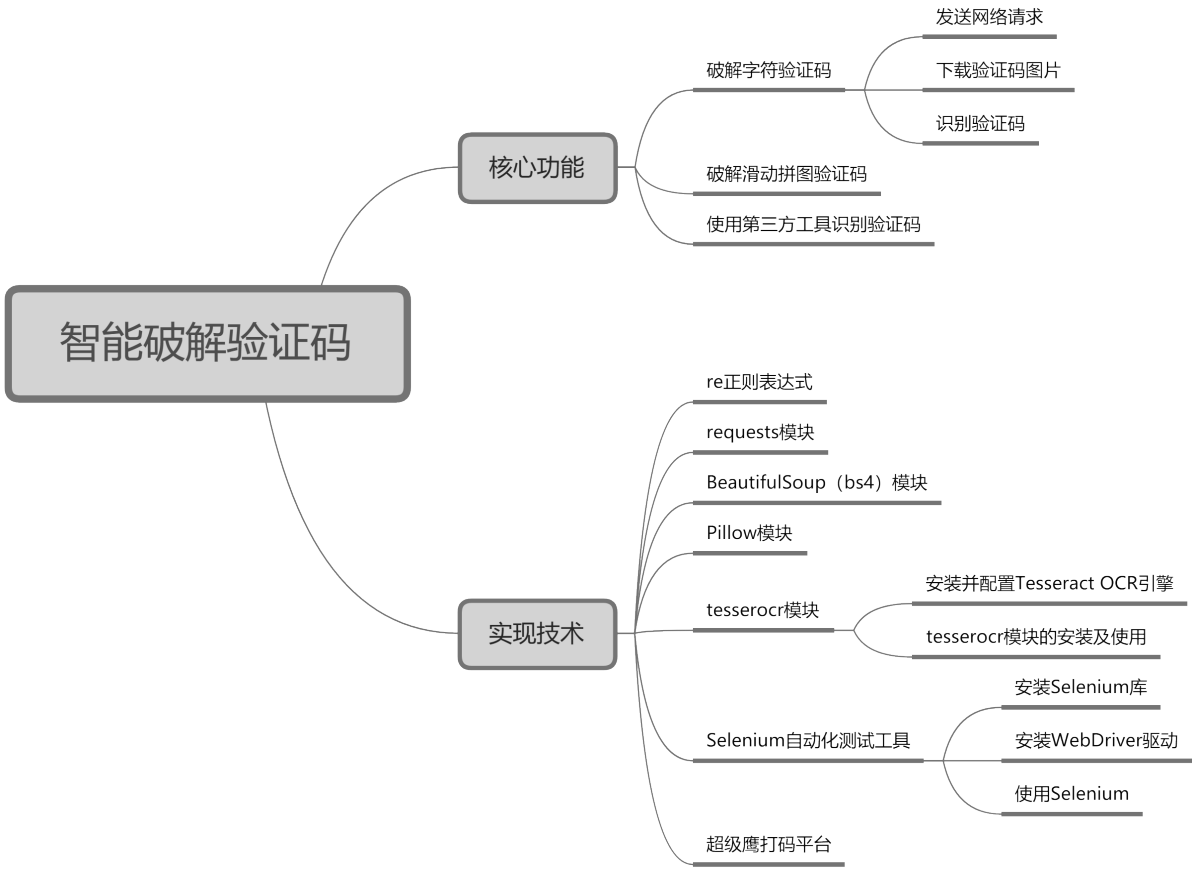
第 1 章

智能破解验证码

——re 正则表达式 + requests + BeautifulSoup（bs4）+ Pillow + tesserocr + selenium

验证码是一种用于确保用户是真人而非自动化程序的技术，常用于防止垃圾邮件发送、暴力破解密码、非法注册账号等恶意行为。在现实生活中，常见的验证码有字符验证码、滑动拼图验证码等。尽管验证码提高了安全性，但也带来了用户体验的问题。本章将使用 Python 网络爬虫技术结合 tesserocr、selenium 等模块开发一个能够自动识别常见验证码的项目。

本项目的核心功能及实现技术如下：



1.1 开 发 背 景

随着互联网技术的飞速发展，信息安全已成为一个重要议题。验证码作为保护网站安全的一种手段，广泛应用于用户登录、注册等场景。然而，随着 AI 技术的进步，传统的验证码机制面临着被自动识别和破解

的风险，这促使了对更高级别验证码识别技术的需求。本项目旨在利用 Python 网络爬虫技术开发一个能够高效、准确地识别各类验证码的应用程序，以更好地研究验证码技术，并推动其升级。

本项目的实现目标如下：

☑ 简洁明了的操作界面，使用户能够轻松体验验证码自动识别功能；
☑ 能够自动识别常见的验证码，如字符验证码、滑动拼图验证码等；
☑ 能够使用常见的第三方打码平台识别验证码。

