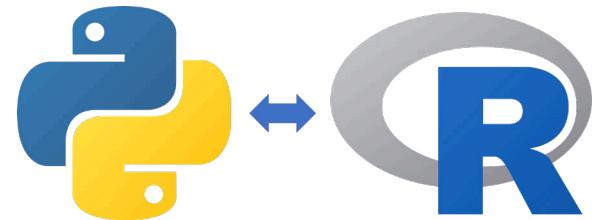


# Phân tích dữ liệu với R/Python: [P1] - TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH PYTHON

ThS. Nguyễn Quang Phúc  
[phucnq@uel.edu.vn](mailto:phucnq@uel.edu.vn)



# KHÁI QUÁT VỀ PYTHON VÀ MÔI TRƯỜNG LẬP TRÌNH

# NỘI DUNG

1. Khái quát về ngôn ngữ lập trình Python
2. Thiết lập môi trường lập trình Python
3. Tạo và thực thi project Python với PyCharm

# 1. Khái quát về ngôn ngữ lập trình Python

## »» Python là gì?

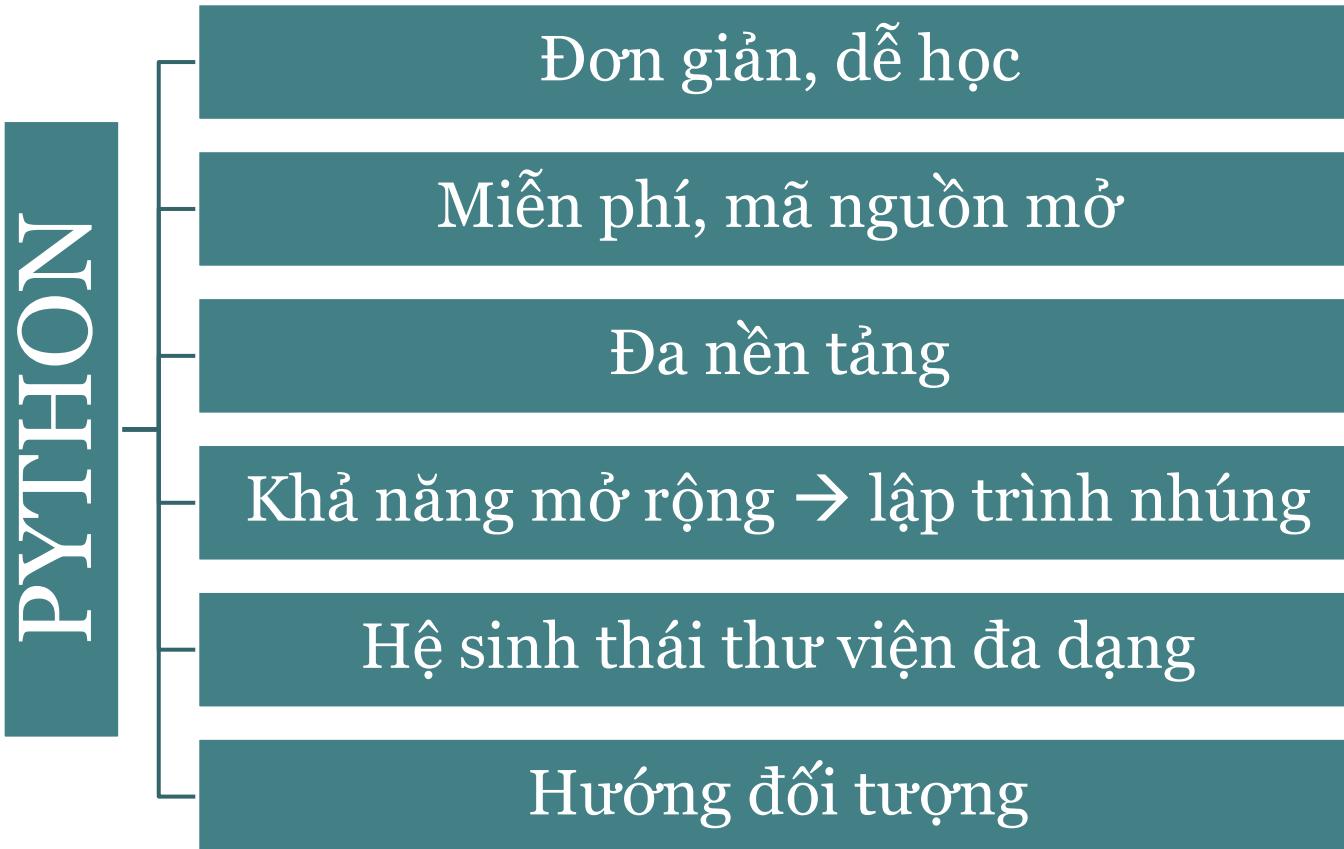
Python là ngôn ngữ lập trình hướng đối tượng, cấp cao, mạnh mẽ, được tạo ra bởi Guido van Rossum.

Phiên bản	Phát hành
– Python 1.0 (bản tiêu chuẩn đầu tiên)	01/1994
– Python 1.6 (bản 1.x cuối cùng)	05/09/2000
– Python 2.0	16/10/2000
– Python 2.7 (bản 2.x cuối cùng)	03/07/2010
– Python 3.0	03/12/2008
– Python 3.9	05/10/2020

<https://www.python.org/doc/versions/>

# 1. Khái quát về ngôn ngữ lập trình Python

## »» Tính năng của Python

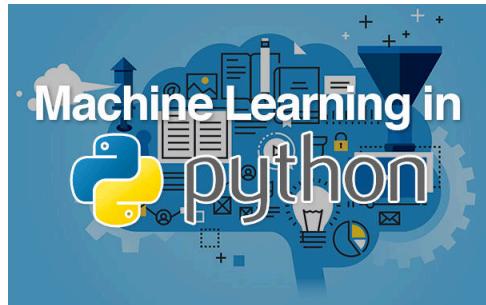


# 1. Khái quát về ngôn ngữ lập trình Python

## »» Python được dùng ở đâu?

Desktop  
App

Tkinter, PyQt, PyGUI, ...



Web

Django, Flask, Pyramid, ...



Mobile  
App

Kivy, BeeWare, ...



# 1. Khái quát về ngôn ngữ lập trình Python

## »» Tại sao là Python?

Jan 2021	Jan 2020	Change	Programming Language	Ratings	Change
1	2	▲	C	17.38%	+1.61%
2	1	▼	Java	11.96%	-4.93%
3	3		Python	11.72%	+2.01%
4	4		C++	7.56%	+1.99%
5	5		C#	3.95%	-1.40%
6	6		Visual Basic	3.84%	-1.44%
7	7		JavaScript	2.20%	-0.25%
8	8		PHP	1.99%	-0.41%
9	18	▲	R	1.90%	+1.10%
10	23	▲	Groovy	1.84%	+1.23%
11	15	▲	Assembly language	1.64%	+0.76%
12	10	▼	SQL	1.61%	+0.10%
13	9	▼	Swift	1.43%	-0.36%
14	14		Go	1.41%	+0.51%
15	11	▼	Ruby	1.30%	+0.24%

<https://www.tiobe.com/tiobe-index/>

# 1. Khái quát về ngôn ngữ lập trình Python

## »» Tài nguyên học liệu

### ➤ Website:

- Learn Python The Hardway: <https://learnpythonthehardway.org/>
- Learn Python Code Cademy:  
<https://www.codecademy.com/learn/python>
- Learn Python Treehouse: <https://teamtreehouse.com/learn-to-code/python>
- Learn Python code mentor: <https://www.codementor.io/learn-python-online>

### ➤ Ebook:

- *Think Python 2nd Edition*, Allen Downey
- *Python for Everybody: Exploring Data in Python 3*, Charles Russell Severance
- *Python for Data Analysis*, Wes McKinney

## 2. Thiết lập môi trường lập trình Python

### »» Cài đặt Python

<https://www.python.org/downloads/>

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

Download the latest version for Mac OS X

Download Python 3.9.1

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases

```
macmall — Python — 72x11
[Macmalls-MacBook-Pro:~ macmall$ python3
Python 3.9.1 (v3.9.1:1e5d33e9b9, Dec 7 2020, 12:10:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## 2. Thiết lập môi trường lập trình Python

### »» Công cụ lập trình Python

- Notepad, SublimeText, ...
- IDLE (Python 3.5)
- PyCharm
- ...

## 2. Thiết lập môi trường lập trình Python

### »» Công cụ lập trình Python

#### ❑ SublimeText

```
Macmails-MacBook-Pro:~ macmail$ python3
Python 3.9.1 (v3.9.1:1e5d33e9b9, Dec  7 2020,
12:10:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for
more information.
```

```
>>> a = 6
>>> b = 8
>>> sum = a + b
>>> print(sum)
14
>>>
```

## 2. Thiết lập môi trường lập trình Python

### »» Công cụ lập trình Python

#### ❑ SublimeText



```
FirstProgram.py
1 a = 6
2 b = 8
3 sum = a + b
4 print(sum)

Line 4, Column 11          Tab Size: 4          Python
```

```
[Macmalls-MacBook-Pro:~ macmall$ cd /Volumes/Data/Works/1.UEL/Python/Demo
[Macmalls-MacBook-Pro:Demo macmall$ python3 FirstProgram.py
14 ←
Macmalls-MacBook-Pro:Demo macmall$ ]
```

## 2. Thiết lập môi trường lập trình Python

### »» Công cụ lập trình Python

SublimeText

Ctrl + B → Build

The screenshot shows a SublimeText window with a dark theme. The top bar has three red, yellow, and green circular icons. The title bar shows 'FirstProgram.py' and 'UNREGISTERED'. The main editor area contains the following Python code:

```
1 a = 6
2 b = 8
3 print(a + b)
4 print("Welcome to UEL")
```

Below the editor, the output pane displays the results of running the program:

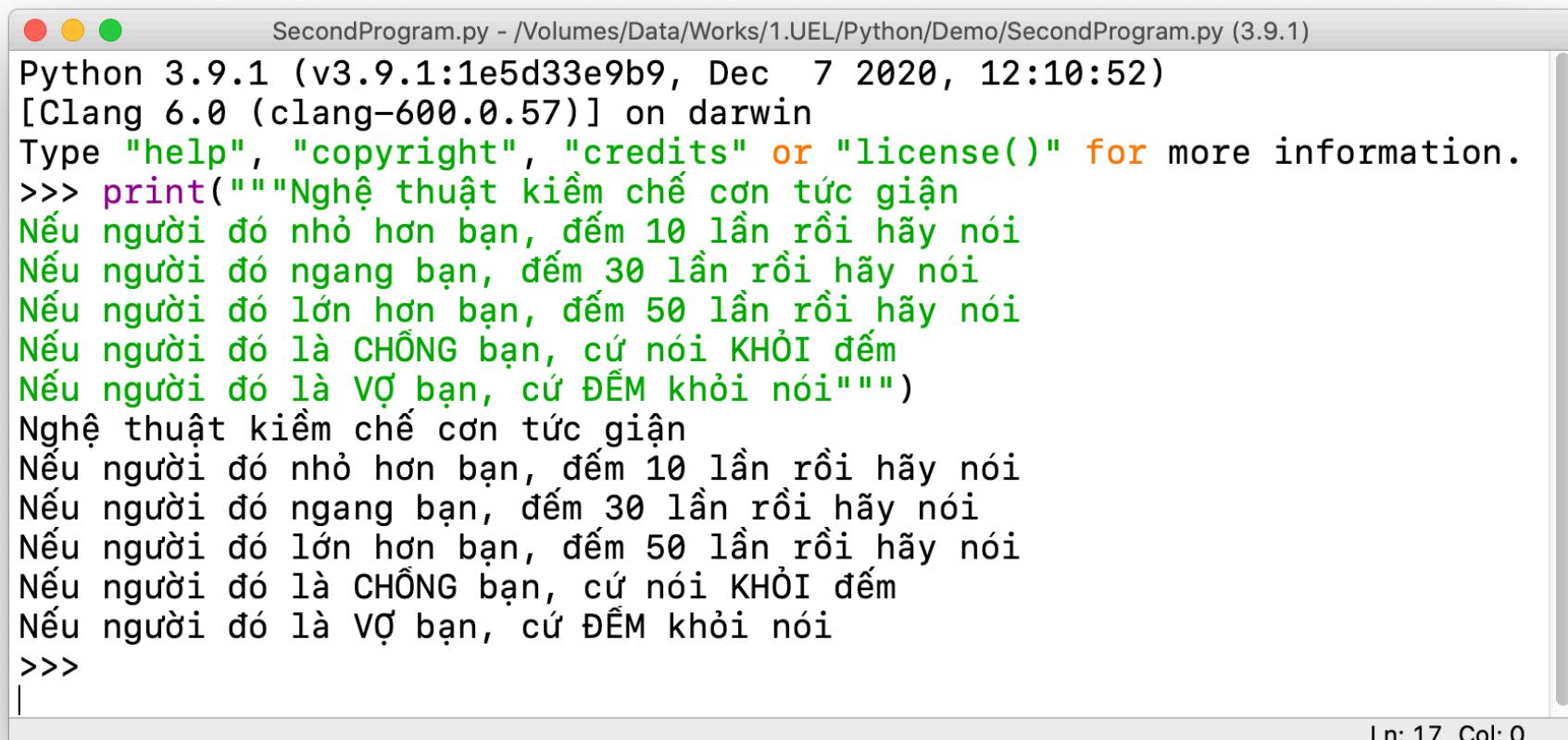
```
14
Welcome to UEL
[Finished in 0.0s]
```

At the bottom left, it says 'Line 4, Column 24'. At the bottom right, it says 'Tab Size: 4' and 'Python'.

## 2. Thiết lập môi trường lập trình Python

### »» Công cụ lập trình Python

#### ❑ IDLE (Python 3.5)



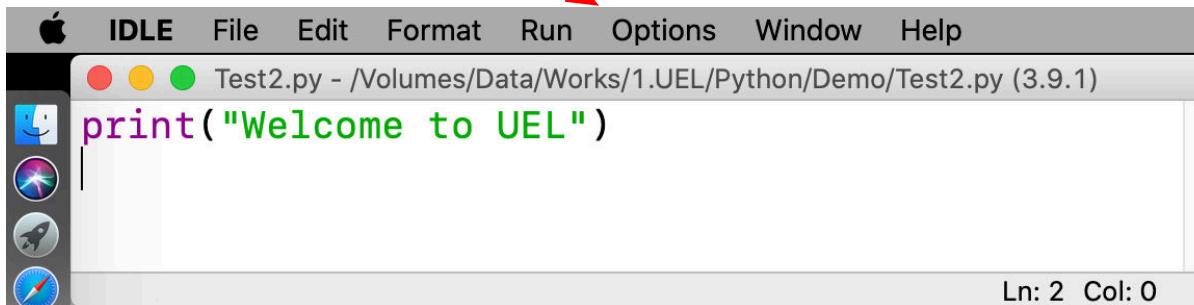
```
SecondProgram.py - /Volumes/Data/Works/1.UEL/Python/Demo/SecondProgram.py (3.9.1)
Python 3.9.1 (v3.9.1:1e5d33e9b9, Dec 7 2020, 12:10:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("""Nghệ thuật kiềm chế cơn tức giận
Nếu người đó nhỏ hơn bạn, đếm 10 lần rồi hãy nói
Nếu người đó ngang bạn, đếm 30 lần rồi hãy nói
Nếu người đó lớn hơn bạn, đếm 50 lần rồi hãy nói
Nếu người đó là CHỒNG bạn, cứ nói KHỎI đếm
Nếu người đó là VỢ bạn, cứ ĐỄM khỏi nói""")
Nghệ thuật kiềm chế cơn tức giận
Nếu người đó nhỏ hơn bạn, đếm 10 lần rồi hãy nói
Nếu người đó ngang bạn, đếm 30 lần rồi hãy nói
Nếu người đó lớn hơn bạn, đếm 50 lần rồi hãy nói
Nếu người đó là CHỒNG bạn, cứ nói KHỎI đếm
Nếu người đó là VỢ bạn, cứ ĐỄM khỏi nói"""
>>>
|
```

Ln: 17 Col: 0

## 2. Thiết lập môi trường lập trình Python

### »» Công cụ lập trình Python

#### IDLE (Python 3.5)



```
Python 3.9.1 (v3.9.1:1e5d33e9b9, Dec  7 2020, 12:10:52)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Volumes/Data/Works/1.UEL/Python/Demo/Test2.py =====
Welcome to UEL
>>> |
```

# 3. Tạo và thực thi project Python với PyCharm

## »» Cài đặt IDE PyCharm

<https://www.jetbrains.com/pycharm/download/#section=mac>

The screenshot shows the PyCharm download page on the JetBrains website. At the top, there's a navigation bar with links for Developer Tools, Team Tools, Learning Tools, Solutions, and Store. A search icon and user profile icons are also present. Below the navigation, there's a main heading "PyCharm" and a "Download" button. On the left, there's a large PyCharm logo icon and some version information: "Version: 2020.3.2", "Build: 203.6682.179", and "30 December 2020". Below this, there are links for "System requirements", "Installation Instructions", and "Other versions". The central part of the page is titled "Download PyCharm" and offers download options for Windows, Mac, and Linux. For Mac, there are two options: "Professional" and "Community". Both options have a "Download" button and a ".dmg (Intel)" dropdown menu. The "Professional" section is described as being for both Scientific and Web Python development, with support for HTML, JS, and SQL. The "Community" section is described as being for pure Python development. Both sections mention availability for Intel and Apple Silicon.

Developer Tools Team Tools Learning Tools Solutions Store

What's New Features Learn Buy

Download

PyCharm

Version: 2020.3.2  
Build: 203.6682.179  
30 December 2020

System requirements  
Installation Instructions  
Other versions

Download .dmg (Intel) ▾

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Community

For pure Python development

Download .dmg (Intel) ▾

Free trial  
Available for Intel and Apple Silicon

Free, open-source  
Available for Intel and Apple Silicon

### 3. Tạo và thực thi project Python với PyCharm

#### »» Cài đặt IDE PyCharm

<https://www.jetbrains.com/community/education/#students>

The screenshot shows the JetBrains website's 'Free License Programs' page. At the top, there's a navigation bar with links for Developer Tools, Team Tools, Learning Tools, Solutions, and Store, along with search and user icons. Below the navigation, there are links for Academic Licensing, Open Source, User Groups, Events Partnership, and Developer Recognition. The main content area features a large heading 'Free Educational Licenses' and a subtext: 'Learn or teach coding with best-in-class development tools from JetBrains!'. To the right is a stylized illustration of a person working on a laptop, with a purple circular background. At the bottom, there are three buttons: 'For students and teachers' (highlighted in blue), 'For schools and universities', and 'For training courses and bootcamp'.

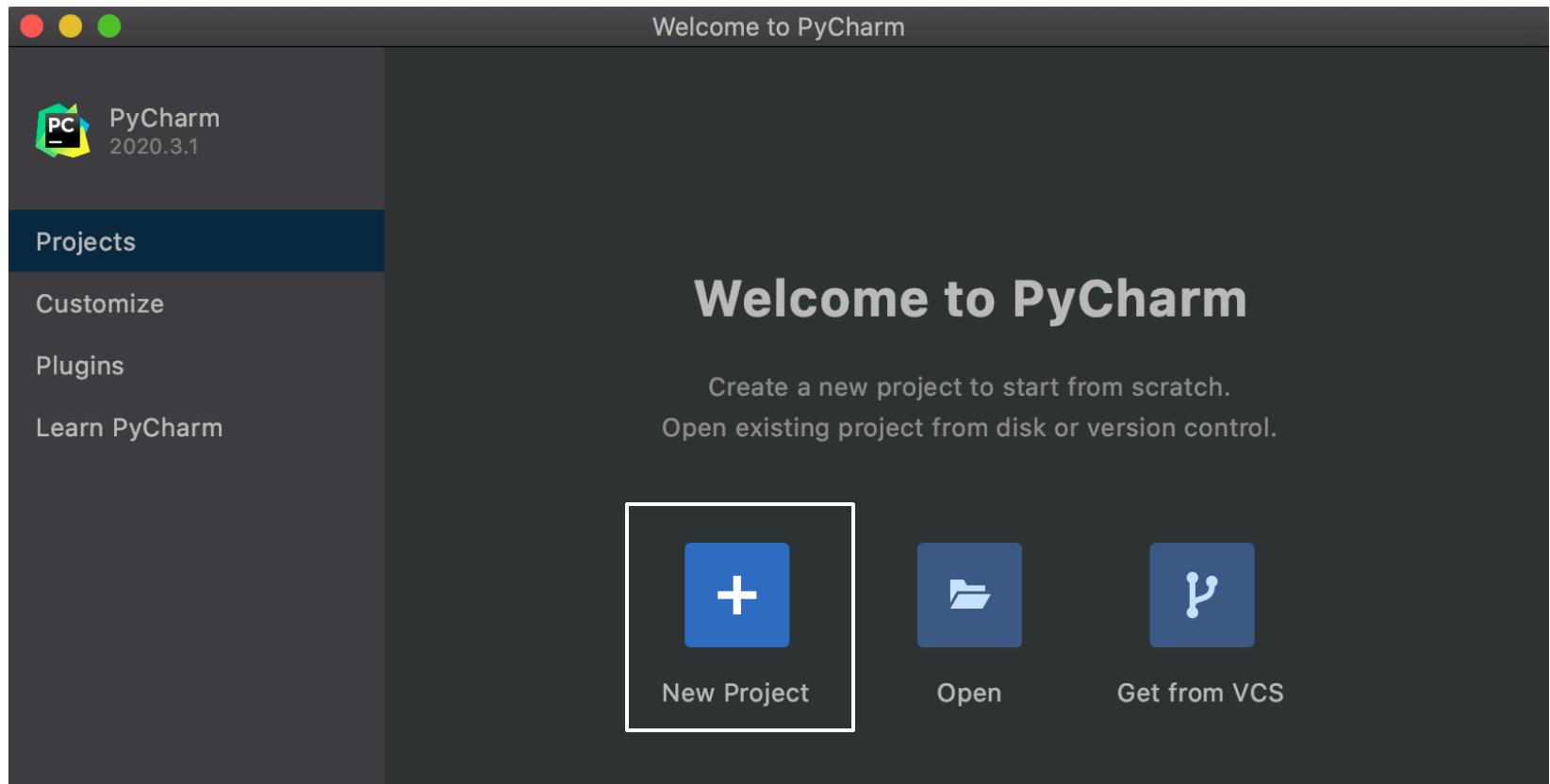
### 3. Tạo và thực thi project Python với PyCharm

#### »» Tạo project với PyCharm



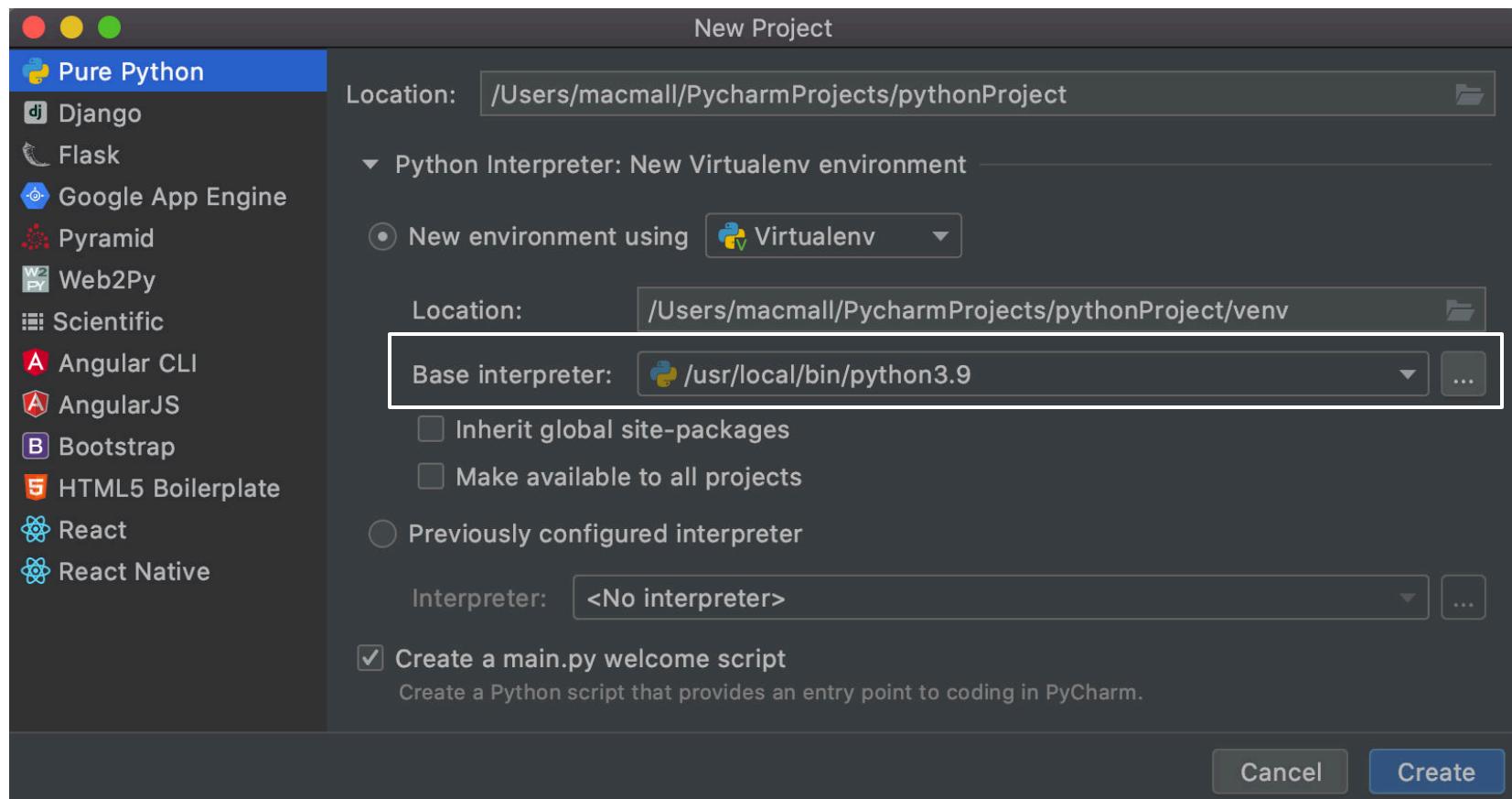
### 3. Tạo và thực thi project Python với PyCharm

#### »» Tạo project với PyCharm



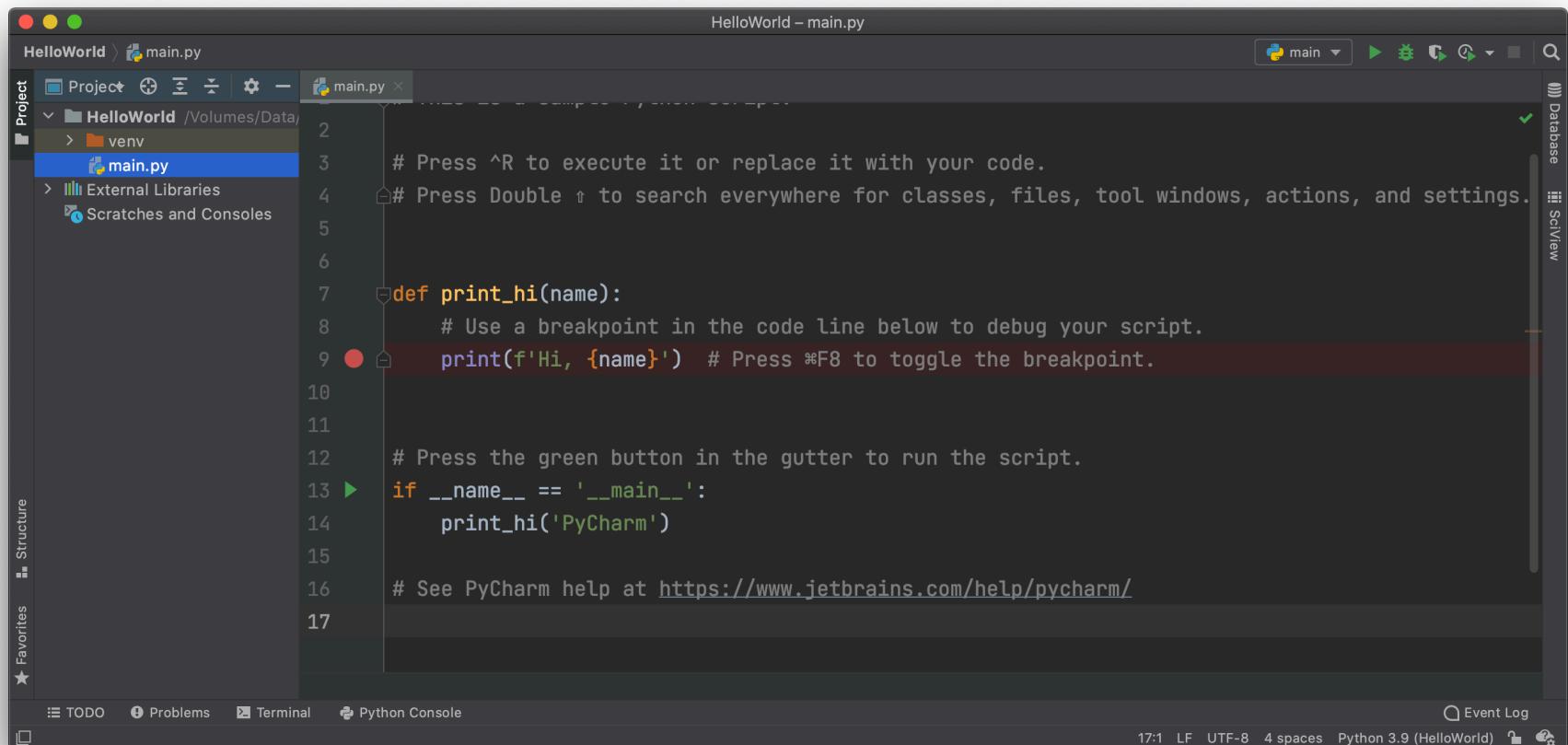
### 3. Tạo và thực thi project Python với PyCharm

#### »» Tạo project với PyCharm



# 3. Tạo và thực thi project Python với PyCharm

## »» Tạo project với PyCharm



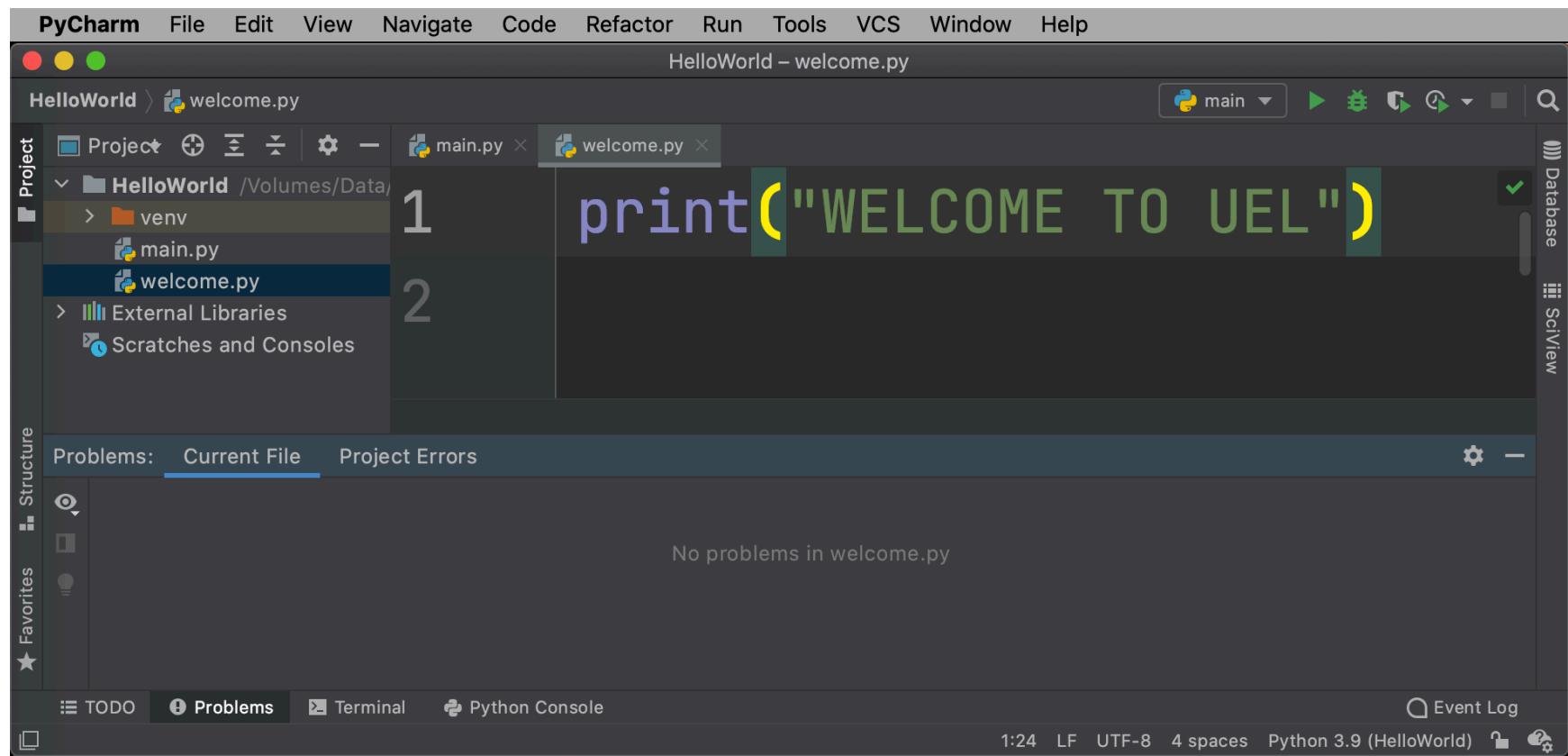
```
HelloWorld - main.py
main.py

