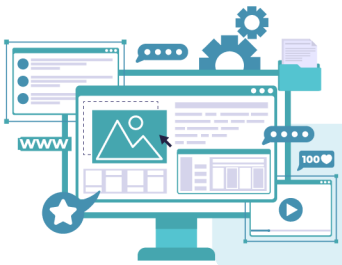
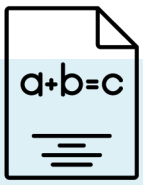


CHAPTER 2



Values and Types

Mỗi giá trị có một kiểu (type) nhất định

```
>>> type(3.2)
<class 'float'>
```

Integer (int): *Số nguyên*; **Floating (float)**: *Số thực*; **String (str)**: *Chuỗi ký tự*

có thể dùng hàm type() để xác định kiểu

Variables and Assignment

Biến (Variable): Một cái tên tham chiếu đến một giá trị.

Phép gán (Assignment): Gán một giá trị cho một biến, sử dụng toán tử bằng "="

```
n = 17
pi = 3.1415926535897931
```

01

Quy tắc đặt tên biến

Tên **có thể chứa chữ cái, số và dấu gạch dưới**.

Phải bắt đầu bằng một **chữ cái** hoặc dấu gạch dưới.

Không được là **từ khóa (keyword) của Python**

Python phân biệt chữ hoa và chữ thường

02

Statements, Expressions, and Operators

```
>>> minute = 59
>>> minute/60
0.9833333333333333
```

Biểu thức (Expression): Sự kết hợp của các giá trị, biến và toán tử mà khi được tính toán sẽ cho ra một giá trị

Câu lệnh (Statement): câu lệnh gán x = 10 hoặc câu lệnh print()

```
x = 2
print(x)
```

Toán tử số học (Arithmetic Operators):

+ - * / *Cộng trừ nhân chia*

****** *Lũy thừa* **%** *Chia lấy dư* **//** *Chia lấy nguyên*

Dấu ngoặc đơn → Lũy thừa → Nhân, Chia → Cộng, Trừ

Toán tử chuỗi: Toán tử + được dùng để nối chuỗi

03

```
>>> first = 10
>>> second = 15
>>> print(first+second)
25
>>> first = '100'
>>> second = '150'
>>> print(first + second)
100150
```

Input and Type Conversions

input(): Lấy dữ liệu từ người dùng và được trả về dưới dạng chuỗi (str).

```
name = input('Enter your name: ')
```

Nếu muốn thực hiện các phép toán số học với dữ liệu lấy từ input(), bạn phải chuyển sang type số int hay float

```
hours = input(prompt)
fhours = float(hours)
```

04

COMMENTS

Bình luận/Ghi chú: Các dòng bắt đầu bằng dấu # (hash mark) là các ghi chú dành cho người lập trình và bị Python bỏ qua hoàn toàn khi chạy chương trình.

Chương 3: Cấu trúc rẽ nhánh (điều kiện)

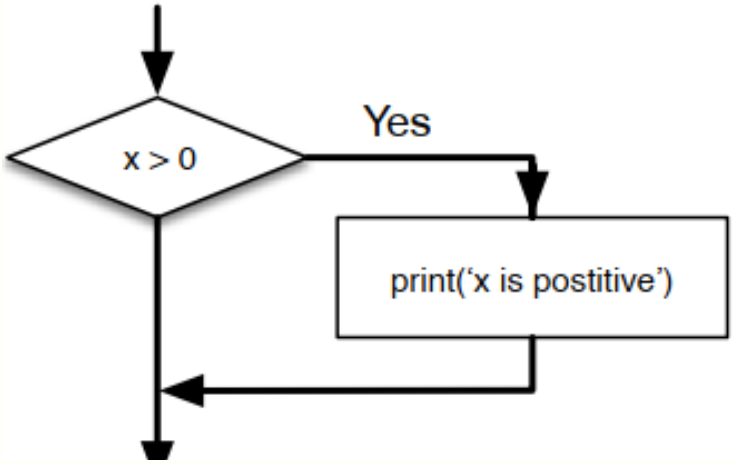
Khái niệm	Toán tử	Ví dụ Python	Kết quả	Mô tả
Bằng nhau	**==**	**5==5**	TRUE	So sánh hai giá trị có bằng nhau hay không
Không bằng	!=	5!=6	TRUE	So sánh hai giá trị có khác nhau hay không
Lớn hơn	>	x>10	False/True	
Nhỏ hơn	<	y<100	False/True	
Lớn hơn hoặc bằng	>=	**a >= 5**	False/True	
Nhỏ hơn hoặc bằng	<=	b <= 0	False/True	
Tính đồng nhất	is	x is y	False/True	Kiểm tra xem hai biến có tham chiếu đến cùng một đối tượng hay không
Không đồng nhất	is not	x is not y	False/True	

