Meta Mind
Explore Knowledge - Open Future

Buổi 4: Python trong Khoa học dữ liệu #3

AGENDA

Meta Mind
Explore Knowledge - Open Future

- Biểu thức chính quy Regex trong Python
- Xử lý ngoại lệ trong Python
- Xử lý file trong Python
- Phân chia module trong Python
- Game Lô Đề Học Pro max
- Tổng kết

Meta Mind
Explore Knowledge - Open Future

Biểu thức chính quy Regex trong Python

BIỂU THỨC CHÍNH QUY TRONG PYTHON

Meta Mind
Explore Knowledge - Open Future

Regular Expression (Regex) là chuỗi các ký tự để tìm kiếm, xác định và thay thế các mẫu trong chuỗi.

Ứng dụng:

- Xử lý văn bản: tìm kiếm, thay thế, phân tích cú pháp văn bản.
- Kiểm tra đầu vào: xác minh định dạng của dữ liệu (ví dụ: email, số điện thoại).
- Xử lý dữ liệu: trích xuất thông tin từ văn bản không cấu trúc.

OTP = ex.search(r'\b\d{6}\b', SMS): tìm và lấy ra mã OTP từ SMS

Ma xac thuc Tiki cua ban la 685708. Ma co hieu luc trong vong 15 phut va can giu bao mat.

Thư viện re và các hàm chức năng

Các hàm sẽ được dùng kết hợp với các biểu thức Regex (slide kế tiếp) để thao tác với chuỗi.

Hàm	Mô tả
search	Hàm này trả về đối tượng khớp nếu có một kết quả khớp được tìm thấy trong chuỗi.
findall	Hàm trả về một danh sách chứa tất cả các kết quả khớp của một mẫu trong chuỗi.
split	Hàm trả về một danh sách trong đó chuỗi đã được phân chia theo mỗi kết quả khớp.
sub	Thay thế một hoặc nhiều kết quả khớp trong chuỗi.

Xây dựng biểu thức chính quy / Metacharacter

Meta Mind
Explore Knowledge - Open Future

Mỗi metacharacter mang 1 ý nghĩa nhất định

Metacharacter	Mô tả	Ví dụ
[ ]	Đại diện cho một tập các ký tự.	"[a-z]"
\	Đại diện cho ký tự đặc biệt.	"\r"
.	Đại diện cho bất kỳ ký tự nào xuất hiện ở một số nơi cụ thể.	"Ja.v."
^	Đại diện cho mẫu có mặt ở đầu chuỗi.	"^Java"
$	Đại diện cho mẫu có mặt ở cuối chuỗi.	"mmc$"
*	Đại diện cho không hoặc nhiều lần xuất hiện của một mẫu trong chuỗi.	"hello*"
+	Đại diện cho một hoặc nhiều lần xuất hiện của một mẫu trong chuỗi.	"hello+"
{ }	Chỉ định số lần xuất hiện của một mẫu trong chuỗi.	"java{2}"
|	Biểu diễn cho cái này hoặc cái kia (điều kiện or).	"python2|python3"
( )	Nhóm các thành phần.	

Xây dựng biểu thức chính quy / Ký tự đặc biệt

Meta Mind
Explore Knowledge - Open Future

Các ký tự đặc biệt được định nghĩa bằng cách đặt trước chúng một dấu \

Ký tự	Mô tả
\A	Trả về một kết quả khớp nếu các ký tự được chỉ định có mặt ở đầu chuỗi.
\b	Trả về một kết quả khớp nếu các ký tự được chỉ định có mặt ở đầu hoặc cuối chuỗi.
\B	Trả về một kết quả khớp nếu các ký tự được chỉ định có mặt ở đầu chuỗi nhưng không ở cuối chuỗi.
\d	Trả về một kết quả khớp nếu chuỗi chứa các chữ số [0-9].
\D	Trả về một kết quả khớp nếu chuỗi không chứa các chữ số [0-9].
\s	Trả về một kết quả khớp nếu chuỗi chứa bất kỳ ký tự khoảng trắng nào.
\S	Trả về một kết quả khớp nếu chuỗi không chứa bất kỳ ký tự khoảng trắng nào.
\w	Trả về một kết quả khớp nếu chuỗi chứa bất kỳ ký tự từ nào.
\W	Trả về một kết quả khớp nếu chuỗi không chứa bất kỳ từ nào.
\Z	Trả về một kết quả khớp nếu các ký tự được chỉ định ở cuối chuỗi.