2
3     # Press ^R to execute it or replace it with your code.
4     # Press Double ↑ to search everywhere for classes, files, tool windows, actions, and settings.
5
6
7     def print_hi(name):
8         # Use a breakpoint in the code line below to debug your script.
9         print(f'Hi, {name}')    # Press *F8 to toggle the breakpoint.
10
11
12     # Press the green button in the gutter to run the script.
13     if __name__ == '__main__':
14         print_hi('PyCharm')
15
16     # See PyCharm help at https://www.jetbrains.com/help/pycharm/
17
```

The screenshot shows the PyCharm interface with a project named "HelloWorld". The "main.py" file is open in the editor. A red dot at line 9 indicates a breakpoint. The code prints "Hi, PyCharm!". The PyCharm status bar at the bottom right shows "17:1 LF UTF-8 4 spaces Python 3.9 (HelloWorld)".

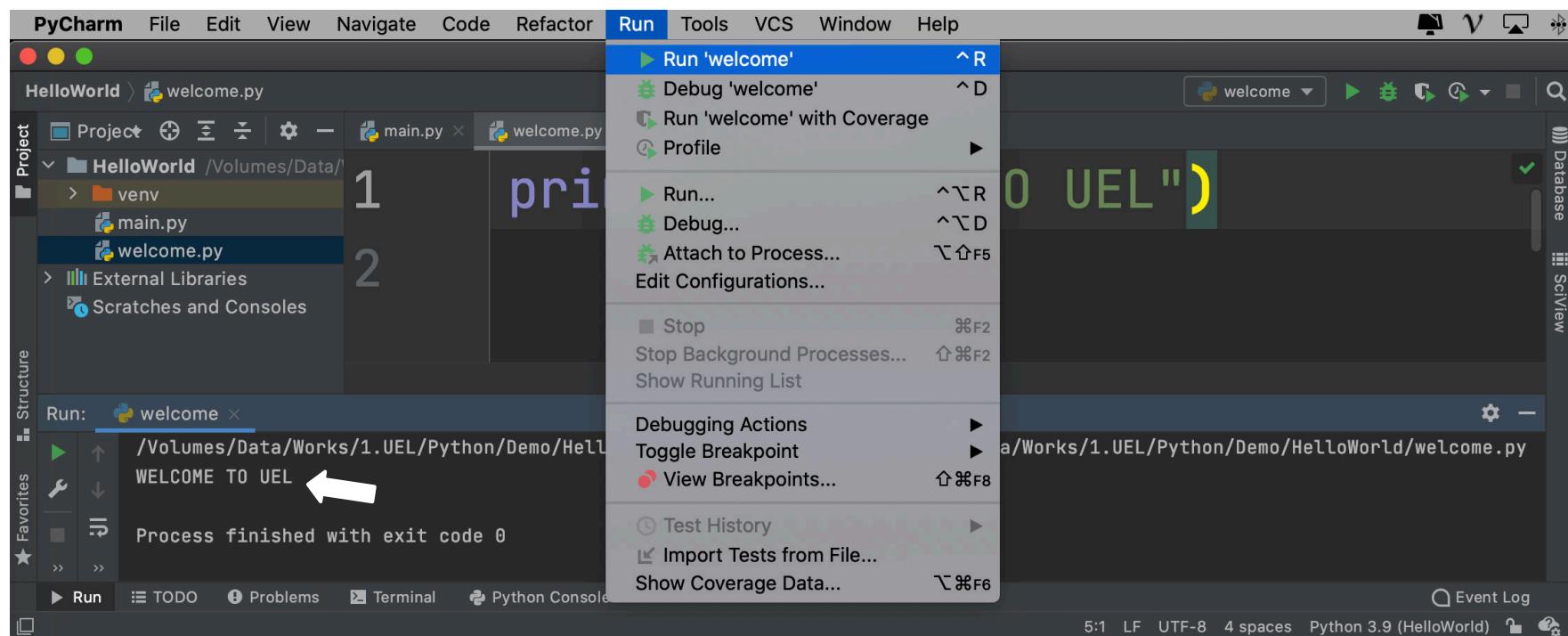
### 3. Tạo và thực thi project Python với PyCharm

#### »» Tạo project với PyCharm



### 3. Tạo và thực thi project Python với PyCharm

#### »» Tạo project với PyCharm



# CÁC KHÁI NIỆM CƠ BẢN

# NỘI DUNG

1. Các kiểu dữ liệu cơ sở
2. Khai báo và sử dụng biến (variable)
3. Chú thích mã nguồn
4. Các toán tử
5. Chương trình nhập liệu từ bàn phím
6. Định dạng dữ liệu xuất ra màn hình
7. Các loại lỗi

# 1. Các kiểu dữ liệu cơ sở

## »» Kiểu số

Kiểu	Mô tả	Ví dụ
<b>int</b>	Số nguyên: nguyên dương và nguyên âm.	6688, -379, ...
<b>float</b>	Số thực	8.9, 6.68, -7.9, ...
<b>complex</b>	Số phức: $z = 6+8j$ , phần thực là 6, phần ảo là 8 $z = \text{complex}(6, 8)$ $z.\text{real} \rightarrow$ phần thực $z.\text{imag} \rightarrow$ phần ảo	$z = 6+8j$

# 1. Các kiểu dữ liệu cơ sở

## »» Kiểu chuỗi

Kiểu	Mô tả	Ví dụ
str	Lưu giá trị chuỗi được bao trong nháy đơn hoặc nháy đôi	“UEL” “I LOVE UEL”

## »» Kiểu luận lý

Kiểu	Mô tả	Ví dụ
bool	Lưu giá trị True/False	b = True

## 2. Khai báo và sử dụng biến

Trong Python *không cần khai báo kiểu dữ liệu cho biến*. Từ giá trị được gán cho biến, Python sẽ tự nội suy ra kiểu dữ liệu của biến.

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** A project named "(DataType)" is open, located at "/Volumes/Data/Works/1.U". It contains a folder "venv" and two files: "data\_type\_example.py" and "main.py".
- Code Editor:** The file "data\_type\_example.py" is selected and displayed. The code defines a variable `v` and prints its type using `print(type(v))`. The code is as follows:

```
v = 68
print(type(v))
v = 6.68
print(type(v))
v = "I LOVE UEL"
print(type(v))
v = False
print(type(v))
```
- Run Tab:** The run configuration "data\_type\_example" is selected. The output window shows the results of running the code:

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

## 2. Khai báo và sử dụng biến

### »» Xóa biến

```
v = 68  
print(v)  
del v ←  
print(v)
```

```
Traceback (most recent call last):  
  File "/data_type_example.py", line 4, in <module>  
    print(v)  
NameError: name 'v' is not defined
```

### 3. Chú thích mã nguồn

#### »»» Tại sao nên chú thích mã nguồn?

Chú thích trong Python là *những ghi chú mà chương trình sẽ bỏ qua trong quá trình thông dịch*.

Tại sao nên chú thích mã nguồn???

### 3. Chú thích mã nguồn

#### »» Chú thích 1 dòng

```
# Đây là chương trình tìm số lớn nhất
a = 6
b = 8
if a >= b:
    print("Số lớn nhất là: " + str(a))
else:
    print("Số lớn nhất là: " + str(b))
```

### 3. Chú thích mã nguồn

#### »» Chú thích nhiều dòng

Sử dụng 3 cặp nháy đôi ("""" ... """") hoặc 3 cặp nháy đơn ("...")

```
"""
```

*Đây là chương trình tìm số lớn nhất  
trong 2 số a và b*

```
"""
```

```
a = 6
```

```
b = 8
```

```
if a >= b:
```

```
    print("Số lớn nhất là: " + str(a))
```

```
else:
```

```
    print("Số lớn nhất là: " + str(b))
```

## 4. Các toán tử

### »» Toán tử số học cơ bản

Toán tử	Mô tả	Ví dụ
+	Cộng	$26 + 2.9 \Rightarrow$ kết quả 28.9
-	Trừ	$4.99 - 5 \Rightarrow$ kết quả -0.01
*	Nhân	$2 * 3.1 \Rightarrow$ kết quả 6.2
/	Chia	$7 / 2 \Rightarrow$ kết quả 3.5
//	Chia lấy nguyên	$7 // 2 \Rightarrow$ kết quả 3
%	Chia lấy dư	$7 \% 2 \Rightarrow$ kết quả 1
**	Lũy thừa	$2 ** 3 \Rightarrow$ kết quả 8

# 4. Các toán tử

## »» Toán tử gán

Toán tử	Mô tả	Ví dụ	Tương đương
=	Giá trị cho biến	$x = 68$	
+=	Cộng và gán	$x = 3, x += 6$ $\rightarrow x = 9$	$x = x + 6$
-=	Trừ và gán	$x = 3, x -= 6$ $\rightarrow x = -3$	$x = x - 6$
*=	Nhân và gán	$x = 3, x *= 6$ $\rightarrow x = 18$	$x = x * 6$

## 4. Các toán tử

### »» Toán tử gán

Toán tử	Mô tả	Ví dụ	Tương đương
$/=$	Chia và gán	$x = 7, x /= 2$ $\rightarrow x = 3.5$	$x = x / 2$
$//=$	Chia lấy nguyên và gán	$x = 7, x //= 2$ $\rightarrow x = 3$	$x = x // 2$
$\%=$	Chia lấy dư và gán	$x = 7, x \%= 2$ $\rightarrow x = 1$	$x = x \% 2$
$**=$	Lũy thừa và gán	$x = 2, x **= 3$ $\rightarrow x = 8$	$x = x ** 3$

## 4. Các toán tử

### »» Toán tử so sánh

Toán tử	Mô tả	Ví dụ
<code>==</code>	Ss bằng	$2 == 2 \Rightarrow \text{True}$
<code>!=</code>	Ss khác	$6 != 8 \Rightarrow \text{True}$
<code>&lt;</code>	Ss bé hơn	$9 < 7 \Rightarrow \text{False}$
<code>&lt;=</code>	Ss bé hơn hoặc bằng	$7 <= 7 \Rightarrow \text{True}$
<code>&gt;</code>	Ss lớn hơn	$7 > 9 \Rightarrow \text{False}$
<code>&gt;=</code>	Ss lớn hơn hoặc bằng	$7 >= 2 \Rightarrow \text{True}$

## 4. Các toán tử

### »» Toán tử so sánh

Toán tử	Mô tả	Ví dụ
<b>is</b>	Trả về <b>True</b> nếu các biến ở hai bên toán tử cùng trả về một đối tượng (hoặc cùng giá trị), ngược lại là <b>False</b>	$x = 6, y = 6$ <code>print(x is y) → True</code>
<b>is not</b>	Trả về <b>False</b> nếu các biến ở hai bên toán tử cùng trả về một đối tượng (hoặc cùng giá trị), ngược lại là <b>True</b>	$x = 6, y = 6$ <code>print(x is not y) → False</code>

\*Lưu ý: toán tử “is” khác với “==”, “is” sẽ so sánh qua id

$x = [1, 2]$       `print(x is y) → False`

$y = [1, 2]$       `print(x == y) → True`

## 4. Các toán tử

### »» Toán tử logic

Toán tử	Mô tả	Ví dụ
<b>and</b>	Toán tử “Và”	$x = 9$ <code>print(x % 3 == 0 <b>and</b> x &gt; 8) → True</code> $2 < x < 11 \rightarrow \text{True}$
<b>or</b>	Toán tử “Hoặc”	$x = 9$ <code>print(x % 2 == 0 <b>or</b> x &gt; 8) → True</code>
<b>not</b>	Toán tử “Phủ định”	$x = 9$ <code>print(<b>not</b> x % 3 == 0) → False</code>

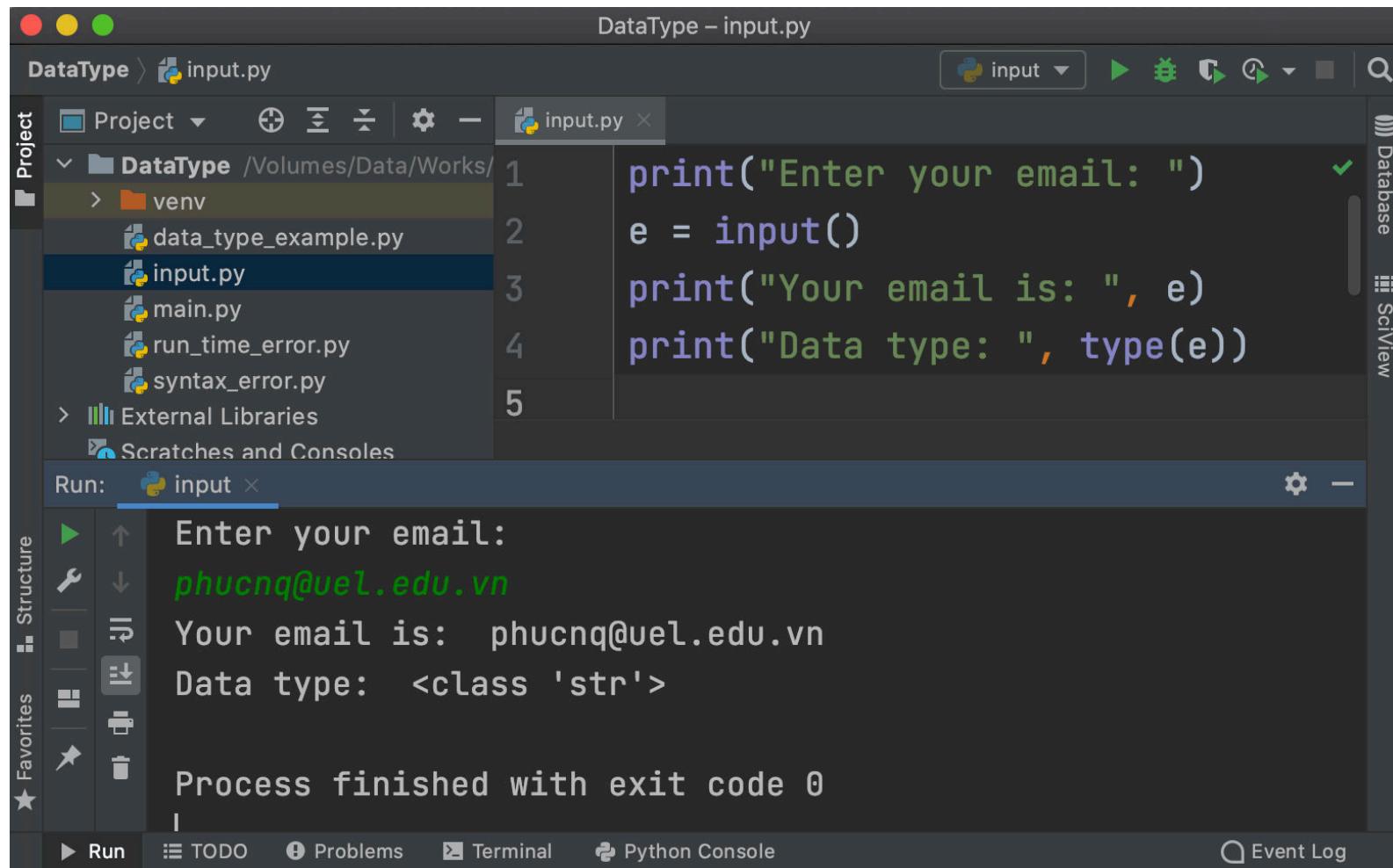
## 4. Các toán tử

### »» Độ ưu tiên các toán tử

Ưu tiên	Toán tử
1	**
2	* / % //
3	+ -
4	<= < > >=
5	== !=
6	= %= /= //= -= += *= **=
7	is, is not
8	not, and, or

# 5. Chương trình nhập liệu từ bàn phím

## »» Nhận giá trị từ bàn phím



The screenshot shows the PyCharm IDE interface with a project named "(DataType)". The "input.py" file is open in the editor, containing the following code:

```
print("Enter your email: ")
e = input()
print("Your email is: ", e)
print("Data type: ", type(e))
```

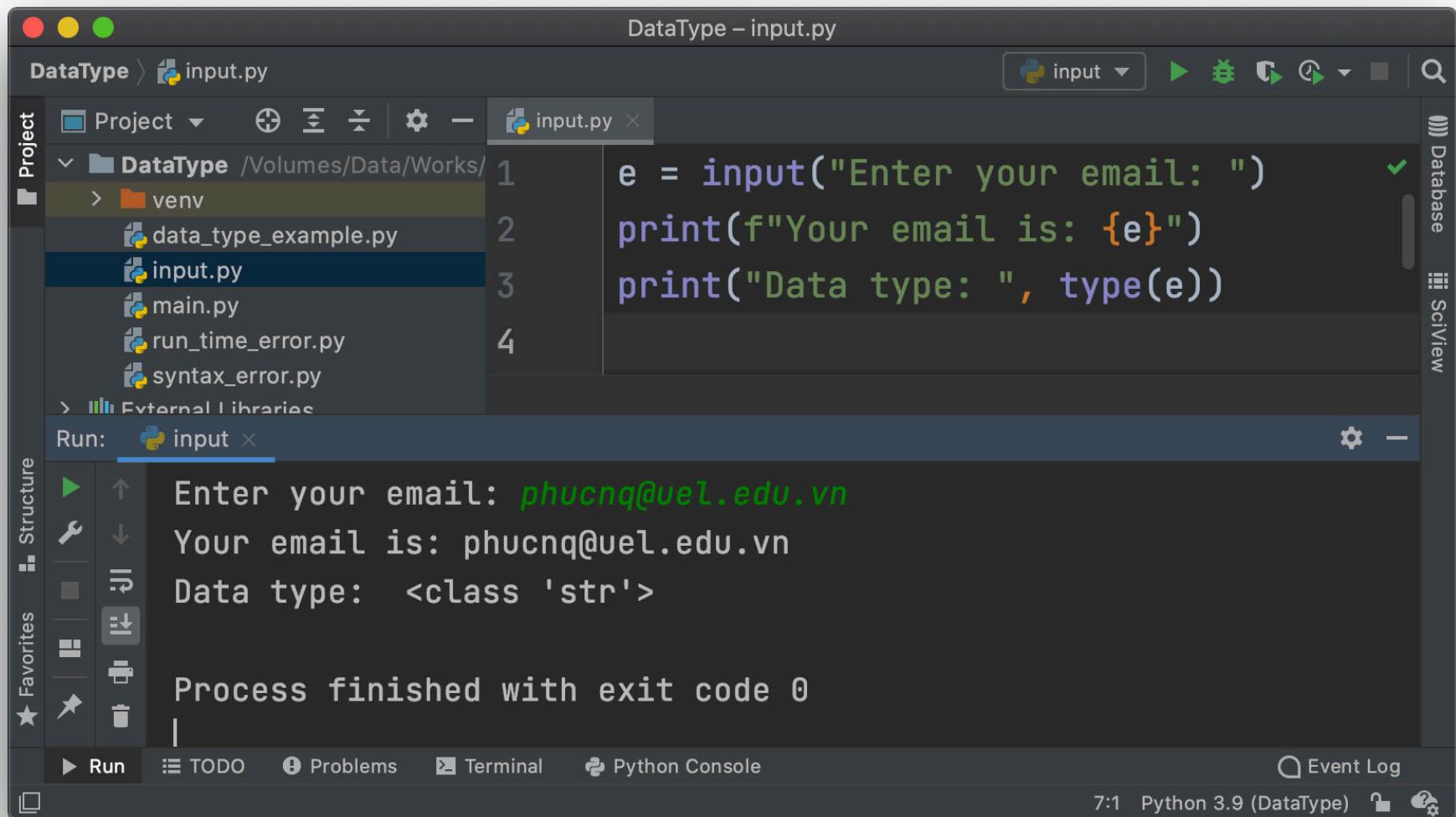
The "Run" tool window at the bottom shows the output of running the script:

```
Enter your email:
phucnq@uel.edu.vn
Your email is: phucnq@uel.edu.vn
Data type: <class 'str'>

