# IE221 - BÀI THỰC HÀNH 2
# Toán tử, từ khóa, biến số, các kiểu dữ liệu

Ngày 26 tháng 9 năm 2025

## Mục lục

# 1. Giới thiệu cơ bản về cú pháp Python

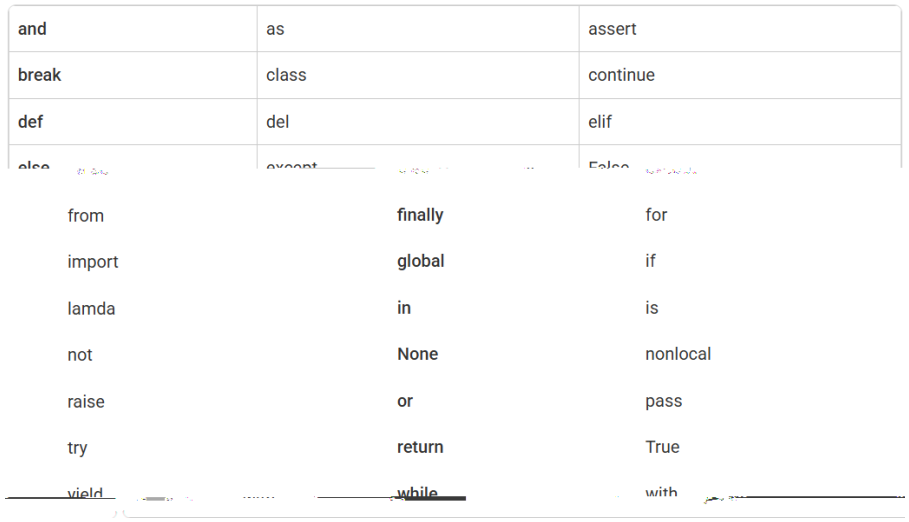
Python là ngôn ngữ lập trình bậc cao, dễ đọc và có quy tắc cú pháp đơn giản. Cú pháp trong Python đóng vai trò như một bộ quy tắc được sử dụng để tạo nên một chương trình Python hoàn chỉnh. Cú pháp của ngôn ngữ lập trình Python có nhiều điểm tương đồng với các ngôn ngữ lập trình khác như Perl, ngôn ngữ C, và Java. Tuy nhiên, giữa các ngôn ngữ này vẫn có những khác biệt nhất định.

Một số nguyên tắc cơ bản trong Python cần lưu ý:

## 1.1 Từ khóa (keywords)

Trong Python, một số từ đã được định nghĩa sẵn và mang ý nghĩa đặc biệt, gọi là từ khóa (keywords). Các từ này được dành riêng cho ngôn ngữ Python, và bạn không thể sử dụng chúng để đặt tên cho biến, hàm, lớp hay bất kỳ đối tượng nào khác trong chương trình của mình. Các từ khóa trong Python luôn được viết bằng chữ thường.

```
import keyword
print(keyword.kwlist)  # In ra danh sách từ khóa
```




Hình 1: Danh sách các từ khóa trong Python

## 1.2 Dòng lệnh và độ thụt dòng lệnh (Indentation)

- Khác với nhiều ngôn ngữ lập trình khác (như C, Java) sử dụng dấu ngoặc nhọn  để xác định các khối mã (code block), Python sử dụng thụt lề (indentation) để thể hiện cấu trúc và phân chia các khối code.
- 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.

## 1.3 Trích dẫn (Quotes)

- Trong Python, dấu nháy được sử dụng để tạo ra các chuỗi ký tự (string).
- Có ba loại dấu nháy chính được chấp nhận: dấu nháy đơn (’), dấu nháy đôi (") và dấu nháy ba (“‘ hoặc """).
- Quan trọng là loại dấu nháy nào dùng để mở chuỗi thì phải dùng chính dấu nháy đó để kết thúc chuỗi.
- Dấu nháy ba thường được sử dụng để tạo ra các chuỗi ký tự kéo dài trên nhiều dòng