Xây dựng biểu thức chính quy / Set

Meta Mind
Explore Knowledge - Open Future

Set là một nhóm các ký tự được đưa ra bên trong một cặp dấu ngoặc vuông [ ].

Set	Mô tả
[arn]	Trả về một kết quả khớp nếu chuỗi chứa bất kỳ ký tự nào được chỉ định trong tập hợp.
[a-n]	Trả về một kết quả khớp nếu chuỗi chứa bất kỳ ký tự nào từ a đến n.
[^arn]	Trả về một kết quả khớp nếu chuỗi chứa các ký tự ngoại trừ a, r và n.
[0123]	Trả về một kết quả khớp nếu chuỗi chứa bất kỳ chữ số nào được chỉ định.
[0-9]	Trả về một kết quả khớp nếu chuỗi chứa bất kỳ chữ số nào trong khoảng từ 0 đến 9.
[0-5][0-9]	Trả về một kết quả khớp nếu chuỗi chứa bất kỳ chữ số nào trong khoảng từ 00 đến 59.
[a-zA-Z]	Trả về một kết quả khớp nếu chuỗi chứa bất kỳ bảng chữ cái nào (chữ thường hoặc chữ hoa).

Thư viện re và một số hàm cho Regex

Meta Mind
Explore Knowledge - Open Future

```
print(re.search(r'\d{5}', "Mã OTP là 12345, có hiệu lực trong 1 phút."))
```

```
<re.Match object; span=(10, 15), match='12345'>
```

```
print(re.findall(r'[A-Z]', "Thành viên gồm: Đăng, Linh, Nhã, Khôi, Tuấn"))
```

```
['T', 'L', 'N', 'K', 'T']
```

Có đánh rơi ai không nhỉ ?

```
print(regex.findall(r'\p{Lu}', "Thàn
['T', 'Đ', 'L', 'N', 'K', 'T']
```

```
print(re.split(r': |, ', "Thành viên gồm: Đăng, Linh, Nhã, Khôi, Tuấn"))
```

```
['Thành viên gồm', 'Đăng', 'Linh', 'Nhã', 'Khôi', 'Tuấn']
```

```
print(re.sub(r'[0-9]', '0', "Mã OTP là 12345, có hiệu lực trong 1 phút."))
```

```
Mã OTP là 00000, có hiệu lực trong 0 phút.
```

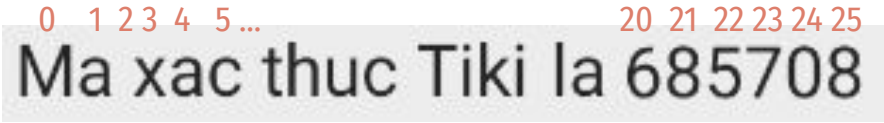
Các thuộc tính của đối tượng re.Match

Meta Mind
Explore Knowledge - Open Future

Thuộc tính:

- ***.span()***: Chứa vị trí bắt đầu và kết thúc của kết quả khớp.
- ***.string***: Chứa chuỗi được truyền vào hàm.
- ***.group()***: Chứa kết quả khớp.

OTP = ex.search(r'\b\d{6}$, SMS): tìm và lấy ra mã OTP từ SMS



OTP***.span()*** = (20,26) ??? 26

OTP***.string*** = "Ma xac thuc Tiki la 685708

OTP***.group()*** = "685708"

Xem thêm regex >>

Meta Mind
Explore Knowledge - Open Future

Xử lý ngoại lệ trong Python

XỬ LÝ NGOẠI LỆ TRONG PYTHON

Meta Mind
Explore Knowledge - Open Future

What?

```
ielts = 6.5
if(ielts >= 6.5)
print("Chúc mừng, bạn đã đủ điểm đi du học!")
```

```
result = 10 / 0
print(result)
```

XỬ LÝ NGOẠI LỆ TRONG PYTHON

Meta Mind
Explore Knowledge - Open Future

What?

Ngoại lệ (Exceptions): Khi một lỗi xảy ra khi chạy chương trình, ta gọi đó là một Exception.

- Lỗi cú pháp (Syntax errors): dẫn đến việc chương trình bị dừng.

```
ielts = 6.5
if(ielts >= 6.5)
print("Chúc mừng, bạn đã đủ điểm đi du học!")
```

```
  File "<ipython-input-14-f160e68b59a3>", line 2
    if(ielts >= 6.5)
                   ^
SyntaxError: expected ':'
```

SyntaxError	Lỗi cú pháp: Sai sót trong cú pháp.

