

Python编程基础

王启睿

内容

- Python中的模块和包
- 一个完整的Python脚本
- 常用模块

内容

- Python中的模块和包
- 一个完整的Python脚本
- 常用模块

Python中的模块和包

- 每一个py文件，或者是一个有__init__.py文件的文件夹，就是一个模块

- 我们使用

```
import <...>
```

来加载一个模块

- 使用 <模块名>.<成员> 的方式使用模块中的成员

Python中的模块和包

- `foo.py`

```
def hello():  
    print('Hello, world')
```

- `main.py`

```
import foo  
  
foo.hello()                # Hello, world
```

Python中的模块和包

- Python有大量自带的包
- 例如，我们希望计算三角函数：

```
import math
```

```
math.cos(math.pi)      # -1.0
```

Python中的模块和包

- Python有大量自带的包
- 例如，我们希望知道时间：

```
import time
```

```
time.ctime()      # 'Tue Jun 28 05:30:28 2016'
```

Python中的模块和包

- Python有大量自带的包
- 例如，我们希望脚本被调用时的命令行参数：

```
import sys
```

```
sys.argv
```


Python中的模块和包

- Python内建包列表：

<https://docs.python.org/library/>

- Python还有大量不是内建的包
- 当你需要某个功能时，善用搜索引擎

比如去年队式平台打包成exe就是通过py2exe这个包实现的

Python中的模块和包 - 安装非内建的包

- 在部分linux发行版上，会有形如

`python3-<package_name>`

的软件包，直接使用`apt-get install`它们即可

注：当系统的包管理软件可以安装某个包时，尽量使用系统的包管理而不是pip，因为前者可以更好地解决依赖关系，特别是在安装重量级的包，如PIL，numpy时

Python中的模块和包 - 安装非内建的包

- 对于任何平台，我们也可以使用pip来安装包

```
pip3 install <package_name>      # 或者
```

```
python3 -m pip install <package_name>
```

- 注意pip本身也是个包，在debian/ubuntu上需要先
apt-get install python3-pip 才能使用它

内容

- Python中的模块和包
- 一个完整的Python脚本
- 常用模块

一个完整的Python脚本

- 当我们在github上面闲逛时，发现别人写的python脚本跟我们写的居然不一样！

一个完整的Python脚本

```
#!/usr/bin/env python3
# -*- coding utf-8 -*-

def main():
    print('Hello, world')

if __name__ == '__main__':
    main()
```

一个完整的Python脚本

```
#!/usr/bin/env python3
```

- *nix系统不是通过扩展名，而是通过文件内容来判断（回忆file指令）
- 脚本文件第一行，以 `#!` 开头的是执行这个脚本的解释器的命令
- 例如shell script往往以 `#!/bin/sh` 或者 `#!/bin/bash` 开头
- 使用 `env` 程序来打开python是因为python的位置不确定

一个完整的Python脚本

```
#!/ -*- coding: <coding> -*-
```

- 告诉解释器/文本编辑器这份脚本存储的编码
- 推荐使用 utf-8 编码
- 如果脚本中出现了非ascii字符，最好注明编码，以保证不出现乱码

一个完整的Python脚本

- `def main():`
- 定义了一个main函数
- 在Python中，main函数和别的函数没有区别
- 只是我们往往会将其作为程序的主函数

一个完整的Python脚本

```
if __name__ == '__main__':  
    main()
```

- 当变量__name__为字符串 '__main__' 的时候，调用
main()
- 一个变量的__name__是它的名字

```
a.__name__          # 'a'
```

一个完整的Python脚本

```
if __name__ == '__main__':  
    main()
```

- 当变量__name__为字符串 '__main__' 的时候，调用main()
- 一个包的__name__是它的名字

```
import math
```

```
math.__name__          # 'math'
```

