KIỂU DỮ LIỆU VÀ TOÁN TỬ

NỘI DUNG

/01 Câu lệnh printf

/02 Python syntax và chú thích

/03 Biến và kiểu dữ liệu

/04 Toán tử

1. Câu lệnh printf:

Câu lệnh printf giúp in ra nội dung lên màn hình, nội dung có thể là xâu kí tự, các đối tượng bất kì trong Python. Trước khi được in ra màn hình, các đối tượng này được chuyển thành xâu kí tự.

1. Câu lệnh printf:

CÚ PHÁP

```
printf(object, sep = seperator, end = end)
```

Các tham số của hàm printf

Object: Các đối tượng trong Python.

sep: Phân cách giữa các đối tượng khi in, nếu không có tham số này thì mặc định sẽ là dấu cách.

end: Chỉ ra kí tự được in ở cuối của dòng, nếu không có tham số này thì mặc định sẽ là dấu xuống dòng.

1. Câu lệnh printf:

EXAMPLE

In ra xâu ký tự (được đặt trong nháy đơn hoặc nháy kép)

```
print("28tech !")
print('28tech !')
```

OUTPUT
28tech 28tech

EXAMPLE

In nhiều object

```
print("28tech", "programming")
print("28tech", "python", "java", "php")
```

OUTPUT
28tech programming 28tech python java php

1. Câu lệnh printf:

EXAMPLE

print có tham số sep

```
print("28tech", "programming", sep = '--')
print("28tech", "python", "java", "php", sep = '#')
```

OUTPUT

```
28tech--programming
28tech#python#java#php
```

EXAMPLE

print có tham số end

```
print('28tech', end ='')
print('python', end = '--')
```

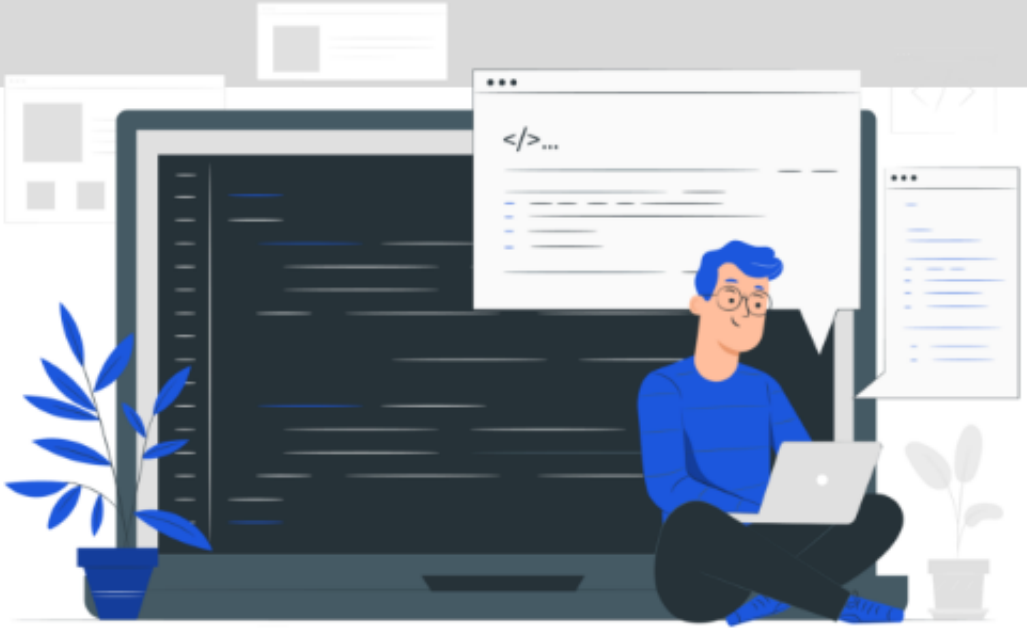
OUTPUT

```
28techpython--
```

2. Python syntax và chú thích:

Các câu lệnh trong ngôn ngữ lập trình Python không có kết thúc bằng dấu chấm phẩy, trong các ngôn ngữ lập trình khác thì thụt lề khi code giúp cho code dễ đọc thì trong Python thụt lề có nhiệm vụ rất quan trọng.

Python sử dụng thụt lề để chỉ ra một khối lệnh mới.



2. Python syntax và chú thích:

Cách chú thích trong Python

Sử dụng dấu # để chú thích trên 1 dòng

```
#This is a comment
print('28tech')
```

OUTPUT

28tech

Đưa nội dung nhiều dòng cần chú thích vào giữa 3 dấu nháy kép

```
"""
This is
a comment
"""
print('28tech')
```

OUTPUT

28tech

3. Biến và kiểu dữ liệu:

3.1 Biến:

Biến là vùng chứa để lưu trữ dữ liệu phục vụ bài toán.