说明

本项目仅用于学习，严禁恶意爬取、滥用资源等行为，以免侵犯他人权益或引发法律纠纷。

1.2 系 统 设 计

1.2.1 开发环境

本项目的开发及运行环境如下：

☑ 操作系统：推荐 Windows 10、Windows 11 或更高版本。
☑ 开发工具：PyCharm 2024（向下兼容）。
☑ 开发语言：Python 3.12。
☑ Python 内置模块：re、urllib。
☑ 第三方模块：requests、BeautifulSoup（bs4）、Pillow、fake_useragent、tesserocr、selenium。
☑ 第三方工具：超级鹰打码识别平台。

1.2.2 业务流程

本项目的实现流程比较简单，使用 3 种技术实现验证码的识别或破解功能。其中：在识别普通的字符验证码时，需要使用 OCR 技术，因此首先需要搭建 OCR 环境，然后下载相应的验证码图片进行识别；在破解滑动拼图验证码时，需要借助 Selenium 自动化测试工具，因此首先需要安装相应的 Selenium 库和 WebDriver 驱动，然后编写代码进行破解；最后，本项目还借助第三方的打码平台识别验证码，这需要注册相应打码平台的账号，下载相应的示例代码，并将自己注册的用户名、密码，以及要识别的验证码图片传入相应的函数中，以实现识别验证码的功能。

本项目的业务流程如图 1.1 所示。

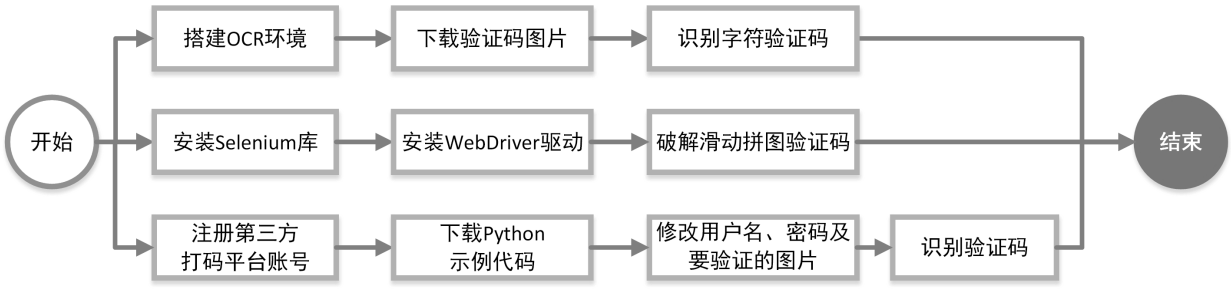


图 1.1 智能破解验证码业务流程

1.2.3 功能结构

本项目的功能结构已经在章首页中给出，具体实现功能如下：

☑ 使用 OCR 技术破解字符验证码；
☑ 使用 Selenium 自动化测试工具破解滑动拼图验证码；
☑ 使用第三方打码平台识别验证码。

1.3 技 术 准 备

1.3.1 技术概览

☑ re 正则表达式：正则表达式（regular expression，常简写为 regex 或者 re），又称规则表达式，它通常被用于检索和替换符合某些规则的文本。在 Python 中，可以使用 re 模块实现正则表达式操作。具体实现时，可以使用 re 模块提供的方法（如 search()、match()、findall()等）进行字符串处理，也可以先使用 re 模块的 compile()方法将模式字符串转换为正则表达式对象，再使用该正则表达式对象的相关方法来操作字符串。例如，本项目在破解滑动拼图验证码时使用正则表达式对页面中的标签进行匹配，代码如下：

```
# 获取空缺滑块样式
verified_style = driver.find_element(By.XPATH,
    '/html/body/div/div[2]/div[1]/div[2]').get_attribute('style')
# 获取空缺滑块 left 值
verified_left =float(re.findall('left: (.*?)px;',verified_style)[0])
# 获取图形滑块 left 值
verify_left =float(re.findall('left: (.*?)px;',verify_style)[0])
```

☑ requests 模块：requests 是 Python 中用于实现 HTTP 请求的第三方模块。在使用该模块之前，需要通过执行 pip install requests 命令进行安装。例如，本项目在破解字符验证码时，使用 requests 模块的 get()函数向指定网址发送网络请求，代码如下：

```
header = {'User-Agent':UserAgent().random}                  # 创建随机请求头
url = 'http://spider.mingribook.com/spider/word/'           # 网页请求地址
# 发送网络请求
response = requests.get(url,header)
```

☑ BeautifulSoup（bs4）：BeautifulSoup（bs4）是一个用于从 HTML 和 XML 文件中提取数据的 Python 库。BeautifulSoup（bs4）提供了一些简单的函数，用于处理导航、搜索、修改分析树等功能。例如，本项目使用 BeautifulSoup（bs4）模块对验证码的 HTML 页面进行解析，以获取验证码图片，代码如下：

```
from bs4 import BeautifulSoup                               # 导入模块，用于解析 HTML
# 省略部分代码……
html = BeautifulSoup(response.text,"html.parser")           # 解析 HTML
src = html.find('img').get('src')
img_url = url+src                                           # 组合验证码图片请求地址
urllib.request.urlretrieve(img_url,'code.png')              # 下载并设置图片名称
```