- Lỗi chia cho 0 (ZeroDivisionError)

```
# Ngoại lệ: Chia một số cho 0
result = 10 / 0
print(result)
```

```
---------------------------------------------------------
ZeroDivisionError                         Traceback
<ipython-input-2-d707037ccc30> in <cell line: 2>()
      1 # Ngoại lệ: Chia một số cho 0
----> 2 result = 10 / 0
      3 print(result)
```

ZeroDivisionError	Lỗi chia cho 0: Thử chia một số cho 0.

XỬ LÝ NGOẠI LỆ TRONG PYTHON

Meta Mind
Explore Knowledge - Open Future

What?

Ngoại lệ (Exceptions): Khi một lỗi xảy ra khi chạy chương trình, ta gọi đó là một Exception.

- Lỗi cú pháp (Syntax errors): dẫn đến việc chương trình bị dừng.

```
ielts = 6.5
if(ielts >= 6.5)
print("Chúc mừng, bạn đã đủ điểm đi du học!")
```

```
  File "<ipython-input-14-f160e68b59a3>", line 2
    if(ielts >= 6.5)
                   ^
SyntaxError: expected ':'
```

- Lỗi chia cho 0 (ZeroDivisionError)

```
# Ngoại lệ: Chia một số cho 0
result = 10 / 0
print(result)
```

```
---------------------------------------------------------
ZeroDivisionError                         Traceback
<ipython-input-2-d707037ccc30> in <cell line: 2>()
      1 # Ngoại lệ: Chia một số cho 0
----> 2 result = 10 / 0
      3 print(result)
```

Exceptions	Meaning
SyntaxError	Lỗi cú pháp: Sai sót trong cú pháp.
TypeError	Lỗi kiểu: Phép toán không phù hợp với kiểu dữ liệu.
NameError	Lỗi tên: Tên biến hoặc hàm không được tìm thấy.
IndexError	Lỗi chỉ số: Chỉ số vượt quá phạm vi của danh sách.
KeyError	Lỗi khóa: Khóa không tồn tại trong từ điển.
ValueError	Lỗi giá trị: Đối số không hợp lệ cho hàm.
AttributeError	Lỗi thuộc tính: Thuộc tính không tồn tại.
IOError	Lỗi I/O: Lỗi khi đọc hoặc ghi tệp.
ZeroDivisionError	Lỗi chia cho 0: Thử chia một số cho 0.
ImportError	Lỗi nhập: Lỗi khi module không thể được nhập.

How?

Bắt và Xử lý Ngoại lệ bằng khối lệnh Try - Except

Bắt ngoại lệ cụ thể:

```
try:
    # Các câu lệnh
except IndexError:
    # Các câu lệnh
except ValueError:
    # Các câu lệnh
```

```
try:
    numbers = [1, 2, 3]
    index = int(input("Nhập chỉ số của phần tử bạn muốn truy cập: "))
    value = numbers[index]
    print("Giá trị tại chỉ số đã nhập là:", value)
except IndexError:
    print("Lỗi: Chỉ số không hợp lệ.")
except ValueError:
    print("Lỗi: Giá trị nhập không phải là một số nguyên.")
```

```
Nhập chỉ số của phần tử bạn muốn truy cập: 3
Lỗi: Chỉ số không hợp lệ.
```

```
Nhập chỉ số của phần tử bạn muốn truy cập: abc
Lỗi: Giá trị nhập không phải là một số nguyên.
```

How?

Khối Else:

- Có thể sử dụng khối else trong khối Try-Except.
- Phải xuất hiện ***sau tất cả*** các khối Except.
- Code bên trong khối Else ***chỉ được thực hiện nếu không xuất hiện bất kỳ một Exception nào.***