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

print(word)
print(sentence)
print(paragraph)
```

### 1.4 Chú thích (Comment)

- Trong lập trình Python, chú thích (comment) là những dòng giải thích hoặc ghi chú do lập trình viên viết trong mã nguồn Python.
- Mục đích chính của chú thích là làm cho mã nguồn dễ đọc và dễ hiểu hơn đối với con người, đồng thời trình thông dịch Python sẽ bỏ qua những dòng này khi thực thi chương trình.
- Dấu thăng (#) được sử dụng để bắt đầu một chú thích. Bất kỳ ký tự nào theo sau dấu thăng (#), cho đến hết dòng, đều được coi là một phần của chú thích và sẽ không được trình thông dịch Python thực thi.

```
    # Đây là một dòng chú thích đầu tiên
print ("Hello World!") # Đây là dòng chú thích thứ hai
```

- Ngoài ra, chuỗi ký tự được đặt trong ba dấu nháy đơn (''') hoặc ba dấu nháy kép (""") cũng có thể được sử dụng như chú thích nhiều dòng vì trình thông dịch Python cũng sẽ bỏ qua chúng

```
"""
This is a multiline
comment.
"""
print("Hello World!")
```

### 1.5 Kết hợp nhiều câu lệnh trên cùng một dòng

- Trong Python, có thể viết nhiều câu lệnh trên cùng một dòng bằng cách sử dụng dấu chấm phẩy (;).
- Tuy nhiên, cần lưu ý rằng, cách này chỉ áp dụng được khi các câu lệnh đó không bắt đầu một khối mã mới (code block).

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

## 2. Biến số (Variables)

Biến dùng để lưu trữ dữ liệu, Python là ngôn ngữ **dynamic typing** (không cần khai báo kiểu). Biến giống như nhãn dán trên hộp, dùng để lưu dữ liệu (số, văn bản) và truy cập qua tên.

1. Quy tắc đặt tên biến:
   - Biến bắt đầu bằng chữ cái (a–z, A–Z) hoặc _. Không bắt đầu bằng chữ số.
   - *Không* dùng ký tự đặc biệt (trừ _).
   - *Không* được trùng với **từ khóa** Python như: class, for, if...
   - Phân biệt chữ hoa, thường (age khác Age)
2. Khai báo và gán giá trị: Sử dụng = để gán.

```
name = "Lan"
age = 25
height = 1.65
is_student = True

print(name, age, height, is_student)
```

## 3. Toán tử (Operators)

Toán tử dùng để thực hiện phép tính hoặc so sánh. Phân loại như sau:

### 3.1 Toán tử số học

`+` (cộng), `-` (trừ), `*` (nhân), `/` (chia), `//` (chia lấy nguyên), `%` (chia lấy dư), `**` (lũy thừa).

```
a = 10
b = 3

print(a + b)  # 13
print(a - b)  # 7
print(a * b)  # 30
print(a / b)  # 3.333...
print(a // b) # 3 (chia lấy nguyên)
print(a % b)  # 1 (dư)
print(a ** b) # 1000 (10^3)
```

### 3.2 Toán tử so sánh

`==` (bằng), `!=` (khác), `>` (lớn hơn), `<` (bé hơn), `>=` (lớn bằng), `<=` (bé bằng)

```
x = 5
y = 10

print(x == y)  # False
print(x != y)  # True
print(x > y)   # False
print(x < y)   # True
```

### 3.3 Toán tử logic

`and` (và), `or` (hoặc), `not` (phủ định).

```
p = True
q = False

print(p and q)  # False
print(p or q)   # True
print(not p)    # False
```

### 3.4 Toán tử gán

`=` (gán), `+=` (cộng gán), `-=` (trừ gán), v.v.

```
z = 5
z += 3  # Tương đương z = z + 3
print(z)  # 8
```

### 3.5 Toán tử khác

Toán tử bit (`&`: AND bit, `|`: OR bit, `~`: NOT bit), toán tử thành viên (`in`: có trong, `not in`: không trong).

