TRƯỜNG ĐẠI HỌC THƯƠNG MẠI

THUONGMAI UNIVERSITY

Chương 2. Các thành phần cơ sở của Python

Số tiết: 8 LT

Nội dung

2.1. Giới thiệu về ngôn ngữ lập trình Python

2.2. Các yếu tố cơ bản của Python

2.3. Các lệnh cơ bản

2.1. Giới thiệu về ngôn ngữ lập trình Python

- 2.1.1 Lịch sử phát triển
- 2.1.2 Đặc điểm của ngôn ngữ Python
- 2.1.3 Cài đặt công cụ lập trình Python
- 2.1.4 Các bước viết và biên dịch chương trình với Python

2.1.1 Lịch sử phát triển

- Được tạo bởi Guido van Rossum, phát triển từ 12/1989 và phát hành năm 1991
- Phiên bản chính thức: 1 (1/1994), 2 (16/10/2000), 3 (3/12/2008) (3.10)
- Là ngôn ngữ phổ biến và được sử dụng cho:
  - Phát triển web (phía server)
  - Phát triển phần mềm
  - Toán học
  - Kịch bản hệ thống
  - Phân tích dữ liệu

2.1.2 Đặc điểm của ngôn ngữ Python

- Là ngôn ngữ lập trình kịch bản, động, hỗ trợ cho các mô hình lập trình khác nhau, bao gồm: lập trình thủ tục, lập trình hướng đối tượng, lập trình chức năng.
- Mã code Python có thể được viết bởi 1 text Editor hoặc 1 môi trường phát triển tích hợp (IDE) như Thony, Pycharm, Netbeans hay Eclipse và được thực thi trong trình thông dịch Python. → xây dựng mẫu nhanh
- Python có thể làm việc trên các nền tảng khác nhau: Windows, Mac, Linux, Raspberry Pi,…
- Python có cú pháp đơn giản, rõ ràng (tương tự tiếng Anh) và cực kỳ linh hoạt (versatile) (giống giả mã)

2.1.2 Đặc điểm của ngôn ngữ Python

- Python sử dụng dòng mới (tương ứng enter) để hoàn thành một câu lệnh (các ngôn ngữ khác thường dùng dấu chấm phẩy (semicolon) hoặc dấu ngoặc đơn (parentheses)
- Python dựa vào thụt lề (indentation), sử dụng khoảng trắng để xác định phạm vi , ví dụ phạm vi của vòng lặp, hàm và lớp (các ngôn ngữ khác thường dùng dấu ngoặc nhọn – curly brackets)
- Thư viện tiêu chuẩn toàn diện cho nhiều tác vụ, cộng đồng phát triển lớn.
- Có thể mở rộng đơn giản qua thông C/C++, đóng gói các thư viện C/C++
- Python cho phép người phát triển viết chương trình với số dòng ít hơn một số ngôn ngữ lập trình khác → tốc độ lập trình nhanh.

2.1.2 Đặc điểm của ngôn ngữ Python

Các nguyên tắc phần mềm ảnh hưởng thiết kế của Python

- Đẹp tốt hơn xấu
- Rõ ràng tốt hơn ngầm/ ẩn (explicit/implicit)
- Đơn giản tốt hơn phức tạp
- Phức tạp tốt hơn lộn xộn (complex/complicate)
- Phẳng hơn tốt hơn là lồng nhau (flat/nested)
- Thưa tốt hơn dày (Sparse/Dense)
- Số lượng có thể đọc
- Các trường hợp đặc biệt không đủ đặc biệt để phá vỡ quy tắc
- Thực tế đánh bại sự trong sáng
- Lỗi không nên trôi qua im lặng trừ khi sự im lặng là rõ ràng
- ….