一个完整的Python脚本

```
if __name__ == '__main__':  
    main()
```

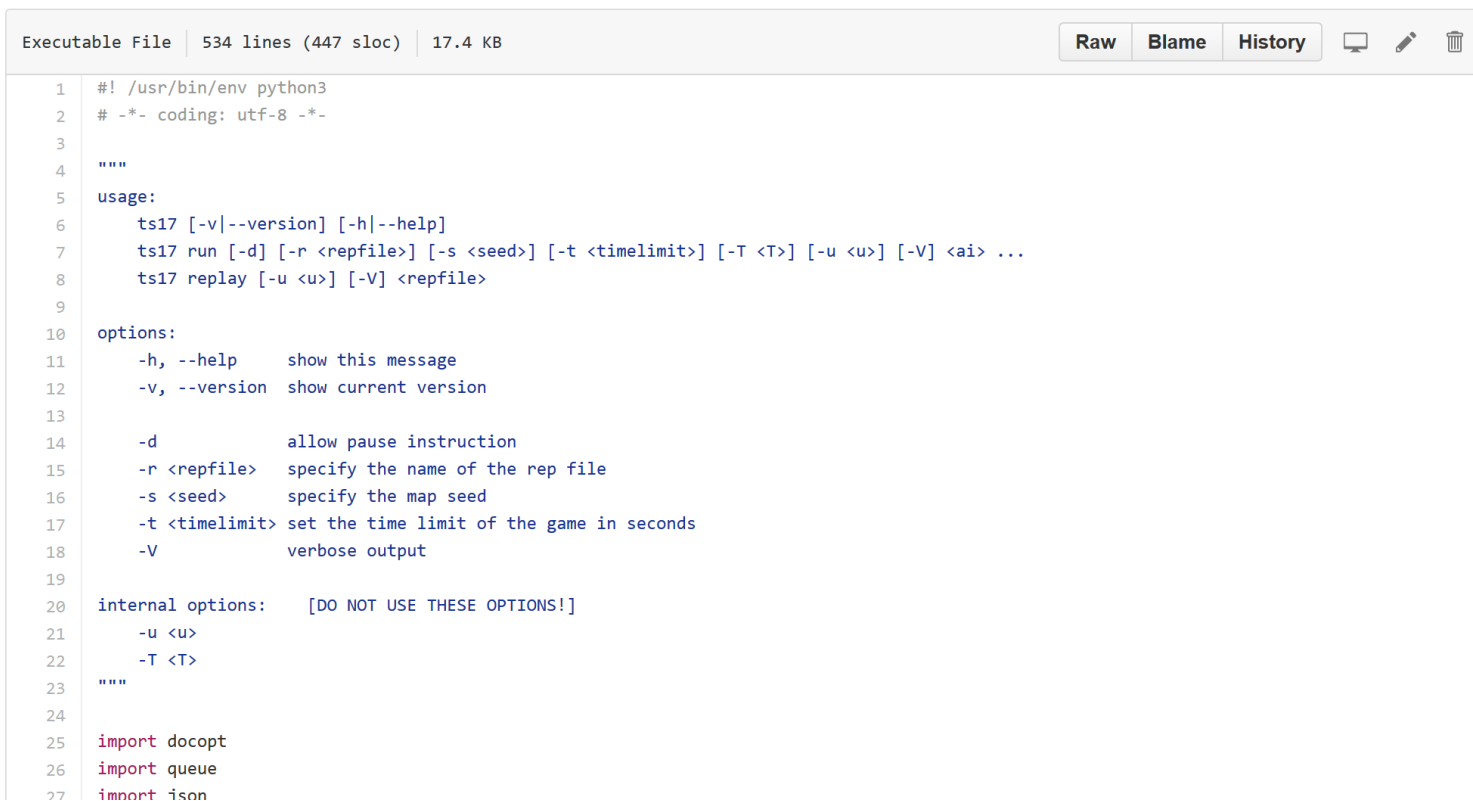
- 当变量__name__为字符串 '__main__' 的时候，说明用户运行的就是这个脚本
- 这样，当用户import这个脚本时，main()就不会被执行了

