TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

CHƯƠNG 2

NGÔN NGỮ LẬP TRÌNH PYTHON

Nghệ An, 2022

Chương 2:
NGÔN NGỮ LẬP TRÌNH PYTHON

NỘI DUNG GIẢNG DẠY:

2.1. Một số đặc điểm của ngôn ngữ lập trình Python

2.2. Môi trường lập trình và thực thi Python

2.3. Cú pháp cơ bản của Python

2.4. Biến và các toán tử trong Python

2.5. Các cấu trúc điều khiển

Chương 2:
NGÔN NGỮ LẬP TRÌNH PYTHON

NỘI DUNG GIẢNG DẠY:

2.1. Một số đặc điểm của ngôn ngữ lập trình Python

2.2. Môi trường lập trình và thực thi Python

2.3. Cú pháp cơ bản của Python

2.4. Biến và các toán tử trong Python

2.5. Các cấu trúc điều khiển

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON

Python là gì?

Python là một ngôn ngữ lập trình thông dịch (interpreted), hướng đối tượng (object-oriented), và là một ngôn ngữ bậc cao (high-level) ngữ nghĩa động (dynamic semantics).

Python hỗ trợ các module và gói (packages), khuyến khích chương trình module hóa và tái sử dụng mã.

Trình thông dịch Python và thư viện chuẩn mở rộng có sẵn dưới dạng mã nguồn hoặc dạng nhị phân miễn phí cho tất cả các nền tảng chính và có thể được phân phối tự do.

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON

Các đặc điểm của Python:

- Ngữ pháp đơn giản, dễ đọc.
- Vừa hướng thủ tục (procedural-oriented), vừa hướng đối tượng (object-oriented)
- Hỗ trợ module và hỗ trợ gói (package)
- Xử lý lỗi bằng ngoại lệ (Exception)
- Kiểu dữ liệu động ở mức cao.
- Có các bộ thư viện chuẩn và các module ngoài, đáp ứng tất cả các nhu cầu lập trình.

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON

- Có khả năng tương tác với các module khác viết trên C/C++ (Hoặc Java cho Jython, hoặc .Net cho IronPython).

- Có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface).

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON

Lịch sử của Python:

Python đã được hình thành vào cuối những năm 1980, và việc thực hiện nó vào tháng 12 năm 1989 bởi Guido van Rossum tại Centrum Wiskunde & Informatica (CWI) ở Hà Lan như là một kế thừa cho ngôn ngữ ABC (tự lấy cảm hứng từ SETL) có khả năng xử lý ngoại lệ và giao tiếp với hệ điều hành Amoeba.

Van Rossum là tác giả chính của Python, và vai trò trung tâm của ông trong việc quyết định hướng phát triển của Python.

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON

Về nguồn gốc của Python, **Van Rossum** đã viết vào năm 1996:

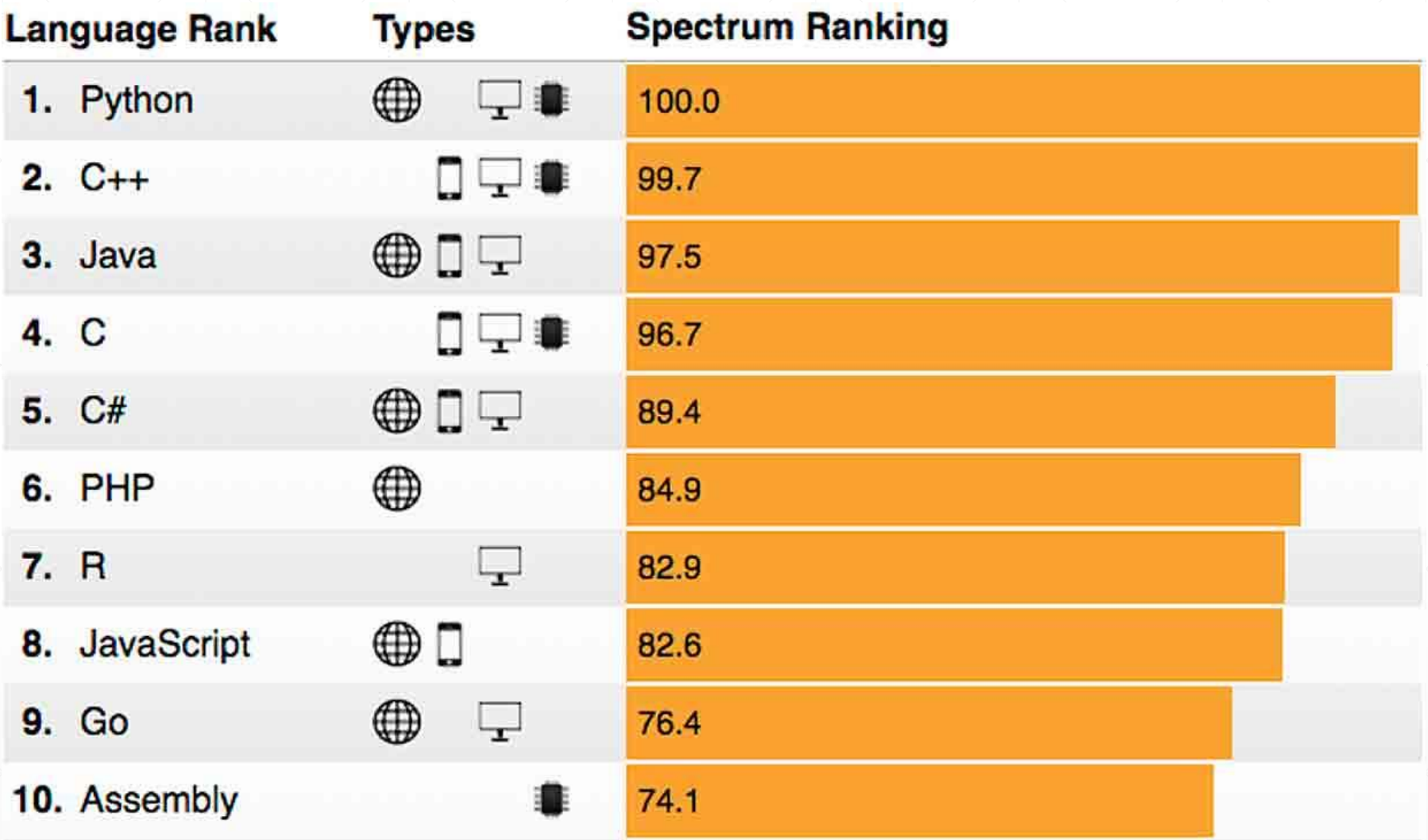
Hơn sáu năm trước, vào tháng 12 năm 1989, tôi đã tìm kiếm một dự án lập trình "sở thích" mà nó đã chiếm đóng tâm trí tôi trong suốt tuần lễ Giáng sinh. Văn phòng của tôi ... sẽ đóng cửa, nhưng tôi đã có một máy tính ở nhà, và không có nhiều thứ khác trên tay. Tôi quyết định viết một bộ thông dịch (interprester) cho ngôn ngữ kịch bản mới mà tôi đã từng nghĩ đến: một hậu duệ của ABC có thể hấp dẫn các hacker Unix/C. Tôi đã chọn Python như là một tiêu đề làm việc cho dự án.