Process finished with exit code 0
```

# 5. Chương trình nhập liệu từ bàn phím

## »» Nhận giá trị từ bàn phím



The screenshot shows a Python development environment with the following details:

- Project:** DataType
- File:** input.py
- Code:**

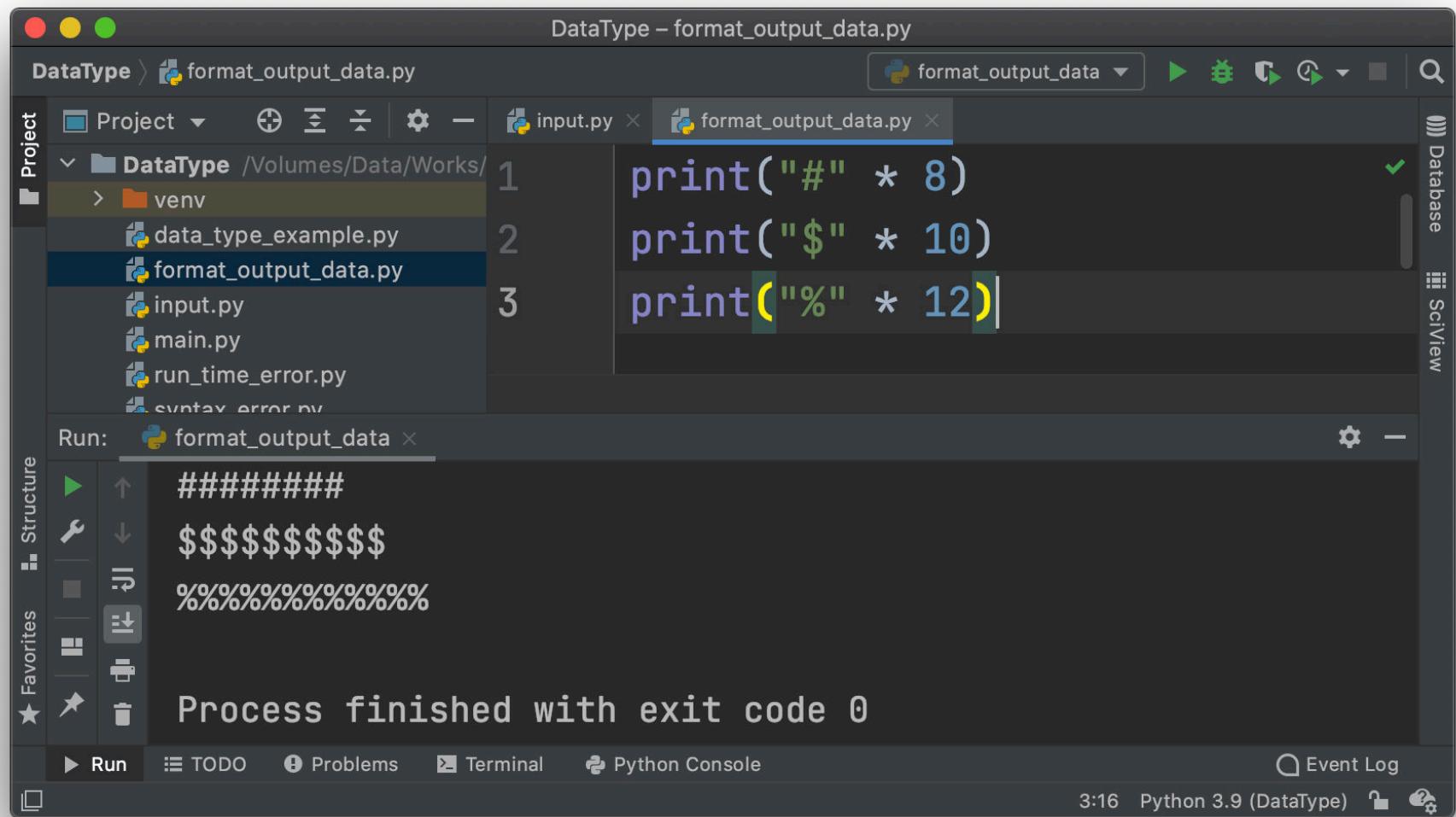
```
1 e = input("Enter your email: ")
2 print(f"Your email is: {e}")
3 print("Data type: ", type(e))
```
- Run:** input
- Output:**

```
Enter your email: phucnq@uel.edu.vn
Your email is: phucnq@uel.edu.vn
Data type: <class 'str'>

Process finished with exit code 0
```
- Bottom Navigation:** Run, TODO, Problems, Terminal, Python Console, Event Log

# 6. Định dạng dữ liệu xuất ra màn hình

## »» Xuất dữ liệu lặp lại



The screenshot shows a PyCharm IDE interface with the following details:

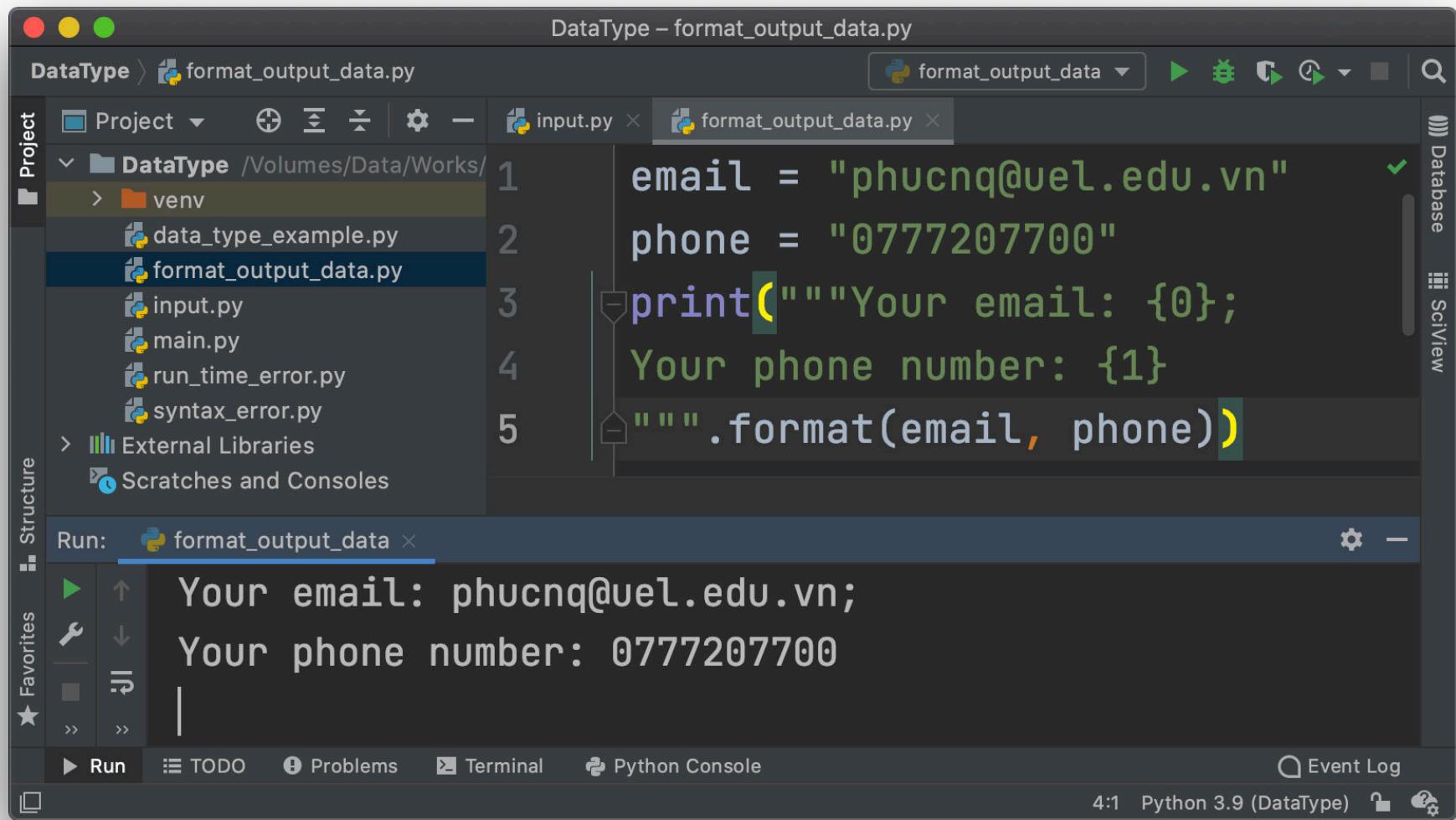
- Title Bar:**(DataType – format\_output\_data.py)
- Toolbar:** Includes icons for Run, Stop, Refresh, and others.
- Project View:** Shows a project named "DataType" containing files: "data\_type\_example.py", "format\_output\_data.py" (selected), "input.py", "main.py", "run\_time\_error.py", and "syntax\_error.py".
- Code Editor:** Displays the following Python code:

```
print("#" * 8)
print("$" * 10)
print("%" * 12)
```
- Run Console:** Shows the output of the run command:

```
#####
$$$$$$$$$#
%%%%%%%%%
```
- Status Bar:** Shows "Process finished with exit code 0".
- Bottom Navigation:** Includes tabs for Run, TODO, Problems, Terminal, Python Console, and Event Log, along with a status bar showing "3:16 Python 3.9 (DataType)" and system icons.

# 6. Định dạng dữ liệu xuất ra màn hình

## »» Xuất dữ liệu sử dụng hàm **format**



```
(DataType) format_output_data.py
```

```
Project Data Type /Volumes/Data/Works/ venv
  > dataType_example.py
  > format_output_data.py
  input.py
  main.py
  run_time_error.py
  syntax_error.py
> External Libraries
Scratches and Consoles
```

```
Run: format_output_data
```

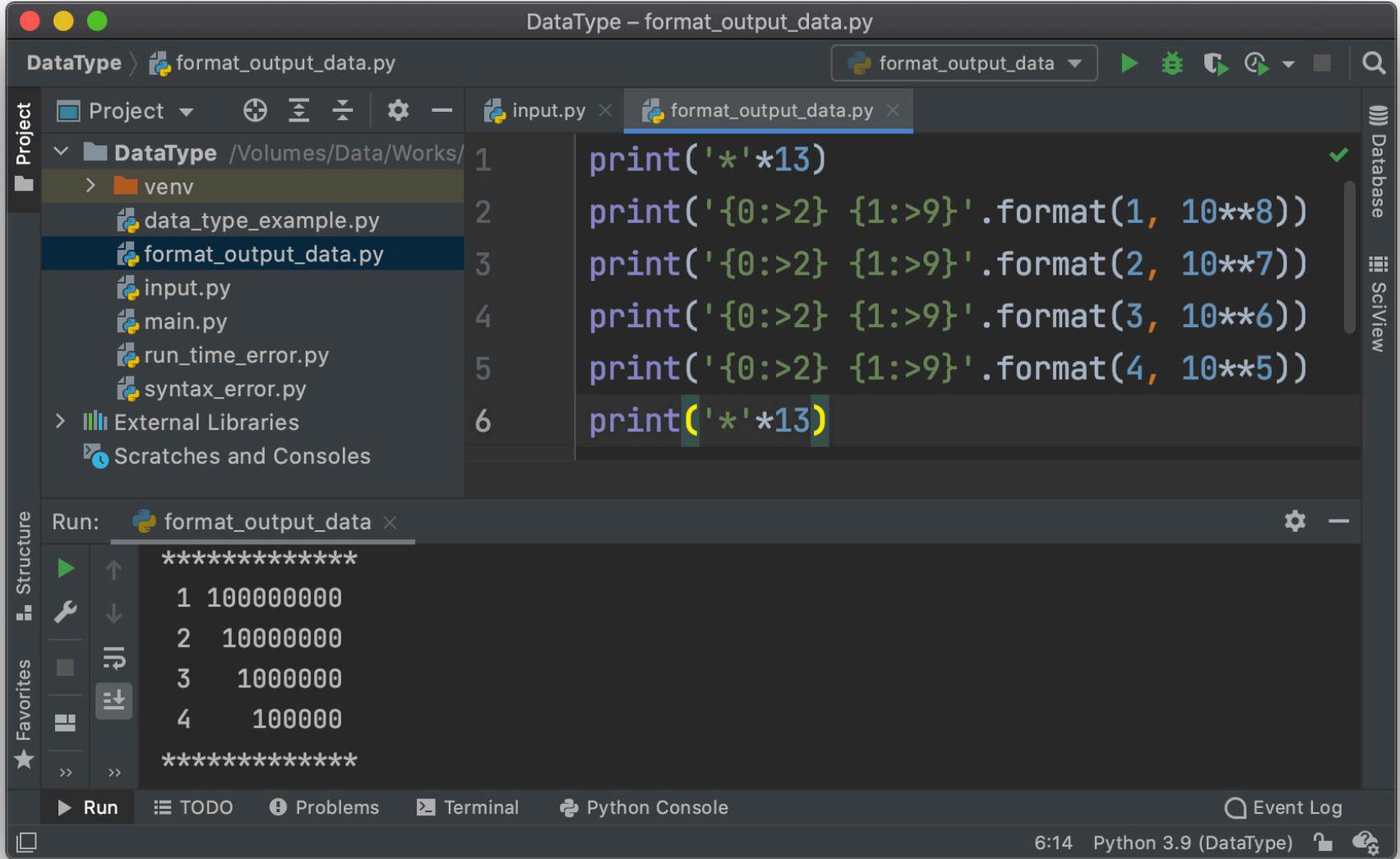
```
1 email = "phucnq@uel.edu.vn"
2 phone = "0777207700"
3 print("""Your email: {0};
4 Your phone number: {1}
5 """.format(email, phone))
```

```
Your email: phucnq@uel.edu.vn;
Your phone number: 0777207700
```

Event Log

# 6. Định dạng dữ liệu xuất ra màn hình

## »» Xuất dữ liệu căn lề sử dụng hàm **format**



The screenshot shows a PyCharm IDE interface with the following details:

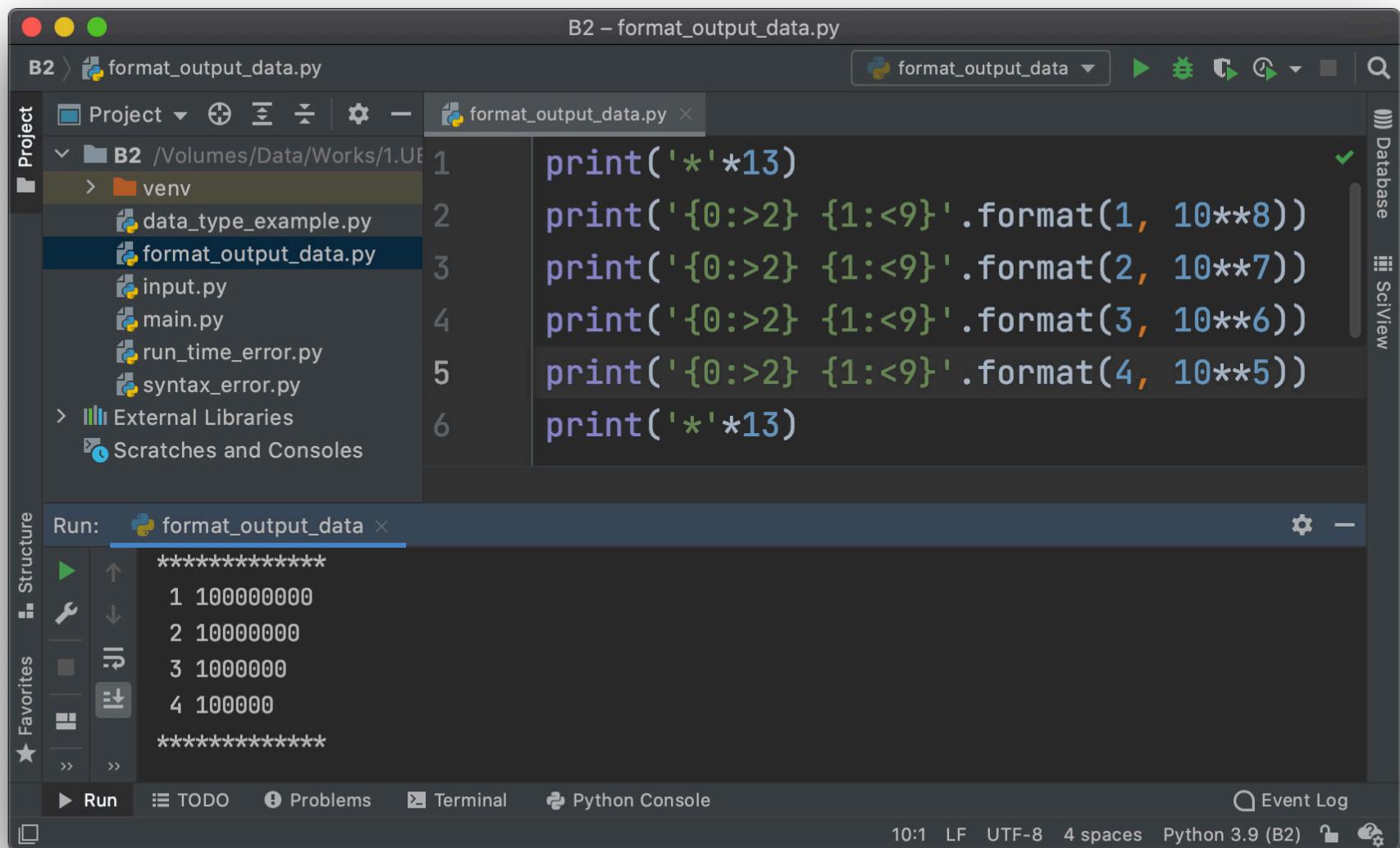
- Project:** The project is named "(DataType)". It contains a folder "(DataType)" with files: "data\_type\_example.py", "format\_output\_data.py" (selected), "input.py", "main.py", "run\_time\_error.py", and "syntax\_error.py". There are also "External Libraries" and "Scratches and Consoles".
- Code Editor:** The file "format\_output\_data.py" is open. The code uses the `format` function to print numbers with specific widths and alignments:

```
1 print('*'*13)
2 print('{0:>2} {1:>9}'.format(1, 10**8))
3 print('{0:>2} {1:>9}'.format(2, 10**7))
4 print('{0:>2} {1:>9}'.format(3, 10**6))
5 print('{0:>2} {1:>9}'.format(4, 10**5))
6 print('*'*13)
```
- Run Tab:** The "Run" tab shows the output of the code:

```
*****
1 1000000000
2 100000000
3 10000000
4 1000000
*****
```
- Bottom Navigation:** The bottom navigation bar includes "Run", "TODO", "Problems", "Terminal", "Python Console", "Event Log", and a status bar showing "6:14 Python 3.9 (DataType)".

# 6. Định dạng dữ liệu xuất ra màn hình

## »» Xuất dữ liệu căn lề sử dụng hàm **format**



```
B2 – format_output_data.py
B2 > format_output_data.py
Project  format_output_data.py
B2 /Volumes/Data/Works/1.UE
  > venv
    data_type_example.py
    format_output_data.py
    input.py
    main.py
    run_time_error.py
    syntax_error.py
  > External Libraries
  Scratches and Consoles

1 print('*'*13)
2 print('{0:>2} {1:<9}'.format(1, 10**8))
3 print('{0:>2} {1:<9}'.format(2, 10**7))
4 print('{0:>2} {1:<9}'.format(3, 10**6))
5 print('{0:>2} {1:<9}'.format(4, 10**5))
6 print('*'*13)

Run: format_output_data
*****
1 1000000000
2 100000000
3 10000000
4 1000000
*****
```

The screenshot shows a PyCharm IDE interface. The top bar displays the title "B2 – format\_output\_data.py". The left sidebar shows a project structure with files like "data\_type\_example.py", "format\_output\_data.py", and "input.py". The main editor window contains a Python script with the following code:

```
print('*'*13)
print('{0:>2} {1:<9}'.format(1, 10**8))
print('{0:>2} {1:<9}'.format(2, 10**7))
print('{0:>2} {1:<9}'.format(3, 10**6))
print('{0:>2} {1:<9}'.format(4, 10**5))
print('*'*13)
```

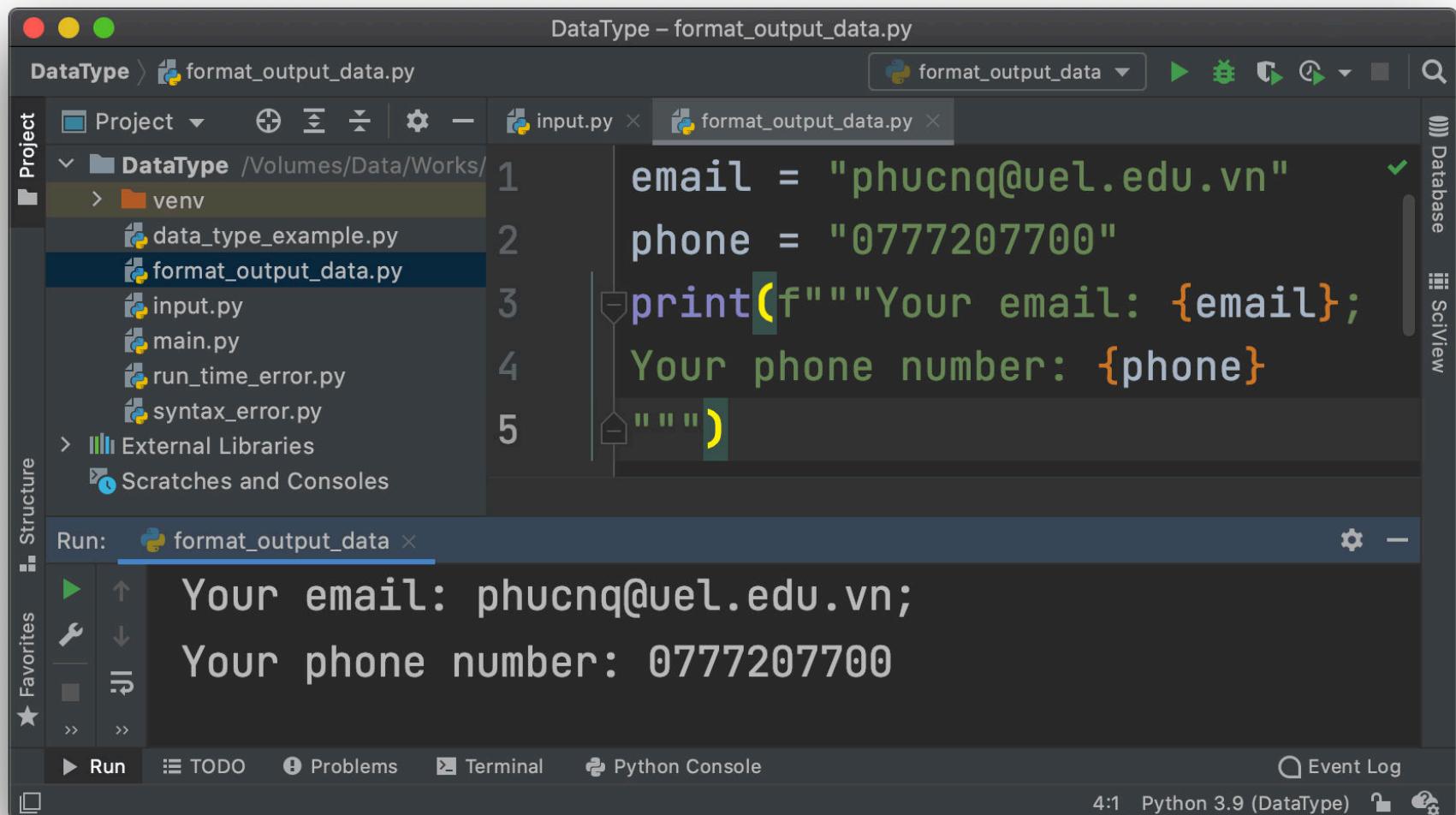
The "Run" tab at the bottom is selected, showing the output of the script:

```
*****
1 1000000000
2 100000000
3 10000000
4 1000000
*****
```

The bottom status bar indicates the current time is 10:1, the line separator is LF, the encoding is UTF-8, there are 4 spaces, and the Python version is 3.9 (B2).

# 6. Định dạng dữ liệu xuất ra màn hình

»»» Xuất dữ liệu sử dụng `print(f"...{variable_name}...")`



```
(DataType) format_output_data.py
```

```
Project Database SciView
```

```
(DataType) /Volumes/Data/Works/
```

```
email = "phucnq@uel.edu.vn"
phone = "0777207700"
print(f"""Your email: {email};
Your phone number: {phone}
""")
```

```
Run: format_output_data
```

```
Your email: phucnq@uel.edu.vn;
Your phone number: 0777207700
```

```
Run TODO Problems Terminal Python Console Event Log
```

```
4:1 Python 3.9 (DataType)
```

# 7. Các loại lỗi

## »» Lỗi cú pháp (Syntax Errors)

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** DataType
- File:** syntax\_error.py
- Code Content:**

```
1     x = 9
2     y = 11
3     x += y
4     y + 2 = x
```

The line `y + 2 = x` is highlighted in red, indicating a syntax error.
- Run Tab:** syntax\_error
- Output:**

```
File "/Volumes/Data/Works/1.UEL/Python/Demo(DataType)/syntax_error.py", line 4
      y + 2 = x
      ^
SyntaxError: cannot assign to operator
```

Process finished with exit code 1
- Status Bar:** 8:1 LF UTF-8 4 spaces Python 3.9 (DataType) Event Log

# 7. Các loại lỗi

## » Lỗi thực thi (Run-time Exceptions)

The screenshot shows the PyCharm IDE interface. The project is named "DataType". The "run\_time\_error.py" file is open in the editor. The code attempts to divide two integers input by the user. A run-time error occurs when the divisor is zero.