CÂU LỆNH ĐIỀU KIỆN

ĐƠN GIẢN:

THỰC HIỆN KHỐI LỆNH NẾU ĐIỀU KIỆN LÀ TRUE.

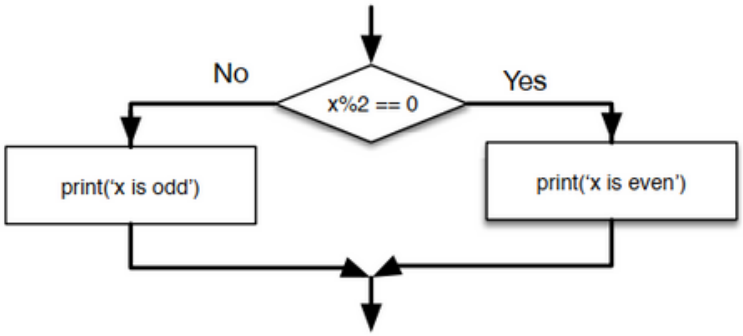
```
if x > 0 :
    print('x is positive')
```



HAI NHÁNH

THỰC HIỆN KHỐI LỆNH IF NẾU TRUE, HOẶC ELSE NẾU FALSE

```
if x%2 == 0 :
    print('x is even')
else :
    print('x is odd')
```

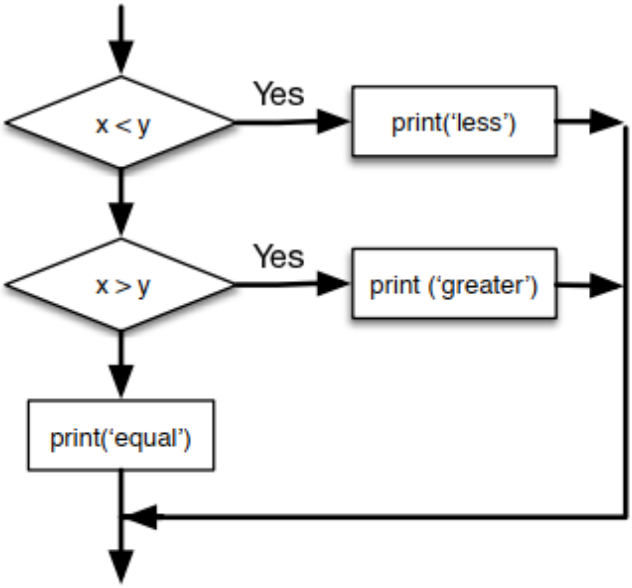


NHIỀU NHÁNH

CHO PHÉP KIỂM TRA NHIỀU ĐIỀU KIỆN MỘT CÁCH TUẦN TỰ.

