命令argparse：

python test.py --<key> <value>

|  |  |  |
| --- | --- | --- |
| ArgumentParser() | | 实例化命令解析器 |
| 实例方法 | add\_argument(\*str, required, type, default, help) | 添加命令参数 |
| parse\_args() | 返回命令的变量空间 |
| add\_argument | \*, nargs='+', default=[] | 多参数捕获 |
| \*, action='store\_true' | 检测到时为True |

协程asyncio：

|  |  |
| --- | --- |
| async def <func>(\*args, \*\*kwargs) | 定义可等待函数，调用 → 协程实例 |
| await sleep(delay) | 返回睡眠协程 |
| run(async) | 等待协程 |
| wait(tasks) | 多协程 → 单协程 |

|  |  |  |
| --- | --- | --- |
| 循环： | | |
| get\_event\_loop() | | 实例化事件循环 |
| 实例方法 | run\_until\_complete(coroutine) | 等待单协程完成，返回执行结果 |
| close() | 关闭事件循环 |

|  |  |  |
| --- | --- | --- |
| 任务： | | |
| Future / Task | |  |
| 实例方法 | result() | 结果 |
| done() | 完成情况 |

退出atexit：

|  |  |
| --- | --- |
| **register(func, \*args, \*\*kwargs)** | 注册退出事件 |

密码cryptography.hazmat：

|  |  |
| --- | --- |
| 序列化primitives.serialization： | |
| **Encoding** | 序列化编码 |
| **PrivateFormat / PublicFormat** | 密钥格式 |
| **NoEncryption()** | 不加密私钥 |
| **load\_pem\_public\_key(data)** | 加载PEM格式公钥 |

|  |  |  |
| --- | --- | --- |
| 非对称primitives.asymmetric： | | |
| **rsa.generate\_private\_key**  **(public\_exponent=65537, key\_size=2048)** | | 生成RSA私钥 (随机) |
| **私钥方法** | **private\_bytes(encoding, format, encryption\_algorithm)** | 序列化RSA私钥 |
| **decrypt(ciphertext, padding)** | 解密 |
| **public\_key()** | 生成RSA公钥 (唯一) |
| **公钥方法** | **public\_bytes(encoding, format)** | 序列化RSA公钥 |
| **encrypt(plaintext, padding)** | 加密 |

系统ctypes：

|  |  |
| --- | --- |
| **CDLL(file)** | **初始化**依赖文件 **(以dict形式查找函数)** |

|  |  |  |
| --- | --- | --- |
| windll： | | |
| **shell32.IsUserAnAdmin()** | | 管理员权限真值 |
| **shell32.ShellExecuteW(None, "runas", sys.executable, \_\_file\_\_, None, 1)** | | 以管理员权限重新执行 |
| **user32.MessageBoxW(0, context, title, code)** | | 创建弹窗 |
| 形参code | **MB\_OK = 0x0**  **MB\_OKCXL = 0x01**  **MB\_YESNOCXL = 0x03**  **MB\_YESNO = 0x04**  **MB\_HELP = 0x4000** | **按钮** |
| **ICON\_INFO = 0x40**  **ICON\_EXCLAIM = 0x30**  **ICON\_STOP = 0x10** | **图标** |