2.1.3 Cài đặt công cụ lập trình Python

- Nhiều máy PC và Mac có thể được cài sẵn Python
- Để check đã cài Python:
  - Vào Command Line → gõ lệnh: python - -version
- Link download: https://www.python.org/
- Các IDE: IDLE, Visual Studio Code, Thony, Pycharm, Netbeans hay Eclipse
- Lưu ý với IDLE: đơn giản nhưng với 2 cửa sổ khác nhau: cửa sổ Shell cho phép nhập mã và xem kết quả trên cùng cửa số (áp dụng với đoạn lệnh nhỏ) hoặc mở cửa sổ mới để viết chương trình riêng. (cửa sổ Shell chạy bằng Enter, cửa sổ mới chạy bằng F5 hoặc thực đơn)

2.1.4 Các bước viết và dịch chương trình

- Viết code →dịch và chạy
- Bằng Command Line (chuyển sang Python bằng lệnh Python) hoặc cửa sổ Shell
- Nếu viết đến đâu dịch và chạy đến đó (thông dịch): chỉ áp dụng đối với đoạn code ngắn (dùng ; để phân tách các câu lệnh) → không lưu và sửa được code, dịch khi hết lệnh và nhấn Enter.
- Nếu viết chương trình thành các file/ module/package
- Bằng Command Line: Python path//tên_tệp.py

2.1.4 Các bước viết và dịch chương trình

- Hoặc chuyển thư mục làm việc sang thư mục chứa tệp python:
- cd /d path
- Python tên_tệp.py
- Bằng cửa sổ Shell (mở cửa sổ mới) → soạn thảo → lưu tệp→ nhấn F5 (thực đơn Run → Run Module)
- Lưu ý: để xóa màn hình ở chế độ Command Line thực hiện:

```
import os
ret=os.system('cls')
```

2.2. Các yếu tố cơ bản của Python

2.2.1. Bảng chữ cái

2.2.2. Từ khóa

2.2.3. Tên gọi

2.2.4. Chú thích

2.2.5. Các kiểu dữ liệu cơ sở

2.2.6. Các đại lượng

2.2.7. Biểu thức

2.2.8. Cấu trúc chương trình

2.2.1. Bảng chữ cái

- Python sử dụng bộ kí tự ASCII, bao gồm:
  - Chữ cái: A → Z, a →z
  - Dấu gạch nối: _
  - Số: 0 -> 9
  - Các dấu phép toán số học: + - * / <> % ^
  - Các cặp dấu ngoặc ( ) { } [ ]
  - Các dấu ngăn cách: dấu cách, dấu nhảy tab, dấu xuống dòng
  - Các dấu khác: # $. , : ; “ ! ‘ ? @
- ***Đặc điểm***
  - Mỗi kí tự tương ứng một mã (số nguyên 0→255)
  - 128 kí tự đầu là kí tự cố định, còn lại là kí tự mở rộng
- ***Lưu ý***: có thể sử dụng tiếng việt trong viết câu lệnh và ghi chú thích

2.2.2. Từ khóa

- ***Khái niệm***: Là một tập các từ dùng riêng cho từng ngôn ngữ lập trình. Mỗi từ khóa có một ý nghĩa và tác dụng cụ thể. Ví dụ như tên kiểu dữ liệu, tên toán tử, tên hàm v.v..
- ***Lưu ý***
  - Không được đặt tên các đối tượng khác trùng tên với từ khóa như biến, hằng, mảng…
  - Từ khóa phải viết bằng chữ thường.

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

2.2.3. Tên gọi

- ***Khái niệm:*** Là một dãy các ký tự đặt cạnh nhau, được dùng để định danh các đối tượng khác nhau trong chương trình như tên biến, tên hàm,…
- ***Quy tắc***
  - Tên chỉ được chứa các chữ cái, chữ số và dấu gạch nối _
  - Tên phải bắt đầu bằng chữ cái hoặc dấu gạch nối
  - Không được trùng với từ khóa
  - Phân biệt giữa chữ hoa và chữ thường

2.2.4. Chú thích

- ***Mục đích***: dùng để giải thích, làm rõ nghĩa cho một câu lệnh nào đó, và được trình dịch bỏ qua khi dịch và chạy chương trình
- ***Vị trí***: Ở bất kỳ đâu trong chương trình, đứng riêng biệt hoặc sau các câu lệnh.
- Phân loại: chú thích trên 1 dòng và chú thích trên nhiều dòng (không thực sự có)
- Trên 1 dòng:

