Chapter 2: Variables, expressions, and statements

01 Value and Type (Giá trị và Kiểu dữ liệu)

Value : kiểu dữ liệu mà chương trình xử lý
Type : là loại định dạng của một giá trị
+ str : string → chuỗi ký tự "Hello,haha,..."
+float : kiểu số thực
+int: integer-> kiểu số nguyên "1,2,3,4,5"

02 Variables

Biến : tức là tên dùng để lưu trữ giá trị trong bộ nhớ.
Vd :
+name: Hello world
+cuoi: haha

03 Variables names and Keywords

Quy tắc đặt tên :
+Không bắt đầu bằng "_"
+Không được trùng với từ khóa pythoon: if, else, for, while, try, import,...
Không chứa khoảng trắng, không bắt đầu bằng số

04 Expressions (Biểu thức) and Order of Operations (Thứ tự thực hiện)

Là tổ hợp biến và giá trị
Thứ tự thực hiện sẽ tuân theo PEMDAS:
Parentheses (ngoặc)
Exponents (lũy thừa)
Multiplication / Division
Addition / Subtraction

05 Operators and Operands (Toán tử và toán hạng)

Toán học : +,-,*,/,//,%,**
Chia lấy phần dư : %
Chia lấy phần nguyên: //
Lũy thừa: **

06 Statements

Mỗi một dòng thực thi chính là một câu lệnh (statements)

07 String Operations (Phép toán trên chuỗi),Asking for Input (Nhập dữ liệu), Comments (chú thích)

- Nối chuỗi : "Hello"+"world"= Hello world
- Lặp chuỗi: "Hello" *4= HelloHelloHelloHello
- Dùng "input()" để nhận chuỗi từ người dùng
- Phần comments nếu bắt đầu bằng "#" thì không được thực thi

08 Choosing Mnemonic Variable Names (Tên dễ nhớ),Debugging (Sửa lỗi)

Chọn tên dễ hiểu để có thể hiểu rõ được chức năng
Vd :
+hour: diễn tả thời gian
Sửa lỗi :

- Syntax Error: sai cú pháp
- Runtime Error: lỗi khi chạy
- Semantic Error: chương trình chạy nhưng ra kết quả sai logic.

Chapter 3: CONDITIONAL EXECUTION

01 Boolean Expressions – Biểu thức logic

Biểu thức đúng sai, trả về True hoặc False

== bằng
!= khác
> lớn hơn
< nhỏ hơn
>= lớn hơn hoặc bằng
<= nhỏ hơn hoặc bằng

```
x=10
print(x>5)
print(x<3)
```

```
PS D:\HocLapTrinh> & C:/User
True
False
PS D:\HocLapTrinh>
```

02 Logical Operators – Toán tử logic

And: đúng nếu cả hai cùng đúng
Or: Đúng ít nhất 1 trong 2 điều kiện
Not: Hàm đảo ngược logic

```
x=10
x>5 and x<20
print(x)
```

```
10
PS D:\HocLapTrinh>
```

```
C3.py > ...
x=200
x>1 or x>300
print(x)
```

```
PS D:\HocLapTrinh> &
200
PS D:\HocLapTrinh>
```

```
x=-100
not(x>0)
print(x)
```

```
PS D:\HocLapTrin
-100
PS D:\HocLapTrin
```

03 Conditional Execution – Câu lệnh if

Câu điều kiện IF sẽ chỉ chạy nếu điều kiện đúng, nếu sai sẽ bỏ qua khối lệnh

```
x=1
if x>7:
    y=x-10
print(x)
print(y)
```

```
1
Traceback (most recent call last):
  File "d:\HocLapTrinh\C3.py", line 5, in <module>
    print(y)
          ^
NameError: name 'y' is not defined
```

```
C3.py > ...
x=17
if x>7:
    y=x-10
print(y)
```

```
PS D:\HocLapTrinh>
7
PS D:\HocLapTrinh>
```