密码getpass：

|  |  |
| --- | --- |
| getuser() | 返回登陆的用户名 |
| getpass(str) | 打印str并读取输入的密码 |

**基本os：**

|  |  |  |
| --- | --- | --- |
| **getcwd()** | **返回工作目录** | |
| **chdir(path)** | **修改工作目录** | |
| **listdir(path)** | **返回path中的文件名的列表** | |
| **name** | **当前操作系统** | |
| **mkdir(folder)** | **创建 (若已存在则报错)** | **单层目录** |
| **rmdir(folder)** | **删除 (目录非空则报错)** |
| **remove(file)** | **删除文件** | |
| **makedirs(path)** | **创建 (若已存在则报错)** | **多层目录** |
| **removedirs(path)** | **删除 (目录非空则报错)** |
| **rename(m,n)** | **将文件m重命名为n** | |
| **system(cmd)** | **输入命令，输出响应** | |
| **popen(cmd)** | **挂起命令响应，并返回IO** | |
| **environ** | **返回环境变量字典** | |
| **getenv(key, default)** | **访问环境变量** | |
| **cpu\_count()** | **返回CPU数量** | |
| **getpid()** | **获取当前进程ID** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **路径path：** | | | |
| **relpath(path)** | **返回相对路径** | | |
| **basename(path)** | **返回文件名** | | |
| **dirname(path)** | **返回目录路径** | | |
| **split(path)** | **返回**  **元组** | **路径、文件名** | |
| **splitext(path)** | **文件名、拓展名** | |
| **getsize(file)** | **返回**  **file** | **占用空间 (单位：Byte)** | |
| **getatime(file)** | **最近访问时间** | **浮点型秒数** |
| **getctime(file)** | **创建时间** |
| **getmtime(file)** | **最新修改时间** |
| **exists(path)** | **判断**  **path** | **是否存在** | |
| **isabs(path)** | **是否为绝对路径 (包含挂载点)** | |
| **isdir(path)** | **是否存在且是目录** | |
| **isfile(path)** | **是否存在且是文件** | |
| **islink(path)** | **是否存在且是快捷方式** | |
| **ismount(path)** | **是否存在且是挂载点 (如"C:/")** | |
| **samefile(path1, path2)** | **判断两路径是否指向同一文件** | | |
| **join(path1, path2)** | **将多个路径连接成一个路径名** | | |

进程multiprocessing：

|  |  |
| --- | --- |
| threading线程： | |
| active\_count() | 返回当前线程数 |
| enumerate() | 返回已启动的线程列表 |

|  |  |
| --- | --- |
| active\_children() | 返回已启动的进程列表 |
| cpu\_count() | 返回CPU数量 |

|  |  |  |
| --- | --- | --- |
| 进程 (封装exe不可用，不可修改全局变量)： | | |
| current\_process() | | 返回当前进程 |
| Process(target, name, args, kwargs, daemon) | | 实例化进程，import主代码 |
| Thread(target, name, args, kwargs, daemon) | | 实例化线程 |
| 实例属性 | name | 进程名称 |
| 实例方法 | start() | 开始运行进程 |
| join() | 阻塞当前进程 |

|  |  |  |
| --- | --- | --- |
| concurrent.futures线程池： | | |
| ThreadPoolExecutor(max\_workers) | | 实例化线程池 |
| 实例方法 | submit(func, \*args, \*\*kwargs) | 生成后台单线程，返回Future线程 |
| map(func, \*iterables, timeout) | 生成后台多线程，返回映射结果迭代器 |
| Future方法 | result(timeout) | 返回异步进程的结果 |

|  |  |  |
| --- | --- | --- |
| 进程池pool (报错无提示)： | | |
| Pool(num\_processes) | | 实例化进程池 |
| ThreadPool(num\_processes) | | 实例化线程池 |
| 实例方法 | close() | 封闭进程池 |
| join() | 阻塞当前进程 |
| terminate() | 结束进程池 |
| imap / imap\_unordered /  starmap\_async(func, iterable) | 生成后台多进程，返回映射结果迭代器 |
| map(func, iterable) / starmap(func, iterable) | 生成阻塞多进程，返回映射结果列表 |
| apply\_async(func, args, kwds) | 生成后台单进程，返回进程 (报错无反馈) |
| apply(func, args, kwds) | 生成阻塞单进程，返回结果 (冗余) |

|  |  |  |
| --- | --- | --- |
| 管道： | | |
| Pipe(duplex) | | 实例化管道，默认全双工，否则返回 (接收端，发送端) |
| 实例方法 | close() | 关闭管道 |
| send(obj) | 即时发送可pickle对象 |
| recv() | 阻塞接收可pickle对象，返回对象 |
| poll(timeout) | 阻塞等待输入，返回是否可输出 (bool) |

|  |  |  |
| --- | --- | --- |
| 服务器： | | |
| Manager() | | 实例化服务器 |
| 实例方法 | register(typeid, callable) | 注册内置类型 |
| Namespace() | 实例化命名空间，NS.var创建/修改变量 |
| list() | 实例化列表 |
| dict() | 实例化字典 |
| Queue() | 实例化队列 |
| Lock() | 实例化互斥锁 |
| RLock() | 实例化可重入锁 (计数锁) |

