TÁC GIẢ: 8 SYNC

CỘNG ĐỒNG

- Kevin Nguyễn
- Nhóm Chia Sẻ Công Nghệ
- Nhóm BlockChain
- Tiktok: 8 Sync
- Youtube: 8 Sync Dev
- Zalo

KHÓA HỌC:

- Fullstack Python
- Fullstack Nextjs
- Fullstack Android-IOS

Tài liệu sẽ được cập nhật định kì và thông báo trong group nên các bạn chú ý nhen .

PYTHON - TRÌNH DIỄN VÀ CÁC CHẾ ĐỘ HOẠT ĐỘNG

Python là một ngôn ngữ dựa trên trình diễn viên. Trong hệ thống Linux, chương trình thực thi Python được cài đặt trong thư mục /usr/bin/. Đối với Windows,

chương trình thực thi (python.exe) được tìm thấy trong thư mục cài đặt (ví dụ: C:\python311).

Hướng dẫn này sẽ giúp bạn hiểu cách Trình Diễn Python hoạt động ở chế độ tương tác và chế độ viết kịch bản. Mã Python được thực thi bằng cách từng câu lệnh một. Trình diễn viên Python có hai thành phần. Trình dịch kiểm tra câu lệnh để kiểm tra cú pháp. Nếu tìm thấy đúng, nó tạo ra một mã byte trung gian. Có một máy ảo Python sau đó chuyển đổi mã byte thành nhị phân cơ bản và thực thi nó. Sơ đồ dưới đây mô tả cơ chế:

TRÌNH DIỄN PYTHON

Trình diễn viên Python có hai chế độ: chế độ tương tác và chế độ viết kịch bản.

TRÌNH DIỄN PYTHON - CHẾ ĐỘ TƯƠNG TÁC

Khi được khởi chạy từ một dòng lệnh mà không có tùy chọn bổ sung, một lời nhắc Python >>> xuất hiện và trình diễn viên Python hoạt động dựa trên nguyên tắc REPL (Đọc, Đánh Giá, In, Lặp). Mỗi lệnh nhập vào trước dấu nhắc Python được đọc, dịch và thực thi. Một phiên tương tác điển hình như sau.

< 8 Sync Dev />

```
>>> price = 100
>>> qty = 5
>>> total = price * qty
>>> total
500
>>> print("Tổng = ", total)
Tổng = 500
```

Để đóng phiên tương tác, nhập ký tự cuối dòng (ctrl+D cho Linux và ctrl+Z cho Windows). Bạn cũng có thể gõ quit() trước dấu nhắc Python và nhấn Enter để trở về dấu nhắc hệ điều hành.

< 8 Sync Dev />

```
>>> quit()
$
```

Bộ môi trường tương tác có sẵn với bản phân phối Python chuẩn không được trang bị các tính năng như chỉnh sửa dòng, tìm kiếm lịch sử, tự động hoàn thành v.v. Bạn có thể sử dụng các phần mềm trình diễn tương tác nâng cao khác như IPython và bpython để có các chức năng bổ sung.

TRÌNH DIỄN PYTHON - CHẾ ĐỘ VIẾT KỊCH BẢN

Thay vì nhập và nhận kết quả của một hướng dẫn một lần như trong môi trường tương tác, bạn có thể lưu một tập hợp các hướng dẫn trong một tệp văn bản, đảm bảo rằng nó có phần mở rộng .py, và sử dụng tên đó làm tham số dòng lệnh cho lệnh Python.

Lưu các dòng sau đây vào prog.py, với sự sử dụng của bất kỳ trình soạn thảo văn bản nào như vim trên Linux hoặc Notepad trên Windows.

< 8 Sync Dev />

```
print("Chương trình đầu tiên của tôi")
price = 100
qty = 5
total = price * qty
print("Tổng = ", total)
```

Khi chúng ta thực thi chương trình trên máy tính Windows, nó sẽ tạo ra kết quả sau:

< 8 Sync Dev />

```
C:\Users\Acer>python prog.py
Chương trình đầu tiên của tôi
Tổng = 500
```

Lưu ý rằng mặc dù Python thực thi toàn bộ tập lệnh một cách liên tục, nhưng bên trong, nó vẫn được thực thi theo dòng.

Trong trường hợp của bất kỳ ngôn ngữ dựa trên trình biên dịch nào như Java, mã nguồn không được chuyển đổi sang mã byte trừ khi toàn bộ mã nguồn không có lỗi. Trong Python, ngược lại, các câu lệnh được thực thi cho đến khi gặp lỗi đầu tiên.