有关 re 正则表达式、requests 模块和 BeautifulSoup（bs4）模块的知识在《Python 从入门到精通（第 3 版）》中有详细讲解，读者如果对这些知识不太熟悉，可以参考该书的相关章节。接下来，我们将对实现本项目时

使用的其他主要技术点进行必要的介绍，包括 Pillow 模块的使用、tesserocr 模块的使用、Selenium 自动化测试工具的使用，以确保读者可以顺利完成本项目。

1.3.2 Pillow 模块的使用

Pillow 是一个开源的 Python 图像处理库，是 Python Imaging Library（PIL）的一个更现代且持续得到维护的分支版本。由于 PIL 在较早版本的 Python 中已停止更新，Pillow 作为其替代品而出现，并提供了对 Python 3 的全面支持。使用该模块之前，需要使用 pip install Pillow 命令进行安装，安装完成后即可使用该模块。

例如，下面的代码使用 Pillow 模块中的 Image 类的 open()方法和 show()方法分别打开和显示指定图片的原图。其中，Image 类主要用于图像的基本操作，如打开、加载、保存、显示图像，以及进行图像模式转换、尺寸调整（缩放）、旋转、裁剪等。使用前，需要导入 Image 类，代码如下：

```
from PIL import Image
```

然后使用 open()方法创建一个图像对象，接下来就可以使用该对象调用其方法实现相应的功能了。例如，下面的代码用于打开并显示一张指定路径下的图片：

```
img = Image.open(tppath + sender.text())
img.show()
```

1.3.3 tesserocr 模块的使用

tesserocr 是一个第三方的 Python 模块，用于调用 Tesseract OCR（optical character recognition，光学字符识别）引擎。Tesseract 是由 Google 维护的一个开源 OCR 引擎，能够识别多种语言的文字，并支持多种输出格式。tesserocr 提供了一种简单的方式来与 Tesseract 进行交互，特别适合于处理图像中的文字提取任务。在使用 tesserocr 模块前，需要安装 Tesseract OCR 引擎并配置环境变量，然后安装 tesserocr 模块，具体步骤如下：

（1）打开 Tesseract OCR 引擎的开源下载地址（https://github.com/UB-Mannheim/tesseract/wiki），选择与自己操作系统匹配的版本（这里以 Windows 64 位操作系统为例），如图 1.2 所示。

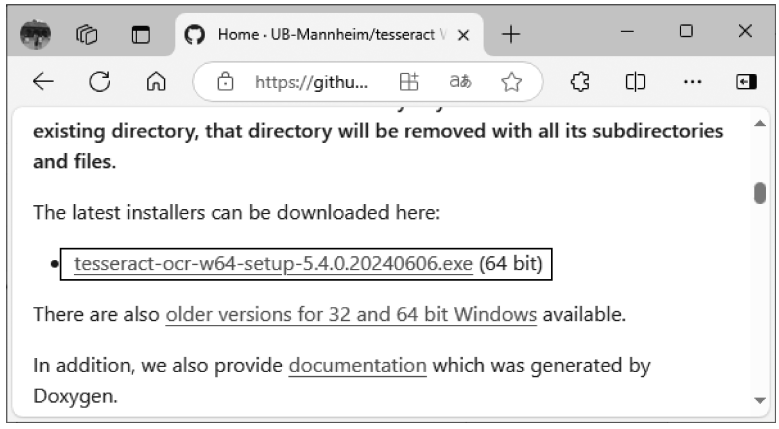


图 1.2　下载 Tesseract OCR 引擎安装文件

（2）Tesseract OCR 引擎安装文件下载完成后，直接双击并按照向导进行安装即可。

（3）配置 Tesseract OCR 引擎环境变量。打开计算机系统的“环境变量”对话框，在“系统变量”的 Path 变量中添加 Tesseract OCR 引擎的默认安装位置，如图 1.3 所示。

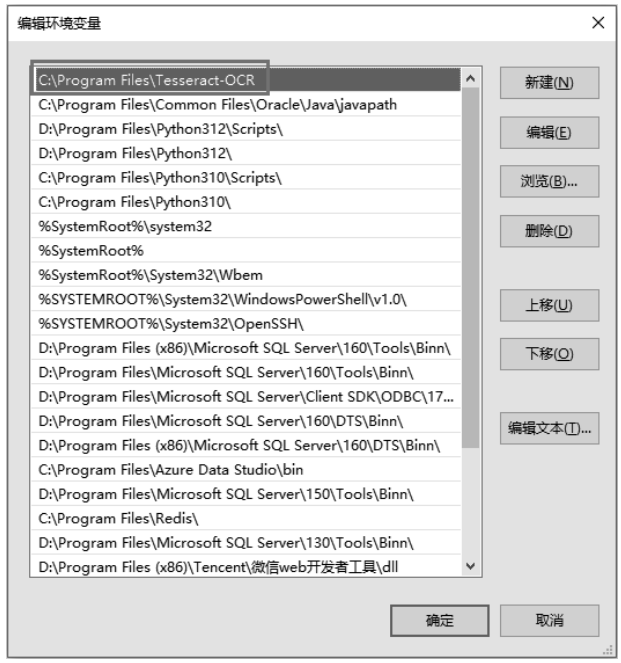


图 1.3　配置 Tesseract OCR 引擎环境变量

（4）在“系统变量”中添加一个新的变量，命名为 TESSDATA_PREFIX，该变量的路径指向 Tesseract OCR 引擎安装目录下的 tessdata 文件夹，如图 1.4 所示。