|  |  |  |
| --- | --- | --- |
| 队列： | | |
| Queue(maxsize) | | 实例化队列 |
| 实例方法 | empty() | 是否队空 |
| full() | 是否队满 |
| put(obj, block, timeout) | 可pickle对象入队 |
| get(obj, block, timeout) | 可pickle对象出队，返回对象 |
| join() | 等待所有任务完成 |
| taskdone() | 标记任务完成 |

|  |  |  |
| --- | --- | --- |
| 互斥锁 (threading同)： | | |
| Lock() | | 实例化互斥锁 |
| RLock() | | 实例化可重入锁 (计数锁) |
| 实例方法 | acquire(block, timeout) | 申请互斥锁并返回状态 (默认阻塞) |
| release() | 释放互斥锁 |

进程psutil：

|  |  |
| --- | --- |
| cpu\_count(logical) | 返回CPU数量 |
| cpu\_percent() | 返回CPU使用率 |
| virtual\_memory() | 返回虚拟内存信息 |
| swap\_memory() | 返回交换内存信息 |
| process\_iter() | 当前所有进程 |

|  |  |  |
| --- | --- | --- |
| Process(pid=None) | | 实例化进程 |
| 实例属性 | pid | 进程ID |
| ppid() | 父进程ID |
| name() | 进程名称 |
| username() | 用户名称 |
| 实例方法 | is\_running() | 运行状态 |
| open\_files() | 打开的文件列表 |

外设pyautogui：

|  |  |
| --- | --- |
| 屏幕： | |
| size() | 返回屏幕宽高 |
| screenshot(region) | 返回屏幕截图 |
| sleep(seconds) | 休眠 |

|  |  |
| --- | --- |
| 鼠标： | |
| position() | 返回位置 |
| moveTo(x, y, duration) | 移动 |
| click(x, y, clicks, button) | 点击 |
| mouseDown(x, y, button) | 按下 |
| mouseUp(x, y, button) | 释放 |
| dragTo(x, y, duration, buttom) | 拖动 |
| scroll(clicks, x, y) | 滚轮 |

|  |  |
| --- | --- |
| 键盘： | |
| typewrite(message, interval) | 输入 |
| press(keys, presses, interval) | 点击 |
| keyDown(key) | 按下 |
| keyUp(key) | 释放 |

|  |  |
| --- | --- |
| 消息框： | |
| alert(text, title) | 提示框 |
| confirm(text, title, buttons) | 选择框，返回选项 |
| prompt(text, title, default) | 输入框，返回文本 |
| password(text, title, default, mask) | 密码框，返回密码 |

封装pyinstaller：

|  |  |  |
| --- | --- | --- |
| 参数： | | |
| -c | 默认 | 使用控制台，无窗口 |
| -D | 创建包含exe、依赖文件的目录 |
| -w | 使用窗口，无控制台 | |
| -F | 打包生成单exe | |

|  |  |
| --- | --- |
| 语句： | |
| -p D:\Anaconda3 | 添加文件搜索路径 |
| -i ico.ico | 设置exe图标 |
| --hidden-import <mod> | 导入自定义模块 |
| --add-data <src>:<dst> | 添加文件到打包后的路径 |

监测pynput：

鼠标mouse：

todo：Controller, Button

|  |  |  |
| --- | --- | --- |
| Listener(on\_move, on\_click, on\_scroll) | | 初始化监听器 |
| 实例方法 | start() | 开始 |
| join() | 阻塞当前线程 |

剪切pyperclip：

|  |  |
| --- | --- |
| copy(str) | 发送文本到剪切板 |
| paste() | 返回剪切板的内容 |

测试pytest：

|  |  |  |
| --- | --- | --- |
| 运行配置： | | |
| main(args) | | 运行测试用例 |
| 形参 | -v / -s | 启用print |
| -n=2 | 多线程运行 |
| -m "smoke" | 运行选中标记 |
| -k EXP | 运行含关键词的测试用例 |
| --reruns=2 | 失败重运行 |
| --html=report.html | 生成html报告 |
| --alluredir report | 生成allure数据 |
| pytest.ini | [pytest]  addopts = -vs  testpaths = .  python\_files = test\_\*.py  python\_classes = Test\*  python\_functions = test\_\*  markers = smoke | |