ĐẶC ĐIỂM CỦA NGÔN NGỮ LẬP TRÌNH PYTHON

- Python 2.0 đã được phát hành vào ngày 16 tháng 10 năm 2000 và có nhiều tính năng mới, bao gồm bộ thu gom rác theo chu kỳ (cycle-detecting garbage) và hỗ trợ Unicode. Với việc phát hành này quá trình phát triển đã được thay đổi và trở nên minh bạch hơn và cộng đồng hậu thuẫn

- Python 3.0 được phát hành năm 2008, sau một thời gian dài thử nghiệm.

- Cho tới năm 2018, Python đang có phiên bản 3.7.



The 2018 Top Programming Languages

Chương 2:
NGÔN NGỮ LẬP TRÌNH PYTHON

NỘI DUNG GIẢNG DẠY:

2.1. Một số đặc điểm của ngôn ngữ lập trình Python

2.2. Môi trường lập trình và thực thi Python

2.3. Cú pháp cơ bản của Python

2.4. Biến và các toán tử trong Python

2.5. Các cấu trúc điều khiển

MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

Python là một ngôn ngữ lập trình dạng thông dịch, do đó có ưu điểm tiết kiệm thời gian phát triển ứng dụng vì không cần phải thực hiện biên dịch và liên kết.

Trình thông dịch có thể được sử dụng để chạy file script, hoặc cũng có thể được sử dụng theo cách tương tác.

Ở chế độ tương tác, ta có thể nhập vào từng biểu thức rồi gõ Enter, và kết quả thực thi sẽ được hiển thị ngay lập tức. Đặc điểm này rất hữu ích cho người mới học, giúp họ nghiên cứu tính năng của ngôn ngữ; hoặc để các lập trình viên chạy thử mã lệnh trong suốt quá trình phát triển phần mềm.

MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

Tùy vào hệ điều hành, việc cài đặt Python có những sự khác nhau tương đối:

- Người dùng Windows có thể tự tải về và chạy file cài đặt trên máy tính với một vài bước cài đặt đơn giản.

- Người dùng Linux và MacOS X có thể dùng các bộ Python đã được cài đặt sẵn theo hệ điều hành (Python là thành phần có sẵn trong Linux và MacOS X).

Trên mỗi nền tảng có một dạng khác nhau của Python. Từ máy tính, tới điện thoai, web, game… tuy nhiên chúng vẫn có những đặc điểm phổ biến giống nhau về mặt cơ bản.

MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

Quá trình thực thi của Python

Tùy thuộc vào chiều hướng bạn nhìn vào mà Python được chia làm hai dạng: từ phía người lập trình và từ phía trình thông dịch Python.

- Từ phía người lập trình:

Ở dạng đơn giản, Python chỉ là 1 file văn bản với nhưng câu lệnh Python được viết bằng bất cứ trình soan thảo nào, miễn là đúng cú pháp quy định và lưu dưới dạng *.py.

Việc chạy cũng vô cùng đơn giản, chỉ đơn giản là thực thi toàn bộ các câu lệnh trong file *.py lần lượt từ trên xuống dưới.

MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

- **Từ phía trình biên dịch Python**: dưới góc nhìn của trình biên dịch, quá trình thực thi file Python trải qua ba bước:

B1 - Viết code Python: phần này đã giải thích ở trước.

B2 - Biên dịch sang bytecode: các câu lệnh sẽ được Python biên dịch sang một dạng thấp hơn đó là dạng bytecode. Thông thường thì quá trình này được thực thi nội tại và không thể thấy (trong một số trường hợp, có thể thấy một file với phần mở rộng là *.pyc).

Để tăng tốc quá trình làm việc, Python sẽ lưu cái file bytecod này và ở lần thực thi tiếp theo, nó sẽ chạy trực tiếp cái file này thay vì phải biên dịch lại.

MÔI TRƯỜNG LẬP TRÌNH VÀ THỰC THI PYTHON

B3 - Máy ảo Python:

Sau khi biên dịch ra bytecode, các file sẽ được thực thi bởi một máy ảo Python (Python Vitual Machine).

Máy ảo Python đơn giản là một vòng lặp lớn mà ở đó bytecode sẽ được thực thi lần lượt. Đây mới là phần thực sự thi hành các lệnh của chương trình. Về mặt kĩ thuật thì đây là bước cuối cùng trong quá trình thông dịch Python.

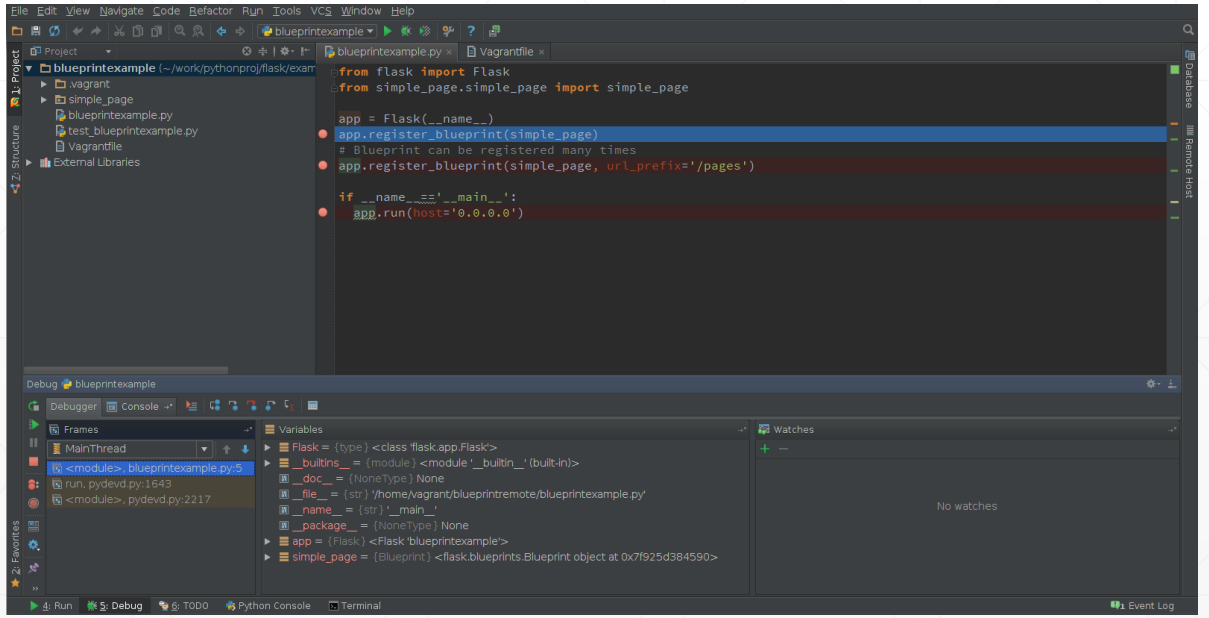
GIỚI THIỆU PYCHARM IDE

PyCharm được phát triển bởi Jet Brains, cung cấp cho người dùng bản Community miễn phí, dùng thử 30 ngày cho phiên bản chuyên nghiệp, $213 – $690 phí đăng ký hằng năm.