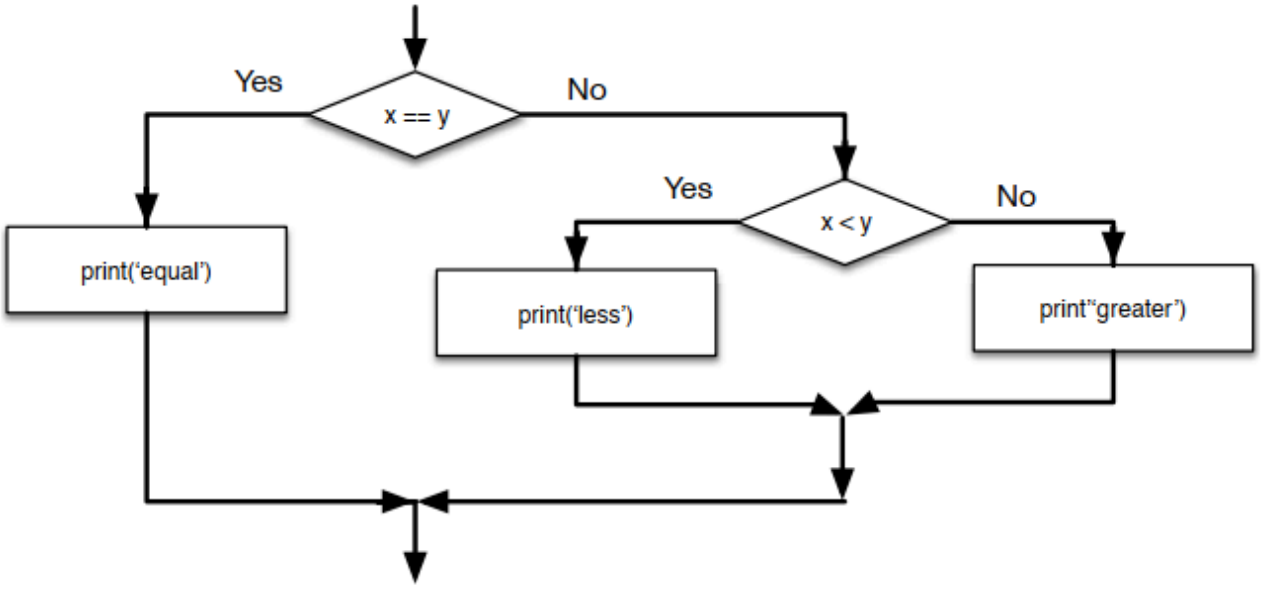
NHIỀU CHIỀU:

```
if x < y:
    print('x is less than y')
elif x > y:
    print('x is greater than y')
else:
    print('x and y are equal')
```



LỒNG NHAU:

```
if x == y:
    print('x and y are equal')
else:
    if x < y:
        print('x is less than y')
    else:
        print('x is greater than y')
```



CẤU TRÚC TRY/EXCEPT

MÔ TẢ: MỘT CƠ CHẾ BẢO VỆ CÂU LỆNH CÓ KHẢ NĂNG GÂY LỖI (NHƯ CHUYỂN ĐỔI KIỂU DỮ LIỆU). NẾU LỖI XẢY RA, CHƯƠNG TRÌNH SẼ NHẢY SANG KHỐI EXCEPT THAY VÌ DỪNG LẠI.

```
inp = input('Enter Fahrenheit Temperature:')
try:
    fahr = float(inp)
    cel = (fahr - 32.0) * 5.0 / 9.0
    print(cel)
except:
    print('Please enter a number')
```

Logical Operators

TOÁN TỬ	MÔ TẢ	VÍ DỤ
AND	TRUE NẾU CẢ HAI ĐIỀU KIỆN ĐỀU TRUE.	`x > 0 and x < 10`
OR	TRUE NẾU MỘT TRONG HAI ĐIỀU KIỆN LÀ TRUE.	`n%2 == 0 or n%3 == 0`
NOT	ĐẢO NGƯỢC KẾT QUẢ CỦA BIỂU THỨC BOOLEAN.	`not (x > 10)`

Chương 4
FUNCTIONS

PYTHON FUNCTIONS

Có hai loại hàm trong Python.
- Hàm tích hợp sẵn được cung cấp như một phần của Python
- print(), input(), type(), float(), int() ...
- Hàm mà chúng ta tự định nghĩa và sau đó sử dụng

Chúng ta coi tên các hàm tích hợp sẵn như là các từ khóa "mới" (tức là, chúng ta tránh sử dụng chúng làm tên biến)

Type Conversions

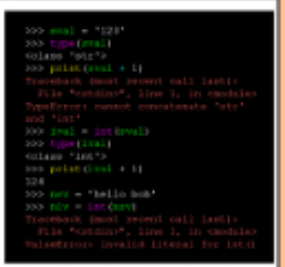
Khi bạn đưa một số nguyên và số thực vào một biểu thức, số nguyên sẽ được chuyển đổi ngầm định thành số thực.
Bạn có thể kiểm soát điều này bằng các hàm tích hợp sẵn int() và float().