```
print("Mời bạn nhập vào 2 số thực hiện phép chia")
a = input("Mời bạn nhập vào số bị chia: ")
b = input("Mời bạn nhập vào số chia: ")
c = int(a)/int(b)
print(a + "/" + b + " = " + str(c))
```

The run console output shows:

```
Mời bạn nhập vào 2 số thực hiện phép chia
Mời bạn nhập vào số bị chia: 6
Mời bạn nhập vào số chia: 0
Traceback (most recent call last):
  File "/Volumes/Data/Works/1.UEL/Python/Demo(DataType/run_time_error.py", line 4, in <module>
    c = int(a)/int(b)
ZeroDivisionError: division by zero

Process finished with exit code 1
```

The status bar at the bottom indicates the environment settings: 11:1 LF UTF-8 4 spaces Python 3.9 (DataType).

## 7. Các loại lỗi

### »» Lỗi nghiệp vụ (Logic Errors)

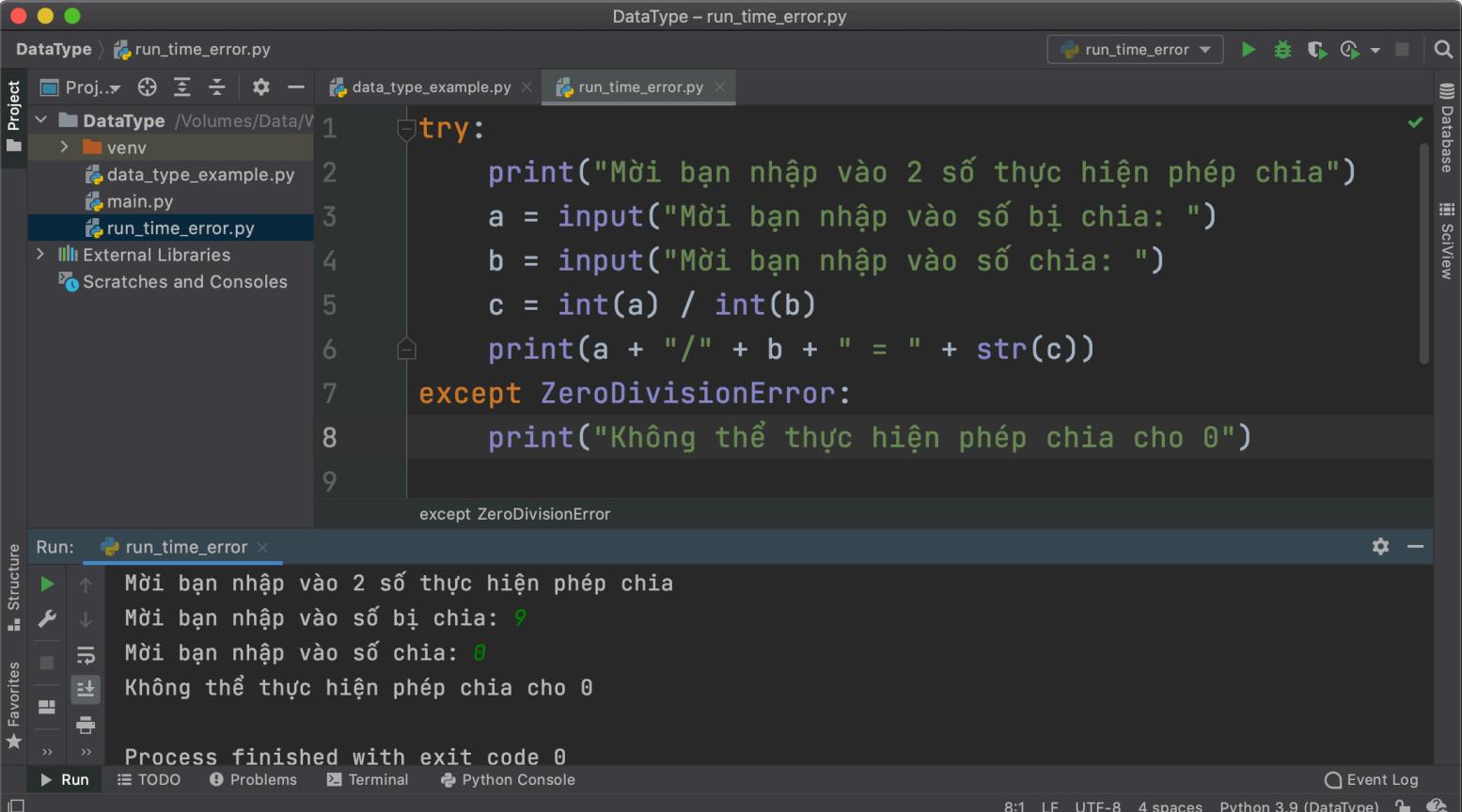
Lỗi về tư duy logic khi lập trình viên viết chương trình → đây là lỗi rất nghiêm trọng.

Lập trình viên nên **cẩn trọng** và **unit test kỹ** trong quá trình phát triển ứng dụng

# 7. Các loại lỗi

## »» Bắt lỗi

Python hỗ trợ **try...except** để bắt lỗi Runtime, giúp báo rõ loại lỗi chương trình đang gặp và vẫn tiếp tục hoạt động khi gặp lỗi.



```
(DataType) run_time_error.py
(DataType) run_time_error.py
1 try:
2     print("Mời bạn nhập vào 2 số thực hiện phép chia")
3     a = input("Mời bạn nhập vào số bị chia: ")
4     b = input("Mời bạn nhập vào số chia: ")
5     c = int(a) / int(b)
6     print(a + "/" + b + " = " + str(c))
7 except ZeroDivisionError:
8     print("Không thể thực hiện phép chia cho 0")
9

except ZeroDivisionError

Run: run_time_error
Mời bạn nhập vào 2 số thực hiện phép chia
Mời bạn nhập vào số bị chia: 9
Mời bạn nhập vào số chia: 0
Không thể thực hiện phép chia cho 0

Process finished with exit code 0
```

The screenshot shows a Python script named `run_time_error.py` in a PyCharm IDE. The code demonstrates the use of a `try...except` block to handle a `ZeroDivisionError`. When run, it prompts the user to enter two numbers for division. If the divisor is zero, it catches the error and prints an appropriate message. The run console at the bottom shows the interaction and the resulting output.

# CẤU TRÚC ĐIỀU KHIỂN VÀ VÒNG LẶP

# NỘI DUNG

1. Boolean Expression trong Python
2. Cấu trúc điều khiển
3. Cấu trúc lặp

# 1. Boolean Expression trong Python

## »»» Boolean Expression

**Boolean** là *kiểu dữ liệu luận lý* chỉ có 2 giá trị: *đúng (True)* hoặc *sai (False)*.

**Boolean Expression** là biểu thức trả về kiểu giá trị luận lý True/False.

# 1. Boolean Expression trong Python

## »»» Boolean Expression

Biểu thức	Kết quả
$6 < 8$	True
$9 \geq 12$	False
$8 * 0 \neq 0$	False

Biểu thức	Kết quả
$x = 6$ $y = 6$	$x \text{ is } y \rightarrow \text{True}$
$x = [1, 2]$ $y = [1, 2]$	$x \text{ is } y \rightarrow \text{False}$

Biểu thức	Kết quả
“UEL” == “UEL”	True
‘Hello’ == ‘hello’	False ( $\text{ord}(\text{'H'}) \rightarrow 72, \text{ord}(\text{'h'}) \rightarrow 104$ )
“Happy” > “Harry”	False ( $\text{ord}(\text{'p'}) \rightarrow 112, \text{ord}(\text{'r'}) \rightarrow 114$ )

# 1. Boolean Expression trong Python

## »»» Boolean Expression

Biểu thức	Kết quả
<code>n = 8</code>	
<code>n &gt; 4 and n &lt; 9</code> Cách viết khác: <code>4 &lt; n &lt; 9</code>	True
<code>n = 6</code>	
<code>n == 4 or n == 6 or n == 8</code> Cách viết khác: <code>n in (4, 6, 8)</code>	True

# 1. Boolean Expression trong Python

## »»» Boolean Expression

\*Lưu ý khi so sánh biểu thức trả giá trị số thực.

```
f1 = 1.11 - 1.10
```

```
f2 = 2.11 - 2.10
```

```
print(f1 == f2) → False (Tại sao?)
```

```
print(f1) → 0.01000000000000009
```

```
print(f2) → 0.00999999999999787
```

→ Đặt một ngưỡng sai số cho phép khi so sánh số thực

## 2. Cấu trúc điều khiển

### »» Cấu trúc “if”, “if ... else ...”

Khi điều kiện đúng → khối lệnh bên trong “if” sẽ được thực thi, điều kiện sai → khối lệnh trong “else” được thực thi (nếu có).

**if** **condition** : **block**

```
dtb = float(input("Nhập điểm của bạn: "))
if dtb >= 5:
    print("Chúc mừng bạn đã qua môn :)")
```

**if** **condition** : **if-block**  
**else:** **else-block**

```
dtb = float(input("Nhập điểm của bạn: "))
if dtb >= 5:
    print("Chúc mừng bạn đã qua môn :)")
else:
    print("Hẹn gặp lại bạn kỳ sau :(")
```

## 2. Cấu trúc điều khiển

»» Cấu trúc “if...else...” viết gọn

*expression-1* if *condition* else *expression-2*

```
dtb = float(input("Nhập điểm của bạn: "))
kq = "Đậu" if dtb >= 5 else "Rớt"
```

## 2. Cấu trúc điều khiển

### »» Cấu trúc “if ...elif”

```
dtb = float(input("Nhập điểm của bạn: "))
if dtb >= 8:
    print("Bạn đạt loại GIỎI")
elif dtb >= 6.5:
    print("Bạn đạt loại KHÁ")
elif dtb >= 5:
    print("Bạn đạt loại TRUNG BÌNH")
else:
    print("Hẹn gặp lại bạn kỳ sau :(")
```

## 2. Cấu trúc điều khiển

### »»» Từ khóa “pass”

```
a = float(input("Nhập hệ số a: "))
b = float(input("Nhập hệ số b: "))
if a == 0:
    Để tránh → gây lỗi
else:
    x = -b/a
    print(f"PT {a}x + {b} = 0 có nghiệm x = {x}")
```

```
a = float(input("Nhập hệ số a: "))
b = float(input("Nhập hệ số b: "))
if a == 0:
    pass
else:
    x = -b/a
    print(f"PT {a}x + {b} = 0 có nghiệm x = {x}")
```

### 3. Cấu trúc lặp

#### »» Vòng lặp “while”

Vòng lặp “while” dùng để thực hiện lặp đi lặp lại các công việc trong khôi lệnh **block** khi điều kiện **condition** còn đúng.

Vòng lặp chỉ kết thúc khi điều kiện không còn thỏa mãn (sai) hoặc gặp lệnh **break**.

```
while condition :  
    block
```

### 3. Cấu trúc lặp

#### »» Vòng lặp “while”

VD: viết chương trình yêu cầu nhập vào số nguyên dương trong đoạn [1..10], nếu nhập sai yêu cầu nhập lại. Nhập đúng thì in giá trị mới nhập ra màn hình.

```
x = -1
while x <= 0 or x > 10:
    x = int(input("Nhập vào giá trị [1..10]: "))
    if x <= 0 or x > 10:
        print("Bạn đã nhập sai. Vui lòng nhập lại!")
    print(f"x = {x}")
```

BT: viết chương trình tính tổng từ 1 đến n, n là giá trị người dùng nhập.

### 3. Cấu trúc lặp

#### »» Vòng lặp “for”

Vòng lặp “for” dùng để lặp tuân tự các công việc, for sử dụng **range(begin, end, step)** để định nghĩa vùng dữ liệu lặp và bước lặp.

**for n in range(10):**

    print(n, end=' ') → 0 1 2 3 4 5 6 7 8 9

**for n in range(1, 10):**

    print(n, end=' ') → 1 2 3 4 5 6 7 8 9

**for n in range(1, 10, 2):**

    print(n, end=' ') → 1 3 5 7 9

**for n in range(10, 0, -1):**

    print(n, end=' ') → 10 9 8 7 6 5 4 3 2 1

**for n in range(2, 11, 2):**

    print(n, end=' ') → 2 4 6 8 10

BT: viết chương trình tính tổng chẵn từ 1 đến n, n là giá trị người dùng nhập.

### 3. Cấu trúc lặp

#### »» Lệnh “break”

Lệnh “break” dùng để thoát (kết thúc) vòng lặp.

while True:

s = input("Bạn đang FA? ")

print("Bạn đã xác nhận: ", s)

e = input("Tiếp tục chương trình không? (y/n): ")

if e == 'n':

    break

print("BYE!")

### 3. Cấu trúc lặp

#### »» Lệnh “continue”

Lệnh “continue” dùng để nhảy sớm tới lần lặp kế tiếp, các lệnh bên dưới “continue” sẽ không được thực thi trong lần lặp hiện tại.

VD: tính tổng các số lẻ từ 1 đến 23, ngoại trừ 5 và 11.

```
tong = 0
for numb in range(1, 24, 2):
    if numb == 5 or numb == 11:
        continue
    tong += numb
print("Tổng là: ", tong)
```

### 3. Cấu trúc lặp

#### »» Vòng lặp “while” kết hợp else

Khi vòng lặp **while** kết thúc bình thường (tức là không phải dùng lệnh break để kết thúc) thì khối lệnh else-block sẽ được thực hiện ngay sau đó.

**while** *condition* :

*while-block*

**else:**

*else-block*

```
i = 1
while i < 8:
    print(i, end=' ')
    i += 1
else:
    print("\ni is no longer less than 8")
```



1 2 3 4 5 6 7  
i is no longer less than 8

### 3. Cấu trúc lặp

#### »» Vòng lặp “for” kết hợp else

Khi vòng lặp **for** kết thúc bình thường (tức là không phải dùng lệnh `break` để kết thúc) thì khối lệnh `else-block` sẽ được thực hiện ngay sau đó.

**for** *expression* :  
    *for-block*

**else:**  
    *else-block*

```
for x in range(8):  
    print(x, end=' ')  
else:  
    print("\nFinally finished!")
```

→ 0 1 2 3 4 5 6 7  
Finally finished!

# 3. Cấu trúc lặp

## »»» Vòng lặp lồng nhau

The screenshot shows a PyCharm IDE interface with the following details:

- Title Bar:** B3 – loop.py
- Project View:** Shows a project named "B3" containing files: venv, boolean\_expression.py, break\_continue.py, if\_else.py, loop.py (selected), main.py, and External Libraries.
- Code Editor:** Displays Python code for printing a diamond pattern:

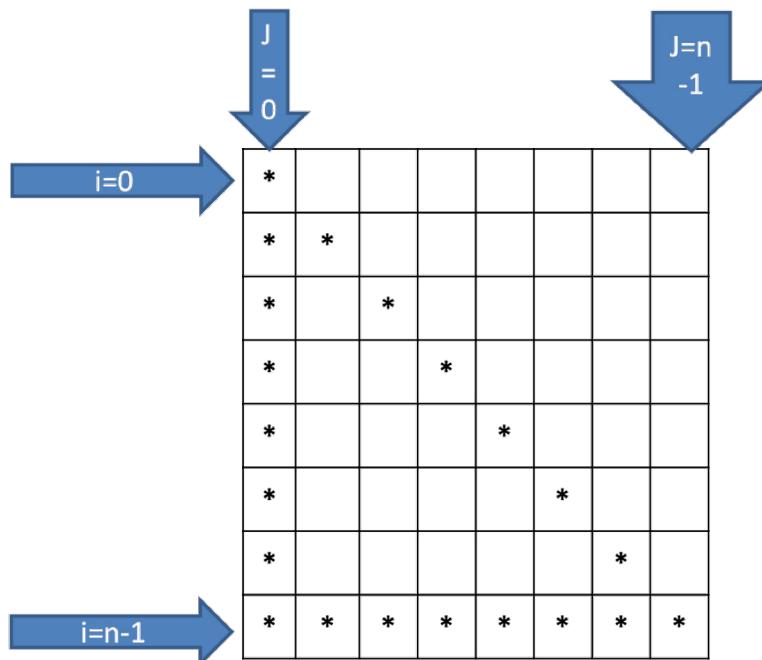
```
n = int(input("Nhập chiều cao: "))
for i in range(n):
    for j in range(n):
        if j == 0 or i == j or j == n - 1:
            print("*", end=' ')
        else:
            print(" ", end=' ')
    print()
```
- Run Tab:** Set to "loop". The run history shows the input "Nhập chiều cao: 6" and the resulting output:

```
*  *
** *
* * *
* * *
* * *
** *
*  *
```
- Bottom Status Bar:** Shows "Shortcuts conflicts: Query Console and 12 more shortcut conflict with macOS shortcuts. Modify these ... (today 8:33 AM)" and "11:1 Python 3.9 (B3)".

### 3. Cấu trúc lắp

# »» Vòng lặp lồng nhau

BT: viết chương trình vẽ các hình sau:



＊＊＊＊

\* \* \* \*

\*

\* \* \* \*

\* \* \* \*

\* \* \* \*

\* \* \*

\*

\* \* \*

\* \* \*

\* \* \* \*

\* \* \*

\*

3

\* \* \*

\* \* \*

# HÀM TRONG PYTHON

# NỘI DUNG

1. Khái quát về hàm trong Python
2. Hàm đệ quy
3. Một số hàm toán học thông dụng

# 1. Khái quát về hàm trong Python

## »»» Khái niệm

Hàm là *một khôi lệnh thực hiện một công việc hoàn chỉnh* (module), *được đặt tên* và được gọi thực thi nhiều lần tại nhiều vị trí trong chương trình.

Hàm còn gọi là chương trình con (Subroutine)

Phân loại hàm:

- ✓ *Hàm thư viện: phải import thư viện trước khi sử dụng hàm (from ... import)*
- ✓ *Hàm do người dùng tự định nghĩa.*

# 1. Khái quát về hàm trong Python

## »»» Ví dụ hàm thư viện, hàm tự định nghĩa

Vd: Hàm thư viện

```
from decimal import *
x = 6.6789
y = Decimal(x)
print(type(x))
print(type(y))
```

Vd: Hàm người dùng  
định nghĩa:

```
def Tong(x, y):
    return x + y

a = 8
b = 9
c = Tong(a, b)
print(f"Tong = {c}")
```

# 1. Khái quát về hàm trong Python

## »» Cú pháp hàm

```
def name (parameter list ):  
    block
```

Hàm được định nghĩa với từ khóa `def`, hàm có thể có đối số hoặc không, có thể có kết quả trả về hoặc không.

# 1. Khái quát về hàm trong Python

## »»» Ví dụ định nghĩa hàm

Vd: Hàm in lời chào

```
def sayWelcome():
    print("Welcome To UEL")
```

Vd: Hàm kiểm tra một số có  
phải là số chẵn

```
def isEven(x):
    if x % 2 == 0:
        return True
    return False
```

```
def phepChia(sobichia, sochia):
    return sobichia / sochia
```

# 1. Khái quát về hàm trong Python

## »»» Cách gọi hàm

Khi gọi hàm cần khai báo đầy đủ các đối số nếu có.

Gọi hàm **không** có kết quả trả về

*FunctionName([parameter])* → ***sayWelcome()***

Gọi hàm **có** kết quả trả về

*result = FunctionName([parameter])*

→ *b = isEven(x)*

→ *x = phepChia(9, 2)*

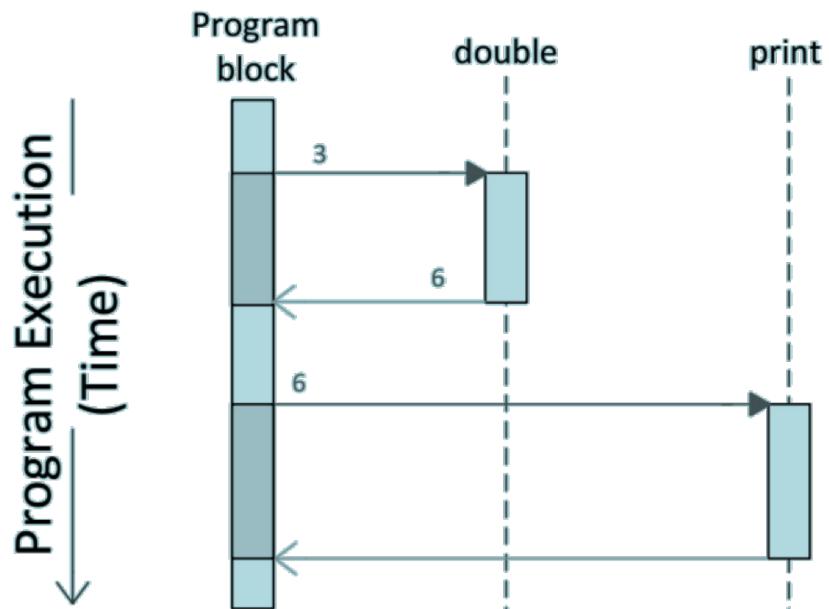
→ hoặc ***phepChia(sochia=2, sobichia=9)***

# 1. Khái quát về hàm trong Python

## »» Cơ chế hoạt động của hàm

Hàm trong Python cũng như trong các ngôn ngữ lập trình khác, đều hoạt động theo cơ chế LIFO (LAST IN FIRST OUT).

```
def double(n):  
    return 2 * n  
x = double(3)  
print(x)
```



# 1. Khái quát về hàm trong Python

## »»» Tạo tài liệu cho hàm

func\_example.py

```
def USLN(x, y):
    """
    Hàm tìm ước số chung lớn nhất của 2 số x, y
    """
    us = 1
    sobe = x if x < y else y
    for i in range(1, sobe + 1):
        if x % i == 0 and y % i == 0:
            us = i
    return us
```

# 1. Khái quát về hàm trong Python

## »»» Tạo tài liệu cho hàm

**func\_example.py**

```
>>> from func_example import *
```

```
>>> help(USLN)
```

```
Help on function USLN in module func_example:
```

```
USLN(x, y)
```

Hàm tìm ước số chung lớn nhất của 2 số x, y

# 1. Khái quát về hàm trong Python

## »»» Biến toàn cục (global variable)

Tất cả các biến khai báo bên trong hàm chỉ có phạm vi ảnh hưởng trong hàm, các biến này gọi là biến **local**. Khi thoát khỏi hàm thì các biến này không thể truy xuất được.

```
x = 8  
def increment():  
    x = 6  
    x = x + 1  
increment()  
print(x) ?
```

```
x = 8  
def increment():  
    global x  
    x = 6  
    x = x + 1  
increment()  
print(x) ?
```

```
x = 8  
def increment():  
    x = x + 1  
increment() ?  
print(x)
```

# 1. Khái quát về hàm trong Python

## »»» Tham số (parameter) mặc định

Ví dụ:

```
def print(self, *args, sep=' ', end='\n', file=None): # known special case of print
    """
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
    """
    pass
```

# 1. Khái quát về hàm trong Python

## »» Tham số (parameter) mặc định

Ví dụ:

```
def lastItem(m, n=0):  
    last = 0  
    for i in range(1, m+n, 1):  
        last = i  
    return last
```

```
x = lastItem(8)  
print("x = {}".format(x)) ? x = 7  
y = lastItem(8, 2)  
print(f'y = {y}') ? y = 9
```

# 1. Khái quát về hàm trong Python

## »»» Lambda expression

**lambda** *parameter list* : *expression*

```
def handle(f, x):  
    return f(x)  
a = handle(lambda x: x % 2 == 0, 9)  
b = handle(lambda x: x % 2 == 0, 8)  
print(f'a = {a}, b = {b}') ?
```

```
def handle(f, x, y):  
    return f(x, y)  
tong = handle(lambda x, y: x + y, 3, 5)  
print(f'Tong = {tong}') ?
```

# 1. Khái quát về hàm trong Python

## »»» Lambda expression

**lambda** *parameter list* : *expression*

```
def handle(f, x):
    return f(x)
def isEven(x):
    return x % 2 == 0
def isOdd(x):
    return x % 2 != 0

r1 = handle(isOdd, 5)
r2 = handle(lambda x: isOdd(x), 5)
r3 = handle(lambda y: isEven(y), 9)
print(f'R = {r1}, {r2}, {r3}') ?
```

## 2. Hàm đệ quy

### »»» Khái niệm

**Đệ quy** là cách thức định nghĩa hàm gọi lại chính nó. Định nghĩa hàm đệ quy cần lưu ý:

1. *Quy luật giải quyết bài toán*
2. *Điều kiện dừng*

Ưu khuyết giữa đệ quy và vòng lặp?

## 2. Hàm đệ quy

### »»» Ví dụ:

- Tính n giai thừa (với n là số nguyên):  $n! = n * (n-1)!$

$$n! = \begin{cases} 1, & \text{if } n = 0 \\ n * (n-1)! & \end{cases}$$

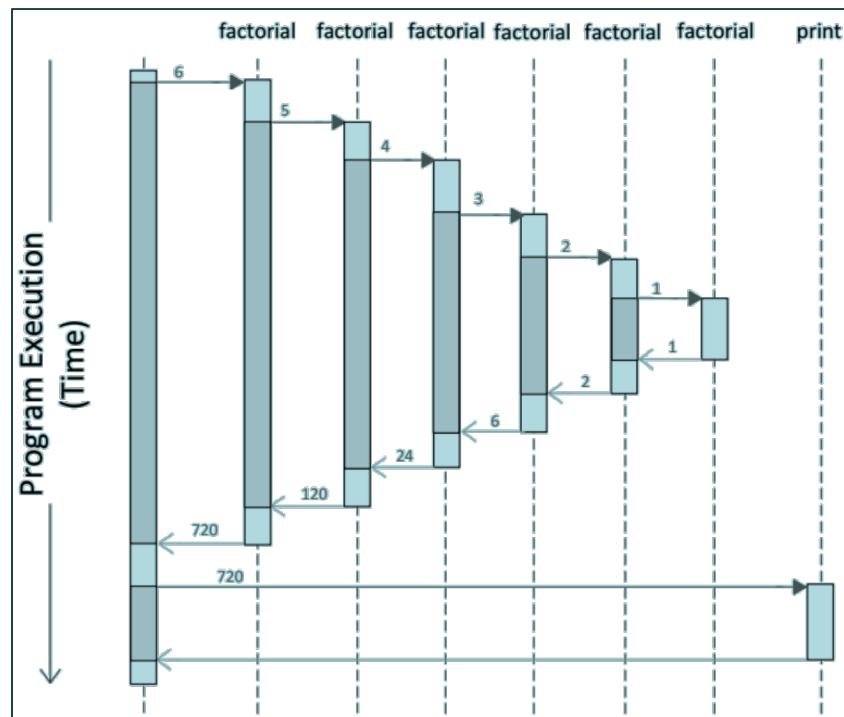
Điều kiện dừng

Quy luật

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

```
result = factorial(6)
print(result)
```

```
def factorial(n):
    return 1 if n == 0 else n * factorial(n - 1)
```



## 2. Hàm đệ quy

### »»» Bài tập:

- Tính số hạng thứ n của dãy Fibonacci:

$$F(1) = 1, F(2) = 1, F(n) = F(n-1) + F(n-2)$$

```
def fibonacci(n):
    if n == 1 or n == 2:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
```

```
def fibonacci(n):
    return 1 if n == 1 or n == 2 else fibonacci(n-1) + fibonacci(n-2)
```

### 3. Một số hàm toán học thông dụng

#### »» Các hàm cơ bản

- `sqrt` → căn bậc 2
- `pow` → lũy thừa
- `log` →  $\log(x) = \log_e x = \ln x$
- `log10` →  $\log_{10} x$
- `exp` →  $e^x$
- `degrees` → đổi radian sang độ
- `radians` → tính radian  $180/\pi * x$
- `fabs` → trị tuyệt đối
- `round` → làm tròn số
- `randrange(start, stop, [step])` → sinh số ngẫu nhiên  $\in [start, stop)$
- `randint(a, b)` → sinh số ngẫu nhiên  $\in [a, b]$

### 3. Một số hàm toán học thông dụng

#### »» Các hàm cơ bản

```
from math import *
print('sqrt(9) = ', sqrt(9))
print('pow(2,3) = ', pow(2, 3))
print('log(4) = ', log(4))
print('log10(100) = ', log10(100))
print('exp(3) = ', exp(3))
print('degrees(0.5235) = ', degrees(0.5235))
print('radians(60) = ', radians(60))
print('fabs(-6) = ', fabs(-6))
```

```
sqrt(9) =  3.0
pow(2,3) =  8.0
log(4) =  1.3862943611198906
log10(100) =  2.0
exp(3) =  20.085536923187668
degrees(0.5235) =  29.994340575098594
radians(60) =  1.0471975511965976
fabs(-6) =  6.0
```

```
print(round(10/3, 2)) → 3.33
```

```
from random import *
print(randrange(1, 10, 2)) → 1 or 3 or 5 or 7 or 9
```

### 3. Một số hàm toán học thông dụng

#### »»» Hàm eval

```
from math import *
x = eval('3*sqrt(16)+pow(2,3)')
print(f'x = {x}')
```

```
def func(f, a, b):
    return f(a, b)
x = eval('func(lambda a, b: a if a > b else b, 6, 8)')
print('x = {}'.format(x))
```

# XỬ LÝ CHUỖI

# NỘI DUNG

1. Cấu trúc chuỗi
2. Các hàm xử lý chuỗi

# 1. Cấu trúc chuỗi

**Chuỗi** là một *tập hợp các ký tự* được đặt trong cặp *nháy đơn* hoặc *nháy đôi*, chuỗi nhiều dòng được đặt trong cặp 3 nháy đơn hoặc đôi.