|  |  |
| --- | --- |
| 报告allure： | |
| allure generate <tmp> -o <report> | 数据 => html |
| allure serve <report> | 浏览html |

|  |  |
| --- | --- |
| 前后置夹具： | |
| setup / teardown | 类方法夹具 |
| setup\_class / teardown\_class | 实例夹具 |
| @fixture(scope, params, autouse) | 夹具修饰器，存放于conftest.py |
| 作用域scope：'function' / 'class' / 'module' / 'package' |
| 参数params：对应函数的request参数 (.param访问) |
| 自动autouse：False时，作为作用目标的参数 (fun -> yield) |

|  |  |
| --- | --- |
| 标签mark： | |
| flaky(reruns, reruns\_delay) | 失败重运行 |
| skip(reason) | 跳过 |
| skipif(\*boolean, reason) |
| parametrize(argnames, argvalues) | 参数传递 (可堆叠) |

请求requests：

|  |  |
| --- | --- |
| 网址： | |
| /robots.txt | 君子协议 |

|  |  |
| --- | --- |
| Request Headers： | |
| User-Agent | 请求载体的身份标识 |
| Connection | 请求完毕后的连接状态 |
| Proxies | 代理 {'http' / 'https': IP\_str} |

|  |  |
| --- | --- |
| Response Headers： | |
| Status | 响应状态 |
| Content-Type | 响应的数据类型 |

|  |  |  |
| --- | --- | --- |
| 响应： | | |
| get(url, params, headers) | | 返回响应实例 |
| post(url, data, headers) | |
| delete(url, params, headers) | |
| put(url, data, params, headers) | |
| 实例属性 | content | bytes |
| text | str |
| headers | Headers |
| status\_code / reason | status |
| 实例方法 | json() | json |
| raise\_for\_status() | 状态码报错 |

from requests.sessions import Session

|  |  |  |
| --- | --- | --- |
| 会话： | | |
| 实例方法 | get(url, params, headers) | 返回响应实例 |
| post(url, data, headers) |

机器人ros：

## humble：

entry\_points={

'console\_scripts': [

'<node\_name> = <mod>:<func>'

],

### 结点rclpy：

|  |  |
| --- | --- |
| init(args) | 初始化 |
| spin(node\_ptr) | **进入结点的事件循环** |
| shutdown() | **清理程序** |
| ok() | **系统状态** |

|  |  |  |
| --- | --- | --- |
| impl.rcutils\_logger.RcutilsLogger(name) | | **日志系统** |
| **属性** | info / warn / error / fatal(msg) | **输出日志** |

|  |  |  |
| --- | --- | --- |
| node.Node(name) | | 结点 |
| **属性** | get\_logger() **-> impl.rcutils\_logger.RcutilsLogger** | **日志系统** |
| **方法** | create\_timer(sec, callback) | **创建定时执行器** |
| create\_publisher(msgT, topic, qos) | **创建发布器** |
| create\_subscription(msgT, topic, callback, qos) | **创建订阅器, callback(msgT)** |
| create\_client(srvT, service) | **创建客户端** |
| create\_service(srvT, service, callback) | **创建服务端**  **callback(srvT.Request) -> srvT.Response** |

|  |  |  |
| --- | --- | --- |
| Timer | | **定时执行器** |
| **属性** | time\_since\_last\_call() | **当前计时** |
| time\_until\_next\_call() | **触发时间** |
| **方法** | reset() | **重置** |
| cancel() | **取消** |

|  |  |  |
| --- | --- | --- |
| **消息：** | | |
| Publisher | | 发布器 |
| **方法** | publish(msg) | **发布** |
|  | | |
| Subscription | | 订阅器 **(自动)** |

|  |  |  |
| --- | --- | --- |
| **服务：** | | |
| Client | | 客户端 |
| **方法** | call\_async(request) | **发送请求** |
| remove\_pending\_request(future) | **终止请求** |
|  | | |
| Service | | 服务端 |
|  | | |
| Future | | 异步响应 |
| **属性** | result() | 结果 |
| done() | 完成情况 |

### 消息\*\_msgs.msg：

#### 标准std\_msgs.msg：

|  |  |  |
| --- | --- | --- |
| String | | 字符串 |
| **属性** | data | 数据 |

### 服务\*\_srvs.srv：

#### 标准std\_srvs.srv：

|  |  |  |
| --- | --- | --- |
| Empty | | 空服务 |
| **属性** | Request | 请求 |
| Response | 响应 |