Khả năng hỗ trợ code toàn diện và phân tích làm cho PyCharm là IDE tốt nhất cho các lập trình Python tất cả các cấp độ.

PyCharm cũng hỗ trợ các ngôn ngữ khác và hoạt động trên nhiều nền tảng, vì vậy thực tế bất cứ ai cũng có thể sử dụng nó.

GIỚI THIỆU PYCHARM IDE



Chương 2:
NGÔN NGỮ LẬP TRÌNH PYTHON

NỘI DUNG GIẢNG DẠY:

2.1. Một số đặc điểm của ngôn ngữ lập trình Python

2.2. Môi trường lập trình và thực thi Python

2.3. Cú pháp cơ bản của Python

2.4. Biến và các toán tử trong Python

2.5. Các cấu trúc điều khiển

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Định danh (identifier) trong Python**

- Một định danh (identifier) trong Python là một tên được sử dụng để nhận diện một biến, một hàm, một lớp, hoặc một đối tượng.

- Một định danh bắt đầu với một chữ cái từ A tới Z hoặc từ a tới z hoặc một dấu gạch dưới (_) được theo sau bởi không hoặc nhiều ký tự, dấu gạch dưới hoặc các chữ số (từ 0 tới 9).

- Python không hỗ trợ các punctuation char chẳng hạn như @, $ và % bên trong các định danh.

- Python là ngôn ngữ lập trình phân biệt chữ hoa - chữ thường.

CÚ PHÁP CƠ BẢN CỦA PYTHON

Một số qui tắc nên được sử dụng trong khi đặt tên các định danh:

- Một định danh là một dãy ký tự hoặc chữ số.

- Không có ký tự đặc biệt nào được sử dụng (ngoại trừ dấu gạch dưới) như một định danh.

- Ký tự đầu tiên có thể là chữ cái, dấu gạch dưới, nhưng không được sử dụng chữ số làm ký tự đầu tiên.

- Từ khóa không nên được sử dụng như là một tên định danh *(phần tiếp theo sẽ trình bày về các từ khóa này)*.

CÚ PHÁP CƠ BẢN CỦA PYTHON

- Tên lớp bắt đầu với một chữ cái hoa. Tất cả định danh khác bắt đầu với một chữ cái thường.

- Bắt đầu một định danh với một dấu gạch dưới đơn chỉ rằng định danh đó là private.

- Bắt đầu một định danh với hai dấu gạch dưới chỉ rằng định danh đó thực sự là private.

- Nếu định danh cũng kết thúc với hai dấu gạch dưới, thì định danh này là một tên đặc biệt được định nghĩa bởi ngôn ngữ (ví dụ như __init__ chẳng hạn).

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các từ khóa trong Python:**

Đây là các từ dành riêng và không thể sử dụng chúng như là các hằng, biến hoặc cho bất kỳ tên định danh nào.

Tất cả từ khóa trong Python là chỉ ở dạng chữ thường.

and	exec	not
assert	finally	or
break	for	pass
class	from	print

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các từ khóa trong Python:**

continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Dòng lệnh và độ thụt dòng lệnh trong Python**

- Python không cung cấp các dấu ngoặc ôm ({}) để chỉ các khối code cho định nghĩa lớp hoặc hàm hoặc điều khiển luồng.

- Các khối code được nhận biết bởi độ thụt dòng code (indentation) trong Python và đây là điều bắt buộc.

- Số khoảng trống trong độ thụt dòng là biến đổi, nhưng tất cả các lệnh bên trong khối phải được thụt cùng một số lượng khoảng trống như nhau.

CÚ PHÁP CƠ BẢN CỦA PYTHON

- Trong Python thì tất cả các dòng liên tiếp nhau mà được thụt đầu dòng với cùng lượng khoảng trống như nhau sẽ tạo nên một khối.

Ví dụ:

```
if True:
    print "True"
else:
    print "False"
```

Khối sau sẽ tạo ra một lỗi:

```
if True:
    print "Answer‟
    print "True"
else:
    print "Answer"
  print "False"
```

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các lệnh trên nhiều dòng trong Python**

Các lệnh trong Python có một nét đặc trưng là kết thúc với một newline (dòng mới). Tuy nhiên, Python cho phép sử dụng ký tự \ để chỉ rõ sự liên tục dòng.

Ví dụ:

```
total = item_one + \
        item_two + \
        item_three
```

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Trích dẫn trong Python**

Python chấp nhận trích dẫn đơn ('), kép (") và trích dẫn tam (''' hoặc """) để biểu thị các hằng chuỗi, miễn là các trích dẫn này có cùng kiểu mở và đóng.

Trích dẫn tam được sử dụng để trải rộng chuỗi được trích dẫn qua nhiều dòng.

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Comment trong Python**

Python hỗ trợ hai kiểu comment đó là comment đơn dòng và đa dòng.

Trong Python, một dấu #, mà không ở bên trong một hằng chuỗi nào, bắt đầu một comment đơn dòng. Tất cả ký tự ở sau dấu # và kéo dài cho đến hết dòng đó thì được coi là một comment và được bỏ qua bởi trình thông dịch.

```
# First comment
print "Hello, Python!" # second comment
```

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Comment trong Python**

Python cũng hỗ trợ kiểu comment thứ hai, đó là kiểu comment đa dòng được cho bên trong các trích dẫn tam.

```
#single line comment
print "Hello Python"
"""This is
multiline comment"""
```

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Sử dụng dòng trống trong Python**

- Một dòng mà chỉ chứa các khoảng trống trắng whitespace, có thể với một comment, thì được xem như là một dòng trống và Python hoàn toàn bỏ qua nó.

- Trong một phiên thông dịch trong chế độ tương tác, bạn phải nhập một dòng trống để kết thúc một lệnh đa dòng.

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các lệnh đa dòng trên một dòng đơn trong Python**

- Dấu chấm phảy (;) cho phép xuất hiện nhiều lệnh trên một dòng đơn.

- Tất cả các lệnh được cung cấp này không bắt đầu một khối code mới.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các nhóm lệnh đa dòng trong Python**

- Một nhóm các lệnh đơn, mà tạo một khối code đơn, **được gọi là suite** trong Python.