一个完整的Python脚本

- 在脚本中放置一个多行字符串但不把它赋值给任何变量，那么什么也不会发生
- 我们有时用这种方法来进行多行注释

一个完整的Python脚本

- 有的时候，我们会在一个文件/函数声明/类声明的开头放置一个多行字符串，例如



The screenshot shows a code editor window with a file named 'Executable File' containing 534 lines (447 sloc) and 17.4 KB. The code is a Python script for a game called 'ts17'. It features a multi-line docstring at the top, followed by imports for 'docopt', 'queue', and 'json'. The docstring describes the usage of the script, including options for help, version, pause instruction, rep file, map seed, time limit, and verbose output. The script is written in Python 3 and uses UTF-8 encoding.

```
1  #! /usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  """
5  usage:
6      ts17 [-v|--version] [-h|--help]
7      ts17 run [-d] [-r <repfile>] [-s <seed>] [-t <timelimit>] [-T <T>] [-u <u>] [-V] <ai> ...
8      ts17 replay [-u <u>] [-V] <repfile>
9
10 options:
11     -h, --help      show this message
12     -v, --version   show current version
13
14     -d              allow pause instruction
15     -r <repfile>    specify the name of the rep file
16     -s <seed>       specify the map seed
17     -t <timelimit>  set the time limit of the game in seconds
18     -V              verbose output
19
20 internal options:  [DO NOT USE THESE OPTIONS!]
21     -u <u>
22     -T <T>
23 """
24
25 import docopt
26 import queue
27 import json
```

一个完整的Python脚本

- 这种字符串被称为 `docstring` , 可以通过

`foo.__doc__`

的方式来获得它

- `help(<...>)` 函数会使用 `docstring`

内容

- Python中的模块和包
- 一个完整的Python脚本
- 常用模块

常用模块

- 善用搜索引擎

常用模块

- `os, sys`

系统相关的函数

- `time`

时间相关的函数

- `math`

数学函数

- `multiprocessing`

多进程

- `threading`

多线程

常用模块

- json

json解析

- re

正则表达式

- pickle

序列化

- requests

HTTP请求

- socket

socket通信

常用模块

- ctypes

与c/c++的接口

- PIL (pillow)

图像处理

- numpy, scipy

数值/科学计算

作业3

- 1. 编写一个StopWatch类，具有如下功能：

`sw.start()` # 开始或继续计时

`sw.stop()` # 暂停计时

`sw.current_time()` # 返回一共经过了多少秒，浮点数表示

`sw.reset()` # 时间清零并停止计时

- 2. （选做）编写一个脚本，以另一个程序为命令行参数，统计那个程序从开始运行到结束用了多少秒

作业3

- 提交方式同作业1
- 暑假结束之前提交即可

作业3（大作业，选做，不必提交）

- 用Python编写一个项目，有实际用途即可
 - 聊天工具
 - 小游戏
 - 验证码生成工具
 -
- 正确地组织自己的项目，对于重要的函数、类应有注释说明，正确地处理特殊的输入情况

作业3 – 如果你想不到好的主意

- 参考项目：清华校园网登陆助手

编写一个tunet.py，能够根据输入（例如命令行参数）实现登录、下线、在线状态检查，可选功能：

断线自动重连；托盘图标和右键菜单；流量使用警告；在线设备查看；判断是否在路由之后（防蹭流量）；使用客户端协议而不是web协议；.....

- 基本功能约需100行，可能需要有对HTTP的基本了解

作业3 – 如果你想不到好的主意

- 参考项目：网络学堂助手

编写一个learn.py，能够根据输入（例如命令行参数）实现后台定时登录网络学堂，并列出新公告/新文件提醒，可选功能：

自动下载所有课件；ddl提醒；一键提交作业；对新版网络学堂的支持；.....

- 这个我没写过不知道工程量，同样需要有对HTTP的基本了解

我还想学.....

- 刷Google的习题

<https://developers.google.com/edu/python/>

- 看廖雪峰的教程

<http://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000>

我还想学.....

- 学习用Python作为网站后端

明天同一时间：科协内部培训系列之Django 林子恒

- 用Python刷OJ

<http://www.spoj.com/>

我还想学.....

- 写一些有趣的项目，或者上github看看别人在拿Python做什么

或者去队式17的repo里找找^{bug}灵感？

- 看看官方的文档

<https://docs.python.org/>

我还想学.....

- 看看别的语言吧

比如Haskell如何？

谢谢大家！



作业3 标程

- 不许偷看！

<https://github.com/EESAST/teamstyle17/blob/master/src/main.py#L46>