04 Alternative Execution – if...else

Đây là câu lệnh theo nhánh đúng sai nếu điều kiện đầu sai thì sẽ xét tiếp điều kiện tiếp theo

```
x=5+10-2+20
if x>20:
    print("Ket qua chinh xac")
else :
    print("Ket qua da sai")
```

```
Ket qua chinh xac
```

```
x=5+10-2+20
if x>50:
    print("Ket qua chinh xac")
else :
    print("Ket qua da sai")
```

```
Ket qua da sai
```

05 Chained Conditionals – if...elif...else

Dùng cho trường hợp cần xét nhiều điều kiện hơn

```
x= float(input("Hay nhap so diem cua ban:"))
if x>9:
    print("Excellent")
elif x<9 and x>7:
    print("Good")
elif x<5:
    print("Bad")
```

```
Hay nhap so diem cua ban:9.5
Excellent
PS D:\HocLapTrinh> & C:/User
Hay nhap so diem cua ban:8
Good
PS D:\HocLapTrinh> & C:/User
Hay nhap so diem cua ban:3.5
Bad
```

06 Nested Conditionals – Điều kiện lồng nhau

Một điều kiện IF nằm trong IF

```
x= float(input("Hay con so cua ban:"))
if x>9:
    if x<20:
        print("Gia tri X nam giua 9 và 20")
```

```
Hay con so cua ban:17
Gia tri X nam giua 9 và 20
```

07 Try / except – Bắt lỗi khi chạy

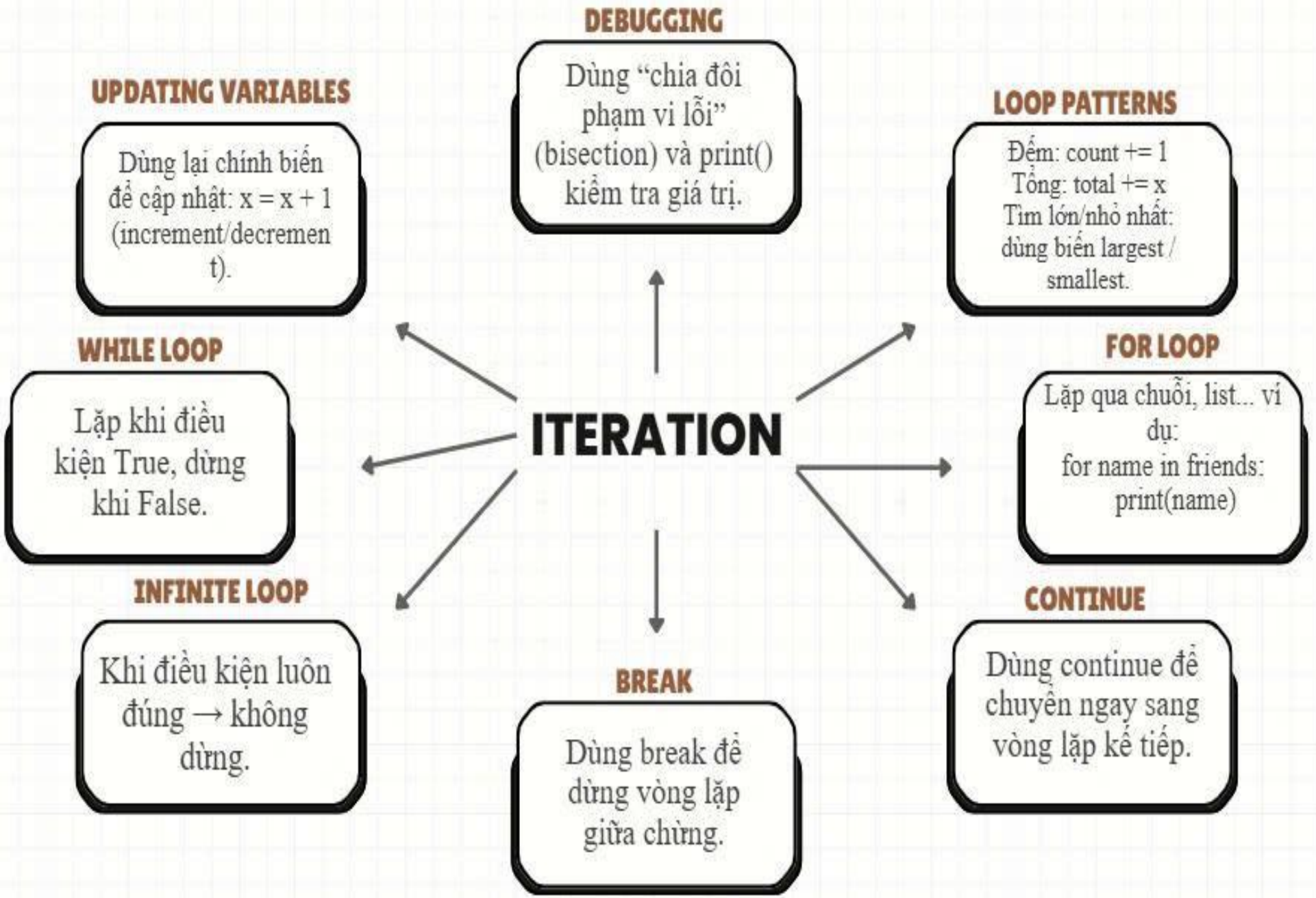
Dùng để xử lý lỗi nhập sai hoặc tính toán sai

```
try:
    x= float(input("Hay nhap gia tri: "))
except:
    print("Error")
```

```
Hay nhap gia tri: chin
Error
PS D:\HocLapTrinh> & C:/
Hay nhap gia tri: 100
```