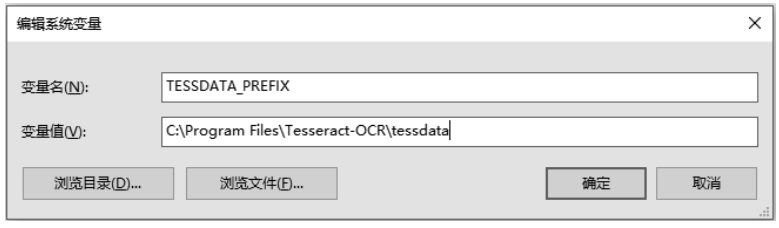


图 1.4　新建 TESSDATA_PREFIX 环境变量

（5）将 Tesseract OCR 安装路径下的 tessdata 文件夹复制到本地的 Python 安装目录中，如图 1.5 所示。这样做主要是为了解决文字识别问题。

图 1.5　将 tessdata 文件夹复制到 Python 安装目录下

（6）下载 tesserocr 模块的离线安装文件。打开 tesserocr 模块离线安装文件的开源下载地址 https://github.com/simonflueckiger/tesserocr-windows_build/releases，选择与自己的 Python 版本相匹配的项进行下载，如图 1.6 所示。

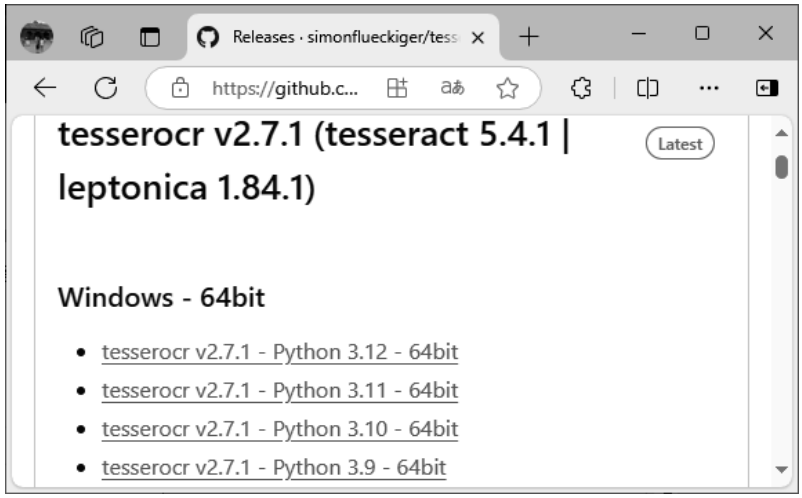


图 1.6　下载 tesserocr 模块的离线安装文件

（7）使用 pip install 命令安装 tesserocr 模块，命令如下：

```
pip install tesserocr 模块离线安装文件所在位置（如：C:\tesserocr-2.7.1-cp312-cp312-win_amd64.whl）
```

完成以上步骤之后，就可以开始使用 tesserocr 模块了，例如，下面的代码展示了如何使用 tesserocr 模块从图像中提取文字，代码如下：

```
# 导入必要的库
import tesserocr
from PIL import Image
# 加载图像
image = Image.open('path/to/image.png')
# 执行 OCR
text = tesserocr.image_to_text(image)
print(text)
```

1.3.4　Selenium 自动化测试工具的使用

Selenium 是一个强大的自动化测试工具，主要用于 Web 应用程序的测试。它支持多种编程语言，包括 Python、Java、C#等。借助 Selenium，开发人员可以模拟用户的浏览器操作，如单击按钮、填写表单、导航页面等。这对于网页自动化测试、数据抓取和自动化任务非常有用。使用 Selenium 自动化测试工具的基本步骤如下：

（1）安装 Selenium 库，命令如下：

```
pip install selenium
```

（2）安装 WebDriver 驱动。Selenium 需要一个浏览器驱动程序（WebDriver）来控制浏览器。常用的浏览器及其对应的 WebDriver 驱动如下：

☑ Chrome：ChromeDriver。

☑ Firefox：GeckoDriver。

☑ Edge：EdgeDriver。

☑ Safari：SafariDriver（内置）。

例如，若要在 Chrome 浏览器中安装 WebDriver 驱动，则需要访问 ChromeDriver 的下载页面 https://sites.google.com/a/chromium.org/chromedriver/downloads?spm=5176.28103460.0.0.77d75d27lMAa7x，下载与本机的 Chrome 浏览器版本匹配的 ChromeDriver 并进行安装。

说明

ChromeDriver 的下载地址是 Google 官方提供的。在访问上述网址时，如果遇到无法打开的情况，读者可以通过正规渠道购买代理 IP 并进行相应设置后尝试重新访问。