Nội dung chú thích

- Trên nhiều dòng:
- C1: dùng dấu # cho từng dòng
- C2: dùng ba dấu nháy đơn hoặc 3 dấu nháy kép (bản chất là khai báo một choỗi trên nhiều dòng mà không gán cho biến nào)
- ‘ ‘ ‘
- Chú thích 1
- Chú thích 2
- ‘ ‘ ‘
- Lưu ý: cách chú thích nhiều dòng này yêu cầu các dòng đều có độ thụt lề (indentation) như nhau

2.2.5. Các kiểu dữ liệu cơ sở

- ***Khái niệm 1 kiểu dữ liệu***: là một tập hợp các giá trị mà một biến thuộc kiểu đó có thể nhận được, và trên đó xác định một số phép toán.
- ***Phân loại***
  - Kiểu dữ liệu cơ sở: kiểu số (int, float, complex), kiểu xâu ký tự (str), kiểu logic (bool), kiểu nhị phân (bytes, bytearray, memoryview), không kiểu (NoneType).
  - Kiểu dữ liệu có cấu trúc: kiểu danh sách (list), kiểu bộ (tuple), kiểu tập hợp (set), kiểu từ điển (dict), kiểu do người dùng tự định nghĩa.
  - Hoặc
  - Kiểu xây dựng sẵn
  - Kiểu do người dùng tự định nghĩa
  - Để xác định kiểu của 1 đối tượng dùng hàm type()

2.2.5. Các kiểu dữ liệu cơ sở

- Kiểu số: gồm 3 loại số nguyên (int), thực (float), phức (complex)
- Int: bao gồm các số nguyên dương hoặc âm, không có phần thập phân và không giới hạn độ dài
- Float: số thực hay số dấu chấm động là số dương hoặc âm có chứa 1 hoặc nhiều chữ số phần thập phần/ có thể biểu diễn bằng cách viết khoa học. (e/E để chỉ lũy thừa của 10)
- Complex: bao gồm các số thực với kí hiệu j để mô tả phần ảo

- X=2
- Y=-3.6
- Z=6e3
- F=-14.5E3
- C=2+5j

2.2.5. Các kiểu dữ liệu cơ sở

- Thứ tự của các phép toán:
- Lũy thừa →nhân, chia (cả // và %)→ cộng, trừ
- Chuyển đổi kiểu dữ liệu (trừ số phức không chuyển về được các kiểu số khác): các kiểu số có thể chuyển về các kiểu khác
- Cú pháp
- Tên_biến= tên_kiểu(biến_cũ)

```
X=1 # int
y=2.8 # float
z=1j # complex
#convert from int to float:
a=float(x)
#convert from float to int:
b=int(y)
#convert from int to complex:
c=complex(x)
print(a)
print(b)
print(c)
print(type(a))
print(type(b))
print(type(c))
```

2.2.5. Các kiểu dữ liệu cơ sở

- Số ngẫu nhiên
- Module: random
- Function: randint, randbytes, randrange, seed,…
- `import random`
- `print(random.randrange(1, 10))`
- from random import randint
- x = randint(1,10)
- print('A random number between 1 and 10: ', x)

2.2.5. Các kiểu dữ liệu cơ sở

- Các hàm toán học
- Module: math
- Functions: sin, cos, tan,exp, log,log10,factorial, sqrt,abs, floor, ceil, …, các hằng pi, e
- from math import abs,sin, pi
- print('Pi is roughly', pi)
- print('sin(0) =', sin(0))
- print(abs(-4.3))
- print(round(3.336, 2))
- Lưu ý: để xem các hàm có trong 1 modle bằng cửa sổ Command Line, ý nghĩa của các hàm
- Import module
- Dir(module)
- Help (module.tên_hàm)

2.2.5. Các kiểu dữ liệu cơ sở

- Kiểu xâu ký tự: được khai báo thông qua cặp dấu nháy đơn hoặc dấu nháy kép.
- Nếu muốn xác định xâu ký tự trên nhiều dòng thì dùng 3 dấu nháy đơn hoặc 3 dấu nháy kép