```
s1 = "Welcome to UEL"  
s2 = 'I Love UEL'  
s3 = "Chào mừng bạn đến với Khóa học Python!"  
s4 = """Chào mừng bạn đến với Khóa học  
Lập trình Python"""
```

```
print(s1)  
print(s2)  
print(s3)  
print(s4)
```

```
Welcome to UEL  
I Love UEL  
Chào mừng bạn đến với Khóa học Python!  
Chào mừng bạn đến với Khóa học  
Lập trình Python
```

Chuỗi trong Python là một *đối tượng*, các thao tác trên chuỗi có thể được thực hiện thông qua việc gọi các phương thức xử lý theo cú pháp:

***object . method name ( parameter list )***

## 2. Hàm xử lý chuỗi

### »»» Upper, Lower, Title

**upper**: chuyển đổi chuỗi v`ề chữ in hoa

```
welcome = "Welcome to UEL"  
print(welcome.upper())
```

WELCOME TO UEL

**lower**: chuyển đổi chuỗi v`ề chữ in thường

```
welcome = "Welcome to UEL"  
print(welcome.lower())
```

welcome to uel

**title**: in hoa ký tự đ`âu mỗi từ

```
welcome = "Welcome to UEL"  
print(welcome.title())
```

Welcome To Uel

Nhóm hàm kiểm tra: **isupper()**, **islower()**, **istitle()**

## 2. Hàm xử lý chuỗi

### »» Capitalize, Swapcase

**capitalize:** in hoa ký tự đầu tiên

```
welcome = "Welcome to UEL"  
print(welcome.capitalize())
```

Welcome to uel

**swapcase:** chuyển đổi hoa → thường, thường → hoa

```
welcome = "Welcome to UEL"  
print(welcome.swapcase())
```

wELCOME TO uel

## 2. Hàm xử lý chuỗi

### »» Căn lề: ljust, rjust, center

```
welcome = "UEL"  
print(welcome.ljust(10))  
print(welcome.ljust(10, "*"))
```

UEL  
UEL\*\*\*\*\*

```
welcome = "UEL"  
print(welcome.rjust(10))  
print(welcome.rjust(10, "*"))
```

UEL  
\*\*\*\*\*UEL

```
welcome = "UEL"  
print(welcome.center(10))  
print(welcome.center(10, "*"))
```

UEL  
\*\*\*UEL\*\*\*

## 2. Hàm xử lý chuỗi

»» Xóa khoảng trắng thừa: strip, lstrip, rstrip

```
welcome = " Welcome to UEL "
print(welcome.strip())
```

Welcome to UEL

```
welcome = " Welcome to UEL "
print(welcome.lstrip())
```

Welcome to UEL

```
welcome = " Welcome to UEL "
print(welcome.rstrip())
```

Welcome to UEL

»» Độ dài chuỗi: len(), \_\_len\_\_()

```
welcome = "Welcome to UEL!"
print(len(welcome)) → 15
print(welcome.__len__()) → 15
```

## 2. Hàm xử lý chuỗi

### »»» **startswith, endswith**

**startswith:** Kiểm tra chuỗi có bắt đầu bằng chuỗi con nào đó hay không. Cú pháp: **startswith(value, start, end)**

**endswith:** Kiểm tra chuỗi có kết thúc bằng chuỗi con nào đó hay không. Cú pháp: **endswith(value, start, end)**

\*Lưu ý: startswith, endswith có phân biệt chữ hoa thường

```
welcome = "Welcome to UEL!"
```

```
print(welcome.startswith("We")) ➔ True
```

```
print(welcome.startswith("We", 3)) ➔ False
```

```
print(welcome.startswith("We", 2, 9)) ➔ False
```

```
print(welcome.endswith("!")) ➔ True
```

```
print(welcome.endswith("L!", -2)) ➔ True
```

## 2. Hàm xử lý chuỗi

### »» TÌM CHUỖI CON: find, rfind

**find**: trả về vị trí đầu tiên tìm thấy chuỗi con, **rfind**: trả về vị trí cuối cùng tìm thấy chuỗi con. Trả về -1 nếu không tìm thấy.

Cú pháp: **find(str, start, end)**, **rfind(str, start, end)**

```
welcome = "No Pain No Gain!"  
print(welcome.find("No")) → 0  
print(welcome.find("No", 2, 6)) → -1  
print(welcome.find("No", 2)) → 8  
print(welcome.rfind("No")) → 8  
print(welcome.rfind("No", -9, -2)) → 8
```

## 2. Hàm xử lý chuỗi

### »»» Tìm chuỗi con: index, rindex

**index, rindex**: tương tự như **find, rfind**. Tuy nhiên, nếu không tìm thấy chuỗi con sẽ trả về exception.

```
welcome = "No Pain No Gain!"  
print(welcome.index("No", 2))  
print(welcome.rindex("No", 5))  
print(welcome.index("Success"))
```

```
8  
8  
Traceback (most recent call last):  
  File "/Volumes/Data/Works/1.UEL/Python/Demo/StringExample/main.py", line 4, in <module>  
    print(welcome.index("Success"))  
ValueError: substring not found
```

## 2. Hàm xử lý chuỗi

### »»» Đếm chuỗi con: count

**count**: đếm số lần xuất hiện của chuỗi con trong chuỗi gốc, không tìm thấy chuỗi con → trả về 0.

Cú pháp: **count(str, start, end)**

```
welcome = "No Pain No Gain!"  
print(welcome.count("No")) → 2  
print(welcome.count("No", 3)) → 1  
print(welcome.count("Success")) → 0
```

## 2. Hàm xử lý chuỗi

### »» Cắt chuỗi: substring

```
welcome = "No Pain No Gain!"
```

```
print(welcome[3:7])      Pain
```

```
print(welcome[:7])      No Pain
```

```
print(welcome[None:7])  No Pain
```

```
print(welcome[3:])       Pain No Gain!
```

```
print(welcome[3:None])  Pain No Gain!
```

```
print(welcome[-5:])     Gain!
```

```
print(welcome[:-9])    No Pain
```

## 2. Hàm xử lý chuỗi

### »» Tách chuỗi: split

```
welcome = "No-Pain-No-Gain!"  
s = welcome.split('-')  
print(s)  
for i in s:  
    print(i)
```

```
[ 'No' , 'Pain' , 'No' , 'Gain! ']  
No  
Pain  
No  
Gain!
```

## 2. Hàm xử lý chuỗi

### »» Nối chuỗi: join

```
welcome = "No-Pain-No-Gain!"  
s = welcome.split('-')  
print(s)  
for i in s:  
    print(i, end=' ')  
print('\n-----')  
s1 = ''  
print("Joined string: ", s1.join(s))
```

```
[ 'No', 'Pain', 'No', 'Gain! ']
```

```
No Pain No Gain!
```

```
-----
```

```
Joined string: No Pain No Gain!
```

## 2. Hàm xử lý chuỗi

### »» Thay thế chuỗi: replace

**replace**: tìm kiếm và thay thế chuỗi con bằng chuỗi mới.

Cú pháp: **replace(oldStr, newStr, max)**

```
welcome = "No Pain No Gain!"  
print(welcome.replace("No", "o"))  
print(welcome.replace("No", "o", 1))
```

```
0 Pain 0 Gain!  
0 Pain No Gain!
```

## 2. Hàm xử lý chuỗi

»» Chuyển đổi chuỗi: maketrans(), translate()

```
welcome = "No Pain No Gain!"  
inputs = "oa"  
outputs = "OA"  
trans = welcome.maketrans(inputs, outputs)  
print(trans)  
print(welcome.translate(trans))
```

{111: 79, 97: 65}

NO PAin NO GAin!

## 2. Hàm xử lý chuỗi

»» Các hàm kiểm tra: **isalnum()**, **isalpha()**, **isdigit()**, **isnumeric()**

print('Coin68'.isalnum())	True
print('Coin68.com'.isalnum())	False
print('Binance'.isalpha())	True
print('68'.isdigit())	True
print('68.5'.isdigit())	False
print('68abc'.isdigit())	False
print("⑩⑬⑮".isdigit())	False
print('86'.isnumeric())	True
print("⑩⑬⑮".isnumeric())	True

# KIỂU DỮ LIỆU TẬP HỢP (COLLECTION)

# NỘI DUNG

1. List
2. Tuple
3. Set
4. Dictionary

# 1. List

## »»» Khái niệm

**List:** là một collection dùng *lưu trữ* các *phân tử* theo *thứ tự*, các *phân tử* có thể thuộc *nhiều kiểu dữ liệu* khác nhau.

Ví dụ:

```
list_1 = []
list_2 = ['Python', 'Golang', 'Ruby']
list_3 = ['Java', 1, 'Swift', 2]
list_4 = [['React native', 'Ionic', 'Flutter'], [1, 2, 3]]
```

# 1. List

## »»» Các thao tác trên List

### Khởi tạo list

```
list_1 = [1, 2, 3] [1, 2, 3]
```

```
list_2 = list([1, 2, 3]) [1, 2, 3]
```

```
list_3 = [i for i in range(4)] [0, 1, 2, 3]
```

```
list_4 = [i for i in range(4) if i % 2 == 0] [0, 2]
```

```
list_5 = [6]*3 [6, 6, 6]
```

# 1. List

## »»» Các thao tác trên List

### Truy xuất phần tử của List

```
print('1. Truy xuất phần tử của List theo chỉ mục')
my_list = [1, 2, 3, 4]
print(my_list[2])      3
print('2. Duyệt list theo tập hợp')
for i in my_list:
    print(i, end=' ')  1 2 3 4
print('\n3. Duyệt list theo chỉ mục')
for i in range(len(my_list)):
    print(my_list[i], end=' ') 1 2 3 4
```

# 1. List

## »»» Các thao tác trên List

### Truy xuất phần tử của List

my_list = ['Python', 'Perl', 'Ruby', 'Golang', 'C#', 'Java', 'Swift']	
print(my_list[:2])	['Python', 'Perl']
print(my_list[None:1])	['Python']
print(my_list[3:6])	['Golang', 'C#', 'Java']
print(my_list[5:])	['Java', 'Swift']
print(my_list[4:None])	['C#', 'Java', 'Swift']
print(my_list[-2:])	['Java', 'Swift']

# 1. List

## »»» Các thao tác trên List

### Cập nhật giá trị phần tử

```
list_1 = [1, 2, 3, 4]
list_2 = ["Python", "Angular", "Ruby"]
print(list_1)  [1, 2, 3, 4]
print(list_2)  ['Python', 'Angular', 'Ruby']
list_1[0] = 9
list_2[1] = "Golang"
print(list_1)  [9, 2, 3, 4]
print(list_2)  ['Python', 'Golang', 'Ruby']
```

# 1. List

## »»» Các thao tác trên List

### Cập nhật giá trị phần tử

```
list_1 = [1, 2, 3, 4]
```

```
list_2 = list_1
```

```
print(list_1)
```

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

```
print(list_2)
```

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

```
list_1[0] = 9
```

```
list_1[1] = 8
```

```
print(list_1)
```

```
[9, 8, 3, 4]
```

```
print(list_2)
```

```
[9, 8, 3, 4] ?
```

list\_1 →

1	2	3	4
0	1	2	3

list\_2 →

1	2	3	4
0	1	2	3

# 1. List

## »» Các thao tác trên List

Cập nhật giá trị phần tử

```
list_1 = [1, 2, 3, 4]
```

```
list_2 = list(list_1)
```

```
print(list_1)
```

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

```
print(list_2)
```

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

```
list_1[0] = 9
```

```
list_1[1] = 8
```

```
print(list_1)
```

```
[9, 8, 3, 4]
```

```
print(list_2)
```

```
[1, 2, 3, 4] ?
```

list\_1 →

1	2	3	4
0	1	2	3

list\_2 →

1	2	3	4
0	1	2	3

# 1. List

## »»» Các thao tác trên List

Cập nhật giá trị phần tử

```
list_1 = [1, 2, 3, 4]
```

```
list_2 = list_1.copy()
```

```
print(list_1)
```

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

```
print(list_2)
```

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

```
list_1[0] = 9
```

```
list_1[1] = 8
```

```
print(list_1)
```

```
[9, 8, 3, 4]
```

```
print(list_2)
```

```
[1, 2, 3, 4] ?
```

list\_1 →

1	2	3	4
0	1	2	3

list\_2 →

1	2	3	4
0	1	2	3

# 1. List

## »»» Các thao tác trên List

Thêm phần tử vào List:

✓ **insert(index, value)**

```
my_list = ["Python", "Angular", "Ruby"]
my_list.insert(1, "Perl")
print(my_list) ['Python', 'Perl', 'Angular', 'Ruby']
```

✓ **append(value)**

```
my_list = ["Python", "Ruby"]
my_list.append(["Perl", "Golang"])
print(my_list) ['Python', 'Ruby', ['Perl', 'Golang']]
```

# 1. List

## »»» Các thao tác trên List

Thêm phần tử vào List:

✓ **extend(value)**

```
my_list = ["Python", "Ruby"]
my_list.extend(["Perl", "Golang"])
print(my_list) ['Python', 'Ruby', 'Perl', 'Golang']
```

✓ **\_\_add\_\_(value)**

```
my_list = ["Python", "Angular", "Ruby"]
print(my_list.__add__([["Perl"])))
['Python', 'Angular', 'Ruby', 'Perl']
```

# 1. List

## »» Các thao tác trên List

Xóa phần tử khỏi List:

✓ `pop(index)`

```
my_list = ["Python", "Angular", "Ruby", "Golang"]
my_list.pop()
print(my_list)      ['Python', 'Angular', 'Ruby']
my_list.pop(1)
print(my_list)      ['Python', 'Ruby']
```

✓ `remove(value)`: xóa phần tử tìm thấy đầu tiên trong List

```
my_list = ["Python", "Angular", "Ruby", "Angular", "Golang"]
my_list.remove('Angular') # del my_list[1]
print(my_list)      ['Python', 'Ruby', 'Angular', 'Golang']
```

# 1. List

## »» Các thao tác trên List

### Đếm phần tử trên List

```
my_list = [3, 9, 7, 9, 5, 9, 8]
print(my_list.count(9)) 3
```

### Kiểm tra sự tồn tại của phần tử trên List

```
my_list = [3, 9, 7, 9, 5, 9, 8]
print(my_list.__contains__(2)) False
print(my_list.__contains__(7)) True
```

```
my_list = [3, 9, 7]
print(3 in my_list) True
print(8 in my_list) False
```

# 1. List

## »»» Các thao tác trên List

### Sắp xếp List

```
my_list = [3, 2, 7, 5, 9, 8]
my_list.sort()
print(my_list)      [2, 3, 5, 7, 8, 9]
my_list_2 = [4, 2, 5, 7, 3, 6]
sorted_list = sorted(my_list_2)
print(sorted_list)  [2, 3, 4, 5, 6, 7]
```

# 1. List

## »»» Các thao tác trên List

### Đảo ngược List

```
my_list = [3, 2, 7, 5, 9, 8]
my_list.sort()
my_list.reverse()
print(my_list)      [9, 8, 7, 5, 3, 2]
my_list_2 = [4, 2, 5, 7, 3, 6]
my_list_2.sort(reverse=True)
print(my_list_2)  [7, 6, 5, 4, 3, 2]
```

```
my_list = ['Python', 'Perl', 'Ruby', 'Golang', 'C#', 'Java', 'Swift']
print(my_list[::-1])
['Swift', 'Java', 'C#', 'Golang', 'Ruby', 'Perl', 'Python']
```

# 1. List

## »»» Các thao tác trên List

### Chuyển đổi string ↔ list

```
s1 = 'madam'  
print(f"s1: {s1}") s1: madam  
s_list = list(s1)  
print(s_list) ['m', 'a', 'd', 'a', 'm']  
s2 = ''.join(s_list)  
print("s2: {}".format(s2)) s2: madam
```

# 1. List

## »»» Các thao tác trên List

### Tạo list đa chiều (ma trận)

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
print(matrix) [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
print(matrix[0]) [1, 2, 3]  
print(matrix[1]) [4, 5, 6]  
print(matrix[2]) [7, 8, 9]
```

```
row = 3  
col = 4  
matrix = [[0]*col]*row  
print(matrix)  
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

## 2. Tuple

### »» Khái niệm

**Tuple:** là một collection tương tự như List. Tuy nhiên, *giá trị của các phần tử không được phép thay đổi.*

Ví dụ:

```
tup_1 = ()  
tup_2 = ('Python', 'Golang', 'Ruby')  
tup_3 = ('Java', 1, 'Swift', 2)  
tup_4 = ([ 'React native', 'Ionic', 'Flutter'], [1, 2, 3])
```

### »» Các thao tác trên Tuple

Tương tự như trên List. Lưu ý,  
*không thể cập nhật giá trị hay  
xóa phần tử.*

~~tup\_2[0] = 'C#'  
del tup\_2[0]~~

### 3. Set

#### »»» Khái niệm

**Set:** là một collection không có thứ tự, không có chỉ mục, loại bỏ các giá trị trùng lặp. Các phần tử có thể là số, chuỗi, tuple nhưng không thể chứa list.

Ví dụ:

```
set_1 = {1, 6, 2, 8, 3}
print(set_1)  {1, 2, 3, 6, 8}
set_2 = {1, 6, 2, 6, 8, 2, 8}
print(set_2)  {8, 1, 2, 6}
set_3 = {1, 5, 'Python', 2, 9, (1, "Golang")}
print(set_3)
{1, 2, 5, 9, (1, 'Golang'), 'Python'}
```

### 3. Set

#### »» Các thao tác trên Set

##### Thêm phần tử

```
set_1 = {6, 5, 8, 3}  
set_1.add(4)  
print(set_1)  {3, 4, 5, 6, 8}
```

##### Xóa phần tử

```
set_1 = {6, 5, 8, 3, 7, 19}  
set_1.pop()  
print(set_1)  {19, 5, 6, 7, 8}  
set_1.discard(8)  
print(set_1)  {19, 5, 6, 7}  
set_1.remove(5)  
print(set_1)  {19, 6, 7}
```

### 3. Set

#### »» Các thao tác trên Set

Toán tử: -, |, &, ^

```
set_1 = {3, 9, 7}
```

```
set_2 = {6, 3, 8}
```

```
print(set_1 - set_2) {9, 7}
```

```
print(set_1 | set_2) {3, 6, 7, 8, 9}
```

```
print(set_1 & set_2) {3}
```

```
print(set_1 ^ set_2) {6, 7, 8, 9}
```

# 4. Dictionary

## »» Khái niệm

**Dictionary:** là một collection không có thứ tự, mỗi phần tử được định nghĩa bởi cặp khóa và giá trị (*key: value*).

Ví dụ:

```
dict1 = {}  
print(dict1)  {}  
  
dict2 = {1: 'Python', 2: 'Ruby'}  
print(dict2)  {1: 'Python', 2: 'Ruby'}  
  
dict3 = {'name': 'Huynh Giao', 'age': 20}  
print(dict3)  {'name': 'Huynh Giao', 'age': 20}  
  
dict4 = dict({'id': 1, 'productName': 'Iphone'})  
print(dict4)  {'id': 1, 'productName': 'Iphone'}  
  
dict5 = dict([(1, 'Python'), (2, 'Ruby')])  
print(dict5)  {1: 'Python', 2: 'Ruby'}
```

## 4. Dictionary

### »» Các thao tác trên Dictionary

#### Khởi tạo dictionary với Comprehension

```
d1 = {x: x**2 for x in range(5)}  
print(d1)  {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}  
  
d2 = {y: y+1 for y in range(10) if y % 2 == 0}  
print(d2)  {0: 1, 2: 3, 4: 5, 6: 7, 8: 9}
```

#### Duyệt các phần tử

```
d = {1: 'Python', 'pythonApp': 'Game, Web',  
     2: "Java", "javaApp": "Web, Mobile"}
```

```
for i in d:
```

```
    print(d[i])
```

```
Python  
Game, Web  
Java  
Web, Mobile
```

## 4. Dictionary

### »» Các thao tác trên Dictionary

#### Truy xuất phàn tử

```
dict1 = {1: 'Python', 2: 'Ruby', 3: 'Perl', 4: 'Golang'}  
print(dict1[1]) Python  
print(dict1[3]) Perl
```

#### Cập nhật giá trị phàn tử

```
dict1 = {1: 'Python', 2: 'Ruby', 3: 'Perl', 4: 'Golang'}  
dict1[2] = 'Java'  
dict1[4] = 'C#'  
print(dict1)  
{1: 'Python', 2: 'Java', 3: 'Perl', 4: 'C#'}
```

# 4. Dictionary

## »»» Các thao tác trên Dictionary

### Thêm phần tử

```
dict1 = {1: 'Python', 2: 'Ruby', 3: 'Perl'}  
dict1[5] = 'Java'  
print(dict1)  
{1: 'Python', 2: 'Ruby', 3: 'Perl', 5: 'Java'}
```

## 4. Dictionary

### »» Các thao tác trên Dictionary

#### Xóa phần tử

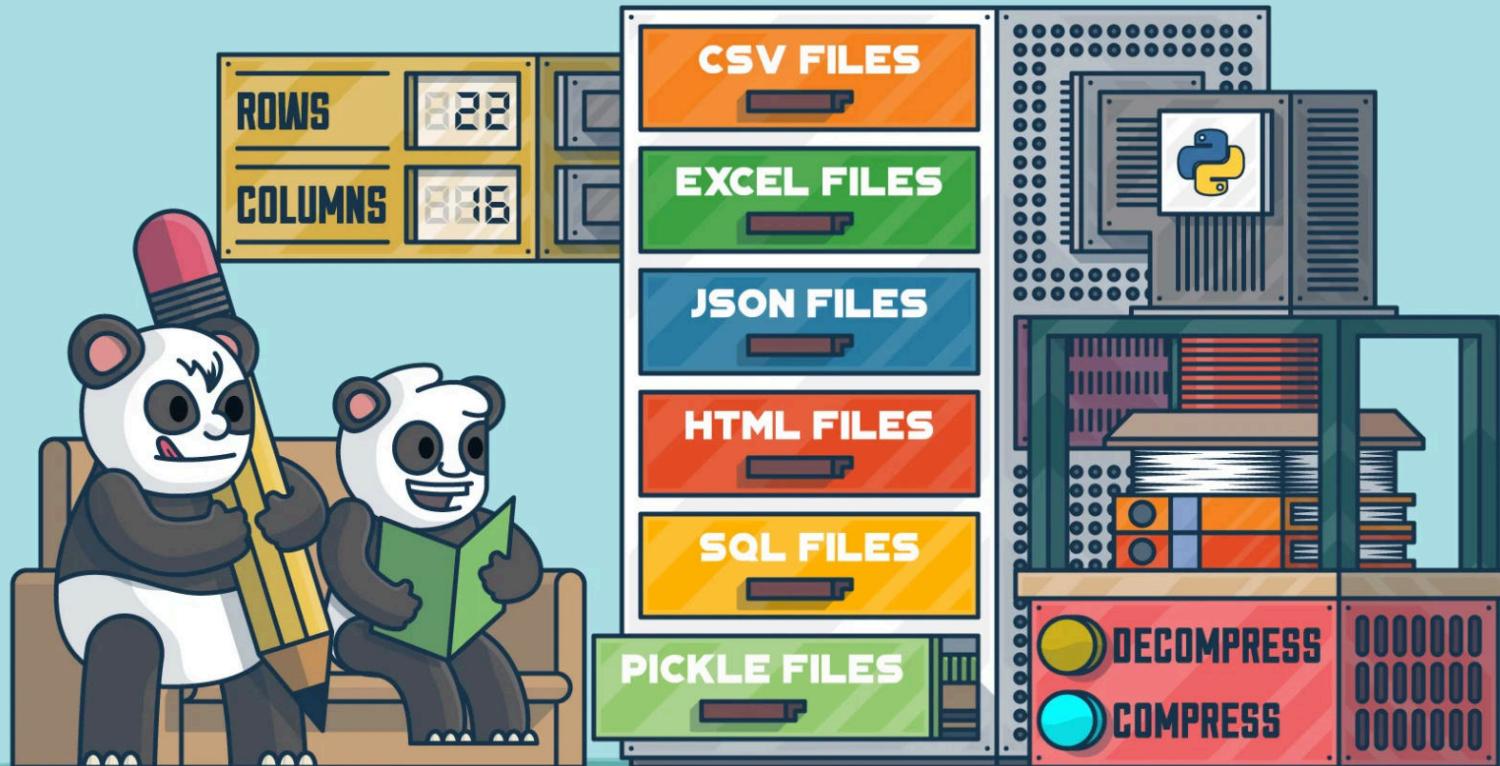
```
dict1 = {1: 'Python', 2: 'Ruby', 3: 'Perl', 4: 'Golang'}
print(dict1.pop(3)) Perl
print(dict1) {1: 'Python', 2: 'Ruby', 4: 'Golang'}
del dict1[2]
print(dict1) {1: 'Python', 4: 'Golang'}
print(dict1.popitem()) (4, 'Golang')
print(dict1) {1: 'Python'}
dict1.clear()
print(dict1) {}
# del dict1
# print(dict1) --> Error
```

# XỬ LÝ DỮ LIỆU TẬP TIN

# NỘI DUNG

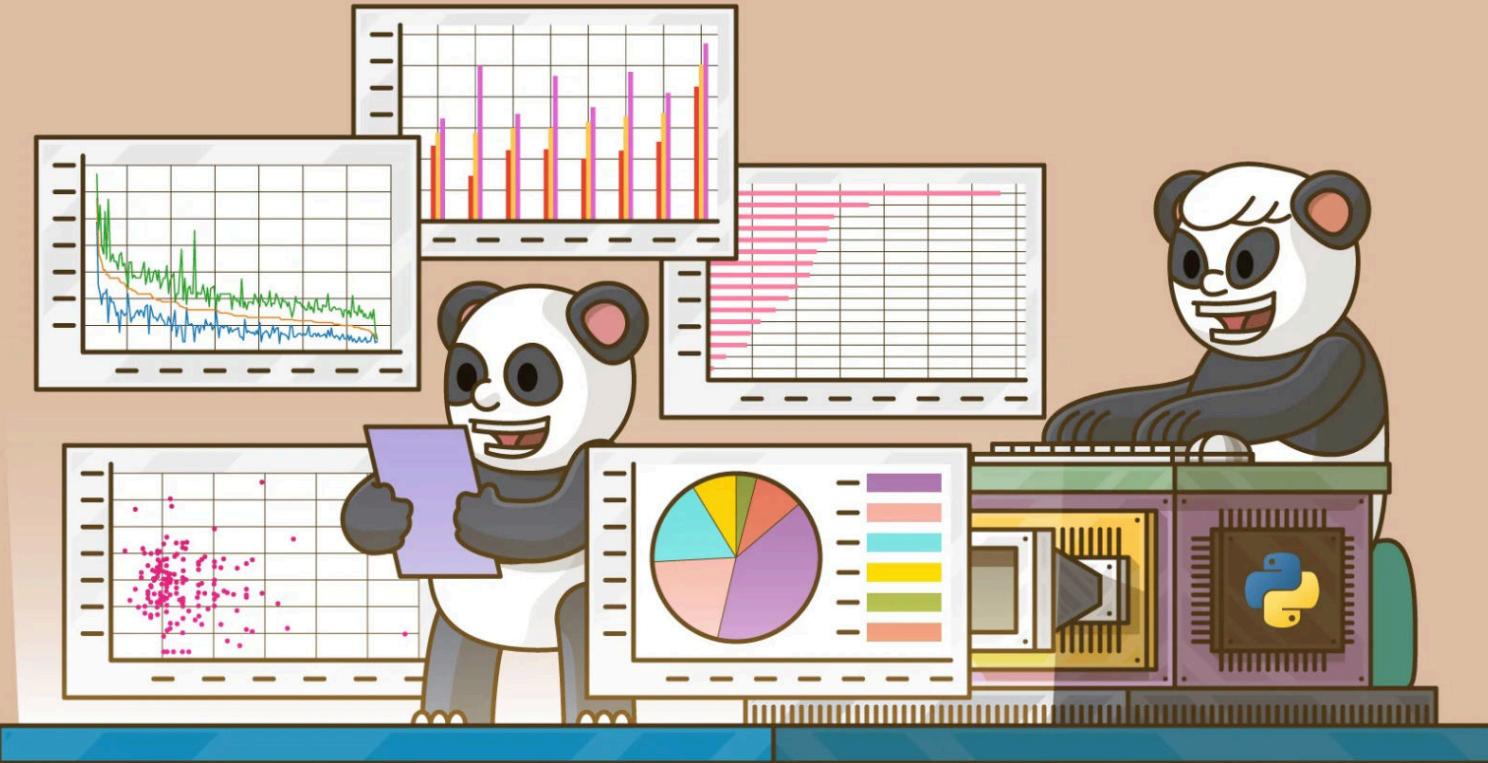
1. Tập tin dữ liệu
2. Xử lý tập tin (file)

# 1. Tập tin dữ liệu



Real Python

# 1. Tập tin dữ liệu



Real Python

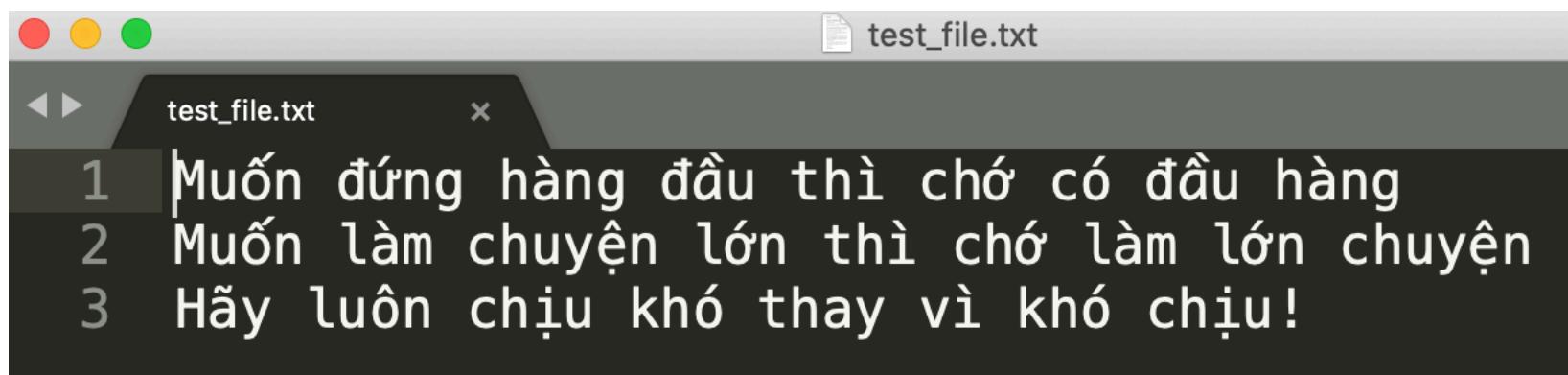
## 2. Xử lý tập tin (file)

### »» Text file (.txt)

#### Ghi dữ liệu

w (write): ghi đè dữ liệu  
a (append): ghi nối tiếp dữ liệu  
...

