# NÂNG CAO HIỆU QUẢ VIẾT CODE VỚI TYPE HINTS VÀ MYPY

*Dinh-Tiem Nguyen và Quang-Vinh Dinh*

## 1 Mở đầu

Làm sao để viết code Python dễ đọc, dễ hiểu và dễ bảo trì hơn? Làm thế nào để có thể kiểm tra tính đúng đắn của code trước khi thực thi chúng? Và liệu có cách nào để phát hiện và sửa lỗi kiểu dữ liệu một cách dễ dàng và nhanh chóng không? Trong bài viết này sẽ hướng dẫn sử dụng Type Hints để xác định và gợi ý kiểu dữ liệu của các biến, giá trị trả về của các hàm, phương thức... Đồng thời kết hợp với Mypy-một công cụ kiểm tra kiểu dữ liệu static mạnh mẽ cho các chương trình Python, giúp nâng cao hiệu quả viết code trong quá trình phát triển dự án.
Yêu cầu:

- Máy tính đã cài đặt python>=3.10, Vscode.
- Đã biết lập trình Python cơ bản.


## 2 Hướng dẫn sử dụng Type Hints và Mypy

### 2.1 Hướng dẫn sử dụng Type Hints

Type Hints là một tính năng đã được tích hợp sẵn trong Python, cho phép ta khi viết code có thể khai báo kiểu dữ liệu của các biến, tham số và giá trị trả về của hàm. Và điều thú vị là khi thực thi chương trình Python trình thông dịch sẽ bỏ qua các gợi ý kiểu dữ liệu trong code. Chúng ta sẽ hiểu rõ hơn thông qua các ví dụ.

#### 2.1.1 Type hints cho biến

Chúng ta sẽ bắt đầu bằng ví dụ tạo biến theo cách thông thường:

```python
# Cách không dùng type hints
name = "AI VIETNAM"
year = 2024

print(f"Welcome to {name} {year}!")
```

```
================ Output ===============
Welcome to AI VIETNAM 2024!


=======================================
```

Để gợi ý về kiểu dữ liệu của một biến, ta có thể sử dụng Type Hints như sau:

```python
name: str = "AI VIETNAM"
year: int = 2024

print(f"Welcome to {name} {year}!")
print(__annotations__)
```

```
================ Output ===============
Welcome to AI VIETNAM 2024!
{'name': <class 'str'>, 'year': <class '
    int'>}
=======================================
```

Trong ví dụ trên chúng ta khai báo kiểu dữ liệu cho name là string, year với kiểu int tuy nhiên khi thực thi chương trình thì đều cho kết quả giống nhau vì Python sẽ bỏ qua phần gợi ý kiểu dữ liệu. Khác biệt duy nhất là khi sử dụng type hints chương trình sẽ tạo ra một thộc tính đặc biệt __annotations__ để chứa thông tin về kiểu dữ liệu của các biến, điều này sẽ giúp IDE(Vscode) đưa ra những gợi ý cú pháp chính xác. Ngoài ra thuộc tính này sẽ là thông tin quan trọng để các công cụ hỗ trợ như Mypy có thể kiểm tra kiểu dữ liệu của chương trình.
Khi sử dụng type hints, có nhiều trường hợp gợi ý kiểu cho biến là kiểu list, tuple, boolean, hoặc biến đó có thể có nhiều kiểu dữ liệu khác nhau.

```python
# Biến với Type Hint là list chứa các số nguyên
numbers: list[int]

# Biến với Type Hint là tuple chứa một chuỗi và một số nguyên
person: tuple[str, int]

# Biến với Type Hint là int hoặc float
value: int|float

# Biến với Type Hint là list chứa chuỗi và tuple chứa số nguyên
data: list[str]|tuple[int, int]

# Biến với Type Hint là Tuple chứa một list các số nguyên và một dict
info: tuple[list[int], dict]

# Biến với Type hint là boolean
is_active: bool
```