- Các lệnh phức hợp như if, while, def, và class cần một dòng header và một suite.

- Các dòng header bắt đầu lệnh (với từ khóa) và kết thúc với một dầu hai chấm (:) và được theo sau bởi một hoặc nhiều dòng để tạo nên một suite.

CÚ PHÁP CƠ BẢN CỦA PYTHON

- **Các nhóm lệnh đa dòng trong Python**

Ví dụ:

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

THẢO LUẬN NHÓM

NỘI DUNG:

Đặc điểm của ngôn ngữ lập trình Python:

1. Phân biệt về ngôn ngữ lập trình biên dịch và ngôn ngữ lập trình thông dịch.

2. So sánh cú pháp của ngôn ngữ lập trình Python với một số ngôn ngữ lập trình thông dụng khác như: C/C++; Java; Pascal.

BÀI TẬP

NỘI DUNG:

1. Cài đặt môi trường lập trình Python trên máy tính cá nhân
2. Viết chương trình hiện thị các nội dung:

“Hello, world!”;

“Họ và tên, Mã số sinh viên, ngành học”

Kết quả code báo cáo trên Github.

BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 2.4; và 2.5.

2. Tìm hiểu về khái niệm biến và toán tử, các loại toán tử trong Python.

3. Nghiên cứu về câu lệnh và các cấu trúc điều khiển trong ngôn ngữ lập trình Python.

Chương 2:
NGÔN NGỮ LẬP TRÌNH PYTHON

NỘI DUNG GIẢNG DẠY:

2.1. Một số đặc điểm của ngôn ngữ lập trình Python

2.2. Môi trường lập trình và thực thi Python

2.3. Cú pháp cơ bản của Python

2.4. Biến và các toán tử trong Python

2.5. Các cấu trúc điều khiển

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Biến**

- Biến là các vị trị bộ nhớ được dành riêng để lưu trữ dữ liệu. Một khi một biến đã được lưu trữ, nghĩa là một khoảng không gian đã được cấp phát trong bộ nhớ đó.

- Dựa trên kiểu dữ liệu của một biến, trình thông dịch cấp phát bộ nhớ và quyết định những gì có thể được lưu trữ trong khu nhớ dành riêng đó.

- Bằng việc gán các kiểu dữ liệu khác nhau cho các biến, chúng ta có thể lưu trữ số nguyên, thập phân hoặc ký tự trong các biến này.

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Gán các giá trị cho biến trong Python**

- Trong Python, chúng ta không cần khai báo biến một cách tường minh.

- Khi gán bất cứ giá trị nào cho biến thì biến đó được khai báo một cách tự động. Phép gán được thực hiện bởi toán tử =.

- Toán hạng trái của toán tử = là tên biến và toán hạng phải là giá trị được lưu trữ trong biến.

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Gán các giá trị cho biến trong Python**

Ví dụ:

```
a = 20              # Một phép gán số nguyên
b   = 100.0         # Một số thực
ten    = "Hoang"    # Một chuỗi
```

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Phép đa gán (multiple assignment) trong Python**

- Python cho phép bạn gán một giá trị đơn cho một số biến một cách đồng thời. Python hỗ trợ hai kiểu đa gán sau:

- Gán giá trị đơn cho nhiều biến.

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

- Gán nhiều giá trị cho nhiều biến

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

Trong trường hợp này, các giá trị sẽ được gán theo thứ tự mà các biến xuất hiện.

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Các kiểu dữ liệu chuẩn trong Python**

- Dữ liệu mà được lưu trữ trong bộ nhớ có thể có nhiều kiểu khác nhau.

Ví dụ: Lương của công nhân đươc lưu trữ dưới dạng một giá trị số còn địa chỉ của họ được lưu trữ dưới dạng các ký tự chữ - số.

Python có nhiều kiểu dữ liệu chuẩn được sử dụng để xác định các hành động có thể xảy ra trên chúng và phương thức lưu trữ cho mỗi kiểu.

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Các kiểu dữ liệu chuẩn trong Python**

Python có 5 kiểu dữ liệu chuẩn là:

- Kiểu Number
- Kiểu String
- Kiểu List
- Kiểu Tuple
- Kiểu Dictionary

Ngoài kiểu Number và kiểu String mà có thể đã được làm quen với các ngôn ngữ lập trình khác thì ở trong Python còn xuất hiện thêm ba kiểu dữ liệu đó là List, Tuple và Dictionary.

Chúng ta sẽ tìm hiểu chi tiết từng kiểu dữ liệu trong một chương riêng.

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Chuyển đổi kiểu trong Python**

- Khi cần thực hiện một số phép chuyển đổi kiểu để thỏa mãn hàm hoặc phương thức nào đó, … Để thực hiện điều này, chúng ta sử dụng tên kiểu như là một hàm.

- Các hàm này trả về một đối tượng mới biểu diễn giá trị đã được chuyển đổi.

Hàm	Miêu tả
int(x [,base])	Chuyển đổi x thành một số nguyên. Tham số base xác định cơ sở nếu x là một chuỗi

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

Hàm	Miêu tả
long(x [,base])	Chuyển đổi x thành một long int. Tham số base xác định cơ sở nếu x là một chuỗi
float(x)	Chuyển đổi x thành một số thực
complex(real [,imag])	Chuyển đổi x thành một số phức
str(x)	Chuyển đổi x thành một chuỗi

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Toán tử trong Python**

- Toán tử là các biểu tượng cụ thể mà thực hiện một số hoạt động trên một số giá trị và cho ra một kết quả.

Ví dụ:

Biểu thức 2 + 3 = 5, thì 2 và 3 được gọi là các toán hạng và dấu + được gọi là toán tử.

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Các loại toán tử trong Python**

Python hỗ trợ các loại toán tử sau:

- Toán tử số học: // , + , - , * , / , % , **;
- Toán tử quan hệ (toán tử so sánh): < , > , <= , >= , == , != , <>;
- Toán tử gán: = , /= , += , -= , *= , %= , **= , //=;
- Toán tử logic: and , or , not;
- Toán tử membership: in , not in;
- Toán tử identify: is , is not;
- Toán tử thao tác bit: & , | , ^ , ~ , << , >>;

BIẾN VÀ CÁC TOÁN TỬ TRONG PYTHON

- **Các loại toán tử trong Python**

Python hỗ trợ các loại toán tử sau:

- Toán tử số học: // , + , - , * , / , % , **;
- Toán tử quan hệ (toán tử so sánh): < , > , <= , <= , == , != , <>;
- Toán tử gán: = , /= , += , -= , *= , %= , **= , //=;
- Toán tử logic: and , or , not;
- Toán tử membership: in , not in;
- Toán tử identify: is , is not;
- Toán tử thao tác bit: & , | , ^ , ~ , << , >>;

THẢO LUẬN NHÓM

NỘI DUNG:

Xác định thứ tự ưu tiên của các toán tử trong Python để mang lại kết quả như mong muốn trong quá trình làm việc.

