Used Code

os module

1. `os.path.dirname`: 返回文件路径
2. `os.path.abspath`: 返回绝对路径
3. `os.path.realpath`: 去除符号链接(symbolic link)
4. `__file__`: 返回当前文件路径

sys module

1. `sys.argv`: 命令行参数
2. `sys.path`: 模块搜索路径
3. `sys.exit()`: 退出程序

collections module

numpy module

运算：

- `np.dot(v1, v2)`: 向量点乘
- `np.linalg.norm(v)`: 向量范数
- `np.sum(a, axis=0)`: 按行求和

其它：

1. `np.random.permutation`
2. `.squeeze`: 删除为1的维度
3. `np.where`: 条件筛选，条件赋值
4. `np.logical_and`: 按元素求交

python

1. 标准容器：`list`，`tuple`，`dict`，`set`
    - `list`: 可变，有序，元素可重复
    - `tuple`: 不可变，有序，元素可重复
    - `dict`: key-value对，key不可重复
    - `set`: 不可重复，无序

2. `list` 方法：
    - `append(x)`：将元素 x 添加到列表的末尾
    - `extend(iterable)`：将可迭代对象的元素添加到列表的末尾
    - `insert(i, x)`：在索引 i 处插入元素 x
    - `remove(x)`：删除列表中第一个值为 x 的元素
    - `pop([i])`：删除索引 i 处的元素，并返回它。如果未指定索引，则删除最后一个元素
    - `clear()`：删除列表中的所有元素
    - `index(x[, start[, end]])`：返回列表中第一个出现的元素 x 的索引位置。如果指定了 start 和 end，则在指定的范围内查找
    - `count(x)`：返回列表中值为 x 的元素的数量
3. `dict` 方法：
    - `get(key[, default])`：返回键 key 对应的值。如果键不存在，则返回默认值 default
    - `pop(key[, default])`：删除键 key 对应的值，并返回它。如果键不存在，则返回默认值 default
    - `popitem()`：删除并返回字典中的一个键值对。字典是无序的，所以不保证删除哪个键值对
    - `clear()`：删除字典中的所有键值对
    - `update([other])`：将字典 other 中的键值对添加到字典中。如果键已存在，则覆盖它
    - `keys()`：返回字典中的所有键
    - `values()`：返回字典中的所有值
    - `items()`：返回字典中的所有键值对
4. 可迭代对象：
    - `zip()`：将多个可迭代对象（如列表、元组等）中的元素打包成一个元组迭代器，每个元组包含来自每个可迭代对象对应位置的元素
    - `enumerate()`：返回元素和索引
    - `reversed()`：返回反向迭代器
    - `sorted(iterable, key=None, reverse=False)`：排序函数
        - `iterable`：需要排序的可迭代对象（如列表、元组、字符串等）
        - `key`：一个函数，用于从每个元素中提取用于排序的关键字。如果未指定，则直接比较元素
        - `reverse`：一个布尔值。如果为 True，则按降序排序；如果

为 False（默认值），则按升序排序

5. `list()`，`dict()`
6. `",".join()`：连成字符串
7. 函数变量
8. 字典解析：

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
def my_function(a, b, c):
    print(a, b, c)
my_function(**my_dict)
```

9. 创建字典：

```
dic_values = {'%g' % (line[0]): line[1:] for line in values}
```

10. `pass`：啥也不干
11. 写入 CSV：

```
with open('file.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerows(data)
```

12. 集合操作：
    - 交：`set1 & set2`
    - 并：`set1 | set2`
    - 差：`set1 - set2`
    - 对称差：`set1 ^ set2`

scikit-learn (keras)

1. `precision_recall_curve`：返回不同阈值下的 precision 和 recall 以及阈值
2. `roc_auc_score`
3. `verbose`：控制输出信息

tensorflow

1. `tf.compact`
2. `tf.zeros`

3. `tf.eye`
4. `tf.shape`
5. `tf.linalg`：对角阵 `tf.eye` 是单位阵

快捷键

1. 多行注释：`Shift + P`
2. 多行前移：`Shift + Tab`
3. 反撤销：`Ctrl + Shift + Z`
4. Fn 与功能键（声音等）冲突：`Fn + F12`

terminal 命令

- `history | grep <关键词>`
- 查看帮助：`info ls` 或 `ls --help`
- `cat 文件`：打开文件

git 命令

- `index`：暂存区
- `working tree`：工作区
- vscode 下 git diff 视图：左边是历史，右边是当前的
- `git log`：`git log` 后一直没有光标问题解决：按 `q`
- `git reflog`：获得版本号
- `git status`
- `git diff <文件名>`
- `git add`
- `git commit`
- `git checkout --<文件名>`：工作区回到和版本库相同状态（用于撤销修改，撤销删除）
- `git reset --hard HEAD^`：版本回退 (HEAD 是指向 master 的指针)
- `git reset --hard <版本号>`：到指定版本
- 追加到上一次提交：（待补充）
- 查看分支：`git branch`
- 创建分支：`git branch <name>`
- 切换分支：`git checkout <name>`
- 创建并切换分支：`git checkout -b <name>`

- 合并某分支到当前分支：`git merge <name>`
- 删除分支：`git branch -d <name>`
- `git clone` 不会把 origin 的分支拉下来
- `git remote -v`

问题：git 比对版本的时候会不会丢失之前工作？

Some technique

1. `non_zero_mask = a > 0`：获得为正的 index
2. `perf = -1`：有些函数会返回 -1

VScode 设置：

1. **设置快捷键**：
    - `Ctrl + K Ctrl + S`
    - 打开 `keybinds.json` 文件：