Function Definition

Trong Python, một hàm là một đoạn mã có thể tái sử dụng, nhận đối số làm đầu vào, thực hiện một số phép toán và sau đó trả về một kết quả hoặc nhiều kết quả.
Chúng ta định nghĩa một hàm bằng cách sử dụng từ khóa def.
Chúng ta gọi/khởi tạo hàm bằng cách sử dụng tên hàm, dấu ngoặc đơn và các đối số trong một biểu thức.

String Conversions

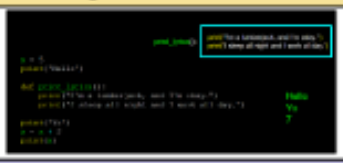
Bạn cũng có thể sử dụng int() và float() để chuyển đổi giữa chuỗi và số nguyên.
Bạn sẽ gặp lỗi nếu chuỗi không chứa ký tự số.



Building our Own Functions

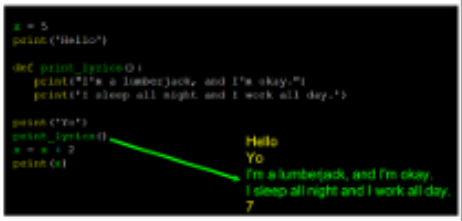
Chúng tôi tạo một hàm mới bằng cách sử dụng từ khóa def theo sau là các tham số tùy chọn trong dấu ngoặc đơn.
Chúng tôi thụt lề phần thân của hàm.
Điều này định nghĩa hàm nhưng không thực thi phần thân của hàm.

Building our Own Functions



Definitions and Uses

Khi chúng ta đã định nghĩa một hàm, chúng ta có thể gọi (hoặc kích hoạt) nó nhiều lần tùy thích.
Đây là mẫu lưu trữ và tái sử dụng.



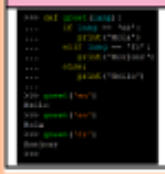
Arguments

Một đối số là giá trị mà chúng ta truyền vào hàm như là đầu vào khi gọi hàm
Chúng ta sử dụng đối số để có thể chỉ định cho hàm thực hiện các loại công việc khác nhau khi chúng ta gọi nó vào những thời điểm khác nhau
Chúng ta đặt các đối số trong dấu ngoặc đơn sau tên của hàm

Parameters

Một tham số là một biến mà chúng ta sử dụng trong định nghĩa hàm. Nó là một "tay cầm" cho phép mã trong hàm truy cập các đối số cho một lần gọi hàm cụ thể.

Parameters



CHƯƠNG 5

LOOPS AND ITERATION

REPEATED STEPS

Các vòng lặp (các bước lặp lại) có các biến lặp lại thay đổi mỗi lần qua vòng lặp. Thường thì các biến lặp lại này đi qua một chuỗi số.

Program:

```
n = 5
while n > 0 :
    print(n)
    n = n - 1
print('Blastoff!')
print(n)
```

Output:

```
5
4
3
2
1
Blastoff!
0
```

A SIMPLE DEFINITE LOOP

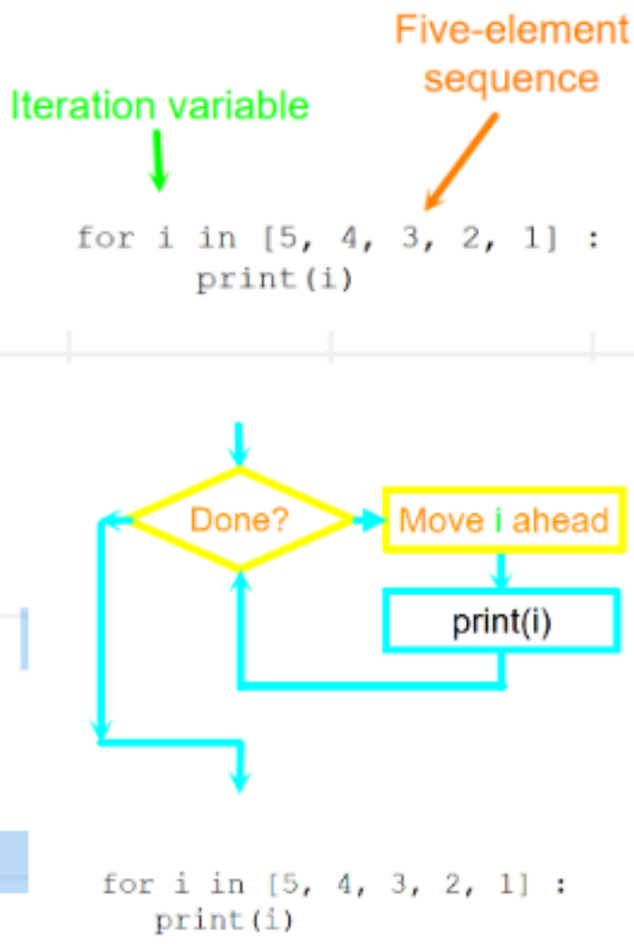
Các vòng lặp xác định (vòng lặp for) có các biến lặp rõ ràng thay đổi mỗi lần qua vòng lặp. Các biến lặp này di chuyển qua chuỗi hoặc tập hợp.