Chương 2:
NGÔN NGỮ LẬP TRÌNH PYTHON

NỘI DUNG GIẢNG DẠY:

2.1. Một số đặc điểm của ngôn ngữ lập trình Python

2.2. Môi trường lập trình và thực thi Python

2.3. Cú pháp cơ bản của Python

2.4. Biến và các toán tử trong Python

2.5. Các cấu trúc điều khiển

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if**

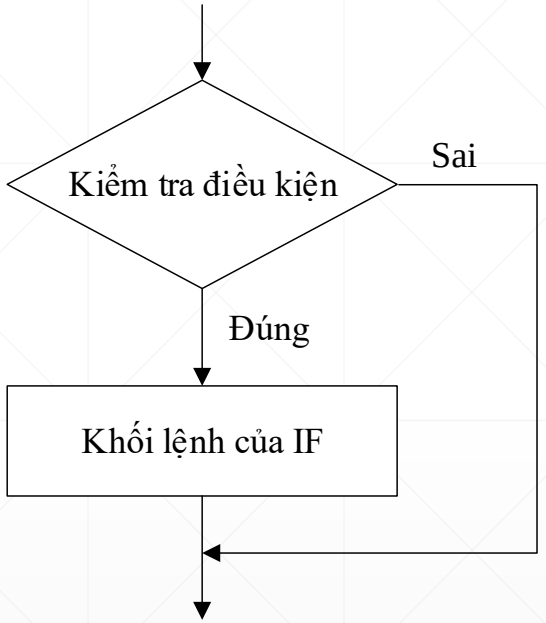
```
if điều kiện:
    khối lệnh
```

- Đánh giá điều kiện và sẽ thực hiện các lệnh khi điều kiện là True. Nếu điều kiện False thì lệnh sẽ không được thực hiện.

- Trong Python, khối lệnh của lệnh if được viết thụt lề vào trong. Khối lệnh của if bắt đầu với một khoảng thụt lề và dòng không thụt lề đầu tiên sẽ được hiểu là kết thúc lệnh if.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Cấu trúc lệnh if**




```
num = 3
if num > 0:
    print(num, "là số dương.")
```

```
num = -1
if num > 0:
    print(num, "là số dương.")
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if...else**

```
if điều kiện:
    Khối lệnh của if
else:
    Khối lệnh của else
```

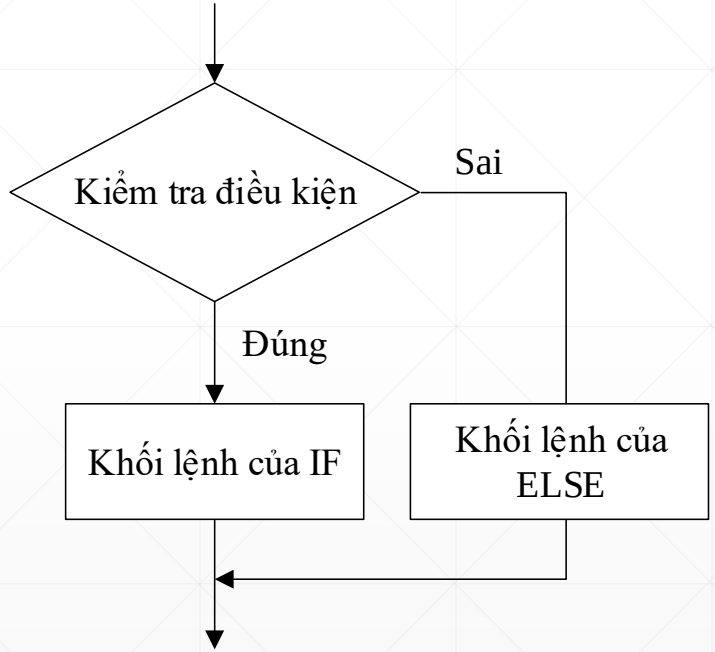
- Lệnh if...else kiểm tra điều kiện và thực thi khối lệnh if nếu điều kiện đúng. Nếu điều kiện sai, khối lệnh của else sẽ được thực hiện.

- Thụt đầu dòng được sử dụng để tách các khối lệnh.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Cấu trúc if...else**

```
if num >= 0:
    print("So duong hoac bang 0")
else:
    print("So am")
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if...elif...else**

```
if điều kiện:
    Khối lệnh của if
elif test expression:
    Khối lệnh của elif
else:
    Khối lệnh của else
```

Lệnh if ... elif ... elif ... là sự thay thế cho câu lệnh switch hay case trong các ngôn ngữ lập trình khác.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if...elif...else**

- elif là viết gọn của else if, nó cho phép chúng ta kiểm tra nhiều điều kiện.

- Nếu điều kiện là sai, nó sẽ kiểm tra điều kiện của khối elif tiếp theo và cứ như vậy cho đến hết.

- Nếu tất cả các điều kiện đều sai nó sẽ thực thi khối lệnh của else.

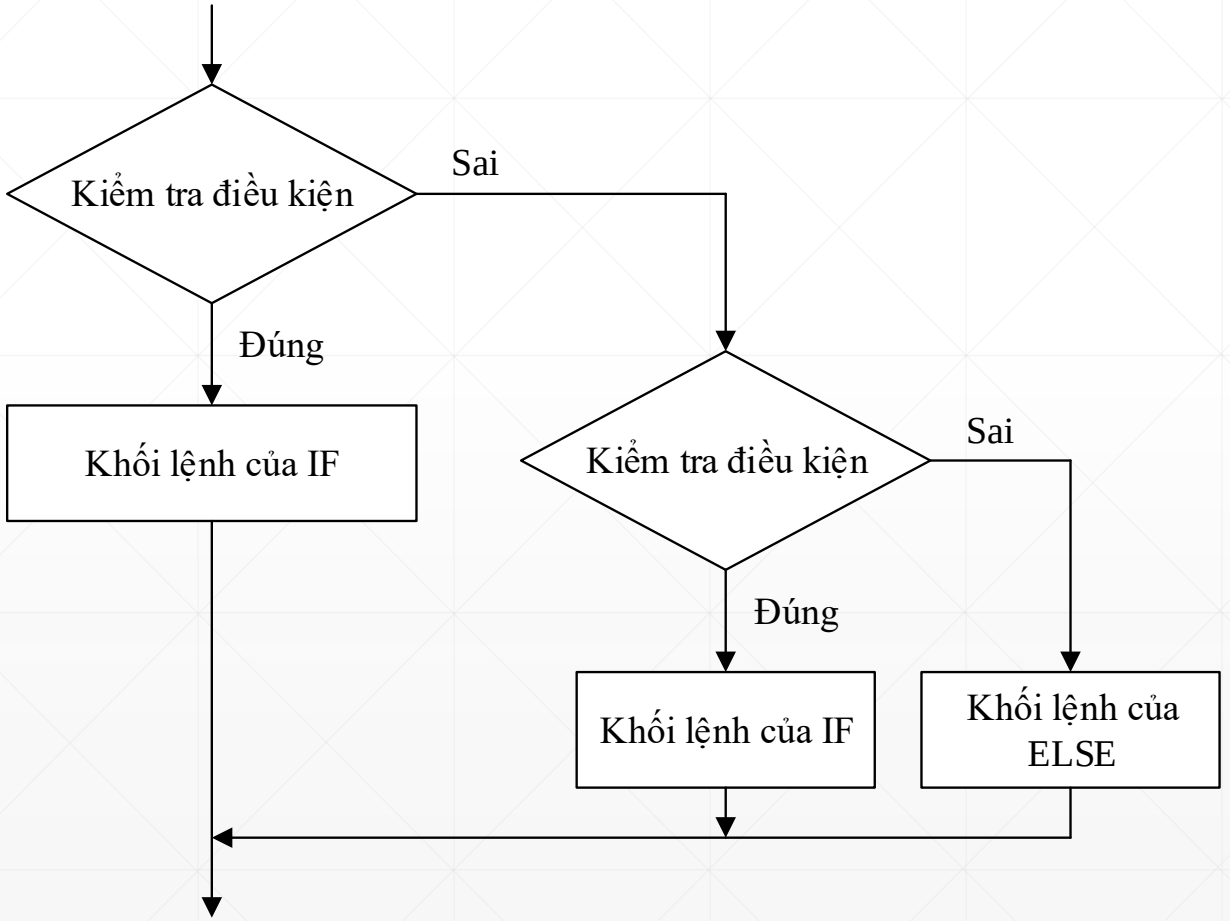
- Chỉ một khối lệnh trong if...elif...else được thực hiện theo điều kiện.