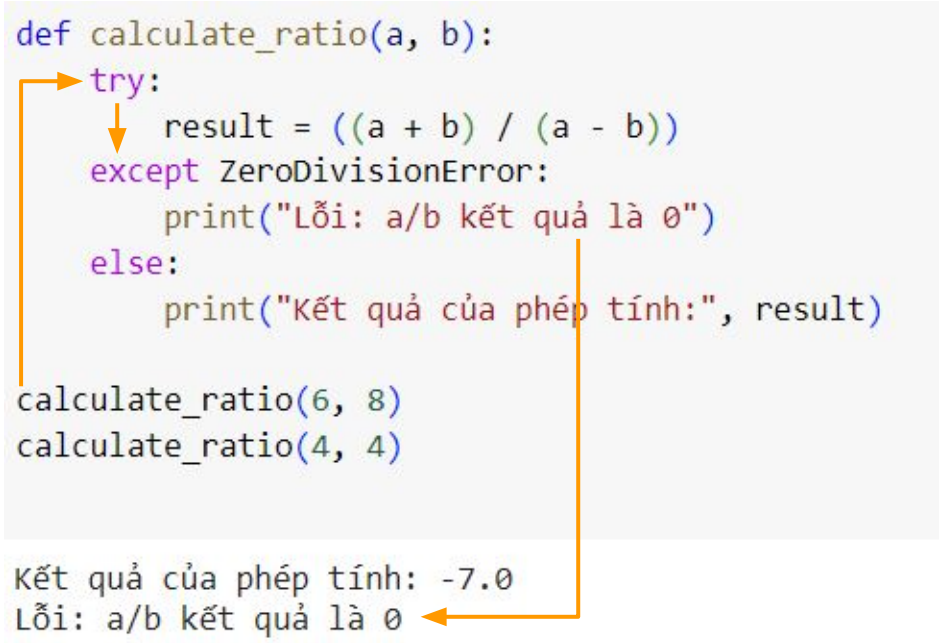
```
def calculate_ratio(a, b):
    try:
        result = ((a + b) / (a - b))
    except ZeroDivisionError:
        print("Lỗi: a/b kết quả là 0")
    else:
        print("Kết quả của phép tính:", result)

calculate_ratio(6, 8)
calculate_ratio(4, 4)
```

```
Kết quả của phép tính: -7.0
Lỗi: a/b kết quả là 0
```

How?

Khối Else:

- Có thể sử dụng khối else trong khối Try-Except.
- Phải xuất hiện ***sau tất cả*** các khối Except.
- Code bên trong khối Else ***chỉ được thực hiện nếu code không bị lỗi nào.***



```
def calculate_ratio(a, b):
    try:
        result = ((a + b) / (a - b))
    except ZeroDivisionError:
        print("Lỗi: a/b kết quả là 0")
    else:
        print("Kết quả của phép tính:", result)

calculate_ratio(6, 8)
calculate_ratio(4, 4)
```

```
Kết quả của phép tính: -7.0
Lỗi: a/b kết quả là 0
```

How?

Khối Finally

- Khối Finally luôn được thực thi trong bất kỳ trường hợp nào.

```
try:
    # Một số mã...

except:
    # Khối tùy chọn
    # Xử lý ngoại lệ (nếu cần)

else:
    # thực hiện nếu không có ngoại lệ

finally:
    # Một số mã... (luôn được thực hiện)
```

```
try:
    num = int(input("Nhập một số nguyên: "))
    result = 10 / num
    print("Kết quả của phép chia:", result)
except ZeroDivisionError:
    print("Lỗi: Không thể chia cho 0.")
else:
    print("In ra nếu không có ngoại lệ")
finally:
    print('Khối finally luôn được thực hiện')
```

```
Nhập một số nguyên: 6
Kết quả của phép chia: 1.6666666666666667
In ra nếu không có ngoại lệ
Khối finally luôn được thực hiện
```

```
Nhập một số nguyên: 0
Lỗi: Không thể chia cho 0.
Khối finally luôn được thực hiện
```

How?

Tự tạo exception với câu lệnh raise

- Câu lệnh Raise sẽ tạo ra và ném ra một Exception cụ thể.
- Cho phép tạo ra các tình huống ngoại lệ theo ý muốn và thông điệp tùy chỉnh.
- Giúp quản lý và xử lý lỗi trong chương trình một cách linh hoạt và hiệu quả.

```
try:
    age = int(input("Nhập tuổi của bạn: "))
    if age < 0:
        raise ValueError("Tuổi không thể là một số âm.")
    elif age > 1000:
        raise ValueError("Bạn đã thành tiên.")
    print("Tuổi của bạn là:", age)

except ValueError as ve:
    print("Lỗi:", ve)
```

```
Nhập tuổi của bạn: 30
Tuổi của bạn là: 30

Nhập tuổi của bạn: -30
Lỗi: Tuổi không thể là một số âm.

Nhập tuổi của bạn: 3000
Lỗi: Bạn đã thành tiên.
```

How?

