

Generic

Generics cho phép bạn **xác định kiểu dữ liệu một cách tổng quát**, giúp **viết code linh hoạt và an toàn hơn** về mặt kiểu dữ liệu – đặc biệt hữu ích khi viết class, hàm làm việc với nhiều kiểu dữ liệu khác nhau.

Python **hỗ trợ Generics**, đặc biệt từ phiên bản Python 3.5 trở đi, nhờ vào **typing module**.

Cú pháp cơ bản với `Generic` trong Python

```
from typing import TypeVar, Generic, List

T = TypeVar("T") # T là một kiểu bất kỳ

class Box(Generic[T]):
    def __init__(self, content: T):
        self.content = content

    def get_content(self) -> T:
        return self.content

# Dùng cụ thể
box_int = Box
box_str = Box[str]("hello")

print(box_int.get_content()) # [] 123
print(box_str.get_content()) # [] "hello"
```

`T` ở đây có thể là `int`, `str`, `List[str]`, hay bất cứ kiểu nào bạn muốn

Ứng dụng thực tế

1. Với hàm:

```
from typing import TypeVar

T = TypeVar("T")

def identity(x: T) -> T:
```

```
return x

print(identity(5))    # int
print(identity("test")) # str
```

2. Với container:

```
from typing import List

def first_item(items: List[T]) -> T:
    return items[0]
```

Một số generic types hay dùng trong `typing`

Type	Ý nghĩa
<code>List[T]</code>	Danh sách chứa các phần tử kiểu <code>T</code>
<code>Dict[K, V]</code>	Dictionary với key là <code>K</code> , value là <code>V</code>
<code>Optional[T]</code>	Có thể là <code>T</code> hoặc <code>None</code>
<code>Union[T1, T2, ...]</code>	Có thể là một trong các kiểu liệt kê
<code>Tuple[T1, T2]</code>	Tuple với các phần tử theo thứ tự kiểu
<code>Callable[[T1, T2], R]</code>	Hàm nhận <code>T1</code> , <code>T2</code> trả về <code>R</code>
<code>Generic[T]</code>	Dùng để tạo class generic

Tác giả: **Đỗ Ngọc Tú**
Công Ty Phần Mềm **VHTSoft**