- Có thể không có hoặc có nhiều elif, phần else là tùy chọn.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if...elif...else**

```
x = int(input("Nhap mot so: "))
if x < 0:
    print('So am')
elif x == 0:
    print('So 0')
elif x == 1:
    print('So 1')
else:
    print('So duong')
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if lồng nhau trong Python**

- Có thể viết lệnh if...elif...else trong một khối lệnh if...elif...else khác, và tạo thành lệnh if lồng nhau.

- Không giới hạn số lệnh được lồng vào lệnh khác.

- Thụt đầu dòng là cách duy nhất để nhận diện mức độ lồng, do đó nó có thể gây rối, nhầm lẫn, nên hạn chế sử dụng nếu có thể.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh if lồng nhau trong Python**

```
num = float(input("Nhập một số: "))
if num >= 0:
    if num == 0:
        print("Số Không")
    else:
        print("Số dương")
else:
    print("Số âm")
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Vòng lặp for trong Python**

```
for bien_lap in chuoi_lap:
    Khối lệnh của for
```

- chuoi_lap là chuỗi cần lặp, bien_lap là biến nhận giá trị của từng mục bên trong chuoi_lap trên mỗi lần lặp.

- Vòng lặp sẽ tiếp tục cho đến khi nó lặp tới mục cuối cùng trong chuỗi.

Khối lệnh của for được thụt lề để phân biệt với phần còn lại của code.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Vòng lặp for trong Python**

```
for chu in `kythuatlaptrinh':
 print('Chữ cái hiện tại:', chu)
#Lặp từ trong chuỗi
chuoi = ['bố','mẹ','em']
for tu in chuoi:
    print(`Dai hoc Vinh', tu)
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Kết hợp for với else**

Khối lệnh của else sẽ được thực thi khi các mục trong chuỗi đã được lặp hết.

```
B = [0, 2, 4, 5]
for b in B:
    print(b)
else:
    print("Đã in hết số.")
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Kết hợp for với else**

Lệnh break có thể được sử dụng để dừng vòng lặp for, lúc này phần else sẽ bị bỏ qua. Hay nói cách khác, phần else trong for sẽ chạy khi không có break nào được thực thi.

- **Hàm range()**

Có thể sử dụng hàm range() để tạo ra một dãy số. Ví dụ, range(100) sẽ tạo một dãy số từ 0 đến 99 (100 số).

Hàm range(số bắt đầu, số kết thúc, khoảng cách giữa hai số) được sử dụng để tạo dãy số tùy chỉnh. Nếu không đặt khoảng cách giữa hai số thì Python sẽ hiểu mặc định nó bằng 1.

BÀI TẬP

NỘI DUNG:

Viết chương trình tìm các số nguyên tố trong khoảng từ 0 đến 100 sử dụng vòng lặp for.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Vòng lặp while trong Python**

```
while điều_kiện_kiểm_tra:
    Khối lệnh của while
```

- Trong vòng lặp while, điều_kiện_kiểm_tra sẽ được kiểm tra đầu tiên.

- Khối lệnh của vòng lặp chỉ được nạp vào nếu điều_kiện_kiểm_tra là True.

- Sau một lần lặp, điều_kiện_kiểm_tra sẽ được kiểm tra lại.

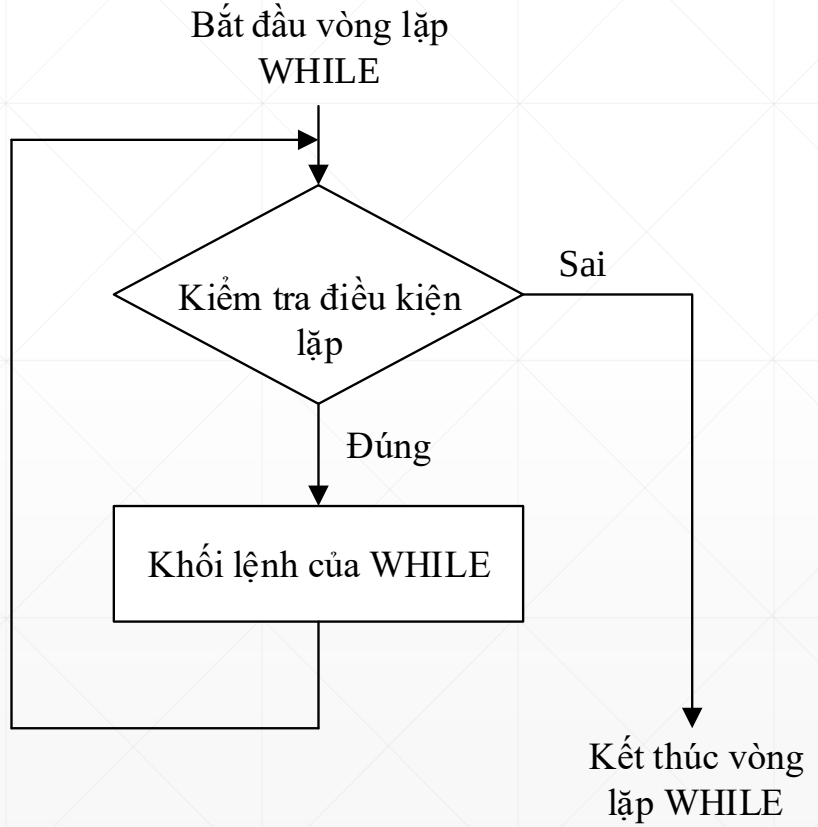
- Quá trình này tiếp tục cho đến khi điều_kiện_kiểm_tra là False.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Vòng lặp while trong Python**

Giống như if hay vòng lặp for, khối lệnh của while cũng được xác định thông qua thụt lề.