```
print(5 & 3)  # 1 (AND bit: 101 & 011 = 001)

fruits = ["apple", "banana"]
print("apple" in fruits)      # True
print("orange" not in fruits) # True
```

## 4. Các kiểu dữ liệu cơ bản (Data Types)

### 4.1 Kiểu số (Numbers)

- int: số nguyên (có thể âm, dương, hoặc 0).
- float: số thực (có dấu thập phân, lưu ý phải dùng dấu phẩy ‘,’).
- complex: số phức (dùng trong toán học cao cấp).

```
a = 20        # int
b = -7        # int âm
c = 3.14    # float
d = 2 + 3j    # complex
e = int(c)

print(type(a), type(b), type(c), type(d), type(e))
```

## 4.2 Chuỗi (String)

Chuỗi là tập hợp các ký tự, được đặt trong " " hoặc ' '.

```
chuoi = "Hello, Python!"
print(len(chuoi))  # 14
print(chuoi.upper())  # "HELLO, PYTHON!"
print(chuoi.split(','))  # ['Hello', ' Python!']
print("Xin chào, " + "IE221")  # Nối chuỗi
print(f"Giá trị: {5 + 3}")  # f-string
```

## 4.3 Danh sách (Lists)

- List là cấu trúc dữ liệu quan trọng nhất trong Python.
- Nó là một danh sách có thứ tự (ordered) và có thể thay đổi (mutable).
- Các phần tử trong list có thể là bất kỳ kiểu dữ liệu nào, thậm chí list chứa list khác.

### 4.3.1 Cú pháp tạo list

```
fruits = ["apple", "banana", "cherry"]
numbers = [1, 2, 3, 4, 5]
mixed = ["Lan", 25, True, 3.14]
empty = []    # list rỗng
```

### 4.3.2 Truy cập phần tử

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0])   # apple
print(fruits[-1])  # cherry (chỉ số âm = từ cuối lên)
```

### 4.3.3 Thay đổi phần tử

```
fruits[1] = "mango"
print(fruits)  # ['apple', 'mango', 'cherry']
```

### 4.3.4 Thêm, xóa phần tử

```
fruits.append("orange")      # thêm vào cuối
fruits.insert(1, "kiwi")     # thêm vào vị trí chỉ định
print(fruits)

fruits.remove("apple")       # xóa theo giá trị
fruits.pop(0)                # xóa theo index
print(fruits)
```

## 4.4 Tuple

Giống list, nhưng bất biến (immutable), Tuple không thay đổi được sau khi tạo và thường dùng cho dữ liệu cố định.

```
tup = (1, 2, 3)
a, b, c = tup  # Unpacking: a=1, b=2, c=3
print(a)  # 1
# So sánh với list: tup[0] = 4     Lỗi (immutable)
```

## 4.5 Từ điển (Dictionaries)

Từ điển lưu dữ liệu dạng key → value giống như "bảng tra cứu". Các key không trùng lặp.

```
student = {"name": "Lan", "age": 25}
print(student["name"])   # Lan

student['lop'] = 'IE221'  # Thêm
del student['age']  # Xóa

print(student.keys())  # dict_keys(['name', 'lop'])
print(student.values()) # dict_values(['Lan', 'IE221'])
print(student.get('lop')) # 'IE221'
```

## 4.6 Tập hợp (Sets)

Set là một tập hợp (collection) trong Python. Các đặc điểm chính:

- Không chứa phần tử trùng lặp
- Không có thứ tự cố định (unordered)
- Có thể thay đổi (mutable) – thêm hoặc xóa phần tử được
- Nhưng các phần tử trong Set phải là bất biến (immutable) → ví dụ: số, chuỗi, tuple được, nhưng list hoặc dict thì không.

