

开发工具

本章中介绍的模块可帮助你编写软件。例如，[pydoc](#) 模块接受一个模块并根据该模块的内容来生成文档。[doctest](#) 和 [unittest](#) 模块包含用于编写自动执行代码并验证是否产生预期的输出的单元测试的框架。

本章中描述的模块列表是：

- [typing](#) --- 对类型提示的支持
 - 有关 Python 类型系统的规范说明
 - 类型别名
 - [NewType](#)
 - 标注可调用对象
 - 泛型 (Generic)
 - 标注元组
 - 类对象的类型
 - 标注生成器和协程
 - 用户定义的泛型类型
 - [Any](#) 类型
 - 名义子类型 vs 结构子类型
 - 模块内容
 - 特殊类型原语
 - 特殊类型
 - 特殊形式
 - 构造泛型类型与类型别名
 - 其他特殊指令
 - 协议
 - 与 IO 相关的抽象基类和协议
 - 函数与装饰器
 - 内省辅助器
 - 常量
 - 一些已被弃用的别名
 - 内置类型的别名
 - [collections](#) 中的类型的别名。
 - 其他具体类型的别名
 - [collections.abc](#) 中容器 ABC 的别名
 - [collections.abc](#) 中异步 ABC 的别名
 - [collections.abc](#) 中其他 ABC 的别名
 - [contextlib](#) ABC 的别名
 - 主要特性的弃用时间线
- [pydoc](#) --- 文档生成器和在线帮助系统

- Python 开发模式
 - Python 开发模式的效果
 - ResourceWarning 示例
 - 文件描述符错误示例
- doctest --- 测试交互式的 Python 示例
 - 简单用法：检查Docstrings中的示例
 - 简单的用法：检查文本文件中的例子
 - 命令行用法
 - 它是如何工作的
 - 哪些文件串被检查了？
 - 文档串的例子是如何被识别的？
 - 什么是执行上下文？
 - 异常如何处理？
 - 选项标记
 - 指令
 - 警告
 - 基本API
 - unittest API
 - 高级 API
 - DocTest 对象
 - Example 对象
 - DocTestFinder 对象
 - DocTestParser 对象
 - TestResults 对象
 - DocTestRunner 对象
 - OutputChecker 对象
 - 调试
 - 肥皂盒
- unittest --- 单元测试框架
 - 基本实例
 - 命令行接口
 - 命令行选项
 - 探索性测试
 - 组织你的测试代码
 - 复用已有的测试代码
 - 跳过测试与预计的失败
 - 使用子测试区分测试迭代
 - 类与函数
 - 测试用例
 - 分组测试
 - 加载和运行测试
 - load_tests 协议
 - 类与模块设定

- setUpClass 和 tearDownClass
 - setUpModule 和 tearDownModule
- 信号处理
- unittest.mock --- 模拟对象库
 - 快速上手
 - Mock 类
 - 调用
 - 删除属性
 - Mock 的名称与 name 属性
 - 附加 Mock 作为属性
 - patch 装饰器
 - patch
 - patch.object
 - patch.dict
 - patch.multiple
 - 补丁方法: start 和 stop
 - 为内置函数打补丁
 - TEST_PREFIX
 - 嵌套补丁装饰器
 - 补丁的位置
 - 对描述器和代理对象打补丁
 - MagicMock 与魔术方法支持
 - 模拟魔术方法
 - MagicMock
 - 辅助对象
 - sentinel
 - DEFAULT
 - call
 - create_autospec
 - ANY
 - FILTER_DIR
 - mock_open
 - 自动 spec
 - 将 mock 封包
 - side_effect, return_value 和 wraps 的优先顺序
- unittest.mock --- 新手入门
 - 使用 mock
 - 模拟方法调用
 - 对象上的方法调用的 mock
 - 模拟类
 - 命名你的 mock
 - 追踪所有的调用
 - 设置返回值和属性

- 通过 mock 引发异常
- 附带影响函数和可迭代对象
- 模拟异步迭代器
- 模拟异步上下文管理器
- 基于现有对象创建模拟对象
- 使用 side_effect 返回每个文件的内容
- 补丁装饰器
- 更多示例
 - 模拟链式调用
 - 部分模拟
 - 模拟生成器方法
 - 对每个测试方法应用相同的补丁
 - 模拟未绑定方法
 - 通过 mock 检查多次调用
 - 处理可变参数
 - 嵌套补丁
 - 使用 MagicMock 模拟字典
 - 模拟子类及其属性
 - 通过 patch.dict 模拟导入
 - 追踪调用顺序和不太冗长的调用断言
 - 更复杂的参数匹配
- test --- Python 回归测试包
 - 为 test 包编写单元测试
 - 使用命令行界面运行测试
- test.support --- 针对 Python 测试套件的工具
- test.support.socket_helper --- 用于套接字测试的工具
- test.support.script_helper --- 用于 Python 执行测试工具
- test.support.bytecode_helper --- 用于测试正确字节码生成的支持工具
- test.support.threading_helper --- 用于线程测试的工具
- test.support.os_helper --- 用于操作系统测试的工具
- test.support.import_helper --- 用于导入测试的工具
- test.support.warnings_helper --- 用于警告测试的工具