Tự tạo exception với câu lệnh raise

- Câu lệnh Raise sẽ tạo ra và ném ra một Exception cụ thể.
- Cho phép tạo ra các tình huống ngoại lệ theo ý muốn và thông điệp tùy chỉnh.
- Giúp quản lý và xử lý lỗi trong chương trình một cách linh hoạt và hiệu quả.

Xem thêm Xử lý ngoại lệ >>

```
try:
    age = int(input("Nhập tuổi của bạn: "))
    if age < 0:
        raise ValueError("Tuổi không thể là một số âm.")
    elif age > 1000:
        raise ValueError("Bạn đã thành tiên.")
    print("Tuổi của bạn là:", age)

except ValueError as ve:
    print("Lỗi:", ve)
```

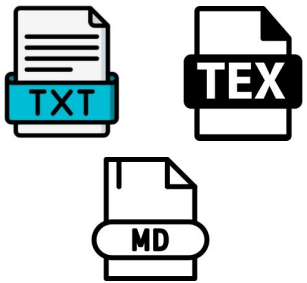
```
Nhập tuổi của bạn: 30
Tuổi của bạn là: 30

Nhập tuổi của bạn: -30
Lỗi: Tuổi không thể là một số âm.

Nhập tuổi của bạn: 3000
Lỗi: Bạn đã thành tiên.
```

Meta Mind
Explore Knowledge - Open Future

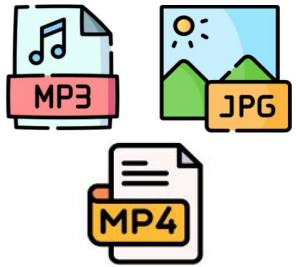
Xử lý file trong Python

What?

- Mọi thứ trong hệ điều hành đều được biểu diễn dưới dạng tệp.
- Là các vị trí có tên trên disk để lưu trữ thông tin.
- Thường có 2 loại file:
  - Tệp văn bản.
  - Tệp nhị phân.



Tệp văn bản



Tệp nhị phân

Why?

- RAM là bộ nhớ thoáng qua (volatile).

  ⇒ Dữ liệu biến mất sau khi quá trình kết thúc.
- Tệp có tính bền vững (persistent).

  ⇒ Dữ liệu được lưu trữ.

?

How: Open a file

1. Mở một file?
2. Đọc hoặc ghi file?
3. Đóng file?

Chương trình đang muốn làm việc với một file cho trước:

- Đó là file nào?
- Muốn thực hiện hoạt động gì với file đó?

open(fileName, mode):

- fileName: xác định đó là file nào.
- mode: xác định hoạt động muốn thực hiện với file đó.
- Trả về một đối tượng File đại diện cho file đã mở.

```
f = open("filetest.txt", "r")
```

```
f = open("filetest.txt", "r")
```

Mode	Ý nghĩa
r	Đọc (mặc định)
w	Ghi. Tạo hoặc xóa một tệp.
x	Tạo độc quyền. Lỗi nếu tệp tồn tại.
a	Thêm vào. Tạo nếu tệp không tồn tại.
t	Mở trong chế độ văn bản. (mặc định)
b	Mở trong chế độ nhị phân.
+	Mở một tệp để cập nhật (rw)

How: Open a file

1. Mở một file?
2. Đọc hoặc ghi file?
3. Đóng file?

XỬ LÝ FILE TRONG PYTHON

Meta Mind
Explore Knowledge - Open Future

- f.read(size) đọc và trả về size bytes
  - Size là tùy chọn.
  - Mặc định đọc toàn bộ nội dung file .
  - Cập nhật con trỏ tệp hiện tại sau khi .read() được gọi.
  - Các tệp quá lớn có thể gây tràn bộ nhớ.
- f.seek(offset) đặt con trỏ hiện tại đến một offset cụ thể
- f.write() ghi vào tệp