例如，使用 Selenium 打开一个网页，然后在一个搜索框中输入文本，并提交搜索，代码如下：

```
# 导入必要的库
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time
# 初始化 WebDriver
driver_path = 'path/to/chromedriver'
driver = webdriver.Chrome(executable_path=driver_path)
# 打开网页
driver.get('https://www.example.com')
# 找到搜索框元素
search_box = driver.find_element(By.NAME, 'q')
# 输入搜索关键词
search_box.send_keys('Selenium')
# 模拟按下 Enter 键
search_box.send_keys(Keys.RETURN)
# 等待页面加载完成
time.sleep(3)
# 获取页面标题
print(driver.title)
# 获取页面源代码
print(driver.page_source)
# 关闭浏览器
driver.quit()
```

1.4 功 能 设 计

1.4.1 破解字符验证码

字符验证码是一种包含数字、字母或者掺杂斑点与混淆曲线的图片验证码。破解此类验证码时，首先需要定位验证码图片在 HTML 代码中的位置，并下载该图片。然后，使用 OCR 技术进行验证码的识别。

下面以 http://spider.mingribook.com/spider/word/地址为例，讲解如何破解字符验证码，步骤如下：

（1）使用浏览器打开测试网页 http://spider.mingribook.com/spider/word/，如图 1.7 所示。

（2）按 F12 键打开浏览器的开发者工具，然后在 HTML 代码中获取验证码图片所在的位置，如图 1.8 所示。

图 1.7　打开字符验证码所在的网页

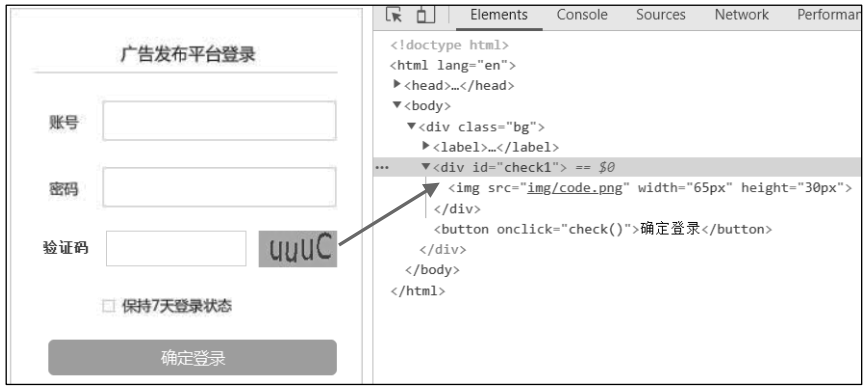


图 1.8　获取验证码图片在 HTML 代码中的位置

（3）向目标网页发送网络请求，并在返回的 HTML 代码中获取验证码图片的下载地址，然后将验证码图片下载到本地。代码如下：

```
import requests                                  # 导入网络请求模块
import urllib.request                            # 导入 urllib.request 模块
from fake_useragent import UserAgent             # 导入随机请求头
from bs4 import BeautifulSoup                    # 导入解析 HTML 的模块
header = {'User-Agent':UserAgent().random}       # 创建随机请求头
url = 'http://spider.mingribook.com/spider/word/'  # 网页请求地址
# 发送网络请求
response = requests.get(url,header)
response.encoding='utf-8'                        # 设置编码方式
html = BeautifulSoup(response.text,"html.parser")  # 解析 HTML
src = html.find('img').get('src')
img_url = url+src                                # 组合验证码图片请求地址
urllib.request.urlretrieve(img_url,'code.png')   # 下载验证码图片
```

上述代码运行后，保存到本地的验证码图片如图 1.9 所示。

图 1.9　验证码图片

（4）使用 Pillow 模块和 tesserocr 模块对下载的验证码图片进行识别。具体实现时，首先通过 Image.open() 方法打开验证码图片，然后通过 tesserocr.image_to_text()方法识别图片中的验证码信息并输出。代码如下：

```
import tesserocr                                 # 导入 tesserocr 模块
from PIL import Image                            # 导入图像处理模块
img =Image.open('code.png')                      # 打开验证码图片
code = tesserocr.image_to_text(img)              # 将图片中的验证码转换为文本
print('验证码为：',code)
```

程序运行结果如下：

```
验证码为：uuuc
```

这里需要注意的是，上面识别的验证码图片只是最常见的一种类型，其中没有任何干扰线。然而，在实际浏览网站时，为了增强安全性，网站通常都会在验证码中增加多条干扰线，如图 1.10 所示。