08 Debugging – Sửa lỗi logic

- Dùng print() để kiểm tra giá trị trong điều kiện.
- Kiểm tra indentation (thụt lề) và dấu : khi viết if, else, elif.


ITERATION
DEBUGGING
Dùng "chia đôi phạm vi lỗi" (bisection) và print() kiểm tra giá trị.
UPDATING VARIABLES
Dùng lại chính biến để cập nhật: x = x + 1 (increment/decrement).
LOOP PATTERNS
Đếm: count += 1
Tổng: total += x
Tìm lớn/nhỏ nhất: dùng biến largest / smallest.
WHILE LOOP
Lặp khi điều kiện True, dừng khi False.
FOR LOOP
Lặp qua chuỗi, list... ví dụ:
for name in friends:
print(name)
INFINITE LOOP
Khi điều kiện luôn đúng → không dừng.
BREAK
Dùng break để dừng vòng lặp giữa chừng.
CONTINUE
Dùng continue để chuyển ngay sang vòng lặp kế tiếp.

FUNCTIONS

Function Calls

Hàm là nhóm lệnh có tên thực hiện 1 tác vụ. Gọi hàm bằng cách viết tên + đối số

Built-in Functions

Python có sẵn nhiều hàm như max(), min(), len().

Type Conversion

int(), float(), str() chuyển đổi giữa các kiểu dữ liệu.

Random Numbers

Dùng import random; ví dụ random.random(), random.randint(1,10).

Why use Functions

Giúp tái sử dụng, gọn cod[illegible]ễ [illegible]ug

Math Function

Dùng import math với math.log10(), math.sin(), math.pi, v.v.

Return

Dùng return để trả kết quả; hàm có return là fruitful, không có là void.

Parameters & Arguments

Hàm nhận giá trị truyền vào (arguments) và gán cho tham số (parameters).

Defining Functions

Dùng def, gồm phần header (tên + tham số) và body (lệnh thụt vào).

Chapter 6
Strings

01 A string is a sequence

- Chuỗi gồm nhiều ký tự, mỗi ký tự có chỉ số (index) bắt đầu từ 0.
- fruit[0] → ký tự đầu tiên ('b').
- fruit[-1] → ký tự cuối ('a').

```
fruit = 'banana'
print(fruit[0])  # b
```

02 Getting the length of a string using len

- Trả về số ký tự trong chuỗi.
- Thường dùng khi duyệt qua chuỗi bằng chỉ số.

```
len('banana')  # 6
```

Thường kết hợp với vòng lặp while:

```
index = 0
while index < len(fruit):
    print(fruit[index])
    index += 1
```

03 Traversal through a string with a loop

Dùng for ngắn gọn hơn, ít lỗi hơn.

```
for ch in 'banana':
    print(ch)
```

04 String slices

Chỉ lấy đến end - 1, không bao gồm ký tự end.

```
s = 'Monty Python'
print(s[0:5])  # Monty
print(s[6:])   # Python
```