How: Read/ Write a file

1. Mở một file?
2. Đọc hoặc ghi file?
3. Đóng file?

```
# Mở tệp "filetest.txt" ở chế độ r+: đọc và cập nhật
f = open("filetest.txt", "r+")
# In ra nội dung của tệp
print (f.read())
```

```
Hi everyone,

Welcome to MetaMind team!

This document is about how to handle your files in Python.

Glad to see you all!
```

```
f.read(17)
```

```
'Hi everyone,\n\nWel'
```

```
# Di chuyển con trỏ đến vị trí thứ 19
f.seek(19)
# Đọc 4 ký tự tiếp theo từ vị trí mới
a = f.read(4)
print (a)
```

```
come
```

```
f.write("\n Ghi vào tệp")
f = open("filetest.txt", "r+")
print (f.read())
```

```
Hi everyone,

Welcome to MetaMind team!

This document is about how to handle your fil

Glad to see you all!
 Ghi vào tệp
```

Meta Mind
Explore Knowledge - Open Future

Các tệp văn bản (text files)

- .readline(): đọc đến khi gặp dấu xuống dòng mới.
  - Có dấu \n ở cuối tệp.
- .readlines(): đọc tất cả các dòng

```
f = open("filetest.txt", "r+")
f.readline()
```

```
'Hi everyone,\n'
```

```
f.readlines()
```

```
['\n',
 'Welcome to MetaMind team!\n',
 '\n',
 'This document is about how to handle your files in Python.\n',
 '\n',
 'Glad to see you all!\n',
 ' Ghi vào tệp']
```

How: Read/ Write a file

1. Mở một file?
2. Đọc hoặc ghi file?
3. Đóng file?

How: Close a file

1. Mở một file?
2. Đọc hoặc ghi file?
3. Đóng file?

- Đóng một tệp sau khi sử dụng.
- Dọn dẹp bộ đệm, cache của hệ điều hành.
- Nếu không đóng, có thể mất dữ liệu khi xảy ra cúp điện.
  - f.close()

- Tự động .close() bằng cách sử dụng with:

```
with open('filetest.txt', 'r+') as f:
  data = f.read()

# other stuffs here, f is closed
```

Xem thêm Xử lý File >>

Meta Mind
Explore Knowledge - Open Future

Phân chia module trong Python

PHÂN CHIA MODULE TRONG PYTHON

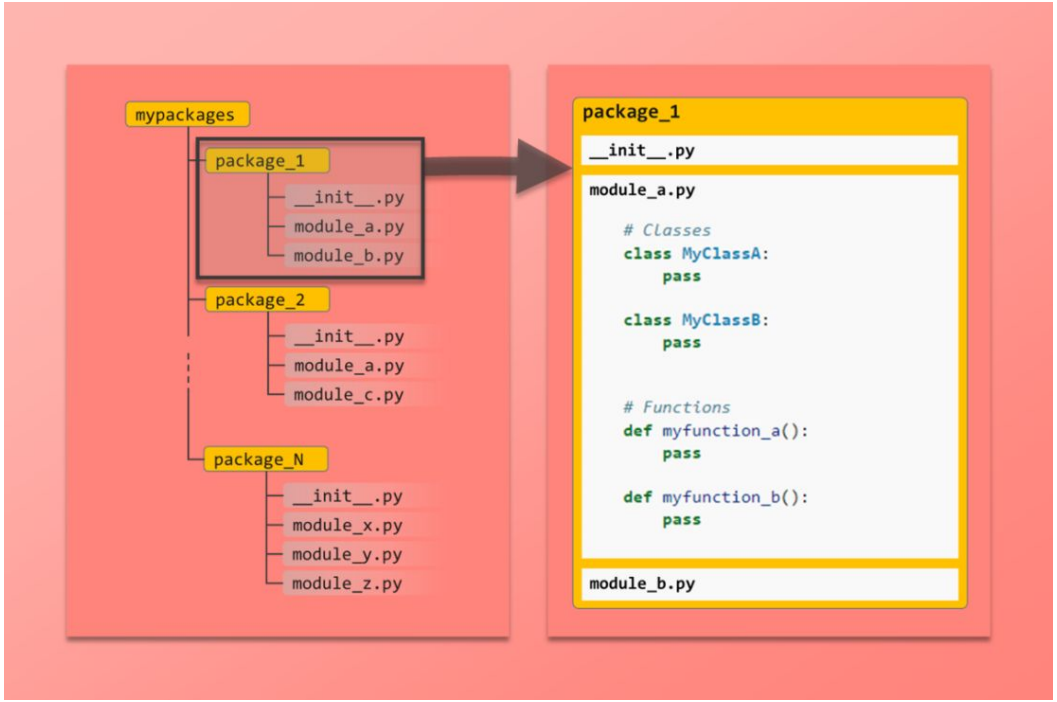
Meta Mind
Explore Knowledge - Open Future

- Module (mô-đun) giúp tổ chức và quản lý mã nguồn một cách hiệu quả.

