

# 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 - PHẠM VI BIẾN

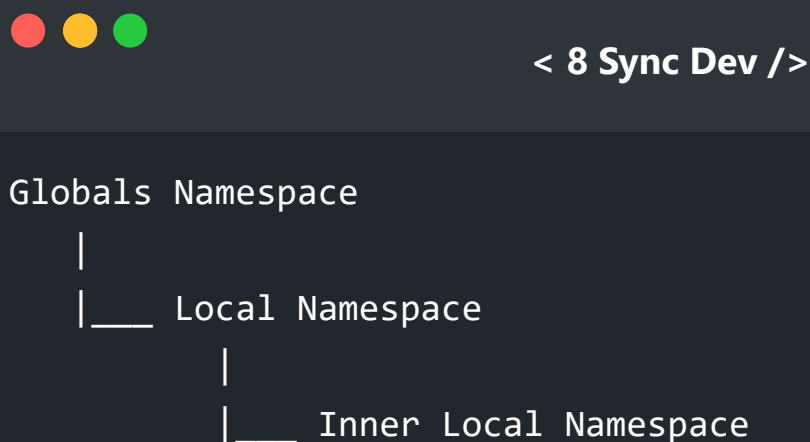
---

Một biến trong Python là tên biểu tượng cho đối tượng trong bộ nhớ máy tính. Python hoạt động dựa trên khái niệm về không gian tên để xác định ngữ cảnh cho các định danh khác nhau như các hàm, biến, v.v. Một không gian tên là một tập hợp các tên biểu tượng được xác định trong ngữ cảnh hiện tại.

Python cung cấp các loại không gian tên sau:

- **Không gian tên tích hợp** chứa các hàm tích hợp và các ngoại lệ tích hợp. Chúng được tải vào bộ nhớ ngay khi trình thông dịch Python được tải và tồn tại cho đến khi trình thông dịch đang chạy.
- **Không gian tên toàn cục** chứa bất kỳ tên nào được xác định trong chương trình chính. Những tên này tồn tại trong bộ nhớ cho đến khi chương trình đang chạy.
- **Không gian tên cục bộ** chứa các tên được xác định bên trong một hàm. Chúng có sẵn cho đến khi hàm đang chạy.

Các không gian tên này được lồng vào nhau. Đồ thị sau đây cho thấy mối quan hệ giữa các không gian tên:



## LOẠI CỦA KHÔNG GIAN TÊN

Tuổi thọ của một biến nhất định được hạn chế trong không gian tên mà nó được xác định. Do đó, không thể truy cập vào một biến có trong không gian tên bên trong từ bất kỳ không gian tên bên ngoài nào.

## HÀM GLOBALS() CỦA PYTHON

Thư viện chuẩn của Python bao gồm một hàm tích hợp là `globals()`. Nó trả về một từ điển các biểu tượng hiện có trong không gian tên toàn cục.

Chạy hàm `globals()` trực tiếp từ dấu nhắc Python.

```
>>> globals()
{'__name__': '__main__', '__doc__': None, '__package__': None,
 '__loader__': <class '_frozen_importlib.BuiltinImporter'>,
 '__spec__': None, '__annotations__': {}, '__builtins__': <module
'builtins' (built-in)>}
```

Có thể thấy rằng mô-đun builtins chứa các định nghĩa của tất cả các hàm tích hợp và ngoại lệ tích hợp được tải.

Lưu mã sau chứa một số biến và một hàm với một số biến nữa bên trong nó.

```
< 8 Sync Dev />

name = '8SyncDev'
marks = 50
result = True
def myfunction():
    a = 10
    b = 20
    return a + b

print(globals())
```

Gọi globals() từ bên trong tập lệnh này sẽ trả về đối tượng từ điển sau:

```
< 8 Sync Dev />

{'__name__': '__main__', '__doc__': None, '__package__': None,
 '__loader__': <_frozen_importlib_external.SourceFileLoader
object at 0x00000169AE265250>, '__spec__': None,
 '__annotations__': {}, '__builtins__': <module 'builtins'
(built-in)>, '__file__': 'C:\\Users\\mlath\\examples\\main.py',
 '__cached__': None, 'name': '8SyncDev', 'marks': 50, 'result':
True, 'myfunction': <function myfunction at 0x00000169AE2104A0>}
```



Không gian tên toàn cục bây giờ chứa các biến trong chương trình và giá trị của chúng cũng như đối tượng hàm trong đó (và không phải là các biến trong hàm).

Bất kỳ biến nào được tạo bên ngoài một hàm có thể truy cập trong bất kỳ hàm nào và vì vậy chúng có phạm vi toàn cục. Dưới đây là một ví dụ để hiển thị việc sử dụng biến toàn cục trong Python:

< 8 Sync Dev />

```
x = 5
y = 10

def sum():
    total = x + y
    return total

print(sum())
```

Điều này sẽ tạo ra kết quả sau:

< 8 Sync Dev />

```
15
```

## HÀM LOCALS() CỦA PYTHON

Thư viện chuẩn của Python cũng bao gồm một hàm tích hợp là `locals()`. Nó trả về một từ điển các biểu tượng hiện có trong không gian tên cục bộ của hàm.