2.2.6. Các đại lượng

- ***Hằng (constant)***: là đại lượng giá trị không thay đổi trong quá trình tính toán.
- Vị trí: tham gia vào các biểu thức tính toán và các lệnh gán
- Giá trị xác định kiểu dữ liệu và các phép toán có thể thực hiện/ thao tác tương ứng.
- Ví dụ:
- Giá trị số: thực hiện các phép toán số học và các hàm trong modle math
- Giá trị xâu: thực hiện các phép toán, hàm với ký tự, xâu ký tự (in, not in, print, len, và các hàm upper, lower, find,…

2.2.6. Các đại lượng

- ***Biến (variable)***: là một đại lượng có giá trị thuộc một kiểu dữ liệu nào đó và giá trị này có thể thay đổi trong thời gian tồn tại của biến .
- Trong Python không có câu lệnh khai báo biến như các ngôn ngữ khác, biến được tạo tại thời điểm nó được gán 1 giá trị ban đầu.
- Cú pháp: tên_biến=giá_trị
- Kiểu dữ liệu của biến được xác định thông qua giá trị mà nó được gán và có thể thay đổi liên tục.

```
x = 4       # x is of type int
print(x)
x = "Sally" # x is now of type str
print(x)
```

- Hoặc xác định kiểu dữ liệu cụ thể cho biến bằng ép kiểu (casting)
- Cú pháp: tên_biến=tên_kiểu(giá_trị)

```
x = str(3)    # x will be '3’
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

2.2.6. Các đại lượng

- Để lấy thông tin kiểu dữ liệu của biến, dùng hàm type()
- x = 5
- y = "John“
- print(type(x))
- print(type(y))
- Lưu ý:
- Tên biến được đặt theo quy tắc đặt tên(bắt đầu bằng chữ hoặc dấu gạch dưới, không bắt đầu bằng số, chứa số, chữ cái và gạch dưới.
- Python phân biệt tên biến chữ hoa và chữ thường

2.2.6. Các đại lượng

- Có thể gán nhiều giá trị cho nhiều biến: số lượng biến phải bằng số giá trị
- x, y, z = "Orange", "Banana", "Cherry“
- print(x); print(y); print(z)
- Có thể gán cùng một giá trị cho nhiều biến
- x = y = z = "Orange“
- print(x); print(y); print(z)
- Giải nén một tập hợp: để gán các giá trị trong một tập hợp (list, tuple…) cho các biến
- fruits = ["apple", "banana", "cherry"]
- x, y, z = fruits
- print(x); print(y); print(z)
- Phân biệt biến toàn cục và biến cục bộ/ địa phương.

2.2.7. Biểu thức

- ***Khái niệm:*** Biểu thức là sự kết hợp giữa các toán hạng và toán tử theo một các phù hợp để diễn đạt một công thức toán học nào đó.

  Trong đó:

  - Các toán hạng có thể là hằng, biến hay lời gọi hàm hoặc là một biểu thức con nào đó.
  - Các toán tử phụ thuộc vào tập các toán tử mà ngôn ngữ hỗ trợ .
- ***Giá trị biểu thức***: được ước lượng và có kiểu dữ liệu phụ thuộc vào các thành phần trong biểu thức.
- ***Vị trí của biểu thức***: xuất hiện trong bất kì một câu lệnh nào của Python
- ***Ví dụ***
  - Vế phải của một câu lệnh gán : ví dụ: x = 5*a ;
  - Làm tham số thực sự của hàm: ví dụ: sqrt(y)
  - Trong câu lệnh điều kiện if, for…

2.2.8. Cấu trúc chương trình

- Mỗi câu lệnh trong Python thường được tổ chức trên 1 dòng và trình thông dịch sẽ đọc và thực hiện từng câu lệnh.
- Mỗi câu lệnh sẽ được xác định kết thúc thông qua dòng mới (enter).
- Nếu có nhiều lệnh trên 1 dòng → dùng dấu chấm phẩy ;
- `print(x); print(y); print(z)`
- Nếu một lệnh ở trên nhiều dòng: 2 loại
- Loại 1: Lệnh nối tiếp ngầm định (explicit), khi có các cặp dấu (),{ },[ ]
- A=[
- [1,2,3],
- [3,4,5]
-]
- Print(A)

```
if (
    person_1 >= 18 and
    person_2 >= 18 and
    person_3 < 18
    ):
     print('2 Persons should have ID Cards')
```