图 1.10　带有干扰线的验证码

在遇到如图 1.10 所示的验证码图片时，我们如果继续使用上述步骤（4）中的代码进行识别，就可能会得到以下结果：

```
验证码为：YSGN.
```

或者

```
验证码为：
```

观察上面的结果，我们发现直接使用 OCR 技术识别带有干扰线的验证码图片时，结果可能会多出一些字符，或者完全无法识别。那么，对于这种验证码图片，我们应如何进行识别呢？这时，一个有效的方法是先对要识别的验证码图片进行灰度处理，然后进行二值化处理，最后使用 OCR 技术进行识别。优化后的代码如下：

```
import tesserocr                              # 导入 tesserocr 模块
from PIL import Image                         # 导入图像处理模块
img =Image.open('code2.jpg')                  # 打开验证码图片
img = img.convert('L')                        # 将彩色图片转换为灰度图片
t = 155                                       # 设置阈值
table = []                                    # 二值化数据的列表
for i in range(256):                          # 循环遍历
    if i <t:
        table.append(0)
    else:
        table.append(1)
img = img.point(table,'1')                    # 将图片进行二值化处理
img.show()                                    # 显示处理后的图片
code = tesserocr.image_to_text(img)           # 将图片中的验证码转换为文本
print('验证码为：',code)                       # 打印验证码
```

运行上述代码后，图 1.10 所示的验证码图片效果会变为如图 1.11 所示的效果。

图 1.11　经过灰度处理和二值化处理后的验证码图片

这时，程序即可正常地对图片中的验证码进行识别，结果如下：

```
验证码为：YSGN
```

1.4.2　破解滑动拼图验证码

滑动拼图验证码是许多网站和 APP 中经常使用的一种图形验证码形式，要求用户将拼图块拖动到正确的位置以完成验证过程。本项目将使用 Python 结合 Selenium 自动化测试工具来破解滑动拼图验证码，测试地址为 http://spider.mingribook.com/spider/jigsaw/。具体实现步骤如下：

（1）在浏览器中打开滑动拼图验证码的测试网页地址 http://spider.mingribook.com/spider/jigsaw/，如图 1.12 所示。

（2）按 F12 键打开浏览器的开发者工具，单击按钮滑块，然后在 HTML 代码中依次定位“按钮滑块”“图形滑块”以及“空缺滑块”对应的 HTML 标签位置，如图 1.13 所示。

图 1.12　滑动拼图验证码

图 1.13　确定滑动拼图验证码的 HTML 代码位置

（3）拖动按钮滑块，完成滑动拼图验证码的校验，此时将显示如图 1.14 所示的 HTML 代码。

图 1.14　验证成功后 HTML 代码变化

对比图 1.13 与图 1.14，可以看出：按钮滑块在默认情况下的位置为 left:0px，图形滑块在默认情况下的位置为 left:10px；而在验证成功后，按钮滑块的 left 值变为 174px，图形滑块的 left 值变为 184px。由此可以总结出整个验证过程中的滑块位置变化情况，如图 1.15 所示。

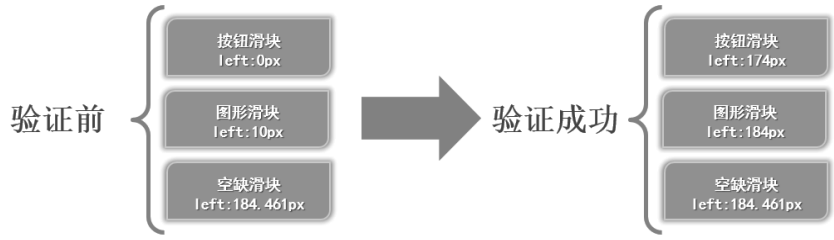


图 1.15　验证过程中的滑块位置变化情况

（4）确定滑动位置变化情况后，在 Python 代码中通过 Selenium 库调用 Selenium 自动化测试工具，完成模拟滑块滑动的操作。代码如下：

```
from selenium import webdriver                                  # 导入 webdriver
from selenium.webdriver.common.by import By
import re                                                       # 导入正则表达式模块

driver = webdriver.Chrome()                                     # 初始化 Chrome WebDriver
driver.get('http://spider.mingribook.com/spider/jigsaw/')       # 打开网页
swiper = driver.find_element(By.XPATH,
    '/html/body/div/div[2]/div[2]/span[1]')                     # 获取按钮滑块
action = webdriver.ActionChains(driver)                         # 创建动作链
action.click_and_hold(swiper).perform()                         # 单击并保持按住滑块
# 滑动 0 距离，保持按住状态，以便获取图形滑块的 left 值
action.move_by_offset(0,0).perform()
# 获取图形滑块的样式
verify_style = driver.find_element(By.XPATH, '/html/body/div/div[2]/div[1]/div[1]').get_attribute('style')
# 获取空缺滑块的样式
verified_style = driver.find_element(By.XPATH, '/html/body/div/div[2]/div[1]/div[2]').get_attribute('style')
# 获取空缺滑块的 left 值
verified_left =float(re.findall('left: (.*?)px;',verified_style)[0])
# 获取图形滑块的 left 值
verify_left =float(re.findall('left: (.*?)px;',verify_style)[0])
action.move_by_offset(verified_left-verify_left,0)              # 滑动指定距离
action.release().perform()                                      # 松开鼠标
```

上述代码运行后，即可自动对 http://spider.mingribook.com/spider/jigsaw/测试网页中的滑动拼图验证码进行操作，并弹出如图 1.16 所示的验证成功提示框，从而实现智能破解滑动拼图验证码的效果。