Trong ví dụ trên, chúng ta đã sử dụng Type Hints để chỉ định kiểu dữ liệu của các biến numbers, person, value, data, và info. Khi chúng ta tạo type hints cho biến mà không gán giá trị như vậy, thì các biến này vẫn chưa thực sự được tạo ra. Mà chỉ có __annotations__ được tạo ra chứa thông tin gợi ý kiểu dữ liệu cho từng biến. Chúng ta có thể kiểm tra điều này bằng cách in biến đó ra để xem thông tin.

```python
print(__annotations__)
print(numbers, person, value, data, info)
```

```
==================================== Output ====================================
{'numbers': list[int], 'person': tuple[str, int], 'value': int | float, 'data': list[
    str] | tuple[int, int], 'info': tuple[list[int], dict], 'is_active': <class 'bool
    '>}
Traceback (most recent call last):
  File "create_type_hints.py", line 20, in <module>
    print(numbers, person, value, data, info, is_active)
NameError: name 'numbers' is not defined
================================================================================
```

### 2.1.2 Type hints cho hàm

Để gợi ý về kiểu dữ liệu của các tham số và giá trị trả về của một hàm, ta có thể sử dụng Type Hints như ví dụ phía dưới. Trong hàm add, chúng ta gợi ý rằng x và y là các số nguyên (int) và hàm trả về là một số nguyên (int). Nhưng nếu ta cố tình sử dụng kiểu dữ liệu float thì chương trình vẫn hoạt động bình thường, không một cảnh báo lỗi nào xuất hiện.

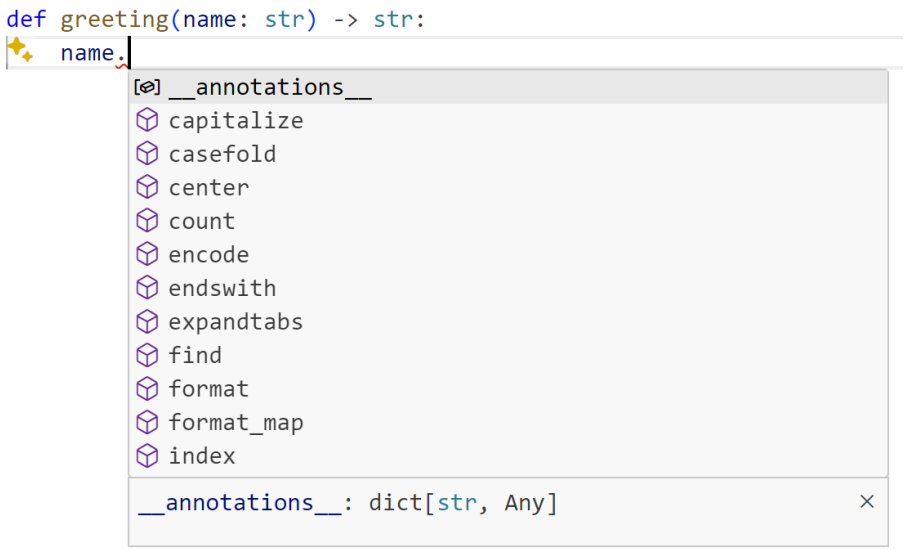
```python
def add(x: int, y: int) -> int:
    return x + y

if __name__ == "__main__":
    print(add(x= 1.5,  y=2))
    print(add.__annotations__)
```

```
================ Output ===============
3.5
{'x': <class 'int'>, 'y': <class 'int'>,
'return': <class 'int'>}

=======================================
```

Ưu điểm của việc sử dụng type hints là giúp code của chúng ta dễ đọc hơn, ngoài ra thì giúp công cụ IDE code gợi ý cú pháp hiệu quả hơn. Thông thường, khi chúng ta tạo hàm, nếu các tham số trong hàm không được gán giá trị mặc định thì IDE không biết kiểu dữ liệu của tham số đó là gì, nên không đưa ra gợi ý code cho chúng ta được. Nhưng khi sử dụng type hints, điều này lại được khắc phục.




Hình 1: Gợi ý code trong Vscode xuất hiện khi sử dụng type hints

### 2.1.3 Type hints cho class

Để sử dụng type hints cho class, đối với thuộc tính chúng ta tạo type hints cho chúng như cách làm với biến, đối với các phương thức thì chúng ta tạo type hints như cách chúng ta làm với hàm.

```python
class Person:
    name: str
    age: int

    def __init__(self, name: str, age: int) -> None:
        self.name = name
        self.age = age

    def greet(self) -> str:
        return f"Xin chào, Tôi là {self.name} năm nay tôi {self.age} tuổi."

if __name__  == "__main__":
    person_1 = Person("Tom", 25)
    print(person_1.greet())
```

```
==================================== Output ====================================
Tôi là Tom năm nay tôi 25 tuổi.
================================================================================
```