```
s = 1 + 2 + 3 + \
 4 + 5 + 6 + \
 7 + 8 + 9
```

- Loại 2: Lệnh nối tiếp rõ ràng (implicit) bằng cách sử dụng dấu \

2.2.8. Cấu trúc chương trình

- Chương trình có thể là 1 tệp, nhiều tệp/ thư mục (module)
- Muốn sử dụng các tệp/ thư mục/ module dùng lệnh import
- Nếu các tệp cùng thư mục
- Cú pháp: import tên_tệp
- Nếu các tệp không cùng thư mục: 2 cách
- Cách 1: Dùng biến sys.path trong module sys
- Import sys
- Sys.path.insert(0,đường_dẫn_thư_mục) hoặc Sys.path.append(đường_dẫn_thư _mục)
- Import tệp.
- Cách 2 sử dụng biến môi trường PYTHONPATH thông qua cửa sổ Command Line
- Mở cửa sổ Command Line và thiết lập đường dẫn
- Cú pháp: set PYTHONPATH=đường_dẫn_thư_mục
- Import sys
- Sau đó trong chương trình import tên_tệp

2.2.8. Cấu trúc chương trình

- Nếu chỉ muốn sử dụng các hàm, các khai báo trong một tệp khác: 2 cách
- Cách 1: import cả tệp
- Import tên_tệp # gọi thư viện/tệp
- tên_tệp.tên_hàm() # sử dụng hàm trong thư viện/tệp
- Cách 2: import trực tiếp đối tượng trực tiếp từ tệp
- Cú pháp: from tên_tệp import tên_hàm/đối_tượng
- Ví dụ
- A=3.2
- Import math
- Print(math.ceil(A))
- Hoặc
- From math import ceil
- Print(ceil(A))

2.2.8. Cấu trúc chương trình

- Nếu muốn đặt lại tên cho tệp/module khi import
- Cú pháp: import tên_tệp as tên_mới
- From tên_tệp import tên_hàm as tên_mới
- Ví dụ: import math as TV_Toan
- Muốn liệt kê các hàm, các biến có trong 1 module/tệp:
- Cú pháp: dir(tên_module)
- Ví dụ:
- Import(math)
- Dir(math)

2.3. Các lệnh cơ bản

- 2.3.1. Lệnh gán
- 2.3.2. Lệnh vào, ra dữ liệu
- 2.3.3. Các lệnh điều khiển rẽ nhánh
- 2.3.4. Các lệnh điều khiển chu trình

Giới thiệu

- ***Khái niệm:*** Lệnh là một chỉ thị yêu cầu máy thực hiện
- ***Phân loại:***
  - Câu lệnh đơn giản: là lệnh không chứa các lệnh khác.Ví dụ: phép gán, lời gọi hàm loại void, lệnh nhảy không điều kiện *goto*
  - Câu lệnh phức hợp/ ghép: khối lệnh, lệnh rẽ nhánh, lệnh lặp-chu trình
- ***Lưu ý***
  - Lệnh được ghi trên 1 dòng và kết thúc bằng dấu Enter. Nếu có nhiều lệnh trên cùng dòng thì mỗi lệnh kết thúc bằng dấu chấm phẩy ;
  - Nếu 1 lệnh kéo dài trên nhiều dòng thì dùng dấu \ để ghép lệnh
  - Khối lệnh/ lệnh ghép dùng dấu hiệu thụt lề để xác định
  - Khối lệnh/ các lệnh ngang hang luôn thụt lề giống nhau

2.3.1. Lệnh gán

- Chức năng: gán 1 giá trị cho một biến
- Cú pháp: tên_biến= giá trị
- `x = “Hello World”`
- `a = 3.5`
- `Li=[3, 5, 7]`
- `Tu=(“apple”,”banana”, “grape”)`
- `Se={“Huyền”, 1990, 3.2}`
- `Di={“Name”:“Huyền”, “YOB”: 1990, “GPA”:3.2}`
- `Có thể gán nhiều giá trị cho nhiều biến`
- `Có thể gán một giá trị cho nhiều biến`

2.3.2.Lệnh vào ra dữ liệu – Hàm input

- Chức năng: để nhập dữ liệu từ bàn phím vào cho biến
- Hoạt động: nhận giá trị nhập từ bàn phím cho đến khi gặp Enter và chuyển về string.
- Cú pháp: tên_biến=input((“Lời_thông báo”])
- Ví dụ: name=input(“Enter your name:”)
- Lưu ý: để nhập đúng kiểu số, cần ép kiểu
- Ví dụ: n =int(input(“Nhap so Nguyen:”))
- Với các phiên bản cũ 2.x trở về trước có them hàm raw_input()

2.3.2.Lệnh vào ra dữ liệu – Hàm input

- Có thể nhập dữ liệu cho nhiều biến trong cùng 1 lệnh: dùng phương thức split hoặc khả năng của List .
- Cách 1: dùng phương thức split(): Input().split(separator,maxsplit)
- Trong đó:
- separator xác định ký tự phân tách xâu nhập vào, mặc định là khoảng trắng
- Maxsplit xác định bao nhiêu lần tách được thực hiện, mặc định là -1 tương ứng với số lần xuất hiện dấu phân tách.
- Ví dụ: x, y, z=input().split() # nhập dữ liệu cho 3 biến và phân tách bằng khoảng trắng
- A,b=input().split(“;”) # nhập dữ liệu cho 2 biến cách nhau bởi dấu ;
- A,b,c=input().split(“,”,2) # nhập dữ liệu cho 3 biến phân tách nhau bằng dấu phẩy và tách bởi 2 lần (tương ứng 3 biến)

2.3.2.Lệnh vào ra dữ liệu – Hàm input()

- Cách 2 dùng khả năng của list