Hãy giả định một lỗi một cách cố ý trong mã trên.

< 8 Sync Dev />

```
print("Chương trình đầu tiên của tôi")
price = 100
qty = 5
total = prive * qty #Lỗi ở câu lệnh này
print("Tổng = ", total)
```

Chú ý biến viết sai prive thay vì price. Hãy thử thực thi lại chương trình như trước -

< 8 Sync Dev />

```
C:\Users\Acer>python prog.py
Chương trình đầu tiên của tôi
Traceback (most recent call last):
  File "C:\Python311\prog.py", line 4, in <module>
    total = prive * qty
NameError: name 'prive' is not defined. Did you mean: 'price'?
```

Lưu ý rằng các câu lệnh trước câu lệnh lỗi được thực thi trước và sau đó thông báo lỗi xuất hiện. Do đó, bây giờ đã rõ ràng rằng kịch bản Python được thực thi

theo cách thông dịch.

TRÌNH DIỄN PYTHON - SỬ DỤNG SHEBANG #!

Ngoài việc thực thi kịch bản Python như trên, chính kịch bản cũng có thể là một tệp tự thực thi trong Linux, giống như một tập lệnh shell. Bạn phải thêm một dòng shebang ở đầu kịch bản. Dòng shebang chỉ ra loại trình diễn Python nào được sử dụng để giải thích các câu lệnh Python trong kịch bản. Dòng đầu tiên của kịch bản bắt đầu bằng #! và tiếp theo là đường dẫn đến trình diễn Python.

Chỉnh sửa kịch bản prog.py như sau −

< 8 Sync Dev />

```
#! /usr/bin/python3.11

print("Chương trình đầu tiên của tôi")
price = 100
qty = 5
total = price * qty
print("Tổng = ", total)
```

Để đánh dấu kịch bản là tự thực thi, sử dụng lệnh chmod

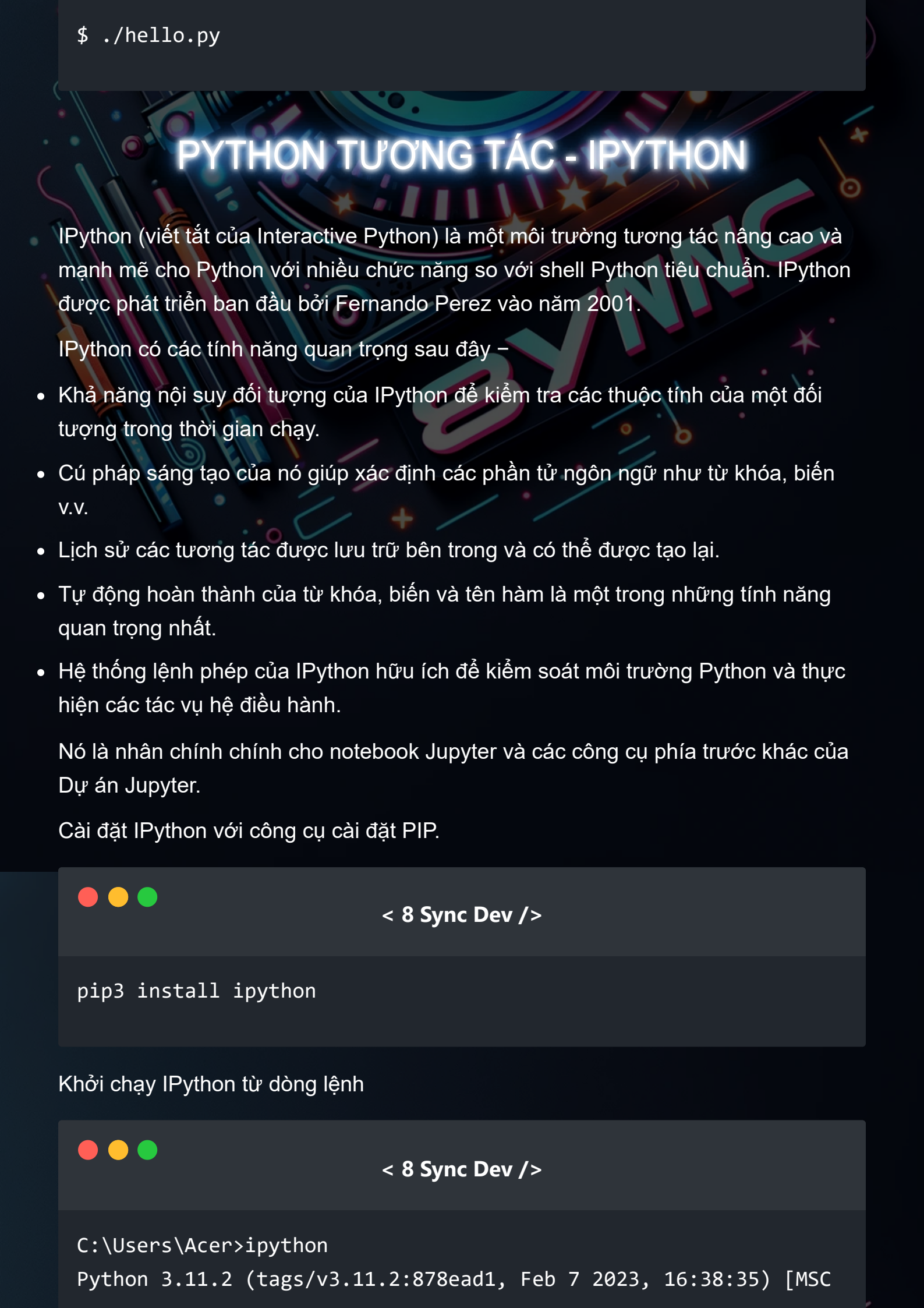
< 8 Sync Dev />

```
$ chmod +x prog.py
```