Trong lớp Person trên, chúng ta gợi ý rằng thuộc tính name là một chuỗi (str), thuộc tính age là một số nguyên (int) và phương thức greet trả về một chuỗi (str).
**Lưu ý:** Việc tạo type hints trong code chương trình là không bắt buộc, không có chúng thì chương trình vẫn chạy bình thường. Vậy thì khi nào nên và không nên sử dụng type hints? Thì sẽ tùy vào từng dự án, nhóm bạn làm việc có muốn sử dụng type hints không. Nhìn chung, type hints có nhiều ưu điểm giúp code chương trình rõ ràng và dễ phát hiện và sửa lỗi bảo trì hơn so với nhược điểm như phải dành thêm nhiều thời gian hơn để viết mã.

## 2.2 Kết hợp Type Hints với Mypy

Type hints giúp code của chúng ta rõ ràng, dễ đọc và dễ sửa lỗi hơn, tuy nhiên việc phát hiện lỗi chúng ta vẫn thực hiện thủ công, người lập trình phải tự đọc hiểu và cố gắng kiểm soát các kiểu dữ liệu để đảm bảo tính logic nhưng điều này có thể gây mất nhiều thời gian mà không hiệu quả.
Công cụ mypy ra đời để khắc phục điều này, nó là một công cụ kiểm tra kiểu dữ liệu static Python, cho phép chúng ta kiểm tra kiểu dữ liệu trong chương trình mà không cần phải thực thi chúng. Nó được phát triển bởi Jukka Lehtosalo và được công bố dưới dạng mã nguồn mở.
Để sử dụng mypy, trước tiên chúng ta cần cài đặt nó thông qua câu lệnh sau:

```
pip install mypy
```

Sau khi cài đặt thành công, chúng ta có thể sử dụng mypy để kiểm tra kiểu dữ liệu trong chương trình của chúng ta bằng cách chạy lệnh:

```
mypy my_script.py
```

Trong đó my_script.py là file chứa code chương trình và các type hints. Ta cần lưu ý là mypy chỉ kiểm tra được cho tệp có đuôi .py, đối với tệp notebook.ipynb chúng ta cần sử dụng công cụ khác như nbqa. Trong phạm vi bài viết này, chúng ta chỉ kiểm tra kiểu đối với file .py.
Chúng ta cùng quay trở lại ví dụ tính tổng hai số, ta tạo type hints cho các tham số là kiểu int, nhưng trong ví dụ, chúng ta truyền giá trị cho hai tham số là kiểu float và int thì chương trình vẫn chạy bình thường. Dưới đây, chúng ta sẽ kiểm tra kiểu bằng mypy.

```
def add(x: int, y: int) -> int:
    return x + y

if __name__ == "__main__":
    print(add(x= 1,  y=2))
    print(add(x= 1.5,  y=2))
    print(add.__annotations__)
```

```
==================================== Output ====================================
mypy add.py
add.py:6: error: Argument "x" to "add" has incompatible type "float"; expected "int"
  [arg-type]
Found 1 error in 1 file (checked 1 source file)
================================================================================
```

Kết quả mypy trả về cho thấy trong file code của chúng ta add.py xuất hiện lỗi kiểu ở dòng 6 tức là dòng print(add(x= 1.5, y=2)), lỗi này do x=1.5 có kiểu là float không khớp với kiểu dữ liệu dự kiến từ type hints là int. Sau khi xác định lỗi, chúng ta dễ dàng sửa lỗi và chương trình sẽ trở lên đúng đắn hơn. Ta thấy mypy thật tiện lợi phải không?

# 3 Bài tập

Viết một chương trình Python để kiểm tra xem một số có phải là số nguyên tố hay không. Số nguyên tố là số nguyên dương lớn hơn 1 và chỉ có hai ước số dương là 1 và chính nó.
Yêu cầu:

- Viết một hàm có tên là is_prime nhận một số nguyên dương n và trả về một giá trị boolean. Nếu n là số nguyên tố, hàm sẽ trả về True, ngược lại trả về False.
- Sử dụng Type Hints để gợi ý về kiểu dữ liệu của tham số và giá trị trả về của hàm.
- Sử dụng mypy để kiểm tra kiểu dữ liệu trong chương trình