```
# taking two input at a time
x, y = [int(x) for x in input("Enter two values: ").split()]
print("First Number is: ", x)
print("Second Number is: ", y)
print()

# taking three input at a time
x, y, z = [int(x) for x in input("Enter three values: ").split()]
print("First Number is: ", x)
print("Second Number is: ", y)
print("Third Number is: ", z)
print()

# taking two inputs at a time
x, y = [int(x) for x in input("Enter two values: ").split()]
print("First number is {} and second number is {}".format(x, y))
print()
```

2.3.2.Lệnh vào ra dữ liệu – Hàm print

- Chức năng: để xuất dữ liệu ra màn hình.
- Cú pháp: print(values,sep=‘ ’, end=‘\n’)
- Trong đó:
- values là các giá trị xâu ký tự hoặc biến cần in giá trị ra màn hình.
- Sep: xác định cách tách các đối tượng khi có nhiều giá trị cần in. Mặc định la ‘ ‘
- End: xác định giá trị được in ở cuối dòng. Mặc định là xuống dòng ‘\n’
- A=5; Print(a)
- Print(“ngon ngu lap trinh Python”)
- Để xuất nhiều thông tin trong cùng một câu lệnh, dùng dấu , để ngăn cách
- a=5; b=7
- print("gia tri cua a va b la:")
- print(a, b,sep=',', end='\n')

2.3.3. Lệnh điều khiển rẽ nhánh If

- Chức năng: cho phép thực hiện các công việc khác nhau tương ứng với các điều kiện khác nhau.
- Cú pháp:

if điều_kiện:

    Khối_lệnh1:

Else:

    Khối_lệnh2

- Hoặc:

If điều_kiện1:

    Khối lệnh 1

Elif điều_kiện2:

    Khối_lệnh2

Else:

    Khối_lệnh3

```
a = 15; b = 50
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
a = 15; b = 50
if b > a:
  print(b," is greater than
“,a)
elif a == b:
  print(a," and”, b, “are
equal")
a = 15; b = 50
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

2.3.3. Lệnh điều khiển rẽ nhánh If

- Lưu ý:
- Nếu sử dụng if khuyết thì khối lệnh có thể viết trên 1 dòng thông qua dấu chấm phẩy
- a=7
- b=2
- `if a>b: print("ket luan"); print(a, "lon hơn",b)`
- Nếu chỉ có 1 công việc cần thực hiện thì có thể dùng cú pháp if… else rút gọn trên cùng 1 dòng:
- `a = 2`
- `b = 8`
- `print("A") if a > b else print("B")`

