trị ngăn nhau bằng dấu phẩy.
- Khác biệt: Tuple dùng dấu ngoặc tròn. Không thể thêm bớt phần tử, không thể thay đổi phần tử. Tuple có thể xem như read-only List

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')

print (tuple)            # Prints complete tuple
print (tuple[0])         # Prints first element of the tuple
print (tuple[1:3])       # Prints elements starting from 2nd till 3rd
print (tuple[2:])        # Prints elements starting from 3rd element
print (tinytuple * 2)    # Prints tuple two times
print (tuple + tinytuple) # Prints concatenated tuple
```

```
('abcd', 786, 2.23, 'john', 70.2)
abcd
(786, 2.23)
(2.23, 'john', 70.2)
(123, 'john', 123, 'john')
('abcd', 786, 2.23, 'john', 70.2, 123,
'john')
```

```
#!/usr/bin/python3


tuple = ( 'abcd', 786 , 2.23  )
list = [ 'abcd', 786 , 2.23 ]
tuple[2] = 1000    # Invalid syntax
list[2] = 1000     # Valid syntax
```

3.5. Kiểu dữ liệu Dictionary (từ điển)

- Bao gồm các cặp key-value (khóa-giá trị)

```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

print (dict['one'])       # Prints value for 'one' key
print (dict[2])           # Prints value for 2 key
print (tinydict)          # Prints complete dictionary
print (tinydict.keys())   # Prints all the keys
print (tinydict.values()) # Prints all the values
```

```
This is one
This is two
{'name': 'john', 'code': 6734, 'dept': 'sales'}
dict_keys(['name', 'code', 'dept'])
dict_values(['john', 6734, 'sales'])
```