Bạn có thể thực thi kịch bản trực tiếp, mà không cần sử dụng nó như một đối số dòng lệnh.

< 8 Sync Dev />

```
$ ./hello.py
```



IPython (viết tắt của Interactive Python) là một môi trường tương tác nâng cao và mạnh mẽ cho Python với nhiều chức năng so với shell Python tiêu chuẩn. IPython được phát triển ban đầu bởi Fernando Perez vào năm 2001.

IPython có các tính năng quan trọng sau đây −

- Khả năng nội suy đối tượng của IPython để kiểm tra các thuộc tính của một đối tượng trong thời gian chạy.
- Cú pháp sáng tạo của nó giúp xác định các phần tử ngôn ngữ như từ khóa, biến v.v.
- Lịch sử các tương tác được lưu trữ bên trong và có thể được tạo lại.
- Tự động hoàn thành của từ khóa, biến và tên hàm là một trong những tính năng quan trọng nhất.
- Hệ thống lệnh phép của IPython hữu ích để kiểm soát môi trường Python và thực hiện các tác vụ hệ điều hành.

Nó là nhân chính chính cho notebook Jupyter và các công cụ phía trước khác của Dự án Jupyter.

Cài đặt IPython với công cụ cài đặt PIP.

< 8 Sync Dev />

```
pip3 install ipython
```

Khởi chạy IPython từ dòng lệnh

< 8 Sync Dev />

```
C:\Users\Acer>ipython
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC
```

```
v.1934
64 bit (AMD64)] on win32
Type 'copyright', 'credits' or 'license' for more information
IPython 8.4.0 -- An enhanced Interactive Python. Type '?' for
help.
In [1]:
```

Thay vì dấu nhắc >>> thông thường như trong trình diễn viên tiêu chuẩn, bạn sẽ thấy hai dấu nhắc quan trọng của IPython được giải thích dưới đây −

- In[1] xuất hiện trước bất kỳ biểu thức nhập nào.
- Out[1] xuất hiện trước Kết quả xuất hiện.


```
In [1]: price = 100
In [2]: quantity = 5
In [3]: total = price * quantity
In [4]: total
Out[4]: 500
In [5]:
```

Hoàn thành tab là một trong những cải tiến hữu ích nhất do IPython cung cấp. IPython hiện ra danh sách phương thức phù hợp khi bạn nhấn phím tab sau dấu chấm trước đối tượng.

IPython cung cấp thông tin (nội suy) của bất kỳ đối tượng nào bằng cách đặt ? trước nó. Điều này bao gồm chuỗi tài liệu, định nghĩa hàm và chi tiết constructor của lớp. Ví dụ, để khám phá đối tượng chuỗi var được xác định ở trên, nhập var? vào dấu nhắc nhập.


```
In [5]: var = "Hello World"
In [6]: var?
Type: str
```

```
String form: Hello World
Length: 11
Docstring
```