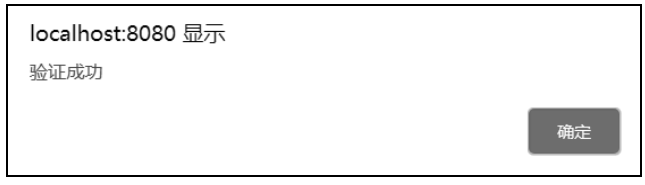


图 1.16　验证成功提示框

1.4.3　第三方平台识别验证码

在 Python 中，除了使用 OCR 技术和 Selenium 自动化测试工具破解验证码，还可以借助第三方平台进行验证码识别。第三方平台通常会提供完善的 API 接口，开发人员可以根据平台文档快速完成开发需求。常

见的第三方验证码识别平台主要分为两类：打码平台和 AI 开发者平台。其中：打码平台由平台官方完成验证码识别，并在较短时间内返回结果，如超级鹰平台；AI 开发者平台则主要采用人工智能技术进行验证码识别，例如百度 AI 以及其他 AI 平台等。

本项目以超级鹰打码平台为例，讲解识别验证码的具体过程。步骤如下：

（1）在浏览器中打开超级鹰打码平台首页（http://www.chaojiying.com/），单击“用户注册”按钮，如图 1.17 所示。

图 1.17　超级鹰打码平台首页

（2）在打开的用户中心页面中填写注册账号的基本信息，并单击“同意以下协议并注册”按钮，如图 1.18 所示。

图 1.18　填写注册账号的基本信息

说明

账号注册完成后，可以联系平台的客服人员，申请免费测试的题分。

（3）账号注册完成后，在超级鹰打码平台网站的顶部导航栏中单击“开发文档”，打开“开发文档”页面，在该页面的“常用开发语言示例下载”中选择 python，如图 1.19 所示。

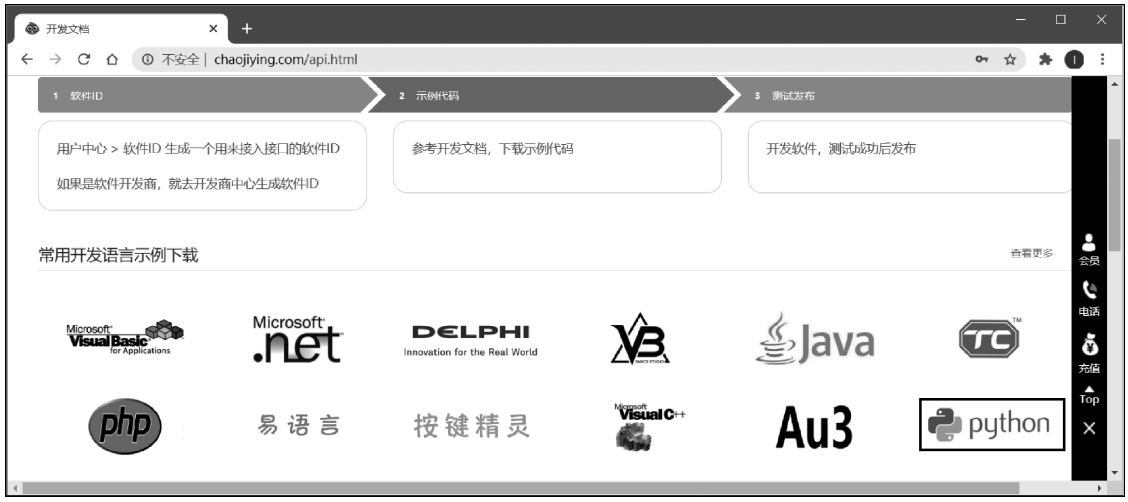


图 1.19　选择开发语言示例

（4）在 Python 语言 Demo 下载页面中可以查看相关的注意事项，单击“单击这里下载”超链接，即可下载使用 Python 结合超级鹰打码平台接口实现的识别验证码示例代码，如图 1.20 所示。

图 1.20　下载示例代码

（5）使用 PyCharm 打开下载的示例代码。该示例代码默认已经封装了识别验证码的功能，具体代码如下：

```
#!/usr/bin/env python
# coding:utf-8
import requests                                          # 导入网络请求模块
from hashlib import md5                                  # 加密

class Chaojiying_Client(object):

    def __init__(self, username, password, soft_id):
        self.username = username                         # 自己注册的账号
        password =  password.encode('utf8')              # 自己注册的密码
        self.password = md5(password).hexdigest()
        self.soft_id = soft_id                           # 软件 ID
        self.base_params = {                             # 组合表单数据
            'user': self.username,
            'pass2': self.password,
            'softid': self.soft_id,
        }
        self.headers = {                                 # 请求头信息
            'Connection': 'Keep-Alive',
            'User-Agent': 'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)',
        }

    def PostPic(self, im, codetype):
        """
        im: 图片字节
        codetype: 题目类型 参考 http://www.chaojiying.com/price.html
        """
        params = {
            'codetype': codetype,
        }
        params.update(self.base_params)                  # 更新表单参数
        files = {'userfile': ('ccc.jpg', im)}            # 上传验证码图片
        # 发送网络请求
        r = requests.post('http://upload.chaojiying.net/Upload/Processing.php',
                data=params, files=files, headers=self.headers)
        return r.json()                                  # 返回响应数据

    def ReportError(self, im_id):
        """
        im_id:报错题目的图片 ID
        """
        params = {
            'id': im_id,
        }
        params.update(self.base_params)
        r = requests.post('http://upload.chaojiying.net/Upload/ReportError.php', data=params, headers=self.headers)
        return r.json()
```

