

Python 运行时服务

本章描述的模块广泛服务于 Python 解释器及其与其环境的交互：

- `sys` --- 系统相关的形参和函数
- `sys.monitoring` --- 执行事件监测
 - 工具标识符
 - 注册和使用工具
 - 事件
 - 本地事件
 - 已弃用的事件
 - 辅助事件
 - 其他事件
 - `STOP_ITERATION` 事件
 - 开启和关闭事件
 - 全局设置事件
 - 针对特定代码对象的事件
 - 禁用事件
 - 注册回调函数
 - 回调函数参数
- `sysconfig` --- 提供对 Python 配置信息的访问
 - 配置变量
 - 安装路径
 - 用户方案
 - `posix_user`
 - `nt_user`
 - `osx_framework_user`
 - 主方案
 - `posix_home`
 - 前缀方案
 - `posix_prefix`
 - `nt`
 - 安装路径函数
 - 其他功能
 - 命令行用法
- `builtins` --- 内置对象
- `__main__` --- 最高层级代码环境
 - `__name__ == '__main__'`
 - 什么是“顶层代码环境”？
 - 惯用法
 - 打包考量

- Python 包中的 `__main__.py`
 - 惯用法
- `import __main__`
- warnings --- 警告信息控制
 - 警告类别
 - 警告过滤器
 - 重复警告的屏蔽准则
 - 警告过滤器的介绍
 - 默认警告过滤器
 - 重写默认的过滤器
 - 暂时禁止警告
 - 测试警告
 - 为新版本的依赖关系更新代码
 - 可用的函数
 - 可用的上下文管理器
 - 上下文管理器的并发安全性
- dataclasses --- 数据类
 - 模块内容
 - 初始化后处理
 - 类变量
 - 仅初始化变量
 - 冻结的实例
 - 继承
 - `__init__()` 中仅限关键字形参的重新排序
 - 默认工厂函数
 - 可变的默认值
 - 描述器类型的字段
- contextlib --- 为 with 语句上下文提供的工具
 - 工具
 - 例子和配方
 - 支持可变数量的上下文管理器
 - 捕获 `__enter__` 方法产生的异常
 - 在一个 `__enter__` 方法的实现中进行清理
 - 替换任何对 try-finally 和旗标变量的使用
 - 将上下文管理器作为函数装饰器使用
 - 单独使用，可重用并可重进入的上下文管理器
 - 重进入上下文管理器
 - 可重用的上下文管理器
- abc --- 抽象基类
- atexit --- 退出处理器
 - atexit 示例
- traceback --- 打印或读取栈回溯信息
 - 模块级函数

- `TracebackException` 对象
- `StackSummary` 对象
- `FrameSummary` 对象
- 使用模块级函数的例子
- 使用 `TracebackException` 的示例
- `__future__` --- Future 语句定义
 - 模块内容
- `gc` --- 垃圾回收器接口
- `inspect` --- 检查当前对象
 - 类型和成员
 - 获取源代码
 - 使用 `Signature` 对象对可调用对象进行内省
 - 类与函数
 - 解释器栈
 - 静态地获取属性
 - 生成器、协程和异步生成器的当前状态
 - 代码对象位标志
 - 缓冲区旗标
 - 命令行界面
- `annotationlib` --- 用于内省标记的功能
 - 注解语义 (Annotation semantics)
 - 类
 - 函数
 - 例程
 - 在元类中使用注解
 - `STRING` 格式的局限性
 - `FORWARDREF` 格式的局限性
 - 内省标注的安全意义
- `site` --- 站点专属的配置钩子
 - `sitecustomize`
 - `usercustomize`
 - Readline 配置
 - 模块内容
 - 命令行界面

参见:

- 参见 [concurrent.interpreters](#) 模块，它以类似方式对外公开了核心运行时功能。