2.3.3. Lệnh điều khiển rẽ nhánh If

- Các câu lệnh trong khối lệnh của if, else ,elif phải tương ứng thụt lề như nhau. (do python dựa vào thụt lề để xác định phạm vi trong code, các ngôn ngữ khác thường dùng { }
- `a = 15`
- `b = 50`
- `if b > a:`
- `print("b is greater than a") # you will get an error`
- Nếu câu lệnh if rỗng (không có nội dung) thì dùng câu lệnh pass.
- `if b > a:`
- `  pass`
- Câu lệnh If có thể lồng nhau
- Không có câu lệnh switch … case trong Python

2.3.4. Lệnh điều khiển chu trình for

- Chức năng: cho phép thực hiện lặp công việc với số lần lặp cho trước hoặc thực hiện công việc trên các phần tử của 1 dãy (list, tuple, dictionary, set, string) (hơi khác so với câu lệnh for trong các NN khác)
- Cú pháp:
- For tên_biến in range (số_lần_lăp):
- Khối_lệnh_lặp
- Ví dụ
- For i in range (10): # in ra 1→10
- Print(i)
- for i in range(4): # in 4 dòng, mỗi dòng 6 ngôi sao
- print('*'*6)
- `fruits = ["apple", "banana", "cherry"]`
- `for x in fruits:`
- `  print(x)`

Statement	Values generated
`range(10)`	0,1,2,3,4,5,6,7,8,9
`range(1,10)`	1,2,3,4,5,6,7,8,9
`range(3,7)`	3,4,5,6
`range(2,15,3)`	2,5,8,11,14
`range(9,2,-1)`	9,8,7,6,5,4,3

2.3.4. Lệnh điều khiển chu trình for

- Để dừng vòng lặp for giữa chừng dùng câu lệnh break. (so sánh)

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

2.3.4. Lệnh điều khiển chu trình for

- Để bỏ qua lần lặp hiện tại và chuyển sang lần lặp kế tiếp sử dụng câu lệnh continue.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Để thực hiện công việc sau khi kết thúc for dùng từ khóa else

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

2.3.4. Lệnh điều khiển chu trình for

- Python cho phép các câu lệnh for lồng nhau
- `adj = ["red", "big", "tasty"]`
- `fruits = ["apple", "banana", "cherry"]`
- `for x in adj:`
- `  for y in fruits:`
- `  print(x, y)`
- Nếu vòng lặp rỗng (không có nội dung), dùng câu lệnh pass.
- `for x in [0, 1, 2]:`
- `  pass`

2.3.4. Lệnh điều khiển chu trình while

- Chức năng: cho phép thực hiện lặp tập hợp các câu lệnh trong khi điều kiện đúng. (không biết trước số lần lặp)
- Cú pháp:

```
while điều_kiện:
    Khối_lệnh_lặp
```

- **Ví dụ:**

```
1.      i = 1
2.      while i < 6:
3.          print(i)
4.          i += 1
```

2.3.4. Lệnh điều khiển chu trình while

- Lưu ý: cần thay đổi giá trị biến điều khiển→ tránh vòng lặp vô hạn
- i=0
- while i<10:
- Print(i)
- Để dừng vòng lặp giữa chừng dùng câu lệnh break.

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Để dừng vòng lặp hiện tại và chuyển sang lần lặp kế tiếp dùng câu lệnh continue

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

2.3.4. Lệnh điều khiển chu trình while

Để thực hiện công việc khi điều kiện sai (kết thúc vòng lặp while) dùng câu lệnh else.

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Nếu vòng lặp while rỗng (không có nội dung) dùng câu lệnh pass

```
a = ‘programing with Python'
i = 0
while i < len(a):
    i += 1
    pass

print('Value of i :', i)
```

2.3.4. Lệnh điều khiển chu trình while

- Python cho phép while lồng nhau

```
a = int(input('Enter a number (-1 to quit): '))

while a != -1:
    a = int(input('Enter a number (-1 to quit): ‘))
```

- Vòng lặp while với list

```
a = [1, 2, 3, 4]

while a:
    print(a.pop())
```

2.3.4. Lệnh điều khiển chu trình while

- So sánh while và for
  - Ý nghĩa
  - Cú pháp
  - Số lần lặp
  - Cách thức thực hiện