Trong Python các bạn không cần phải khai báo biến cũng như chỉ rõ kiểu dữ liệu của biến đó, biến sẽ được tạo và xác định kiểu tự động (dynamic typing) khi bạn gán giá trị cho nó.

Để biết kiểu dữ liệu của biến các bạn có thể sử dụng hàm type().

EXAMPLE

```
a = 100
print(type(a))
s = "28tech"
print(type(s))
```

OUTPUT

```
<class 'int'>
<class 'str'>
```

3. Biến và kiểu dữ liệu:

3.1 Biến:

Chú ý khi sử dụng biến

- Không được đặt tên biến có chứa dấu cách, kí tự đặc biệt, bắt đầu bằng chữ số.
- Tên biến trong Python phân biệt hoa thường (case sensitive)

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

Kiểu dữ liệu số: Trong Python có 3 kiểu dữ liệu số: Số nguyên (Integer), Số thực dấu phẩy động (Floating-point numbers) và Số phức (Complex numbers)

EXAMPLE

```
a = 100
b = 5.2
c = 5 + 3j
print(type(a), type(b), type(c), sep = '\n')
```

OUTPUT

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

a) Số nguyên:

Đối với số nguyên thì trong Python không có giới hạn về giá trị mà số nguyên có thể lưu, bạn có thể xử lý số nguyên lớn với Python.


```
a = 12893812831231238182319238912389123819239128938123 8123
print(type(a))
print(a)
```

OUTPUT

```
<class 'int'>
12893812831231238182319238912389123819239128938123 8123
```

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

a) Số nguyên:

Trong Python thì các số nguyên thường được in ra dưới dạng cơ số 10 nhưng bạn cũng có thể in ra các số ở hệ 2, 8, 16.

Tiền tố (Prefix)	Ý nghĩa (Interpretation)	Cơ số (Base)
'0b' hoặc '0B'	Hệ nhị phân (Binary)	2
'0o' hoặc '0O'	Hệ bát phân (Octal)	8
'0x' hoặc '0X'	Hệ 16 (Hexadecimal)	16

EXAMPLE

```
a = 0b1101
print(a)
b = 0o123
print(b)
c = 0x22A
print(c)
```

OUTPUT

```
13
83
554
```

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

b) Số thực:

Số thực là số âm, dương kèm theo phần thập phân.

Trong Python giá trị số thực có thể lưu lớn nhất xấp xỉ 1.8*10^308, các giá trị lớn hơn giá trị này được mô tả bởi chuỗi inf (Infinity). Giá trị số thực nhỏ nhất có thể lưu là 5.0*10^-324, các giá trị nhỏ hơn số này được coi là 0.

EXAMPLE

```
a = 3.5
print(a)
b = 3e5
print(b)
d = 1.9e308
print(d)
e = 5.4e-325
print(e)
```

OUTPUT

```
3.5
3000000.0
inf
0.0
```

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

b) Số thực:

In số thực với số lượng chữ số sau dấu phẩy xác định.

EXAMPLE

```
a = 28.0412323
print('%.2f' % a)
print(round(a, 2))
print('{:.2f}'.format(a))
```

OUTPUT

```
28.04
```

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

c) Số phức:

Số phức gồm phần thực (real part) và phần ảo (imaginary part) đi kèm j.

Bạn có thể trích xuất phần thực, ảo của số phức x bằng x.real và x.imag.

EXAMPLE

```
a = 3 + 5j
print(a.real)
print(a.imag)
```

OUTPUT
3.0
5.0

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

d) Kiểu đúng sai:

Kiểu bool chỉ lưu 2 giá trị True hoặc False

EXAMPLE

```
a = True
b = False
print(type(a))
print(type(b))
```

OUTPUT

```
<class 'bool'>
<class 'bool'>
```

Chú ý: Các giá trị được xác định là True trong Python: xâu khác rỗng, các số khác 0.

EXAMPLE

```
print(bool(100))
print(bool(0))
print(bool('28tech'))
print(bool(''))
```

OUTPUT

```
True
False
True
False
```

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

e) Kiểu xâu kí tự (str):

Xâu kí tự trong Python được đặt trong nháy đơn hoặc nháy kép trên 1 dòng, trong trường hợp xâu có nhiều dòng ta đặt giữa 3 nháy kép.

EXAMPLE

```
s = '28tech'
t = """python 28tech
programming"""
print(s)
print(t)
```

OUTPUT

```
28tech
python 28tech
programming
```

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

f) Ép kiểu:

Khi bạn muốn chỉ định kiểu cho một biến nào đó bạn có thể ép kiểu cho nó. Quá trình casting trong Python được hoàn thành bằng cách sử dụng constructor.