Khối lệnh bắt đầu với thụt lề đầu tiên và kết thúc với dòng không thụt lề đầu tiên liền sau khối.



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Vòng lặp while trong Python**

Trong Python mọi giá trị khác 0 đều là True, None và 0 được hiểu là False.

Đặc điểm này của while có thể dẫn đến trường hợp là while có thể không chạy vì ngay lần lặp đầu tiên điều_kiện_kiểm_tra đã False.

→ Khi đó, khối lệnh của while sẽ bị bỏ qua và phần code ngay sau đó sẽ được thực thi.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Vòng lặp while trong Python**

```
n = int(input("Nhập n: "))          # Nhập số n tùy ý
tong = 0                            # Khai báo và gán giá trị cho tong
i = 1                               # Khai báo và gán giá trị cho biến đếm i
while i <= n:
    tong = tong + i
    i = i+1                         # Cập nhật biến đếm
print("Tổng là", tong)
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Kết hợp while với else**

Khối lệnh của else sẽ được thực hiện khi điều kiện của while là False.

```
dem = 0
while dem < 3:
    print("Đang ở trong vòng lặp while")
    dem = dem + 1
else:
    print("Đang ở trong else")
```

BÀI TẬP

NỘI DUNG:

Viết chương trình tìm các số nguyên tố trong khoảng từ 0 đến 100 sử dụng vòng lặp while.

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh break trong Python**

Lệnh break kết thúc vòng lặp chứa nó và truyền điều khiển đến lệnh tiếp theo sau khối code của vòng lặp đó.

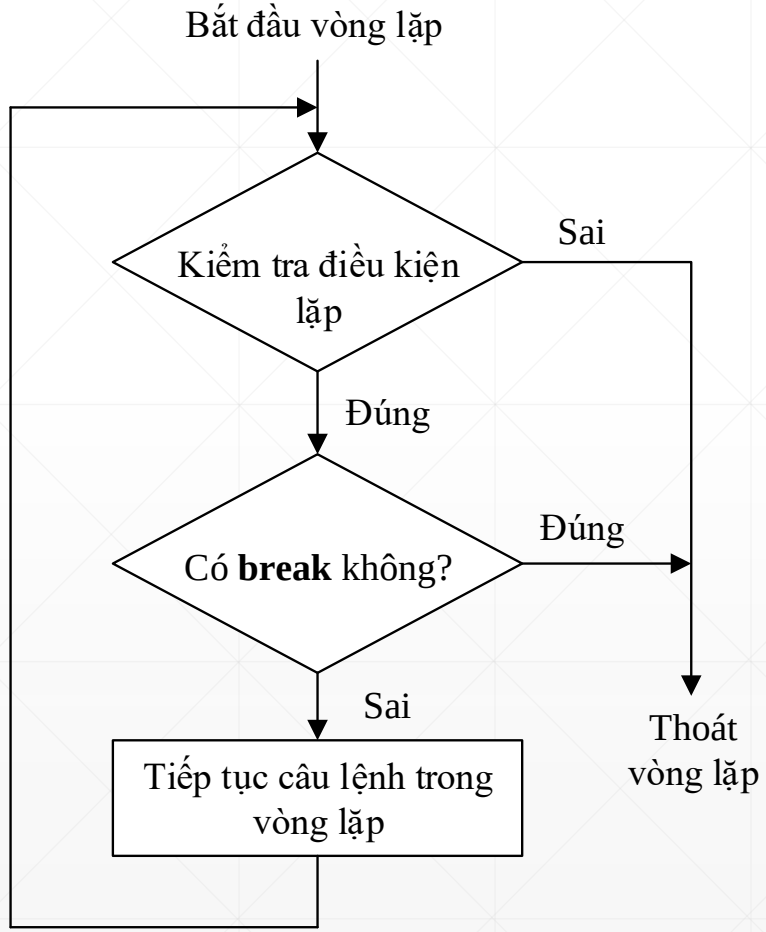
Nếu lệnh break ở trong một vòng lặp lồng nhau (vòng lặp bên trong một vòng lặp khác), break sẽ chấm dứt vòng lặp trong cùng.

```
break
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh break trong vòng lặp for trong Python**

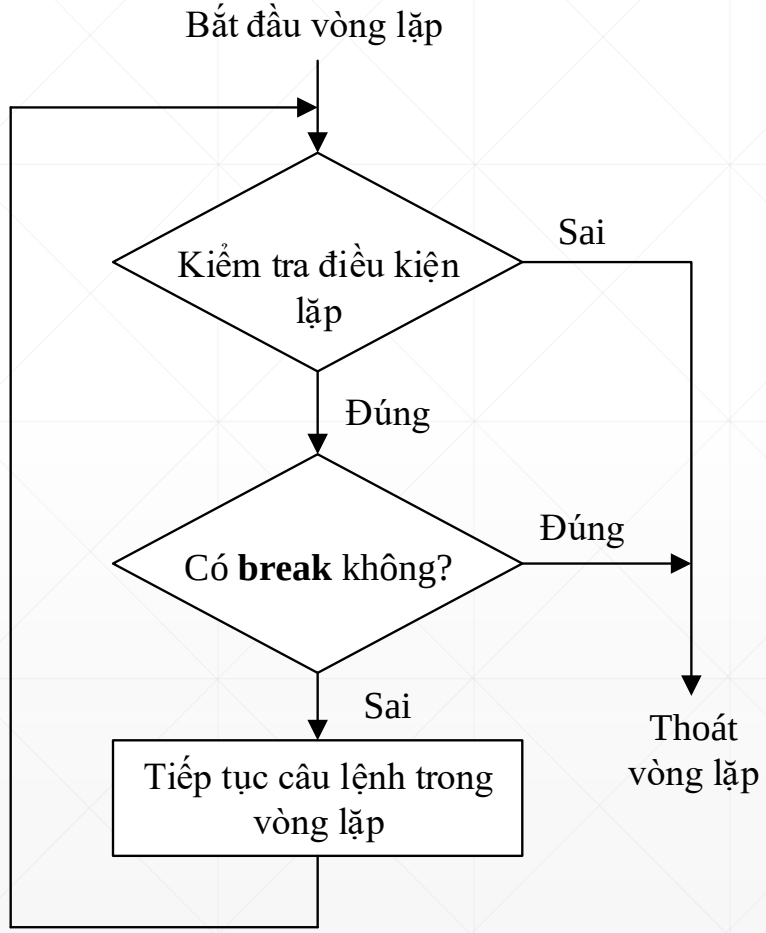
```
for var in sequence:
    #khối code bên trong vòng lặp for
    if dieu_kien:
        break
    #code khác bên trong vòng lặp for
#code bên ngoài vòng lặp for
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh break trong vòng lặp while Python**

```
while dieu_kien_kiem_tra:
    #code bên trong vòng lặp while
    if dieu_kien:
        break
    #code khác bên trong vòng lặp while
#code bên ngoài vòng lặp while
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh continue trong Python**

Lệnh continue được sử dụng để bỏ qua phần còn lại của code bên trong vòng lặp, áp dụng cho lần lặp hiện tại.