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Blastoff!')
```

Biến lặp "lặp lại" qua chuỗi (tập hợp có thứ tự)

Khối (nội dung) mã được thực thi một lần cho mỗi giá trị trong chuỗi

Biến lặp di chuyển qua tất cả các giá trị trong chuỗi



BREAKING OUT OF A LOOP

Câu lệnh break kết thúc vòng lặp hiện tại và nhảy đến câu lệnh ngay sau vòng lặp

Nó giống như một bài kiểm tra vòng lặp có thể xảy ra ở bất kỳ đâu trong thân của vòng lặp.

```
while True:
    line = input('> ')
    if line == 'done' :
        break
    print(line)
print('Done!')
```

```
> hello there
hello there
> finished
finished
> done
Done!
```

Finishing an Iteration with continue

Câu lệnh continue kết thúc vòng lặp hiện tại và nhảy lên đầu vòng lặp để bắt đầu vòng lặp tiếp theo.

```
while True:
    line = input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```

```
> hello there
hello there
> # don't print this
> print this!
print this!
> done
Done!
```

```
while True:
    line = raw_input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```

FINDING THE LARGEST VALUE

Chúng tôi tạo một biến chứa giá trị lớn nhất mà chúng tôi đã thấy cho đến nay. Nếu số hiện tại mà chúng tôi đang xem lớn hơn, nó sẽ là giá trị lớn nhất mới mà chúng tôi đã thấy cho đến nay.

```
largest_so_far = -1
print('Before', largest_so_far)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num > largest_so_far :
        largest_so_far = the_num
    print(largest_so_far, the_num)

print('After', largest_so_far)
```

```
python largest.py
Before -1
9 9
41 41
41 12
41 3
74 74
74 15
After 74
```

COUNTING IN A LOOP

Để đếm số lần chúng ta thực hiện một vòng lặp, chúng ta giới thiệu một biến đếm bắt đầu từ 0 và cộng thêm một vào nó mỗi lần qua vòng lặp.

```
zork = 0
print('Before', zork)
for thing in [9, 41, 12, 3, 74, 15] :
    zork = zork + 1
    print(zork, thing)
print('After', zork)
```

```
python countloop.py
Before 0
1 9
2 41
3 12
4 3
5 74
6 15
After 6
```

INDEFINITE LOOPS

Các vòng lặp while được gọi là "vòng lặp không xác định" vì chúng tiếp tục cho đến khi một điều kiện logic trở thành False.

Các vòng lặp mà chúng ta đã thấy cho đến nay khá dễ để kiểm tra xem chúng có kết thúc hay không, hoặc nếu chúng sẽ là "vòng lặp vô hạn".

Đôi khi, thật khó để chắc chắn rằng một vòng lặp sẽ kết thúc.

Definite Loops

Thường thì chúng ta có một danh sách các mục trong một tệp - thực chất là một tập hợp hữu hạn các thứ.

Chúng ta có thể viết một vòng lặp để chạy vòng lặp một lần cho mỗi mục trong một tập hợp bằng cách sử dụng cấu trúc for của Python.

Các vòng lặp này được gọi là "vòng lặp xác định" vì chúng thực hiện một số lần chính xác.

Chúng ta nói rằng "các vòng lặp xác định lặp qua các thành viên của một tập hợp."

THE IS AND IS NOT OPERATORS

Python có toán tử is có thể được sử dụng

trong các biểu thức logic

Có nghĩa là "là giống nhau"

Tương tự như, nhưng mạnh hơn ==

is not cũng là một toán tử logic