```
f = open('test_file.txt', 'w', encoding='utf-8')
f.write("Muốn đứng hàng đầu thì chờ có đầu hàng\n")
f.write("Muốn làm chuyện lớn thì chờ làm lớn chuyện\n")
f.write("Hãy luôn chịu khó thay vì khó chịu!")
f.close()
```



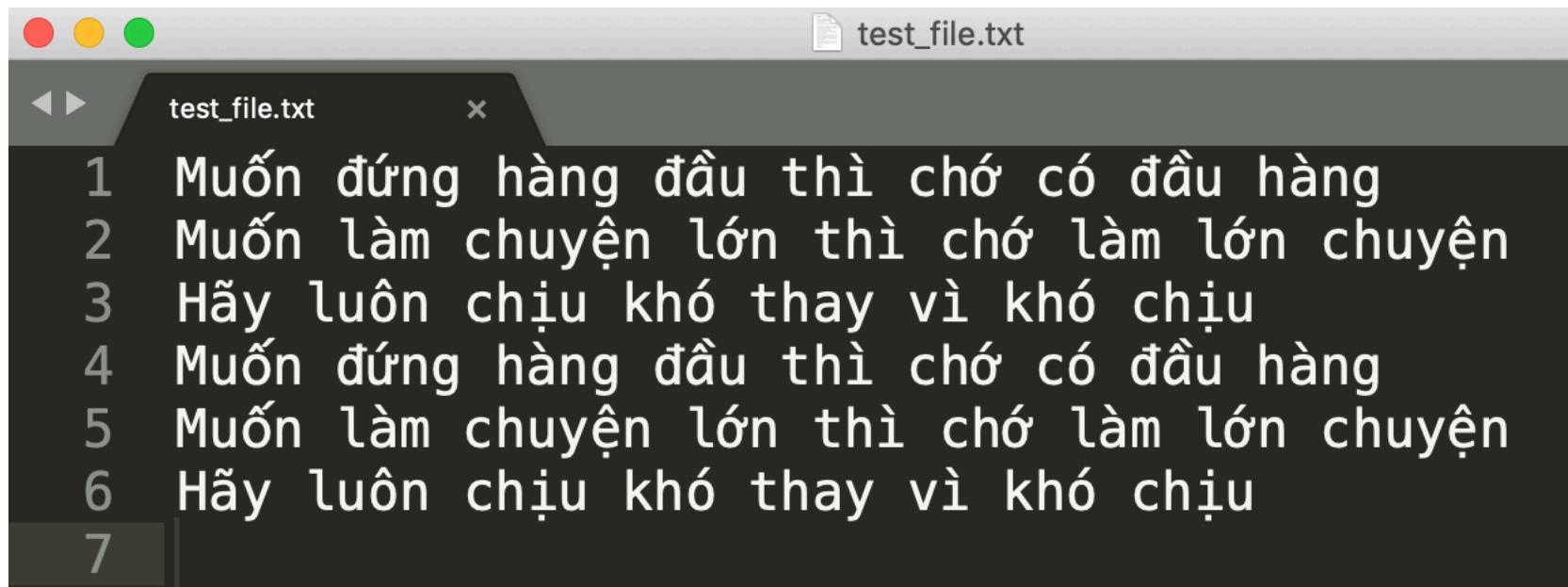
## 2. Xử lý tập tin (file)

### »» Text file (.txt)

#### Ghi dữ liệu

w (write): ghi đè dữ liệu  
a (append): ghi nối tiếp dữ liệu  
...

```
with open('test_file.txt', 'a', encoding="utf-8") as f:  
    f.write("Muốn đứng hàng đầu thì chờ có đầu hàng\n")  
    f.write("Muốn làm chuyện lớn thì chờ làm lớn chuyện\n")  
    f.write("Hãy luôn chịu khó thay vì khó chịu\n")
```



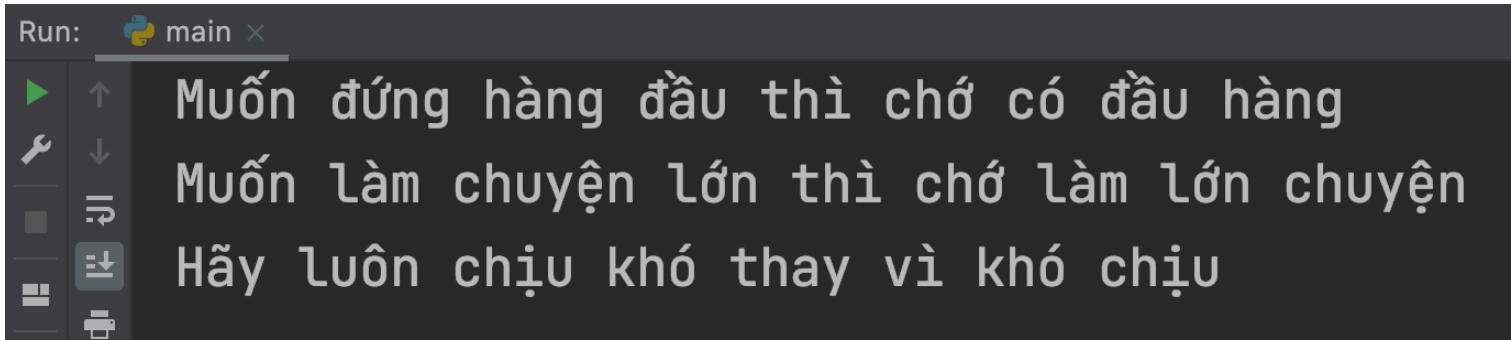
## 2. Xử lý tập tin (file)

### »» Text file (.txt)

#### Đọc dữ liệu

```
f = open('test_file.txt', 'r', encoding="utf-8")
for line in f:
    print(line.strip())
f.close()
```

```
with open('test_file.txt', 'r', encoding="utf-8") as f:
    print(f.read())
```



The screenshot shows a code editor interface with a dark theme. In the top left, there's a 'Run' button and a tab labeled 'main'. The main area contains the following Python code:

```
with open('test_file.txt', 'r', encoding="utf-8") as f:
    print(f.read())
```

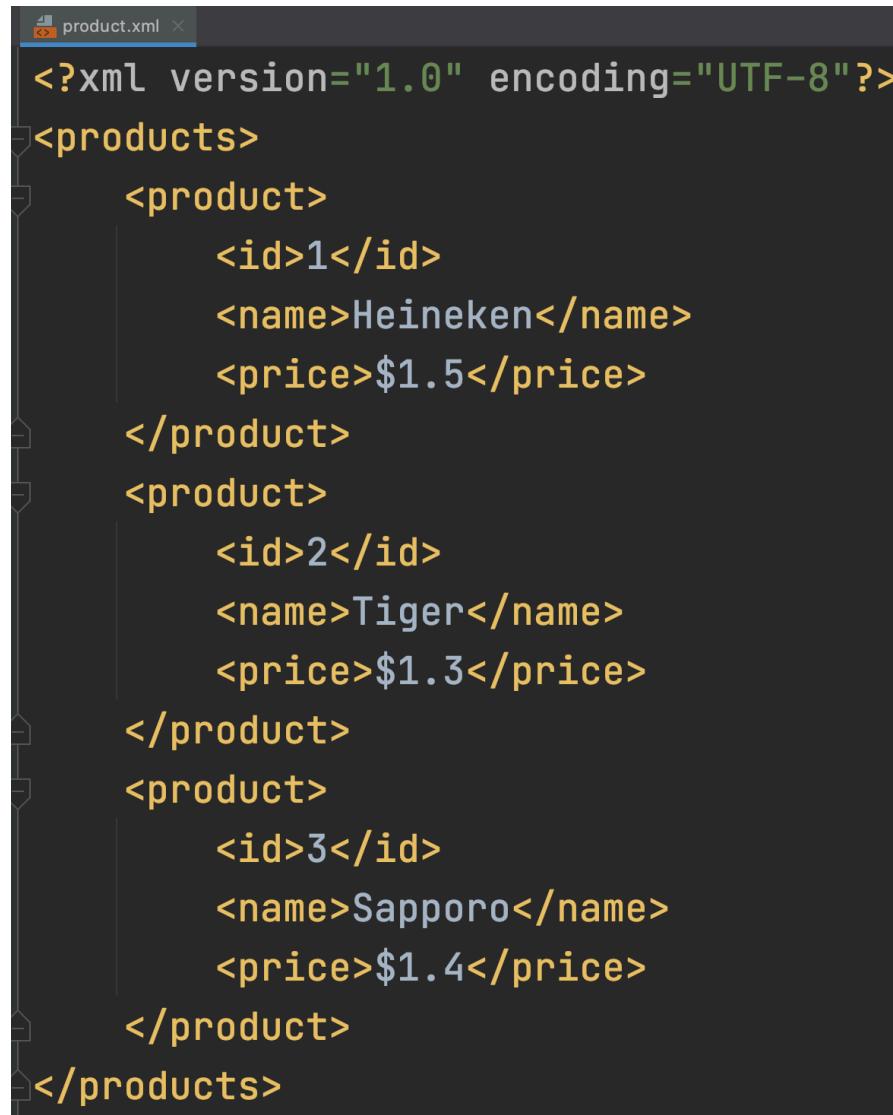
When run, this code prints the following three lines of Vietnamese text to the console:

Muốn đứng hàng đầu thì chờ có đầu hàng  
Muốn làm chuyện lớn thì chờ làm lớn chuyện  
Hãy luôn chịu khó thay vì khó chịu

## 2. Xử lý tập tin (file)

### » XML file (.xml)

Vd: cấu trúc  
File XML



```
<?xml version="1.0" encoding="UTF-8"?>
<products>
    <product>
        <id>1</id>
        <name>Heineken</name>
        <price>$1.5</price>
    </product>
    <product>
        <id>2</id>
        <name>Tiger</name>
        <price>$1.3</price>
    </product>
    <product>
        <id>3</id>
        <name>Sapporo</name>
        <price>$1.4</price>
    </product>
</products>
```

## 2. Xử lý tập tin (file)

### »» XML file (.xml)

```
from xml.dom.minidom import parse
```

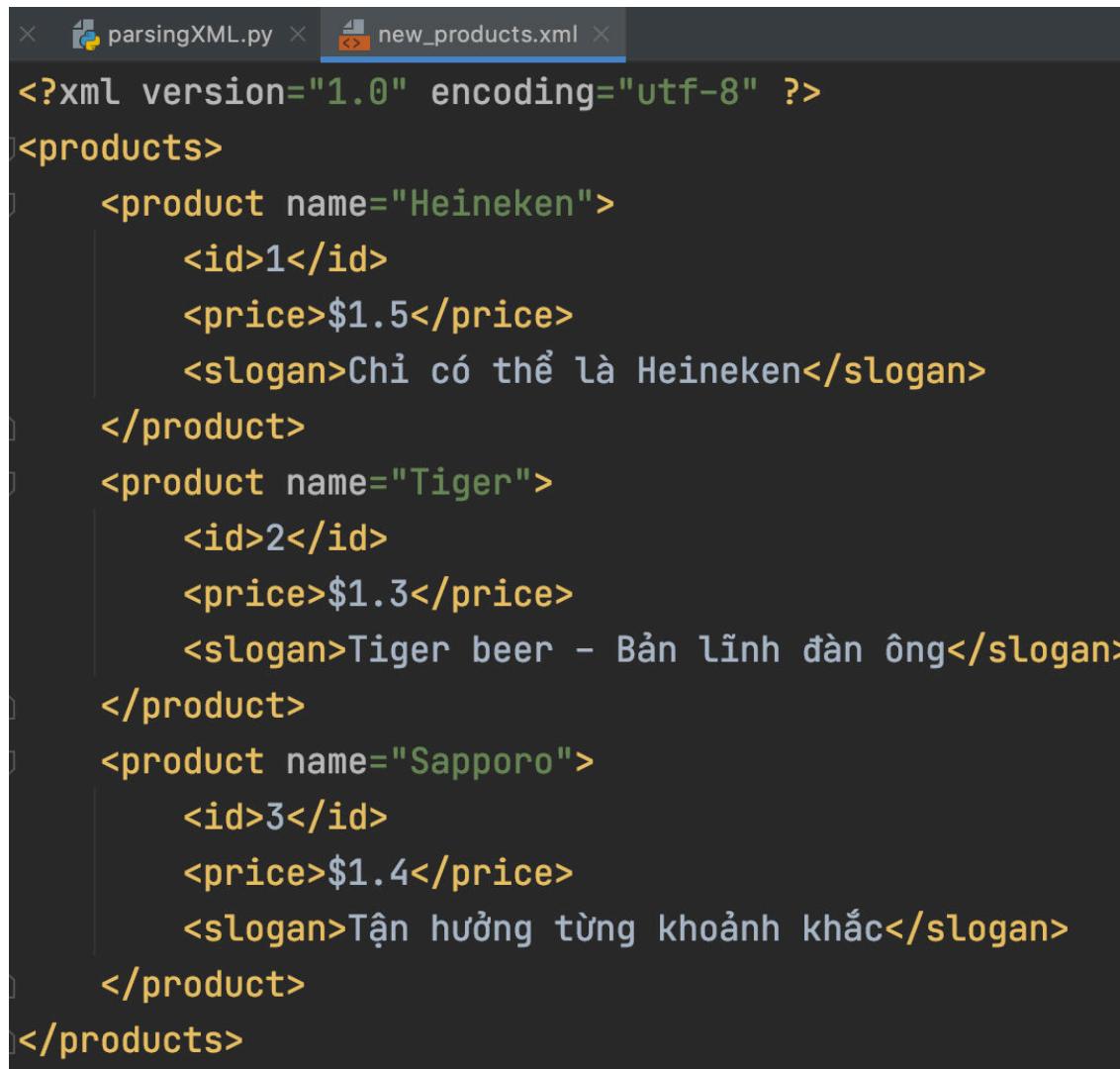
```
DOMTree = parse("product.xml")
elements = DOMTree.documentElement
```

```
products = elements.getElementsByTagName("product")
for p in products:
    pro_id = (p.getElementsByTagName("id")[0]).childNodes[0].data
    pro_name = (p.getElementsByTagName("name")[0]).childNodes[0].data
    pro_price = (p.getElementsByTagName("price")[0]).childNodes[0].data
    print(pro_id + " -> " + pro_name + " -> " + pro_price)
```

```
1 -> Heineken -> $1.5
2 -> Tiger -> $1.3
3 -> Sapporo -> $1.4
```

## 2. Xử lý tập tin (file)

### »»» XML file (.xml)



```
<?xml version="1.0" encoding="utf-8" ?>
<products>
    <product name="Heineken">
        <id>1</id>
        <price>$1.5</price>
        <slogan>Chỉ có thể là Heineken</slogan>
    </product>
    <product name="Tiger">
        <id>2</id>
        <price>$1.3</price>
        <slogan>Tiger beer - Bản lĩnh đàn ông</slogan>
    </product>
    <product name="Sapporo">
        <id>3</id>
        <price>$1.4</price>
        <slogan>Tận hưởng từng khoảnh khắc</slogan>
    </product>
</products>
```

## 2. Xử lý tập tin (file)

### » XML file (.xml)

```
import xml.etree.ElementTree as ET
tree = ET.parse("new_products.xml")
root = tree.getroot()
print("root_tag: ", root.tag)
print("root_attribute: ", root.attrib)
for child in root:
    print(child.tag, child.attrib)
print(root[1][2].text)
for p in root.iter('product'):
    print(p.attrib)
for p in root.findall('product'):
    p_id = p.find('id').text
    p_name = p.get('name')
    p_price = p.find('price').text
    p_slogan = p.find('slogan').text
    print(p_id + " - " + p_name + " - " + p_price + " - " + p_slogan + "")
```

```
root_tag: products
root_attribute: {}
product {'name': 'Heineken'}
product {'name': 'Tiger'}
product {'name': 'Sapporo'}
Tiger beer - Bán lĩnh đòn ông
{'name': 'Heineken'}
{'name': 'Tiger'}
{'name': 'Sapporo'}
1 - Heineken - $1.5 - 'Chỉ có thể là Heineken'
2 - Tiger - $1.3 - 'Tiger beer - Bán lĩnh đòn ông'
3 - Sapporo - $1.4 - 'Tận hưởng từng khoảnh khắc'
```

## 2. Xử lý tập tin (file)

### »» XML file (.xml)

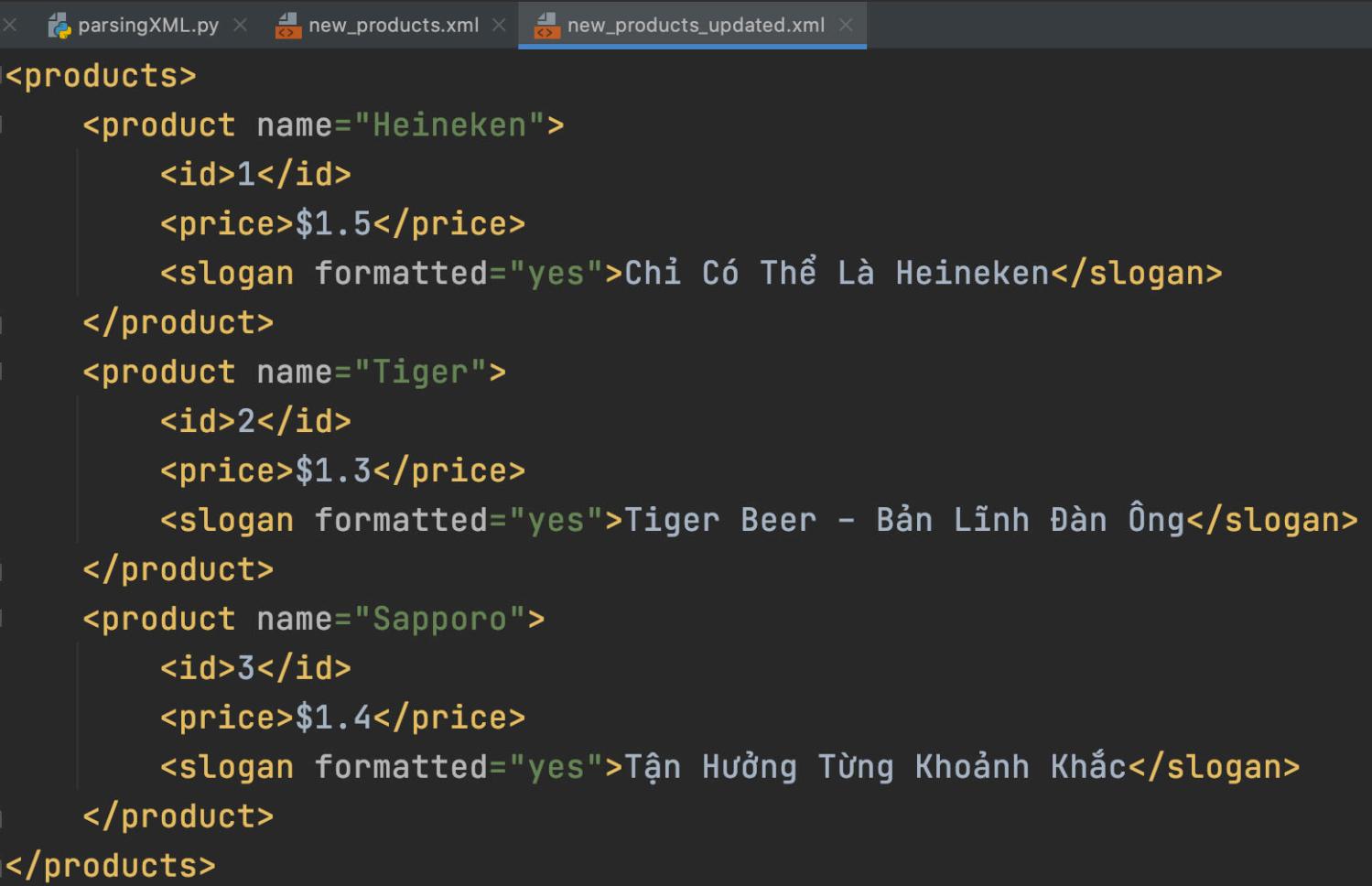
#### Modify XML file

```
import xml.etree.ElementTree as ET
tree = ET.parse("new_products.xml")
root = tree.getroot()
for slogan in root.iter('slogan'):
    formatted_slogan = slogan.text.title()
    slogan.text = formatted_slogan
    slogan.set('formatted', 'yes')
tree.write('new_products_updated.xml', encoding='utf-8')
```

## 2. Xử lý tập tin (file)

### »»» XML file (.xml)

#### Modify XML file



The screenshot shows a code editor with three tabs open:

- parsingXML.py
- new\_products.xml
- new\_products\_updated.xml (selected tab)

The content of new\_products\_updated.xml is as follows:

```
<products>
    <product name="Heineken">
        <id>1</id>
        <price>$1.5</price>
        <slogan formatted="yes">Chỉ Có Thể Là Heineken</slogan>
    </product>
    <product name="Tiger">
        <id>2</id>
        <price>$1.3</price>
        <slogan formatted="yes">Tiger Beer - Bản Linh Đàm Ông</slogan>
    </product>
    <product name="Sapporo">
        <id>3</id>
        <price>$1.4</price>
        <slogan formatted="yes">Tận Hưởng Từng Khoảnh Khắc</slogan>
    </product>
</products>
```

## 2. Xử lý tập tin (file)

### »»» Json file (.json)

```
{  
    "name": "Heineken",  
    "price": "$1.5",  
    "slogan": "....."  
}
```

```
import json  
  
data = {'products': []}  
data['products'].append({  
    'name': 'Heineken',  
    'price': '$1.5',  
    'slogan': 'Chỉ Có Thể Là Heineken'  
})  
data['products'].append({  
    'name': 'Tiger',  
    'price': '$1.3',  
    'slogan': 'Tiger Beer - Bản Lĩnh Đàm Ông'  
})  
data['products'].append({  
    'name': 'Sapporo',  
    'price': '$1.4',  
    'slogan': 'Tận Hưởng Từng Khoảnh Khắc'  
})  
  
with open('data.txt', 'w', encoding='utf8') as outfile:  
    json.dump(data, outfile, ensure_ascii=False)
```

## 2. Xử lý tập tin (file)

### »»» Json file (.json)

```
import json

with open('data.txt', encoding='utf8') as
    json_file:
        data = json.load(json_file)
        for p in data['products']:
            print('Name: ' + p['name'])
            print('Price: ' + p['price'])
            print('Slogan: ' + p['slogan'])
            print()
```

Name: Heineken

Price: \$1.5

Slogan: Chỉ Có Thể Là Heineken

Name: Tiger

Price: \$1.3

Slogan: Tiger Beer - Bản Lĩnh Đàn Ông

Name: Sapporo

Price: \$1.4

Slogan: Tận Hưởng Từng Khoảnh Khắc

## 2. Xử lý tập tin (file)

### »» Json file (.json)

```
import json
jsonObject = {
    'name': 'Heineken',
    'price': '$1.5',
    'slogan': 'Chỉ Có Thể Là Heineken'
}
jsonString = json.dumps(jsonObject)
print(jsonString)
```