Nếu bỏ start → mặc định là 0, bỏ end → đến cuối chuỗi:

05 Strings are immutable

Không thể thay đổi ký tự trực tiếp

```
greet = 'Hello'
# greet[0] = 'J' ✕
greet = 'J' + greet[1:]  # 'Jello'
```

06 Looping and counting

```
word = 'banana'
count = 0
for letter in word:
    if letter == 'a':
        count += 1
print(count)
```

Có thể dùng word.count('a') cho ngắn.

07 The in operator

```
'a' in 'banana'     # True
'seed' in 'banana'  # False
```

Dùng để kiểm tra chuỗi con có trong chuỗi chính không.

08 String comparison

So sánh theo thứ tự bảng chữ cái (lexicographical order).
Chữ hoa < chữ thường theo mã ASCII.

```
word = 'Pineapple'
if word < 'banana':
    print("comes before banana")
```

09 string methods

```
['capitalize', 'casefold', 'center', 'count', 'encode',
'endswith', 'expandtabs', 'find', 'format', 'format_map',
'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit',
'isidentifier', 'islower', 'isnumeric', 'isprintable',
'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower',
'lstrip', 'maketrans', 'partition', 'replace', 'rfind',
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',
'split', 'splitlines', 'startswith', 'strip', 'swapcase',
'title', 'translate', 'upper', 'zfill']
```

10 Phân tích chuỗi (Parsing)

Dùng find() và slicing để trích thông tin:

```
data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
atpos = data.find('@')
sppos = data.find(' ', atpos)
host = data[atpos+1 : sppos]
print(host)  # uct.ac.za
```

Chapter 7 FILES

1.Persistence

Biến tạm mất khi tắt máy, file giúp lưu dữ liệu vĩnh viễn (trên ổ cứng).

Tệp văn bản (.txt), tệp dữ liệu (.csv),...

2. Opening files

```
fhand = open('mbox.txt')
print(fhand)
```

- Trả về đối tượng file handle.
- Mặc định mode 'r' → đọc.

files handle một đối tượng cho phép bạn đọc hoặc ghi vào tệp.

3. Text files and lines

mở, đọc, duyệt, lọc và đếm các dòng trong file văn bản bằng Python.

Mỗi dòng kết thúc bằng ký tự \n.
→ Dùng .strip() để bỏ dòng trống:

4. Reading files

Dùng phương thức read() trên tay cầm tệp.

Trả về toàn bộ nội dung dưới dạng một chuỗi lớn.

```
fhand = open('mbox.txt')
for line in fhand:
    print(line.strip())
```

5. Searching through a file

- Sử dụng vòng lặp để duyệt từng dòng và dùng cấu trúc if với các phương thức chuỗi (startswith(), find(), in) để lọc các dòng phù hợp.
- Dùng startswith() để lọc những dòng cần.

```
fhand = open('mbox.txt')
for line in fhand:
    if line.startswith('From:'):
        print(line.strip())
```

6. Letting the user choose the file name

Cho phép người dùng nhập tên tệp (input()).

Quan trọng là phải đóng tệp sau khi sử dụng: handle.close().

```
fname = input('Enter file name: ')
fhand = open(fname)
count = 0
for line in fhand:
    if line.startswith('Subject:'):
        count += 1
print('There were', count, 'subject lines in', fname)
```

7. Using try, except, and open

Dùng khối try/except để xử lý trường hợp người dùng nhập tên tệp không hợp lệ hoặc tệp không tồn tại, tránh làm sập chương trình.

```
fname = input('Enter file name: ')
try:
    fhand = open(fname)
except:
    print('File cannot be opened:', fname)
    quit()
```

8. Writing files

'w': ghi đè.

'a': nối thêm vào cuối file.

Luôn đóng file sau khi ghi (close()).

```
fout = open('output.txt', 'w')
print('Hello file!', file=fout)
fout.close()
```

LISTS

.KHÁI NIỆM:
-List là một cấu trúc dữ liệu cho phép lưu nhiều giá trị trong một biến bên trong[] và phân cách bởi dấu phẩy.
-List có thể thay đổi và có thể rỗng.

