socket & pwntools & logging & Python 封装

什么是socket

- 是计算机网络通信中的概念，**Socket**是应用程序和网络协议栈之间的接口
- 一个socket包含ip地址＋端口号
- 每个socket都与特定的协议关联
- ▼ 主要的两种socket类型

  ▼ TCP**(Transmission Control Protocol)**协议是一种面向连接的、可靠的、基于字节流的协议。通过 TCP协议，可以在两端建立**可靠的**双向通信连接，确保数据按顺序传输且无丢失。

  ▼ UDP **(User Datagram Protocol)** 是一种无连接、不保证数据包顺序和可靠性的协议。它发送的是独立的消息包（datagram），适用于对实时性要求较高但对可靠性要求不高的场景。
- 工作流程：1.服务器端：创建socket → 绑定地址和端口 → 监听连接 → 接受链接 → 数据通信 → 关闭连接

  2.客户端：创建socket — > 连接服务器 → 数据通信 → 关闭连接

python部署socket服务

- 部署socket服务，即让服务端接收客户端的连接，并处理通信请求。可以使用python中的cocket模块
- 步骤（同上服务端工作流程）
- 常用socket method：

socket( ): 建立一个新socket【family：地址】【type：传输协议】

bind((host,port)):绑定服务器与端口

listen( ):使服务器进入监听状态

accept( ):接受一个客户端连接，返回一个新的 socket 对象和客户端的地址。

connect((host,port)):客户端使用该方法连接服务器。

send( ):发送数据

recv( ):接收数据

close( ):关闭socket连接

pwntools

Pwntools 是一个 CTF 框架和漏洞开发库。它用 Python 编写，旨在快速进行原型设计和开发，并旨在使漏洞编写尽可能简单

调用python pwn库连接远程服务

Server:

```
from pwn import*

//使用listen创建一个监听器，等待客户端连接
server = listen(//port)
client = server.wait_for_connection()

//消息收发
client.send()
response = client.recv()

//完成通信后关闭连接
client.close()
```

Client：

```
from pwn import*

//使用remote连接到服务器的地址和端口
```

```
client = remote('localhost', //port)

//消息收发
client.send()
response = client.recv()

//完成通信后关闭连接
client.close()
```

logging

logging是Python标准库里用于记录日志的模块

主要作用

- 记录程序运行：记录代码的执行过程、
- debug：快速定位bug的位置

官方文档的惯例用法示例

```
# myapp.py
import logging
import mylib
logger = logging.getLogger(__name__)

def main():
    logging.basicConfig(filename='myapp.log', level=logging.I
    logger.info('Started')
    mylib.do_something()
    logger.info('Finished')

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

```
# mylib.py
import logging
logger = logging.getLogger(__name__)
```

```
def do_something():
    logger.info('Doing something')
```

运行*myapp.py* ，在 *myapp.log* 中可以看到

```
INFO:__main__:Started
INFO:mylib:Doing something
INFO:__main__:Finished
```

logging支持多种日志级别（DEBUG、INFO、WARNING、ERROR、CRITICAL）

```
import logging

# 配置日志
logging.basicConfig(level=logging.INFO, format='%(asctime)s -

# 记录不同级别的日志
logging.debug('这是调试信息')
logging.info('这是一般信息')
logging.warning('这是警告信息')
logging.error('这是错误信息')
logging.critical('这是严重错误信息')
```

Python封装

居然———在同一个目录下的文件里的函数可以通过

```
from file_name  import*
```

调用（我好像学了个假python

剩下的关于Python封装和Python面向对象还没有了解太多，之后再学一点再补充笔记吧