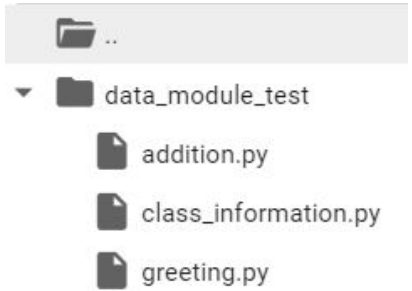
- Giúp chia code thành các phần nhỏ hơn, tiện lợi để duy trì và phát triển, tăng tính tái sử dụng code và giảm rủi ro của lỗi logic.

PHÂN CHIA MODULE TRONG PYTHON

- Một Module thường chứa một tập hợp các hàm, lớp, biến, và các câu lệnh thực thi được.
- Một mô-đun trong Python thường là một tệp riêng biệt có phần mở rộng " .py ", có thể được chứa trong một package.
- Để sử dụng một mô-đun trong Python, chúng ta sử dụng lệnh import và chỉ định tên của mô-đun muốn sử dụng.



Import ...



greeting.py

```
1 def hello(name):
2     print('Hello, ' + name)
```

- *import data_module_test, ...*
- *import data_module_test.addition as add, ...*

Sử dụng bất cứ source file nào dưới dạng như một Module.

Import sẽ trỏ tới đường dẫn của module đó.

Từ khoá " *as* " dùng để đặt tên thay thế cho module để gọn sử dụng

Khi muốn sử dụng, sẽ bắt đầu từ vị trí import, dùng ký hiệu " . " để đi tới các mục con bên trong.

```
import data_module_test.greeting as meta

meta.hello("Khoi")
```

```
Hello, Khoi
```

From ... import ...

Meta Mind
Explore Knowledge - Open Future

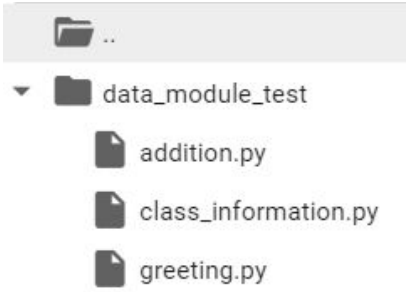
- *from data_module_test.greeting import hello, ...*

  Import cụ thể một thuộc tính nào đó thay vì import cả 1 module.

- *from data_module_test.greeting import **

  Import tất cả thuộc tính trong module.

- *from data_module_test.greeting import hello as hi*



greeting.py

```
1 def hello(name):
2     print('Hello, ' + name)
3
4 def goodbye(name):
5     print('See you, ' + name)
```

```
from data_module_test.greeting import *

hello("Khoi")
```

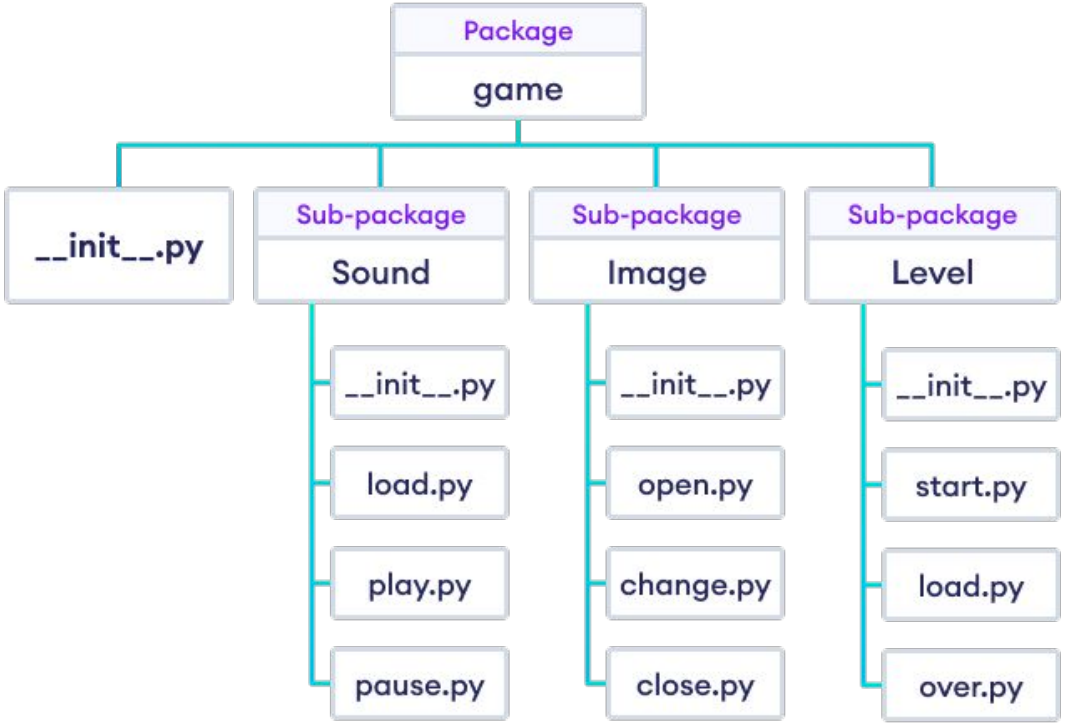
```
Hello, Khoi
```

```
from data_module_test.greeting import hello as hi

hi("Khoi")
```

```
Hello, Khoi
```