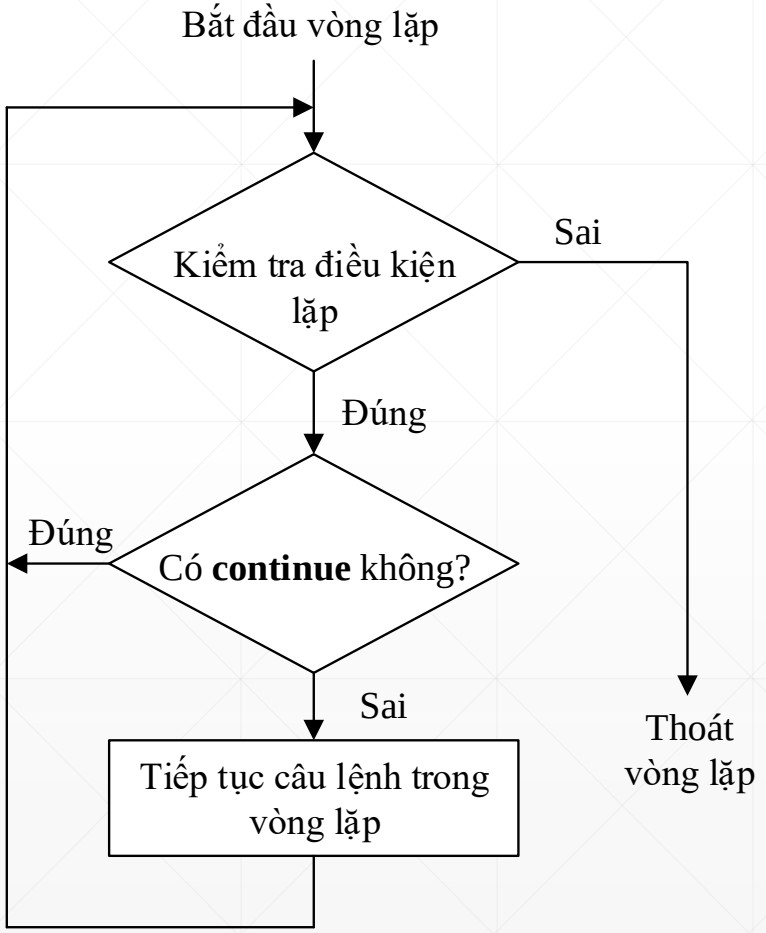
→ Nghĩa là vòng lặp không chấm dứt, nó sẽ tiếp tục với lần lặp kế tiếp.

```
continue
```

CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh continue trong vòng lặp for**

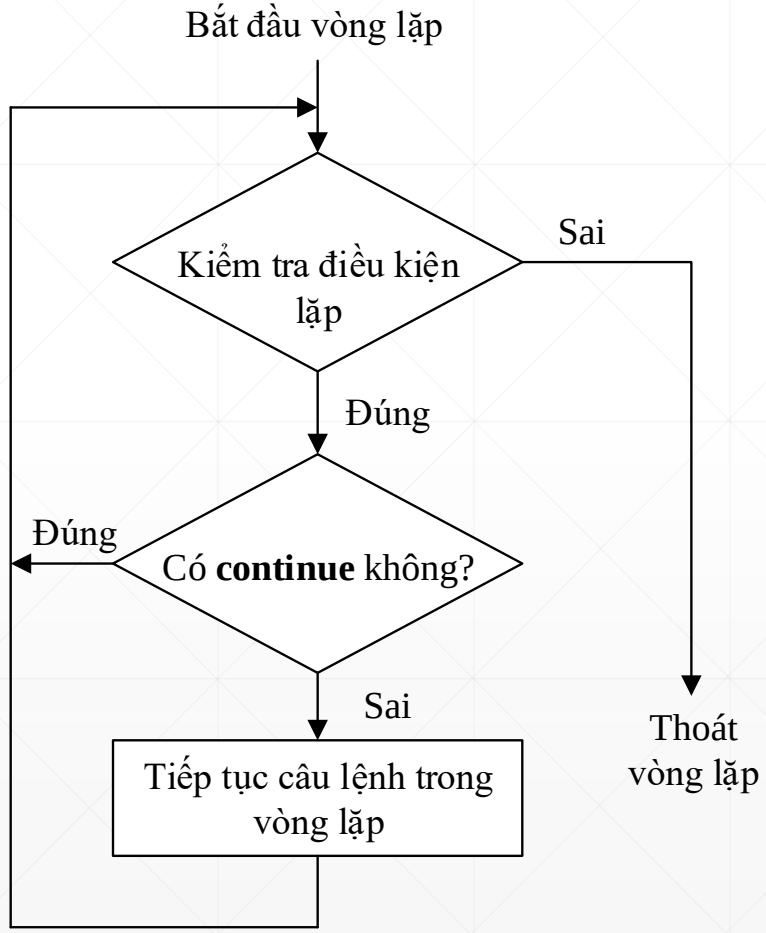
```
for var in sequence:
    #khối code bên trong vòng lặp for
    if dieu_kien:
       continue
    #code khác bên trong vòng lặp for
#code bên ngoài vòng lặp for
```



CÁC CẤU TRÚC ĐIỀU KHIỂN

- **Lệnh continue trong vòng lặp while**

```
while dieu_kien_kiem_tra:
    #code bên trong vòng lặp while
    if dieu_kien:
    continue
    #code khác bên trong vòng lặp while
#code bên ngoài vòng lặp while
```



BÀI TẬP

NỘI DUNG:

1. Viết chương trình tìm tất cả các số chia hết cho 7 nhưng không phải bội số của 5, nằm trong đoạn 2000 và 3200 (tính cả 2000 và 3200). Các số thu được sẽ được in thành chuỗi trên một dòng, cách nhau bằng dấu phẩy.

2. Viết chương trình tìm dãy số Fibonacci trong khoảng từ 0 đến 1000.

BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 3.1; 3.2; và 3.3.
2. Tìm hiểu về khái niệm hàm trong lập trình Python.
3. Nghiên cứu về các phương thức truyền tham số trong lập trình hàm.