### 4.6.1 Cách tạo Set

```
# Set rỗng
empty_set = set()

# Set với phần tử
fruits = {"apple", "banana", "cherry", "apple"}
print(fruits)  # {'apple', 'banana', 'cherry'} (apple chỉ xuất hiện 1 lần)
```

### 4.6.2 Các thao tác cơ bản

```
fruits = {"apple", "banana", "cherry"}
fruits.add("mango")
print(fruits)   # {'apple', 'banana', 'cherry', 'mango'}

fruits.remove("banana")
print(fruits)   # {'apple', 'cherry', 'mango'}

fruits.discard("orange")  # không báo lỗi nếu không có phần tử
```

```
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}

print(A | B)   # Hợp: {1, 2, 3, 4, 5, 6}
print(A & B)   # Giao: {3, 4}
print(A - B)   # Hiệu: {1, 2}
print(A ^ B)   # Phần tử chỉ có ở 1 tập hợp: {1, 2, 5, 6}
```

## 5. Bài tập

Hãy thực hiện các yêu cầu sau:

**Câu 1**: Tạo biến với các kiểu dữ liệu khác nhau:

`so_nguyen = 42, so_thuc = 3.14159, chuoi = "Kỹ thuật lập trình Python", danh_sach = [1, 2.5, "ba", True], tu_dien = 'mon_hoc': 'IE221', 'nam': 2023, tup = (1, 2, 3)`

Code mẫu gợi ý:

```
# Bài tập 1
so_nguyen = 42
print(f"Giá trị: {so_nguyen}, Kiểu: {type(so_nguyen)}")
# Tiếp tục tương tự cho các biến còn lại...
```

**Câu 2**: Sử dụng toán tử để tính diện tích hình chữ nhật

- Sử dụng toán tử số học để tính diện tích (dài * rộng).
- Nhập chiều dài và rộng từ người dùng bằng input().
- In kết quả bằng f-string, ví dụ: "Diện tích hình chữ nhật là: xx cm²".
- Xử lý input là số thực (sử dụng float(input(...))).

Code mẫu gợi ý:

```
# Bài tập 2
chieu_dai =
chieu_rong =
dien_tich =
print(f"Diện tích hình chữ nhật là: {dien_tich}")
```

**Câu 3**: Thao tác với list/dict/tuple theo yêu cầu bên dưới (Yêu cầu: Chạy code, thay MSSV/tên bằng thông tin cá nhân.)

- List: Tạo list gồm (MSSV, tên, lớp). Thêm phần tử năm học bằng append(). Slicing 2 phần tử đầu (vd: sinh_vien[0:2]) và in ra.
- Dict: Tạo dict gồm điểm Toán, Lý. Thêm 1 key Hóa vào dict đã tạo với điểm bằng 9. Tìm giá trị 'Toan' bằng key và in ra.
- Tuple: Tạo tuple thong_tin = ('IE221', 2023). Unpacking: lop, nam = thong_tin, in ra lop và nam.

Code mẫu gợi ý:

```
# Bài tập 3 - List
sinh_vien =
sinh_vien.append(...)
print("List đầy đủ:", sinh_vien)
print("Slicing 2 phần tử đầu:", sinh_vien[0:2])

# Dict
diem_so =
...
print("Dict đầy đủ:", diem_so)
print("Giá trị Toan:", diem_so.get('Toan'))

# Tuple
thong_tin = ('_G0 062G0 021', 20G0 023)
lop, nam = thong_tin
print("Lớp:", lop, "Năm:", nam)
```

**Câu 4**: Hãy trả lời các câu hỏi sau:

1. Phân tích từng case sau, nếu sai thì nêu lý do và sửa (chạy code để kiểm tra):
   - Case 1: age = 20; print(age).
   - Case 2: 1age = 20
   - Case 3: if = "condition"
   - Case 4: x = 5; x = "five"
   - Case 5: tup = (1,2,3); tup[0] = 4

2. Phân biệt list và tuple (Nêu giống và khác). Cho ví dụ minh họa so sánh bằng code.

3. Lập bảng so sánh các kiểu dữ liệu trong Python (Kiểu dữ liệu, Có thứ tự (Ordered), Cho phép trùng lặp, Mutable (có thể thay đổi), Cú pháp tạo, Truy cập phần tử, Ứng dụng chính, Ví dụ)

**Nộp bài:** Chạy code bài tập trên Colab, trả lời lý thuyết (bao gồm bảng so sánh), ghép lại thành 1 PDF. Đặt tên MSSV_HoVaTen_Lab2.pdf và nộp theo deadline trên Courses.