Package

- *Package như là một thư mục trên máy tính.*
- *Chứa một tệp __init__.py và nhiều module .py khác, cùng với các thư mục con.*



Xem thêm Phân chia Module trong Python >>

Meta Mind
Explore Knowledge - Open Future

Game Lô Đề Học Pro Max

GAME LÔ ĐỀ HỌC PRO MAX

Meta Mind
Explore Knowledge - Open Future

Đề bài: Từ Game Lô Đề Pro. Bạn hãy nâng cấp Game Lô Đề Pro max với yêu cầu như sau:

- **Chức năng đăng nhập:**
  - Nếu là tài khoản admin sẽ có menu riêng gồm quản lý tài khoản, nạp tiền tài khoản, thống kê và đăng xuất.
  - Nếu là tài khoản thường thì hiển thị menu thường gồm: chơi lô, đổi mật khẩu, thống kê và đăng xuất.
- **Chức năng admin:**
  - **Quản lý tài khoản:** tạo tài khoản (không được phép tạo trùng username) và xóa tài khoản (không được phép xóa tài khoản admin). Thông tin tài khoản gồm username, password, tổng tiền và được lưu vào trong file taikhoan.txt, mỗi tài khoản 1 dòng.
  - **Nạp tiền:** Chức năng nạp tiền sẽ cập nhật số tiền tài khoản trong file taikhoan.txt.
  - **Thống kê:** Chức năng thống kê sẽ thống kê có bao nhiêu tài khoản, số lượt chơi lô là bao nhiêu, tỷ lệ kèo thắng/thua.
- **Chức năng người dùng:**
  - **Chức năng đổi mật khẩu:** Người chơi có thể đổi mật khẩu của họ
  - **Chức năng chơi lô:** Mỗi lần chơi lô, thông tin sẽ tự được lưu vào file choilo.txt các thông tin: thời gian chơi, username, số lô cược, tiền cược, kết quả quay số, kết quả trúng lô, mỗi lần chơi là 1 dòng.
  - **Chức năng thống kê:** Cho phép người dùng xem thống kê số lần chơi lô, tổng tiền chơi lô thắng, tổng tiền chơi lô thua, tỉ lệ chơi lô thắng/thua.
- **Xây dựng module:** Chuyển toàn bộ các hàm trên vào thành 1 Module LoDeHoc, xây dựng Module XuLyFile với đầu vào là đường dẫn file và đầu ra là list dữ liệu đọc được từ file. Cả 2 Module nằm trong thư mục my_lib.

Meta Mind
Explore Knowledge - Open Future

Tổng kết

TỔNG KẾT

Meta Mind
Explore Knowledge - Open Future

Hoàn thành Form điểm danh sau (10'): Form điểm danh

Bài tập sau buổi học:

- *Regex*
- *Xử lý ngoại lệ trong Python*
- *Đọc tệp tin trong Python*
- *Ghi tệp tin trong Python*
- *Game Lô Đề Học Pro Max*

Yêu cầu:

- Đặt tên folder theo mẫu DSMMC_<Buổi học>_<Tên> (Ví dụ: *DSMMC_Buoi4_DangNH*)
- Hoàn thành các bài lab sau đó nộp lên git cá nhân , sau đó submit link folder git trên Form bài tập
- Hạn nộp: Trước ngày 18/04/2024

THANK YOU !