0x01,验证码识别
0x02,一个关于http协议中的小特性

0x01,验证码识别:

1,图灵测试

- 简单来说，区别人和机器
- 指测试者与被测试者（一个人和一台机器）隔开的情况下，通过一些装置（如键盘）向被测试者随意提问。
- 进行多次测试后，如果有超过30%的测试者不能确定出被测试者是人还是机器，那么这台机器就通过了测试，并被认为具有人类智能。

2,验证码的出现

- 早期是没有验证码的
- 暴力破解的出现，需要区别人和机器
- 反爬虫
- 验证码的诞生
- 一步一步变的十分复杂

3,错误的验证机制

bad case :

```
if ($_SESSION['code']==$_POST['code']){
    echo 'pass validation';
}
```

4,验证码识别技术

- 验证码图像识别技术主要是操作图片内的像素点，通过对图片的像素点进行一系列的操作，最后输出验证码图像内的每个字符的文本矩阵。
    - 读取图片
    - 图片降噪
    - 图片切割
    - 图像文本输出
- 字符识别,字符识别主要以机器学习的分类算法来完成，字符识别的算法比如有KNN（K邻近算法）和SVM（支持向量机算法）。
    - 获取字符矩阵
    - 矩阵进入分类算法
    - 输出结果

5,一些开源识别的库或框架

- tesseract.js 直接在前端完成识别验证
- pytesseract 在服务端完成的识别验证
- demo演示

0x02,一个关于http协议中的小特性

1, RFC文档定义 http range

- Range 的规范定义如下：
    - 所有HTTP可传输的资源，通常是以一系列字节的形式传输的
    - 资源的前500个字节： bytes=0-499
    - 第二段500字节： bytes=500-999
    - 如果最后一个字节位置被指定但是小于首字节位置，那么这个字节范围是无效的
    - 请求最后500个字节 (9500-9999)： bytes=-500 OR bytes=9500-
    - 只请求第一个和最后一个字节： bytes=0-0,-1

2, 可以实现什么

- 断点上传
- 断点下载
- 中断之后可以在中断位置继续下载/上传，而不必重新开始的原因就是利用了支持range的特性
- 下次继续下载/上传读取记录的偏移位置即可支持断点下载/上传

3，基于python的多线程文件下载器实现

- 理论在大型文件下载，带宽充足的情况下，可增加数十倍下载速度
- 原理是多线程对目标文件分块下载
    - 发送head请求获取目标文件总大小，以及当前是否支持分块下载(详情:http协议header头range及response的content-range)
    - 下载前创建一个和要下载文件一样大小的文件
    - 根据之前获得的文件大小分块多线程,各个线程下载不同的数据块
- 小型文件可能看不出加速效果，在大型文件上就会拉大差距
- 多线程下载维护临时文件偏移位置比维护单一进程的临时文件偏移位置要复杂的多
- demo 演示
- py nice_download.py -u http://mirrors.aliyun.com/centos/7.4.1708/os/x86_64/isolinux/initrd.img -t 100 -p yes