（6）在确认用户名已经完成充值的情况下，创建超级鹰打码平台的实例对象，并传入相应的用户名、密码，以及软件 ID，然后指定要识别的验证码图片，最后使用 PostPic()方法识别图片，并输出识别结果。代码如下：

```
if __name__ == '__main__':
    # 用户中心>>软件 ID 生成一个替换 96001
    chaojiying = Chaojiying_Client('超级鹰用户名', '超级鹰用户名对应的密码', '96001')
    im = open('a.jpg', 'rb').read()                      # 使用本地图片文件路径替换 a.jpg
    # 1902 验证码类型  官方网站>>价格体系 3.4+版 print 后要加()
    print(chaojiying.PostPic(im, 1902))
```

这里使用超级鹰打码平台示例代码提供的验证码图片，运行程序，结果如下：

```
{'err_no': 0, 'err_str': 'OK', 'pic_id': '3109515574497000001', 'pic_str': '7261', 'md5': 'cf567a46b464d6cbe6b0646fb6eb18a4'}
```

在上述结果中，pic_str 对应的值即为识别到的验证码信息。

另外，在步骤（6）中使用 PostPic()方法识别验证码图片时，使用了参数 1902，它表示要识别的验证码类型。超级鹰打码平台支持的常用验证码类型及其说明如表 1.1 所示。

表 1.1 常用验证码类型及其说明

验证码类型	说　明
1902	常见 4～6 位英文数字
1101～1020	1～20 位英文数字
2001～2007	1～7 位纯汉字
3004～3012	1～12 位纯英文
4004～4111	1～11 位纯数字
5000	不定长汉字英文数字
5108	8 位英文数字（包含字符）
5201	拼音首字母，计算题，成语混合
5211	集装箱号由 4 位字母和 7 位数字组成
6001	计算题
6003	复杂计算题
6002	选择题四选一（ABCD 或 1234）
6004	问答题，智能回答题
9102	单击两个相同的字，返回：x1,y1|x2,y2
9202	单击两个相同的动物或物品，返回：x1,y1|x2,y2
9103	坐标多选，返回 3 个坐标，如：x1,y1|x2,y2|x3,y3
9004	坐标多选，返回 1～4 个坐标，如：x1,y1|x2,y2|x3,y3

1.5 项 目 运 行

通过前述步骤，我们设计并完成了“智能破解验证码”项目的开发。接下来，我们将运行该项目以检验开发成果。如图 1.21 所示，在 PyCharm 的左侧项目结构中，展开“智能破解验证码”的项目文件夹，然后分别选中 chaojiying.py、charCode.py 和 puzzleCode.py 文件，右击，在弹出的快捷菜单中选择 Run 'charCode'，即可成功运行该项目。

说明

运行项目之前，一定要确保本机已安装 BeautifulSoup（bs4）、requests、Pillow、tesserocr 和 selenium 等相关的模块。如果尚未安装这些模块，请使用 pip install 命令进行安装。

智能破解验证码的运行效果如图 1.22 所示。

本项目的核心功能是智能破解验证码，主要使用了 requests、BeautifulSoup（bs4）、Pillow、tesserocr 和 selenium 等模块。其中：requests、BeautifulSoup（bs4）和 Pillow 模块用于向指定网站发送请求，并获取验

证码图片；tesserocr 模块主要用于通过 OCR 技术识别普通的字符验证码图片；selenium 模块主要用于通过 Selenium 自动化测试工具破解滑动拼图验证码。此外，本项目还借助了第三方打码平台来识别验证码。

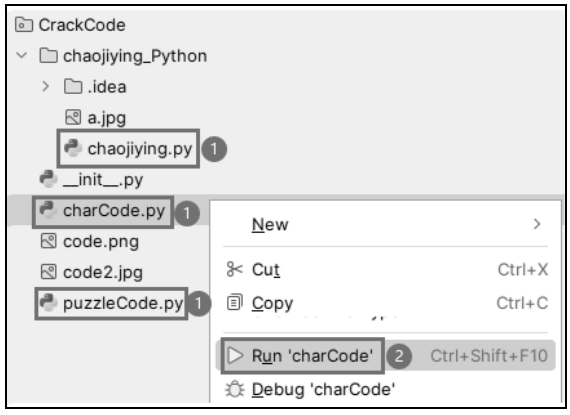


图 1.21　PyCharm 中的项目文件

图 1.22　成功运行项目

1.6　源 码 下 载

本章详细地讲解了如何编码实现“智能破解验证码”项目的各项功能，但给出的代码都是代码片段，而非完整源码。为方便读者学习，本书提供了完整的项目源码，读者可以扫描右侧二维码进行下载。

源码下载