```
smallest = None
print('Before')
for value in [3, 41, 12, 9, 74, 15] :
    if smallest is None :
        smallest = value
    elif value < smallest :
        smallest = value
    print(smallest, value)

print('After', smallest)
```

FINDING THE SMALLEST VALUE

Chúng ta vẫn có một biến là giá trị nhỏ nhất

cho đến nay. Lần đầu tiên qua vòng lặp, giá trị

nhỏ nhất là None, vì vậy chúng ta lấy giá trị

đầu tiên làm giá trị nhỏ nhất.

```
smallest = None
print('Before')
for value in [9, 41, 12, 3, 74, 15] :
    if smallest is None :
        smallest = value
    elif value < smallest :
        smallest = value
    print(smallest, value)
print('After', smallest)
```

```
python smallest.py
Before
9 9
9 41
9 12
3 3
3 74
3 15
After 3
```

FINDING THE AVERAGE IN A LOOP

Một giá trị trung bình chỉ kết hợp các mẫu đếm và

tổng, sau đó chia khi vòng lặp hoàn tất.

```
count = 0
sum = 0
print('Before', count, sum)
for value in [9, 41, 12, 3, 74, 15] :
    count = count + 1
    sum = sum + value
    print(count, sum, value)
print('After', count, sum, sum / count)
```

```
python averageloop.py
Before 0 0
1 9 9
2 50 41
3 62 12
4 65 3
5 139 74
6 154 15
After 6 154 25.666
```

CHAPTER 6
STRING

01.

INTRODUCTION

Chuỗi là một chuỗi các ký tự

Chuỗi được đánh dấu bởi “”

Dấu + nghĩa là "**nối**"

Một chuỗi chứa số, nó vẫn là một chuỗi

Chuỗi chỉ chứa số thì có thể **ép kiểu thành số** bằng cách dùng int()

02.

INPUT - INDEX

input()
khi input() luôn nhập vào kiểu str nếu muốn dùng số cần ép kiểu sang số

Index bắt đầu từ 0

[]
dấu ngoặc vuông [] để lấy ký tự tại thứ tự đó, bắt đầu từ 0

len()
trả về số lượng ký tự có trong một chuỗi.

```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print(letter)
a

>>> x= len(fruit)
>>> print(x)
6
```

Chuỗi là Bất biến không thể thay đổi một chuỗi mà phải gán biến mới hay tạo ra một chuỗi mới.

03.

VÒNG LẶP

Dùng lập kết hợp **index - len - if ...** để đi qua từng kí tự có thể để print() hay kiểm tra ...

```
index = 0
while
  index < len(fruit) :
  letter = fruit[index]
  print(letter)
  index = index+ 1
>>>b
>>>a
>>>...
>>>a
```

04.

SLICING - IN

s[n:m]
Cắt lát chuỗi ký tự từ mục n cho đến (nhưng không bao gồm) mục m

in
là một toán tử Boolean (trả về True hoặc False). Cho biết một chuỗi con của một chuỗi khác hay không

```
'n' in fruit
>>> True
'm' in fruit
>>> False
```

05.

STRING LIBRARY

== ; < ; >
So sánh chuỗi
str.capitalize()
str.center()
lower() - upper()
Chữ thường / hoa
find(substring)
Tìm vị trí chuỗi con
replace(old, new)
Thay thế
strip() - rstrip()- lstrip()
‘Xóa ‘
startswith()-endswith()
bắt đầu bằng

PYTHON

CHATER 7: FILES

1 PERSISTENCE

Bộ nhớ chính (Main memory) dùng cho các biến tạm thời và truy cập rất nhanh, nhưng sẽ bị mất khi chương trình kết thúc.

Bộ nhớ thứ cấp (Secondary memory) chậm hơn, nhưng nó bền bỉ hơn và dữ liệu được lưu trữ trong các tệp tin ngay cả khi chương trình kết thúc hay ca tắt nguồn.

2 OPENING A FILES

Để đọc hoặc ghi một tệp, trước tiên phải "mở".

Phương thức open() sẽ trả về một "file handle" một dạng đối tượng để tương tác tệp.

