



# add\_window\_to\_document()方法的作用 (2025-05-09 11:40:23)

*User*

list类型和typing库的List类型区别

---

*Assistant*

---

## list类型和typing库的List类型区别

### 基本区别

#### 1. list :

- Python内置数据类型
- 用于创建和操作列表对象
- 运行时实际使用的数据结构

#### 2. typing.List :

- 类型注解工具 '不是实际数据类型
- 用于静态类型检查和IDE提示
- 不影响程序运行时行为

# 使用方式

```
# 使用内置list
my_list = [1, 2, 3] # 创建列表实例

# 使用typing.List进行类型注解
from typing import List
def process_numbers(numbers: List[int]) -> List[str]:
    return [str(num) for num in numbers]
```

## 主要优势

1. **typing.List**提供泛型支持：
  - 可以指定列表元素类型：`List[int]`、`List[str]` 等
  - 有助于静态类型检查器分析代码
2. 版本差异：
  - Python 3.9+可以直接使用内置list作为泛型：`list[int]`
  - 旧版本必须使用typing.List

使用typing.List主要是为了提高代码可读性和可维护性，帮助开发者和工具捕获潜在类型错误。

---

### User

```
def process_numbers(numbers: list[int]) -> list[str]:
    return [str(num) for num in numbers]
```

这样不行吗

---

### Assistant

---

这样是可以的，但有版本要求：

## Python 3.9及以上版本

```
def process_numbers(numbers: list[int]) -> list[str]:  
    return [str(num) for num in numbers]
```

这种写法完全正确 °Python 3.9引入了对内置容器类型的泛型类型注解支持 ’允许直接对内置类型如 `list` 、`dict` 、`tuple` 等使用方括号语法指定元素类型 °

## Python 3.8及以下版本

对于旧版本Python ’必须使用typing模块 :

```
from typing import List  
  
def process_numbers(numbers: List[int]) -> List[str]:  
    return [str(num) for num in numbers]
```

## 建议做法

如果你的项目必须兼容3.9以下版本 ’使用typing模块更安全 °如果只需支持3.9+ ’可以直接使用内置类型的泛型语法 ’代码会更简洁 °

也可以通过条件导入确保代码兼容性 :

```
import sys  
if sys.version_info >= (3, 9):  
    ListType = list  
else:  
    from typing import List as ListType
```

---