```
{"name": "Heineken", "price": "$1.5", "slogan": "Ch\u01ec9 C\u000f3 Th\u01ec3 L\u000e0 Heineken"}
```

## 2. Xử lý tập tin (file)

### »»» Json file (.json)

```
import json
jsonObject = {
    'name': 'Heineken',
    'price': '$1.5',
    'slogan': 'Chỉ Có Thể Là Heineken'
}
jsonString = json.dumps(jsonObject, ensure_ascii=False)
print(jsonString)
```

```
{"name": "Heineken", "price": "$1.5", "slogan": "Chỉ Có Thể Là Heineken"}
```

## 2. Xử lý tập tin (file)

### »»» Json file (.json)

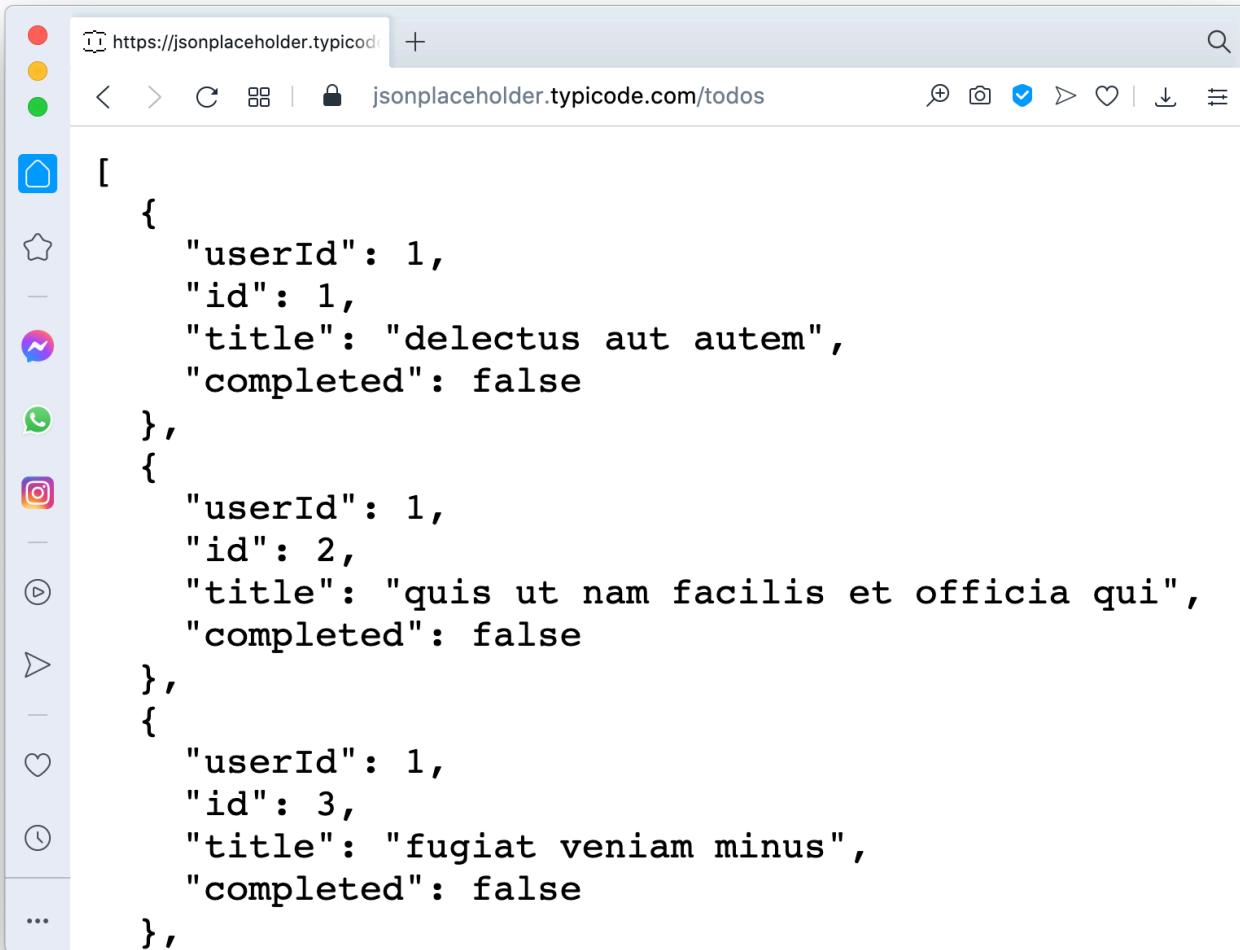
```
import json
jsonObject = {
    'name': 'Heineken',
    'price': '$1.5',
    'slogan': 'Chỉ Có Thể Là Heineken'
}
jsonString = json.dumps(jsonObject, ensure_ascii=False, indent=3)
print(jsonString)
```

```
{
    "name": "Heineken",
    "price": "$1.5",
    "slogan": "Chỉ Có Thể Là Heineken"
}
```

## 2. Xử lý tập tin (file)

### »»» Json file (.json)

#### Đọc dữ liệu Json trả về từ API



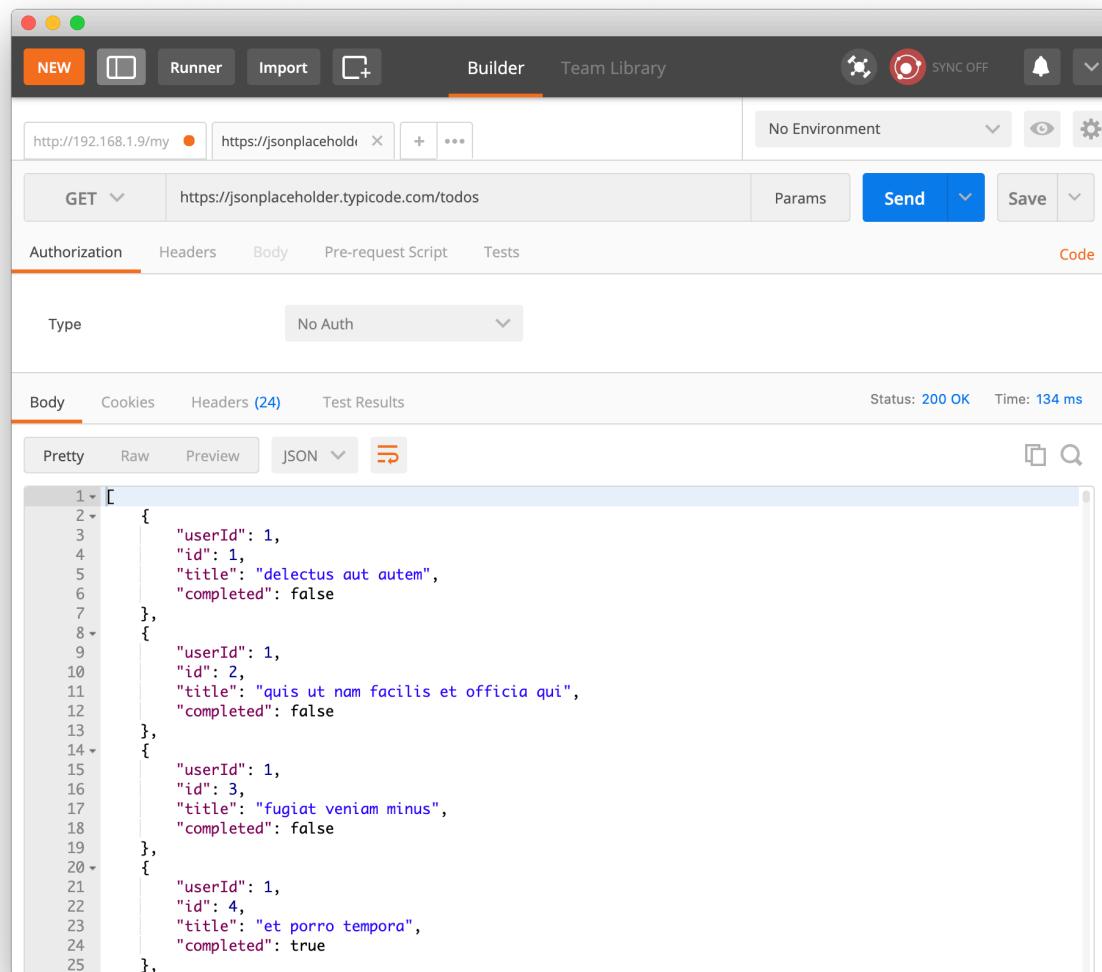
The screenshot shows a web browser window with the URL <https://jsonplaceholder.typicode.com/todos>. The page displays a JSON array of three todo items. Each item is an object with properties: userId, id, title, and completed.

```
[{"userId": 1, "id": 1, "title": "delectus aut autem", "completed": false}, {"userId": 1, "id": 2, "title": "quis ut nam facilis et officia qui", "completed": false}, {"userId": 1, "id": 3, "title": "fugiat veniam minus", "completed": false}]
```

## 2. Xử lý tập tin (file)

### »»» Json file (.json)

#### Đọc dữ liệu Json trả về từ API



The screenshot shows the Postman application interface. At the top, there are tabs for NEW, Runner, Import, and Builder, with 'Builder' selected. Below the tabs, the URL is set to 'https://jsonplaceholder.typicode.com/todos'. The method is set to 'GET'. On the right side, there are buttons for Params, Send, Save, and Code. Under the Authorization tab, it says 'No Auth'. In the Body section, the response is displayed as a JSON array:

```
[{"userId": 1, "id": 1, "title": "delectus aut autem", "completed": false}, {"userId": 1, "id": 2, "title": "quis ut nam facilis et officia qui", "completed": false}, {"userId": 1, "id": 3, "title": "fugiat veniam minus", "completed": false}, {"userId": 1, "id": 4, "title": "et porro tempora", "completed": true}]
```

## 2. Xử lý tập tin (file)

### »»» Json file (.json)

#### Đọc dữ liệu Json trả về từ API

```
import json
import requests

response = requests.get("https://jsonplaceholder.typicode.com/todos")
# todos = json.loads(response.text)
todos = response.json()
print(todos[0])
for t in todos[:2]:
    jsonObj = json.dumps(t, ensure_ascii=False, indent=3)
    print(jsonObj)
```

## 2. Xử lý tập tin (file)

### »»» Json file (.json)

#### Đọc dữ liệu Json trả về từ API

```
{'userId': 1, 'id': 1, 'title': 'delectus aut autem', 'completed': False}  
{  
    "userId": 1,  
    "id": 1,  
    "title": "delectus aut autem",  
    "completed": false  
}  
{  
    "userId": 1,  
    "id": 2,  
    "title": "quis ut nam facilis et officia qui",  
    "completed": false  
}
```

## 2. Xử lý tập tin (file)

### »»» Xml → Json

```
import xmltodict, json
obj = xmltodict.parse("""
<employee>
    <name>Khánh Hưng</name>
    <role>Lập trình viên</role>
    <age>30</age>
</employee>
""")
print(json.dumps(obj, ensure_ascii=False, indent=3))
```

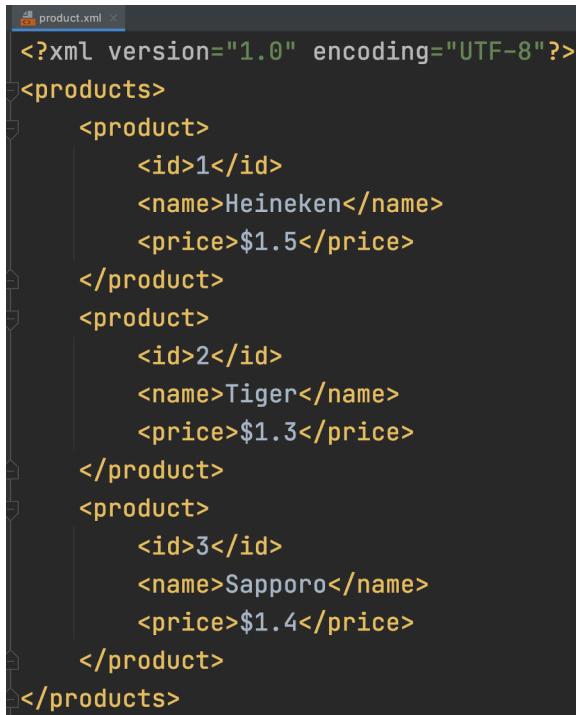
```
{
    "employee": {
        "name": "Khánh Hưng",
        "role": "Lập trình viên",
        "age": "30"
    }
}
```

## 2. Xử lý tập tin (file)

### Xml → Json

```
import xmltodict, json
```

```
with open("product.xml", "r", encoding="utf8") as f:  
    obj = xmltodict.parse(f.read())  
    print(json.dumps(obj, indent=3))
```



```
<?xml version="1.0" encoding="UTF-8"?>  
<products>  
    <product>  
        <id>1</id>  
        <name>Heineken</name>  
        <price>$1.5</price>  
    </product>  
    <product>  
        <id>2</id>  
        <name>Tiger</name>  
        <price>$1.3</price>  
    </product>  
    <product>  
        <id>3</id>  
        <name>Sapporo</name>  
        <price>$1.4</price>  
    </product>  
</products>
```

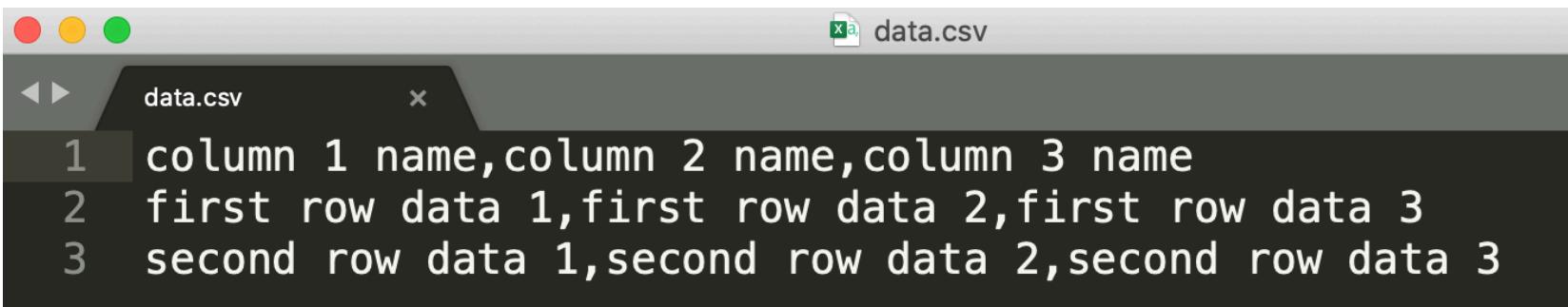


```
{  
    "products": {  
        "product": [  
            {  
                "id": "1",  
                "name": "Heineken",  
                "price": "$1.5"  
            },  
            {  
                "id": "2",  
                "name": "Tiger",  
                "price": "$1.3"  
            },  
            {  
                "id": "3",  
                "name": "Sapporo",  
                "price": "$1.4"  
            }  
        ]  
    }  
}
```

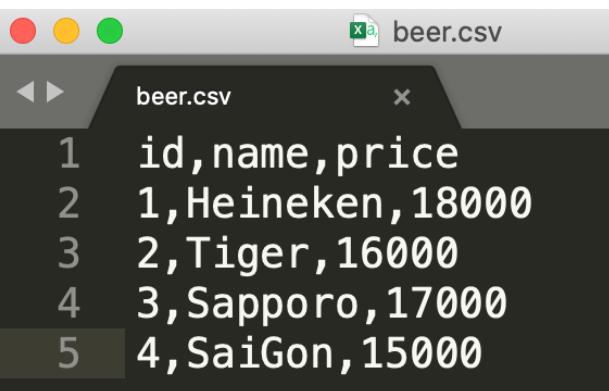
## 2. Xử lý tập tin (file)

»»» Excel file (.csv (comma-separated values) / .xlsx)

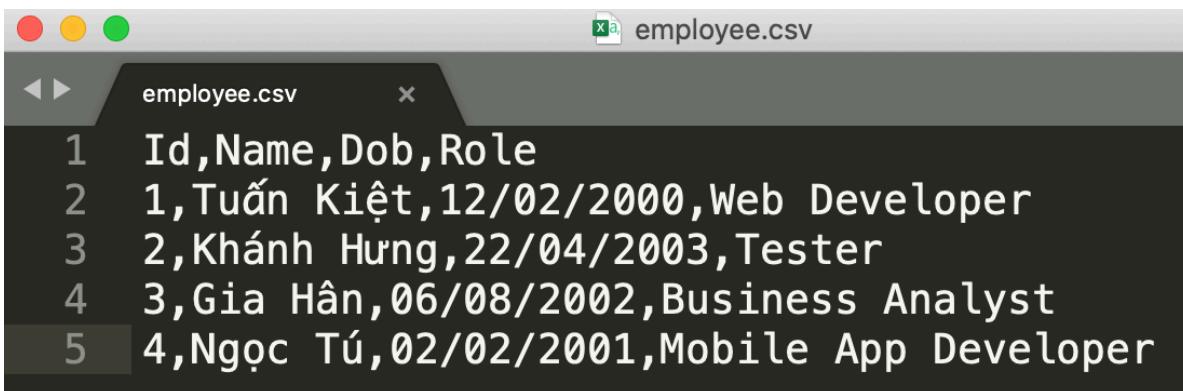
Cấu trúc tập tin .csv



```
data.csv
1 column 1 name,column 2 name,column 3 name
2 first row data 1,first row data 2,first row data 3
3 second row data 1,second row data 2,second row data 3
```



```
beer.csv
1 id,name,price
2 1,Heineken,18000
3 2,Tiger,16000
4 3,Sapporo,17000
5 4,SaiGon,15000
```



```
employee.csv
1 Id,Name,Dob,Role
2 1,TuẤn Kiệt,12/02/2000,Web Developer
3 2,Khánh Hưng,22/04/2003,Tester
4 3,Gia Hân,06/08/2002,Business Analyst
5 4,Ngọc Tú,02/02/2001,Mobile App Developer
```

## 2. Xử lý tập tin (file)

### »» Excel file (.csv (comma-separated values) / .xlsx)

#### Reading .csv file

```
import csv
with open('employee.csv', newline='') as f:
    reader = csv.reader(f, delimiter=',', quoting=csv.QUOTE_NONE)
    for row in reader:
        print(' - '.join(row))
        # print(row[0] + ' * ' + row[1] + ' * ' + row[2] + ' * ' + row[3])
```

Id - Name - Dob - Role

1 - Tuấn Kiệt - 12/02/2000 - Web Developer

2 - Khánh Hưng - 22/04/2003 - Tester

3 - Gia Hân - 06/08/2002 - Business Analyst

4 - Ngọc Tú - 02/02/2001 - Mobile App Developer

## 2. Xử lý tập tin (file)

»»» Excel file (.csv (comma-separated values) / .xlsx)

Reading .csv file (using pandas library)

```
import pandas  
df = pandas.read_csv('employee.csv')  
print(df)
```

	<b>Id</b>	<b>Name</b>	<b>Dob</b>	<b>Role</b>
0	1	Tuấn Kiệt	12/02/2000	Web Developer
1	2	Khánh Hưng	22/04/2003	Tester
2	3	Gia Hân	06/08/2002	Business Analyst
3	4	Ngọc Tú	02/02/2001	Mobile App Developer

## 2. Xử lý tập tin (file)

### »» Excel file (.csv (comma-separated values) / .xlsx)

#### Writing .csv file

```
import csv

with open('new_employee.csv', mode='w') as f:
    employee_writer = csv.writer(f, delimiter=',', quotechar='"')
    employee_writer.writerow(['Id', 'Name', 'Dob'])
    employee_writer.writerow(['1', 'Tú Linh', '02/02/2002'])
    employee_writer.writerow(['2', 'Nam Giao', '03/04/2000'])
    employee_writer.writerow(['3', 'Huỳnh Anh', '05/11/2001'])
```

The screenshot shows a code editor with two tabs: 'read\_write\_csv.py' and 'new\_employee.csv'. The Python script 'read\_write\_csv.py' contains the code provided in the previous block. The 'new\_employee.csv' tab shows the generated CSV file with the following data:

	Id, Name, Dob
1	1, Tú Linh, 02/02/2002
2	2, Nam Giao, 03/04/2000
3	3, Huỳnh Anh, 05/11/2001

## 2. Xử lý tập tin (file)

### »» Excel file (.csv (comma-separated values) / .xlsx)

#### Writing .csv file

```
import csv
with open('new_employee.csv', mode='w') as f:
    fieldnames = ['Id', 'Name', 'Dob']
    writer = csv.DictWriter(f, fieldnames=fieldnames)
    writer.writeheader()
    writer.writerow({'Id': '1', 'Name': 'Tú Linh', 'Dob': '02/02/2002'})
    writer.writerow({'Id': '2', 'Name': 'Nam Giao', 'Dob': '03/04/2000'})
    writer.writerow({'Id': '3', 'Name': 'Huỳnh Anh', 'Dob': '05/11/2001'})
```

The screenshot shows a terminal window with two tabs: 'read\_write\_csv.py' and 'new\_employee.csv'. The 'new\_employee.csv' tab is active, displaying the contents of the CSV file. The file contains four rows of data:

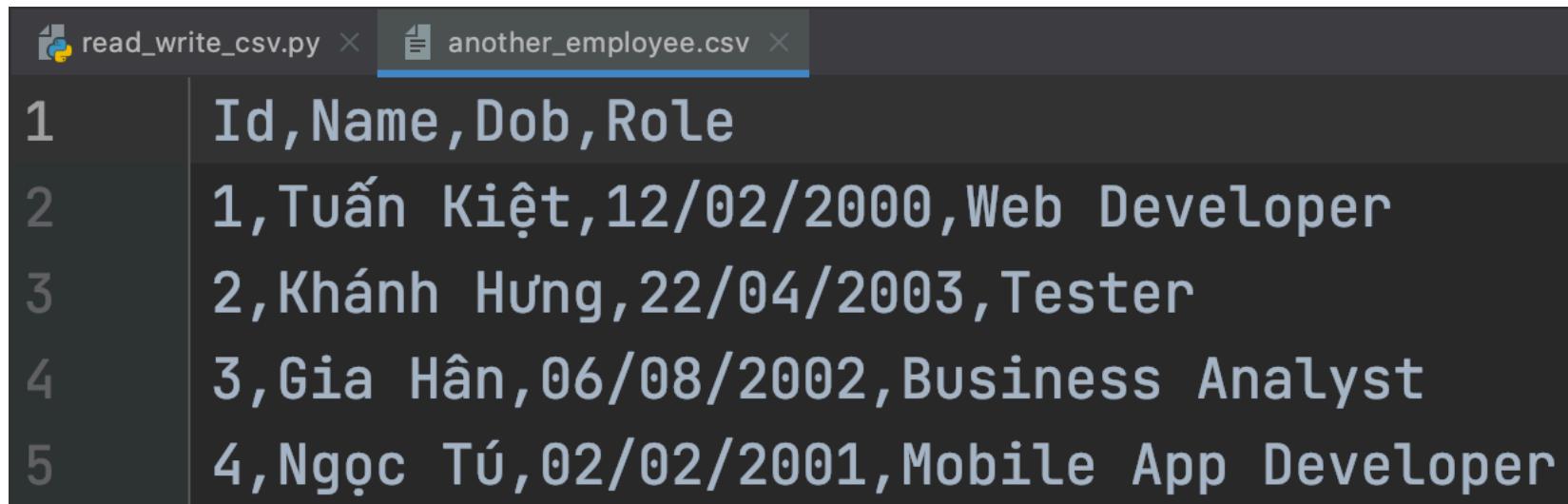
	Id, Name, Dob
1	1, Tú Linh, 02/02/2002
2	2, Nam Giao, 03/04/2000
3	3, Huỳnh Anh, 05/11/2001

## 2. Xử lý tập tin (file)

»» Excel file (.csv (comma-separated values) / .xlsx)

Writing .csv file (using pandas library)

```
import pandas  
df = pandas.read_csv('employee.csv', index_col="Id")  
df.to_csv('another_employee.csv')
```



The screenshot shows a code editor with two tabs: 'read\_write\_csv.py' and 'another\_employee.csv'. The 'another\_employee.csv' tab is active, displaying a CSV file with five rows of employee data:

	Id	Name	Dob	Role
1	1	Tuấn Kiệt	12/02/2000	Web Developer
2	2	Khánh Hưng	22/04/2003	Tester
3	3	Gia Hân	06/08/2002	Business Analyst
4	4	Ngọc Tú	02/02/2001	Mobile App Developer

## 2. Xử lý tập tin (file)

### »» Excel file (.csv (comma-separated values) / .xlsx)

#### Writing .xlsx file

```
import xlsxwriter as xr

workbook = xr.Workbook("Products.xlsx")
worksheet = workbook.add_worksheet()

# Modify column width
worksheet.set_column('A:A', 5)
worksheet.set_column('B:B', 20)
worksheet.set_column('C:C', 15)

bold = workbook.add_format({'bold': True})

# Add header
worksheet.write('A1', 'Id', bold)
worksheet.write('B1', 'Name', bold)
worksheet.write('C1', 'Price', bold)
```

```
# Add first row
worksheet.write('A2', '1')
worksheet.write('B2', 'Heineken')
worksheet.write('C2', '19000')

# Add second row
worksheet.write('A3', '2')
worksheet.write('B3', 'Tiger')
worksheet.write('C3', '18000')

# Insert image
worksheet.insert_image('B5', 'beer.png')

workbook.close()
```

## 2. Xử lý tập tin (file)

»»» Excel file (.csv (comma-separated values) / .xlsx)

Writing .xlsx file

The screenshot shows a Microsoft Excel spreadsheet titled "Products". The table has columns labeled "Id", "Name", and "Price". The data is as follows:

	A	B	C	D	E	F	G
1	Id	Name	Price				
2	1	Heineken	19000				
3	2	Tiger	18000				

Below the table, there is a graphic of two glasses toasting.

## 2. Xử lý tập tin (file)

### »» Excel file (.csv (comma-separated values) / .xlsx)

#### Reading .xlsx file

```
from openpyxl import load_workbook  
wb = load_workbook('Products.xlsx')  
print(wb.sheetnames)  
ws = wb[wb.sheetnames[0]]  
for row in ws.values:  
    for value in row:  
        print(value.center(9), end="")  
    print("")
```

['Sheet1']		
	Name	Price
1	Heineken	19000
2	Tiger	18000

## 2. Xử lý tập tin (file)

»» Excel file (.csv (comma-separated values) / .xlsx)

### Reading .xlsx file

\$ pip install xlrld → .xls

\$ pip install openpyxl → .xlsx