```
handle = open(filename, mode)
```

filename: Là một chuỗi (string) chứa tên của tệp.
mode:
' r': Chế độ Read (Đọc).
'w': Chế độ Write (Ghi).

3 READING FILES

- Một tệp văn bản (text file) là một chuỗi các ký tự được lưu trữ.
- Từ góc nhìn của Python, một tệp văn bản là một chuỗi các dòng và xử lý nó theo từng dòng một.
- Vì thế cách đơn giản nhất để đọc một tệp là sử dụng vòng lặp for.
- Vòng lặp for sẽ tự động xử lý tệp theo từng dòng hoặc dùng.read().

4 READING AND SEARCHING THROUGH A FILES

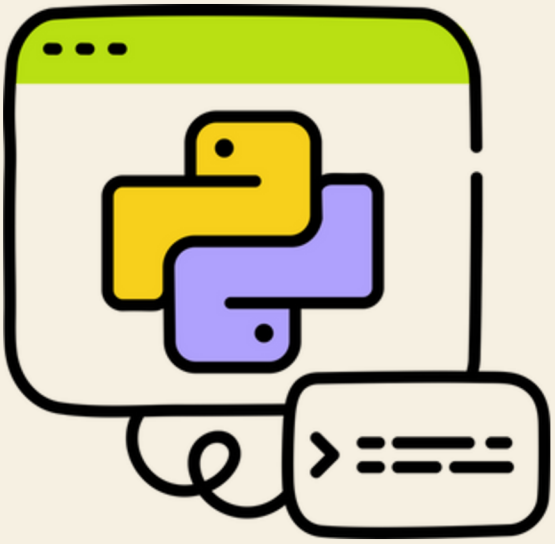
Có thể kết hợp việc đọc tệp theo từng dòng với các phương thức của chuỗi (như .startswith(),.find(), .strip , ... để tìm kiếm thông tin cụ thể.

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

5 CHOOSE THE FILE NAME AND WRITING FILES

- input() để hỏi người dùng muốn mở tệp nào.
- Sử dụng try, except để xử lý khi tệp không tồn tại.
- Để write một tệp cần open() ở mode "w" và sử dụng .write() trên line ; "/n" để xuống dòng và .close() lại để lưu.

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



PYTHON

CÁC PHƯƠNG THỨC VÀ CÚ PHÁP

Phép toán:

+ (Nối): Nối hai danh sách lại với nhau.

*** (Lặp lại):** Lặp lại các phần tử của danh sách nhiều lần.

Cắt lát (Slices): Cú pháp cắt lát [n:m] hoạt động giống hệt như với chuỗi,

Lists and Functions

len(): Trả về số lượng phần tử.

max(): Trả về phần tử lớn nhất.

min(): Trả về phần tử nhỏ nhất.

sum(): Trả về tổng của tất cả các phần tử *(khi danh sách chỉ chứa số).*

Lists and Strings

.split(): Tách một chuỗi thành một danh sách các chuỗi con

.join(): Nối các phần tử của một danh sách thành một chuỗi duy nhất

delimiter phân tách các giá trị từ dấu chấm , phẩy ...

CHƯƠNG 8: LISTS

DANH SÁCH LÀ MỘT CHUỖI

Giống như chuỗi (string), danh sách (list) là một chuỗi các giá trị và các giá trị trong một danh sách có thể thuộc bất kỳ kiểu nào

Các giá trị được đặt trong dấu ngoặc vuông [] và cách nhau bằng dấu phẩy.

```
[10, 20, 30], ['apple', 'banana', 'cherry']
```

DANH SÁCH LÀ KHẢ BIẾN

Với Chuỗi (String) là bất biến

Còn Danh sách (List) là khả biến. Có thể thay đổi nội dung của nó sau khi nó được tạo ra.

```
numbers = [1, 2, 3]
numbers[0] = 99
numbers bây giờ là [99, 2, 3]
```

VÒNG LẶP

- có thể dùng vòng lặp for để duyệt qua và xử lý từng phần tử trong danh sách.