2. CÁC PHƯƠNG THỨC CỦA LIST
-append (X): thêm một phần tử mới vào cuối danh sách
-extend(x): lấy một danh sách làm đối số và thêm tất cả các phần tử
-+insert(i,x)=>chèn phần tử vào một vị trí i.
+sort()=>sắp xếp tăng dần
+count(x)=>đếm số lần xuất hiện
+reverse()=>đảo ngược thứ tự
+index(x)=> trả về vị trí đầu tiên của x

```
test='one two there'
abc=test.split()
print(abc)
```

. CÁC HÀM TÍCH HỢP CỦA LIST
- max()=> phần tử lớn nhất
- min()=> phần tử nhỏ nhất
- len()=> đếm số phần tử
- sum()=> tính tổng các phần tử
- in và not in để kiểm tra các phần tử => true or fale

. CÁC PHƯƠNG THỨC ĐỂ XÓA CỦA LIST
--pop(x)=>xóa và trả về phần tử x (nếu không có x pop sẽ tự động trả về phần tử cuối)
-remove(x)=>xóa phần tử đầu tiên có giá trị x
-del my_list[i]=>xóa phần tử tại vị trí i
-clear=> để xóa toàn bộ list..

. DANH SÁCH VÀ CHUỖI
-split()=>tách strin thành list
- join() => nối từ list lại thành string.dùng sep làm dấu nối

.CÁC PHƯƠNG THỨC CỦA LIST
- append(x) => sẽ thêm một phần tử mới vào cuối danh sách
- extend(x)=> lấy một danh sách làm đối số và thêm các phần tử
+insert(i,x)=>chèn phần tử vào một vị trí i.
- sort(x)=>sắp xếp tăng dần
- count(x)=>đếm số lần xuất hiện
- reverse()=>đảo ngược thứ tự
- index(x)=> trả về vị trí đầu tiên của x

```
my_list=[0,1,2,3,4,5]
print(my_list[::2]) # => 0,2,4
```

```
my_list=['a','b','c','d']
print(my_list[0]) # a
print(my_list[-1]) # d
```

Dictionaries

KHÁI NIỆM
- Từ điển giống như một danh sách nhưng tổng quát hơn.
- Từ điển là một kiểu dữ liệu lưu trữ các cặp khóa - giá trị có thể thay đổi và khong có thứ tự.

TỪ ĐIỂN VÀ TẬP TIN
- Chuyển đổi từ điển thành định dạng chuỗi hoặc JSON rồi ghi vào tập tin
- Đọc dữ liệu từ tập tin rồi chuyển thành kiểu dữ liệu từ điển

VÒNG LẶP VÀ TỪ ĐIỂN
- Lặp qua các khóa sử dụng cú pháp "for key in dictionary:"
- Lặp qua các giá trị sử dụng cú pháp "for value in dictionary.values():"
- Lặp qua các cặp khóa và giá trị sử dụng phương thức ".items()" để trả về đối tượng lặp

CÁC CÚ PHÁP CỦA TỪ ĐIỂN
- {} : Cú pháp của từ điển.
- Hàm dict():=> sử dụng để tạo từ điển, thường dùng để tạo từ các đối tượng có thể lặp lại.

```
my_dict = {"a": 1, "b": 2, "c": 3}
for key in my_dict:
    print(f"Khóa: {key}, Giá trị: {my_dict[key]}")
```

```
my_dict = {"a": 1, "b": 2, "c": 3}
for value in my_dict.values():
    print(f"Giá trị: {value}")
```

CÁC PHƯƠNG THỨC CƠ BẢN CỦA TỪ ĐIỂN
- Update(), pop(), clear() : dùng để sửa đổi từ điển
-Fromkeys() : để tạo từ điển mới
- Keys(), values(), items(), get() : để truy cập và lặp qua dữ liệu

```
>>> inventory = dict.fromkeys(["apple", "orange", "banana", "mango"], 0)

>>> inventory
{'apple': 0, 'orange': 0, 'banana': 0, 'mango': 0}
```