```
import pandas as pd  
df = pd.read_excel('Sales.xlsx')
```

	Week	Sales_Volume	Price	Ads_Cost
0	1	350	5.50000	3.30000
1	2	460	7.50000	3.30000
2	3	350	8.00000	3.00000
3	4	430	8.00000	4.50000
4	5	350	6.80000	3.00000

## 2. Xử lý tập tin (file)

»»» Multimedia file (.jpg, .png, .mp3, .mp4, ...)

### Reading image

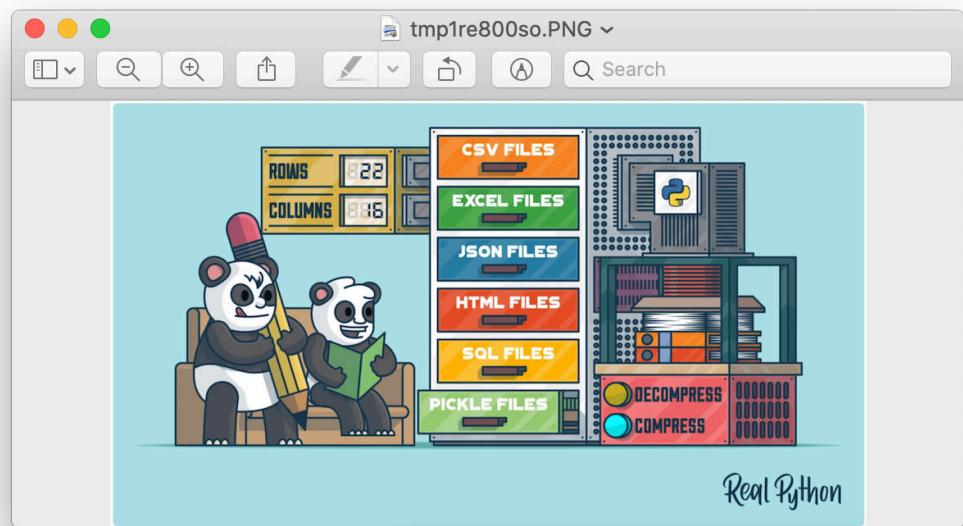
```
from PIL import Image
```

```
# Read image  
img = Image.open('python.png')
```

```
# Output Images  
img.show()
```

```
# Prints format of image  
print(img.format) PNG
```

```
# Prints mode of image  
print(img.mode) RGBA
```



## 2. Xử lý tập tin (file)

»»» Multimedia file (.jpg, .png, .mp3, .mp4, ...)

### Playing sound

```
from playsound import playsound
```

```
playsound("music.mp3")
```



## 2. Xử lý tập tin (file)

### »» Multimedia file (.jpg, .png, .mp3, .mp4, ...)

#### Reading video

```
import cv2
import numpy as np
# Create a VideoCapture object and read
from input file
cap = cv2.VideoCapture('swing.mp4')
# Check if camera opened successfully
if not cap.isOpened():
    print("Error opening video file")
```

```
# Read until video is completed
while cap.isOpened():
    # Capture frame-by-frame
    ret, frame = cap.read()
    if ret:
        # Display the resulting frame
        cv2.imshow('Frame', frame)
        # Press Q on keyboard to exit
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    # Break the loop
    else:
        break
# When everything done, release the video
capture object
cap.release()
# Closes all the frames
cv2.destroyAllWindows()
```

# GUI FRAMEWORKS

# NỘI DUNG

1. Framework hỗ trợ phát triển ứng dụng với Python
2. GUI Frameworks

# 1. Framework hỗ trợ phát triển ứng dụng với Python

Desktop  
App

Tkinter, PyQt, PyGUI, ...



Web

Django, Flask, Pyramid, ...



Mobile  
App

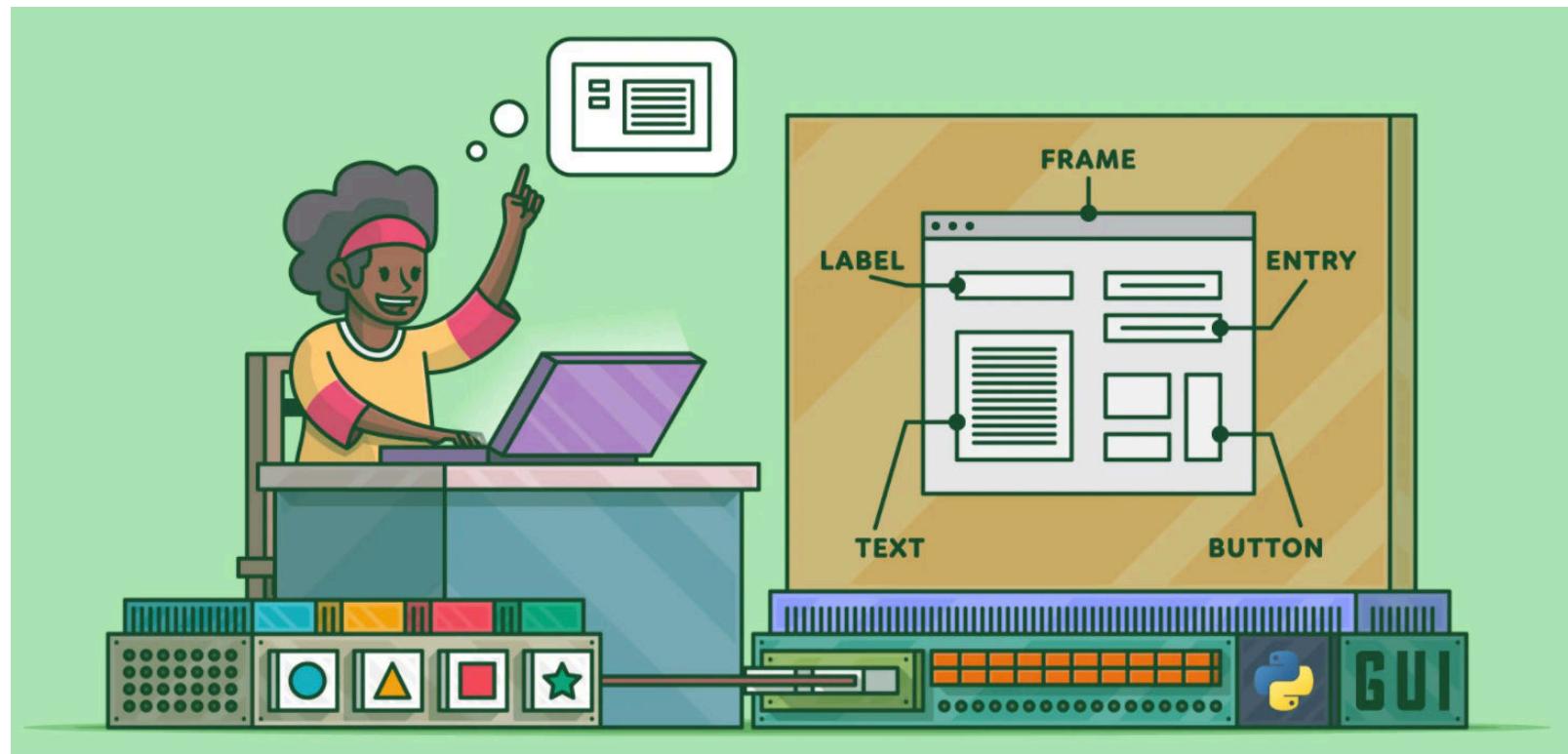
Kivy, BeeWare, ...



## 2. GUI Frameworks

### »» Tkinter (Desktop App)

Tkinter là một GUI framework được tích hợp vào bộ thư viện chuẩn của Python, hoạt động đa nền tảng.



## 2. GUI Frameworks

### »» Tkinter (Desktop App)

#### Tạo cửa sổ màn hình giao diện

```
from tkinter import *
window = Tk()
window.title('Tkinter - GUI framework')
window.resizable(height=True, width=True)
window.minsize(500, 350)
# Center Screen
window.update_idletasks()
width = window.winfo_width()
height = window.winfo_height()
x = (window.winfo_screenwidth()//2) - (width//2)
y = (window.winfo_screenheight()//2) - (height//2)
window.geometry(f'{width}x{height}+{x}+{y}')
# Draw UI
window.mainloop()
```



## 2. GUI Frameworks

### »»» Tkinter (Desktop App)

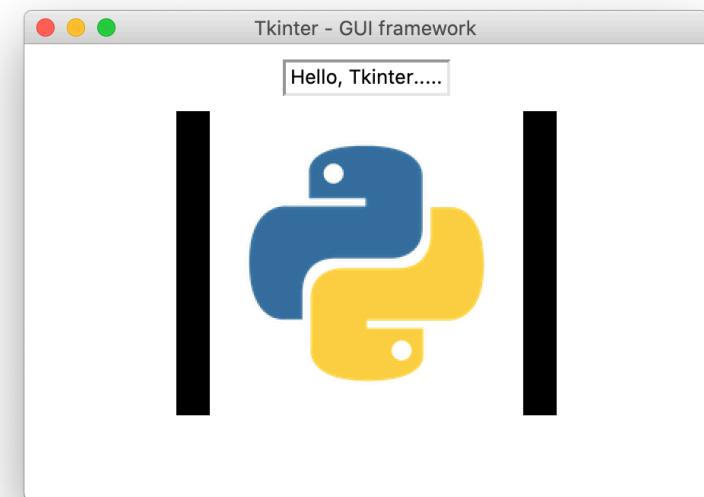
**Label:** hiển thị text và image, không cho phép chỉnh sửa.

```
from tkinter import *
from PIL import Image, ImageTk
window = Tk()
window.title('Tkinter - GUI framework')
window.resizable(height=True, width=True)
window.minsize(450, 300)

Label(window, text='Hello, Tkinter.....', justify=CENTER,
      relief=SUNKEN).pack(pady=10)

image = Image.open('Python_Logo.png')
image = image.resize((200, 200), Image.ANTIALIAS)
photo = ImageTk.PhotoImage(image)
frame = Frame(window, width=250, height=200, background='black')
frame.pack_propagate(0)
frame.pack()
Label(frame, image=photo).pack()

window.mainloop()
```



## 2. GUI Frameworks

### »»» Tkinter (Desktop App)

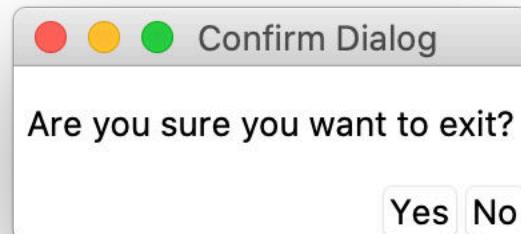
**Button:** control cho phép người dùng tương tác với ứng dụng thông qua sự kiện click (nhấn chuột)

```
from tkinter import *

window = Tk()
window.title('Confirm Dialog')

label = Label(window, text='Are you sure you want to exit?')
label.pack(pady=10)
Button(window, text='No').pack(side=RIGHT)
Button(window, text='Yes', command=window.quit).pack(side=RIGHT)

window.mainloop()
```



## 2. GUI Frameworks

### »» Tkinter (Desktop App)

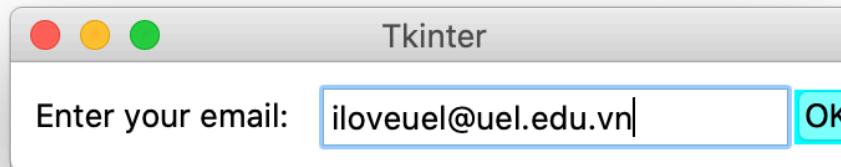
Entry: control cho phép người dùng nhập liệu.

```
from tkinter import *

window = Tk()
window.title('Tkinter')

Label(window, text="Enter your email:").pack(side=LEFT, pady=10, padx=5)
email = StringVar()
email.set('iloveuel@uel.edu.vn')
Entry(window, textvariable=email, width=20).pack(side=LEFT)
Button(window, text="OK", highlightbackground='cyan').pack(side=RIGHT, pady=10)

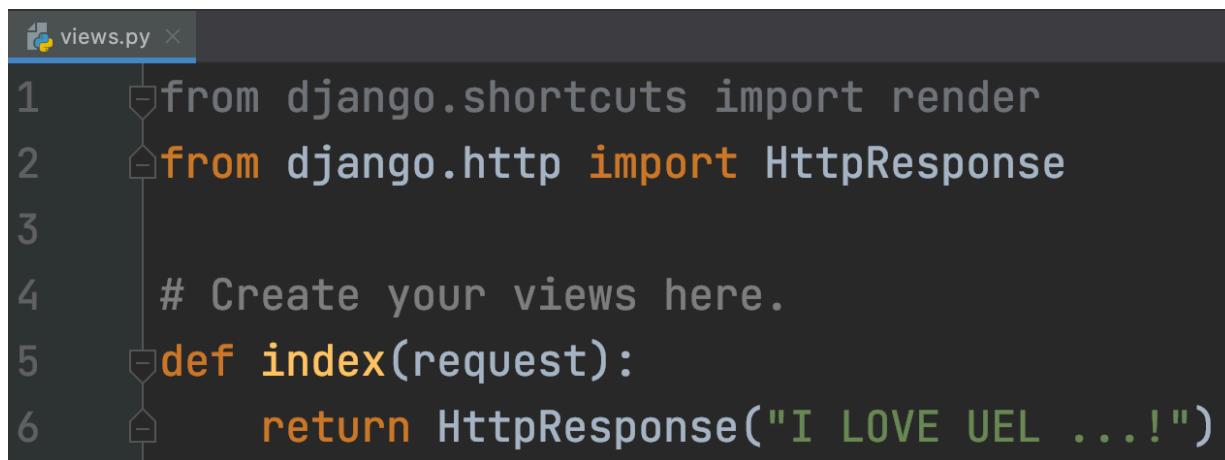
window.mainloop()
```



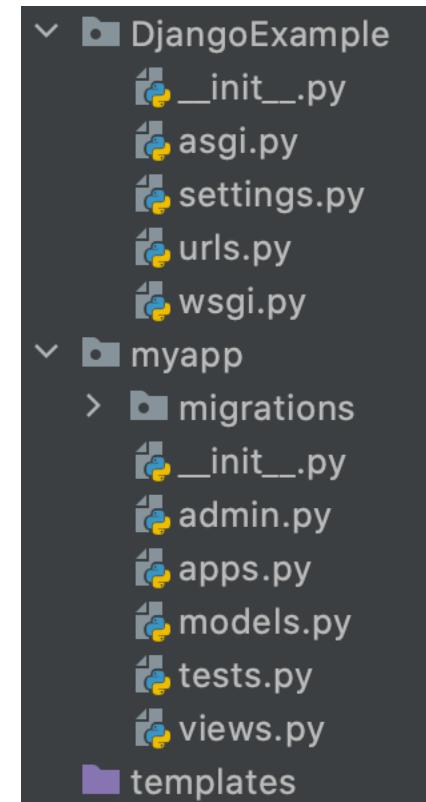
## 2. GUI Frameworks

### »»» Django (Web)

\$ python manage.py startapp **myapp**



```
views.py x
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3
4 # Create your views here.
5 def index(request):
6     return HttpResponseRedirect("I LOVE UEL ...!")
```



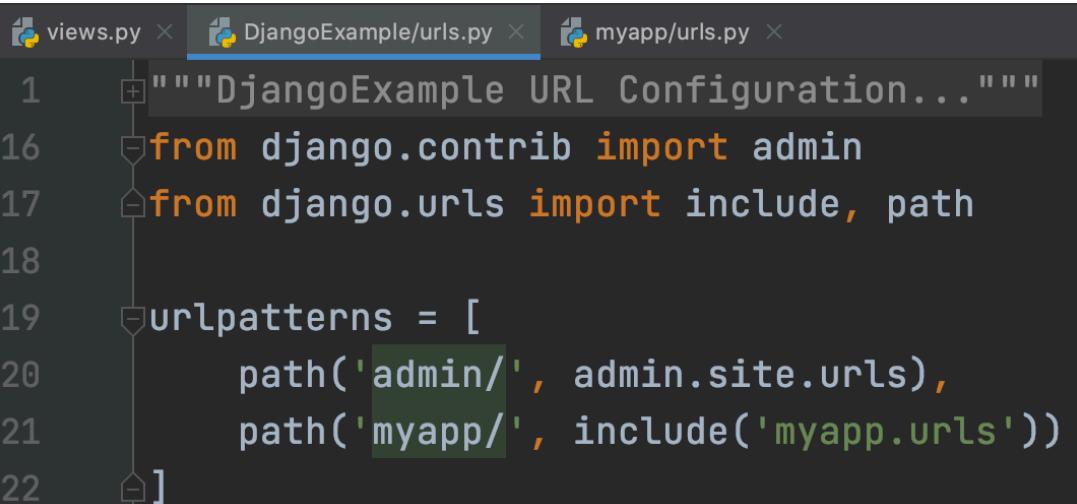
## 2. GUI Frameworks

### »»» Django (Web)

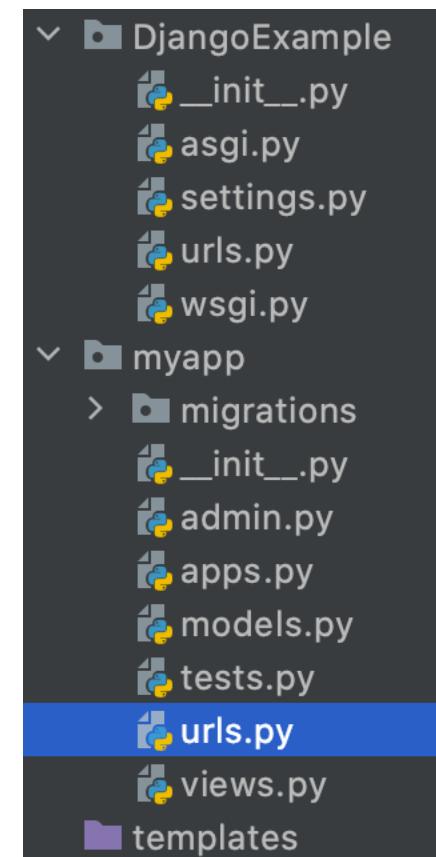
\$ python manage.py startapp myapp



```
views.py × DjangoExample/urls.py × myapp/urls.py ×
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index, name='index'),
6 ]
```



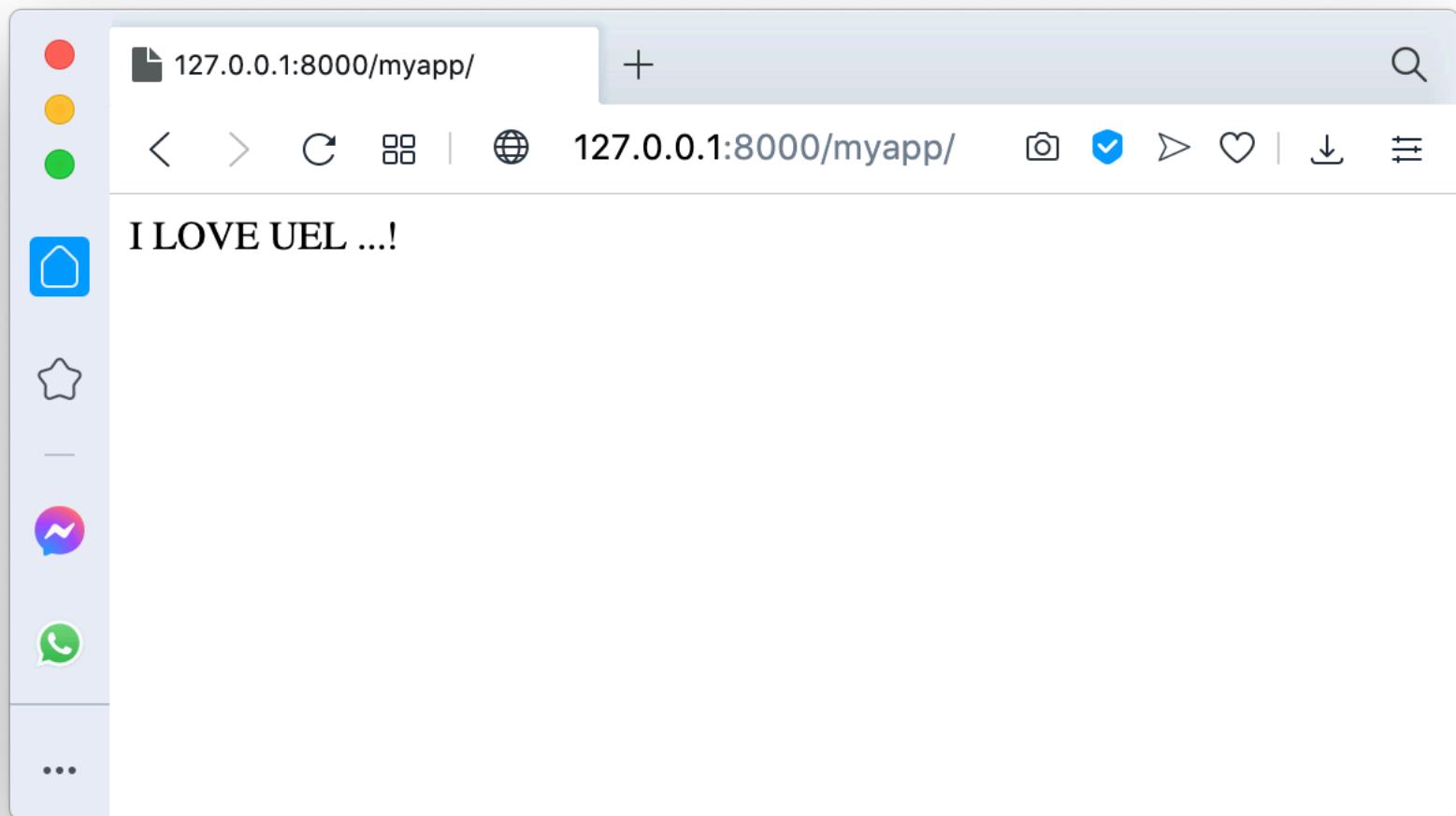
```
views.py × DjangoExample/urls.py × myapp/urls.py ×
1 """DjangoExample URL Configuration"""
16 from django.contrib import admin
17 from django.urls import include, path
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('myapp/', include('myapp.urls'))
22 ]
```



## 2. GUI Frameworks

### »»» Django (Web)

```
$ python manage.py runserver
```



## 2. GUI Frameworks

### »»» Kivy (Mobile App)

<https://kivy.org/doc/stable/gettingstarted/installation.html>

The screenshot shows a web browser displaying the Kivy documentation at <https://kivy.org/doc/stable/gettingstarted/installation.html>. The page title is "Getting Started » Installing Kivy". The left sidebar has a dropdown for "Version" set to "stable" and a "Quick search" input field. The main content area starts with a note about installing Kivy 2.0.0, mentioning Python 3.6 - 3.9 support. It then provides a table for platform-specific installation details:

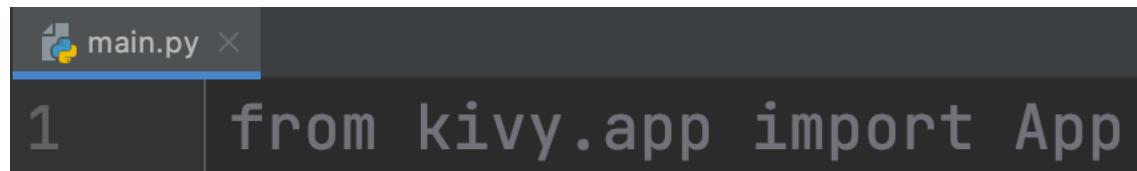
	Platform	Installation	Packaging
	Windows	pip	PyInstaller
	OS X	pip, Kivy.app	Kivy.app, PyInstaller
	Linux	pip, PPA	—
	RPi	pip	—
	Android	python-for-android	python-for-android
<b>iOS</b>	iOS	kivy-ios	kivy-ios
	Anaconda	conda	—

The right sidebar contains a "Table Of Contents" section with links to various installation and setup guides.

## 2. GUI Frameworks

### »»» Kivy (Mobile App)

\$ pip install kivy[base] kivy\_examples



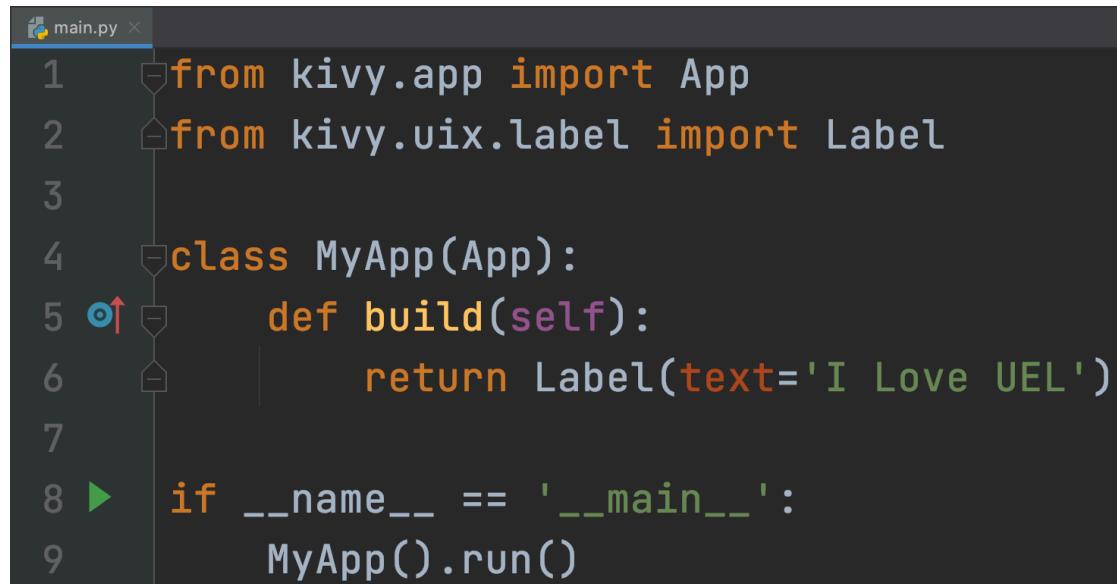
```
main.py ×  
1 | from kivy.app import App
```

```
Terminal: Local +  
(venv) Macmalls-MacBook-Pro:KivyExample macmall$ python main.py  
[INFO ] [Logger      ] Record log in /Users/macmall/.kivy/logs/kivy_21-04-23_39.txt  
[INFO ] [Kivy        ] v2.0.0  
[INFO ] [Kivy        ] Installed at "/Volumes/Data/Works/1.UEL/Python/Demo/KivyExample/venv/lib/python3.9/site-packages/kivy/__init__.py"  
[INFO ] [Python       ] v3.9.1 (v3.9.1:1e5d33e9b9, Dec  7 2020, 12:10:52)  
[Clang 6.0 (clang-600.0.57)]  
[INFO ] [Python       ] Interpreter at "/Volumes/Data/Works/1.UEL/Python/Demo/KivyExample/venv/bin/python"  
[INFO ] [Factory     ] 186 symbols loaded  
[INFO ] [Image       ] Providers: img_tex, img_imageio, img_dds, img_sdl2, img_pil (img_ffpyplayer ignored)
```

## 2. GUI Frameworks

### »»» Kivy (Mobile App)

\$ python **main.py**



```
main.py x
1 from kivy.app import App
2 from kivy.uix.label import Label
3
4 class MyApp(App):
5     def build(self):
6         return Label(text='I Love UEL')
7
8 if __name__ == '__main__':
9     MyApp().run()
```

The image shows a code editor window titled "main.py". The code defines a simple Kivy application named "MyApp". It imports the necessary modules, defines the application class, and specifies the build function to return a label with the text "I Love UEL". A conditional statement checks if the script is run directly to run the application.



# Q & A