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

f) Ép kiểu:

int() : Ép kiểu sang số nguyên

EXAMPLE

```
a = int(123)
b = int('123')
c = int(123.222)
print(a, b, c)
```

OUTPUT

123 123 123

EXAMPLE

```
a = int('1234aaa')
print(a)
```

OUTPUT

ValueError: invalid literal for int() with base 10: '1234aaa'

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

f) Ép kiểu:

float() : Ép kiểu sang số float

EXAMPLE

```
a = float(50)
b = float('50.22')
print(a, b )
```

OUTPUT

```
50.0 50.22
```

EXAMPLE

```
a = float('50.22a')
print(a)
```

OUTPUT

```
ValueError: could not convert string
to float: '50.22a'
```

3. Biến và kiểu dữ liệu:

3.2 Kiểu dữ liệu:

f) Ép kiểu:

str() : Ép kiểu sang string

EXAMPLE

```
a = str(230)
b = str(3234.44)
c = str(True)
print(a, b, c)
```

OUTPUT

```
230 3234.44 True
```

4. Toán tử:

Các loại toán tử trong Python

Toán tử gán	Assignment operator
Toán tử toán học	Arithmetic operator
Toán tử so sánh	Comparision operator
Toán tử logic	Logical operator
Toán tử nhận dạng	Identity operator
Toán tử thành viên	Membership operator
Toán tử bit	Bitwise operator

4. Toán tử:

a) Toán tử gán:

CÚ PHÁP

a = b

Ý nghĩa: Gán giá trị của b cho a

EXAMPLE

```
a = 100
b = a
print(b)
```

OUTPUT

100

EXAMPLE

Gán giá trị cho nhiều biến trong cùng 1 câu lệnh

```
a, b, c = 100, 200, 300
print(a, b, c)
```

OUTPUT

100 200 300

EXAMPLE

Hoán vị giá trị của 2 biến trong Python

```
a = '28tech'
b = 'python'
a, b = b, a
print(a, b)
```

OUTPUT

python 28tech

4. Toán tử:

b) Toán tử toán học:

Toán tử	Ý nghĩa	Ví dụ	Kết quả
+	Cộng	a = 100 + 200	300
-	Trừ	a = 100 - 200	-100
*	Nhân	a = 100 * 200	20000
/	Chia thập phân	a = 100 / 200	0.5
//	Chia nguyên	a = 300 // 200	1
%	Chia dư	a = 15 % 2	1
**	Lũy thừa	a = 2**10	1024

4. Toán tử:

c) Toán tử so sánh:

Khi bạn sử dụng toán tử so sánh thì kết quả của phép so sánh sẽ trả về True hoặc False

Toán tử	Ý nghĩa	Ví dụ	Kết quả
==	So sánh bằng	100 == 100	True
>	Lớn hơn	200 > 300	False
>=	Lớn hơn hoặc bằng	200 >= 100	True
<	Nhỏ hơn	100 < 50	False
<=	Nhỏ hơn hoặc bằng	100 <= 200	True
!=	Khác	50 != 20	True

4. Toán tử:

d) Toán tử logic:

Bạn có thể sử dụng các toán tử and, or, not để kết hợp nhiều biểu thức so sánh.

Toán tử	Ý nghĩa	Ví dụ	Kết quả
and	Toán tử và	(20 == 20) and (1 <0)	False
or	Toán tử hoặc	(10 < 20) or (20 == 50)	True
not	Toán tử phủ định	not(20 == 20)	False

4. Toán tử:

d) Toán tử logic:

CHÚ Ý

- Đối với cổng and: Trả về True nếu các mệnh đề thành phần đều là True, False trong các trường hợp còn lại.
- Đối với cổng or: Trả về True nếu 1 trong các mệnh đề thành phần là True, False khi mọi mệnh đề thành phần đều False.

4. Toán tử:

e) Toán tử nhận dạng:

Toán tử nhận dạng được sử dụng để so sánh 2 đối tượng chứ không phải so sánh 2 giá trị.

Toán tử	Ý nghĩa
is	Trả về True nếu 2 đối tượng là giống nhau
is not	Trả về True nếu 2 đối tượng là khác nhau

4. Toán tử:

e) Toán tử nhận dạng:

Toán tử thành viên được dùng để kiểm tra sự tồn tại của một đối tượng trong list, xâu, tuple...

Toán tử	Ý nghĩa	Ví dụ	Kết quả
in	Trả về true nếu đối tượng kiểm tra tồn tại	'a' in 'abcd'	True
not in	Trả về true nếu đối tượng kiểm tra không tồn tại	'a' not in '28tech'	True