## noetic：

### 结点rospy：

浏览器selenium：

网页webdriver：

|  |  |  |
| --- | --- | --- |
| xpath表达式： | | |
| 标签 | \* | 任意 |
| 层级 | /root/tag | 单个 |
| //tag | 多个 |
| 值 | /text() | 内容 |
| /@attrib | 属性 |
| 条件 | tag[idx] | 索引 [1, ] |
| tag[@attrib="val"] | 基于属性 |

|  |  |  |
| --- | --- | --- |
| 浏览器： | | |
| Edge() | | 返回浏览器实例 |
| 实例属性 | title | 返回网页标题 |
| page\_source | 返回网页源代码 |
| 实例方法 | get(url) | 发送get请求 |
| back() | 后退 |
| quit() | 关闭浏览器 |
| find\_element(by, value) /  find\_elements(by, value) | 返回页面元素 |
| by：'id' / 'xpath' / 'link text'  / 'name' / 'tag name' / 'class name' |

|  |  |  |
| --- | --- | --- |
| 元素： | | |
| 实例属性 | tag\_name | 标签 |
| text | 内容 |
| 实例方法 | get\_attribute(attr) | 属性 |
| click() | 点击元素 |
| send\_keys(\*value) | 输入内容 |

安装setuptools：

|  |  |
| --- | --- |
| find\_packages(where, exclude, include) | 搜索路径下的包 |
| setup(name, version, author, description, packages: list,  data\_files=(dst, src: list)) | 提供命令行工具 |

|  |  |
| --- | --- |
| 命令行： | |
| python setup.py sdist | 构建项目分发包 |

套接字socket：

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 (进程ID)： | | | |
| http | 80 | https | 443 |

回环地址：127.0.0.1，localhost，本机IP

|  |  |  |
| --- | --- | --- |
| 套接字 address = (host, port)： | | |
| socket(family, type) | | 实例化套接字 |
| 形参family | AF\_INET | 服务器之间的网络通信 |
| AF\_INET6 | IPv6 |
| AF\_UNIX | Unix系统进程间通信 |
| 形参type | SOCK\_STREAM | TCP：流式socket |
| SOCK\_DGRAM | UDP：数据报式socket |
| 实例方法 | bind(address) | 绑定地址到套接字 |
| listen(backlog) | 指定可挂起的连接数，监听客户端请求 |
| accept() | 等待传入连接，返回 (TCP连接实例，客户端地址) |
| connect(address) | 向服务端请求连接 (状态改变) |
| close() / shutdown(2) | 关闭套接字 |
| send(byte) | 发送TCP数据 |
| recv(nbyte) | 接收TCP数据 |
| sendto(byte, address) | 发送UDP数据 |
| recvfrom(bufsize) | 接收UDP数据，返回 (数据，对方地址) |
| getsockname() | 返回自身地址 |
| getpeername() | 返回对方地址 |
| settimeout(float) | 设置超时时间 |
| setblocking(bool) | 设置阻塞模式 |

命令subprocess：

|  |  |
| --- | --- |
| 响应： | |
| check\_output(str) | 返回cmd响应gbk比特流 |

|  |  |  |
| --- | --- | --- |
| 子进程： | | |
| Popen(str, shell=True) | | 挂起命令响应 |
| 实例方法 | wait() | 等待进程完成 |
| kill() | 结束进程 |
| poll() | 返回进程返回值 |

系统sys：

|  |  |  |
| --- | --- | --- |
| executable | Python执行程序 | |
| argv | Python命令行参数 | |
| path | Python搜索模块路径的列表 | |
| stdout | 指向控制台，可通过赋值修改输出方向 | |
| modules | 当前已导入模块的字典 | |
| version\_info | 当前环境的 Python版本 | |
| getsizeof(obj) | 返回变量占用字节数 | |
| getdefaultencoding() | 返回当前默认字符串编码 | |
| exit(status) | 0 | 通过引发异常退出程序 |
| int | 用作系统退出状态 |
| obj | 被打印和系统退出状态为1 |
| gettrace() | 返回调试器 | |