Sửa đổi mã trên để in ra từ điển của không gian tên toàn cục và cục bộ từ bên trong hàm.


< 8 Sync Dev />

```
name = '8SyncDev'
marks = 50
result = True

def myfunction():
    a = 10
    b = 20
    c = a + b
    print("globals():", globals())
    print("locals():", locals())
    return c

myfunction()
```


Kết quả sẽ cho thấy locals() trả về một từ điển các biến và giá trị của chúng hiện có trong hàm.



< 8 Sync Dev />

```
globals(): {'__name__': '__main__', '__doc__': None,
 '__package__': None, '__loader__':
<_frozen_importlib_external.SourceFileLoader object at
0x00000169AE265250>, '__spec__': None, '__annotations__': {},
 '__builtins__': <module 'builtins' (built-in)>, '__file__':
'C:\\Users\\mlath\\examples\\main.py', '__cached__': None, '
name': '8SyncDev', 'marks': 50, 'result': True, 'myfunction':
<function myfunction at 0x00000169AE2104A0>}
locals(): {'a': 10, 'b': 20, 'c': 30}
```


Vì cả hai hàm globals() và locals() đều trả về từ điển, bạn có thể truy cập vào giá trị của một biến từ không gian tên tương ứng với phương pháp get() của từ điển hoặc toán tử chỉ mục.



< 8 Sync Dev />

```
print(globals()['name']) # hiển thị 8SyncDev
print(locals().get('a')) # hiển thị 10
```


Dưới đây là một ví dụ đơn giản để hiển thị việc sử dụng biến cục bộ trong Python:



< 8 Sync Dev />

```
def sum(x, y):
    total = x + y
    return total

print(sum(5, 10))
```




< 8 Sync Dev />

15

## XUNG ĐỘT PHẠM VI TRONG PYTHON

Nếu một biến có cùng tên xuất hiện trong phạm vi toàn cục cũng như phạm vi cục bộ, trình thông dịch Python ưu tiên biến trong không gian tên cục bộ.



< 8 Sync Dev />

```
marks = 50 # đây là biến toàn cục
def myfunction():
    marks = 70 # đây là biến cục bộ
    print(marks)

myfunction()
print(marks) # in giá trị toàn cục
```

Sẽ tạo ra kết quả sau:



< 8 Sync Dev />

70

50

Nếu bạn cố gắng thay đổi giá trị của một biến toàn cục từ bên trong một hàm, Python sẽ ném ra `UnboundLocalError`.



< 8 Sync Dev />

```
marks = 50 # đây là biến toàn cục
```

```
def myfunction():
```

```
    marks = marks + 20
```

```
    print(marks)
```

```
myfunction()
```

```
print(marks) # in giá trị toàn cục
```

Sẽ tạo ra kết quả sau:



< 8 Sync Dev />

```
marks = marks + 20
```

```
^^^^^
```

```
UnboundLocalError: cannot access local variable 'marks' where it  
is not associated with a value
```

Để sửa đổi một biến toàn cục, bạn có thể cập nhật nó bằng cú pháp từ điển, hoặc sử dụng từ khóa `global` để tham chiếu đến nó trước khi sửa đổi.



### < 8 Sync Dev />

```
var1 = 50 # đây là biến toàn cục
var2 = 60 # đây là biến toàn cục
def myfunction():
    "Thay đổi giá trị của các biến toàn cục"
    globals()['var1'] = globals()['var1'] + 10
    global var2
    var2 = var2 + 20

myfunction()
print("var1:", var1, "var2:", var2) # hiển thị các biến toàn cục
với giá trị đã thay đổi
```

Sẽ tạo ra kết quả sau:

### < 8 Sync Dev />

```
var1: 60 var2: 80
```

Cuối cùng, nếu bạn cố gắng truy cập vào một biến cục bộ trong phạm vi toàn cục, Python sẽ ném ra `NameError` vì biến trong phạm vi cục bộ không thể truy cập ở bên ngoài phạm vi đó.

### < 8 Sync Dev />

```
var1 = 50 # đây là biến toàn cục
var2 = 60 # đây là biến toàn cục
def myfunction(x, y):
    total = x + y
    print("Total là biến cục bộ: ", total)

myfunction(var1, var2)
print(total) # Điều này sẽ tạo ra NameError
```



Sẽ tạo ra kết quả sau:



< 8 Sync Dev />

```
Total là biến cục bộ: 110
```

```
Traceback (most recent call last):
```

```
  File "C:\Users\user\examples\main.py", line 9, in <module>
```

```
    print(total) # Điều này sẽ tạo ra NameError
```

```
    ^^^^^
```

```
NameError: name 'total' is not defined
```