```
cheeses = ['Cheddar', 'Edam', 'Gouda']
for cheese in cheeses:
 print(cheese)
```

Các cú pháp

.append(): Thêm một phần tử trong () mới vào cuối danh sách.

.sort(): Sắp xếp các phần tử của danh sách.

.pop(index): Xóa và trả về phần tử tại vị trí index. Nếu không có index, nó sẽ xóa và trả về phần tử cuối cùng.

.remove(): Xóa phần tử trong () đầu tiên nó tìm thấy trong danh sách.

.extend(): Nối danh sách trong () vào cuối danh sách hiện tại.

SỰ KHẢ BIẾN

- Khi truyền một danh sách vào một biến, biến đó sẽ nhận được một “tham chiếu” đến danh sách gốc, chứ không phải một bản sao.
- Do đó, nếu biến đó thực hiện bất kỳ thay đổi nào trên danh sách, nó đang thực sự thay đổi danh sách gốc ở bên ngoài hàm.

PYTHON CHAPTER 9

SO SÁNH KHÁI NIỆM

Tính năng	List (Python)	NumPy Array (NumPy)
Kiểu dữ liệu	Linh hoạt, có thể chứa nhiều kiểu dữ liệu khác nhau (số, chuỗi, object...).	**Đồng nhất**. Tất cả phần tử phải cùng một kiểu (tất cả là số nguyên hoặc số thực).
Bộ nhớ	Lưu trữ dữ liệu có thể bị **phân mảnh** trên bộ nhớ.	Lưu trữ dữ liệu trong một **khối bộ nhớ liền kề**. hiệu quả hơn
Hiệu năng	Chậm hơn với các phép toán số học.	**Rất nhanh** cho các phép toán số học.
Mục đích	Đa dụng, lưu trữ chung.	Tính toán khoa học, xử lý số liệu, machine learning.

Integer Indexing: Chọn các phần tử cụ thể: a[[1, 3]]

Boolean Indexing: Lọc mảng dựa trên điều kiện: a[a > 3] (chọn các phần tử lớn hơn 3)

HOẠT ĐỘNG 2D

Hoạt động	List	NumPy Array
Khởi tạo	list_2d = [[1, 2], [3, 4]]	array_2d = np.array([[1, 2], [3, 4]])
Truy cập	list_2d[1][0] (>>> 3)	array_2d[1, 0] (>>> 3)
Nhân ma trận	Dùng 3 vòng lặp for lồng nhau	np.dot(A, B) hoặc A @ B

Tự động "mở rộng" các mảng có hình dạng (shape) nhỏ hơn để chúng tương thích với các mảng lớn hơn trong các phép toán[47]

SO SÁNH CÚ PHÁP

Hoạt động	List	NumPy Array
Khởi tạo	list[start:end:step] data = [6, 5, 7]	import numpy as np array[start:end:step] a_data = np.array([6, 5, 7])
Truy cập (Index)	Giống nhau: data[0]	Giống nhau: a_data[0]
Cắt lát (Slicing)	Giống nhau: data[0:3]	Giống nhau: a_data[0:3]
Cập nhật phần tử	Giống nhau: data[1] = 4	Giống nhau: a_data[1] = 4
Thêm (Append)	Có thể sửa tại chỗ: data.append(4)	Phải tạo mảng mới: a_new = np.append(a_data, 4)
Chèn (Insert)	Có thể sửa tại chỗ: data.insert(0, 4)	Phải tạo mảng mới: a_new = np.insert(a_data, 0, 4)
Toán tử +	**Nối**	**Cộng từng phần tử**
Toán tử *	**Lặp lại**	**Nhân từng phần tử**
Sắp xếp (Sort)	data.sort() data.sort(reverse=True)	a_data.sort() a_data = a_data[::-1] (đảo)
Xóa (Delete)	data.pop(2) (index) data.remove(5) (theo giá trị)	np.delete(a_data, 2) (index) np.where(a_data==5))[0][0] (tìm)
Tìm Index	data.index(9)	np.where(a_data == 9)[0][0]
Đếm (Count)	data.count(7)	(a_data == 7).sum()
Tính tổng	sum(data)	np.sum(a_data)
Lấy độ dài	len(data)	len(a_data) hoặc a_data.shape