```
{
    "key": "ctrl+1",
    "command": "editor.action.insertSnippet",
    "args": {
        "snippet": "(需要插入代码)"
    },
    "when": "editorTextFocus"
}
```

    - `command` 除了以上还有 `type` 选项，用于直接插入指定的文本，不支持设置光标位置
2. **Markdown 在插入行间公式后不显示的问题**：
    - 参考链接: GitHub Issue #1032
    - 解决方案：禁用 `markdown-math` 扩展，使用 `@builtin markdown` 禁用。

网络相关：

1. `netstat -ano` 是一个 Windows 命令，用于显示网络连接、路由表、网络接口统计信息以及网络协议的详细信息。以下是该命令的输出解释：

- `-a`：显示所有连接和监听端口。

- `-n`：以数字形式显示地址和端口号。
- `-o`：显示与每个连接关联的进程 ID (PID)。

你可以在命令提示符中运行以下命令来查看网络连接：

```
netstat -ano
```

这将输出类似以下内容：

```
Proto  Local Address          Foreign Address        State           PID
TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       1234
TCP    192.168.1.100:139      192.168.1.101:445      ESTABLISHED     5678
...
```

- `Proto`：协议类型（TCP 或 UDP）。
- `Local Address`：本地地址和端口号。
- `Foreign Address`：远程地址和端口号。
- `State`：连接状态（如 LISTENING、ESTABLISHED 等）。
- `PID`：进程 ID，可以在任务管理器中找到对应的进程。

如果你需要进一步分析某个特定的连接，可以使用 `tasklist` 命令结合 PID 来找到对应的进程：

```
tasklist /FI "PID eq 1234"
```

这将显示 PID 为 1234 的进程信息。

2. HTTP与HTTPS区别
   HTTP（HyperText Transfer Protocol）和 HTTPS（HyperText Transfer Protocol Secure）是用于在 Web 浏览器和 Web 服务器之间传输数据的协议。以下是它们的主要区别：
   i. **安全性**：
   - **HTTP**：数据以明文形式传输，容易被中间人攻击（如窃听、篡改）。
   - **HTTPS**：使用 SSL/TLS 协议对数据进行加密，确保数据在传输过程中不被窃听和篡改。

   ii. **端口**：

- **HTTP**：默认使用端口 80。
- **HTTPS**：默认使用端口 443。

iii. **证书**：

- **HTTP**：不需要证书。
- **HTTPS**：需要由受信任的证书颁发机构（CA）颁发的 SSL/TLS 证书。

iv. **性能**：

- **HTTP**：由于没有加密和解密过程，性能稍微高一些。
- **HTTPS**：由于需要加密和解密数据，性能稍微低一些，但现代硬件和优化技术已经使得这种差异非常小。

v. **SEO**：

- **HTTP**：搜索引擎对 HTTP 网站的排名可能较低。
- **HTTPS**：搜索引擎（如 Google）更倾向于对 HTTPS 网站进行更高的排名。

vi. **浏览器支持**：

- **HTTP**：现代浏览器会标记 HTTP 网站为“不安全”。
- **HTTPS**：现代浏览器会显示安全锁图标，表示网站是安全的。

总结：

- **HTTP** 适用于不需要保护敏感数据的普通网站。
- **HTTPS** 适用于需要保护敏感数据（如登录信息、支付信息）的安全网站。

common knowledge

`__init__.py` 作用：

1. 使得文件夹成为一个 package
2. 定义包的初始化行为：`__init__.py` 文件可以包含代码，在导入包时自动执行。你可以在其中初始化包的状态或导入其他模块
3. 控制包的导出内容：可以使用 `__init__.py` 中的 `__all__` 列表来定义当使用 `from package import *` 时导出的模块或函数
4. 简化导入路径：你可以在 `__init__.py` 中导入包内的模块，使外部使用时路径更加简洁

文档里方括号含义：

方括号 `[]` 通常表示可选参数。这意味着在调用函数时，可以选择性地提供这些参数；如果不提供，则使用默认值。

例如：`dict.get(key[, default])`