

# 目录

前言	1.1
概述	1.2
常见框架	1.3
Appium	1.3.1
uiautomator2	1.3.2
facebook-wda	1.3.3
AirTest	1.3.4
常见问题	1.4
附录	1.5
参考资料	1.5.1

# 移动端自动化测试概览

- 最新版本: v0.8
- 更新时间: 20210317

## 简介

总结安卓和iOS等移动端自动化测试开发心得，包括常见框架Appium、uiautomator2、facebook-wda、AirTest等。以及一些常见问题的总结。

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### Gitbook源码

- [crifan/mobile\\_automation\\_overview](#): 移动端自动化测试概览

如何使用此Gitbook源码去生成发布为电子书

详见：[crifan/gitbook\\_template](#): demo how to use crifan gitbook template and demo

### 在线浏览

- 移动端自动化测试概览 [book.crifan.com](http://book.crifan.com)
- 移动端自动化测试概览 [crifan.github.io](https://crifan.github.io/mobile_automation_overview/)

### 离线下载阅读

- 移动端自动化测试概览 PDF
- 移动端自动化测试概览 ePUB
- 移动端自动化测试概览 MOBI

## 版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 [admin 艾特 crifan.com](mailto:admin@crifan.com)，我会尽快删除。谢谢合作。

## 鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 crifan 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

## 更多其他电子书

本人 crifan 还写了其他 100+ 本电子书教程，感兴趣可移步至：

[crifan/crifan\\_ebook\\_readme: Crifan的电子书的使用说明](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2021-03-18 20:54:04

## 移动端自动化概述

此处针对移动端自动化测试进行简单概要的介绍：

- 移动端自动化测试概览
  - 移动端：主要指的是安卓和iOS设备
  - 自动化测试
    - 根据用途和场景分
      - 自动化测试
        - 典型用途：测试移动端的app的功能是否满足预期
      - 自动化操作
        - 典型用途：模拟人的手指去操作屏幕，点击元素等操作，以便于自动化一套操作流程
  - 概览
    - 介绍总体概况
    - 有哪些主流的库

## 自动化测试 vs 自动化操作

移动端的自动化领域，根据用途和场景可以分为2类：

- 自动化测试
  - 又称：
    - 移动端测试
  - 侧重于：测试（移动端，主要指手机中）app的功能是否有问题
    - 比如
      - app是否会崩溃
      - 功能是否符合预期
        - 往往涉及到断言assertion，期望特定的输出
          - 举例
            - 输入非法手机号，点击注册
            - 希望：弹框提示 非法手机号
  - 自动化操作
    - 又称：
      - 自动化抓包
    - 侧重于：模拟人的手去操作手机
      - 更多关注的是：
        - 页面上有哪些元素
        - 以及如何处理到这些元素
          - 比如
            - 模拟人手去点击
              - 解放双手，写自动化脚本，实现自己的功能
              - 举例

- 自动化操作：每天定时收取支付宝蚂蚁深林中的能量
- 提取元素中的内容
  - 保存出来
    - 就属于 抓包，保存特定数据 的方面了
    - 所以也可以叫做：自动化抓包
  - 举例：
    - 自动化抓包：天猫app中的 热卖商品信息的爬取和保存

说明：

- 此文主要侧重于介绍：自动化操作 = 自动化抓包
- 不论是 自动化操作 还是 自动化测试 其使用的底层框架都是一样的
  - 比如facebook-wda用于iOS的自动化操作和测试

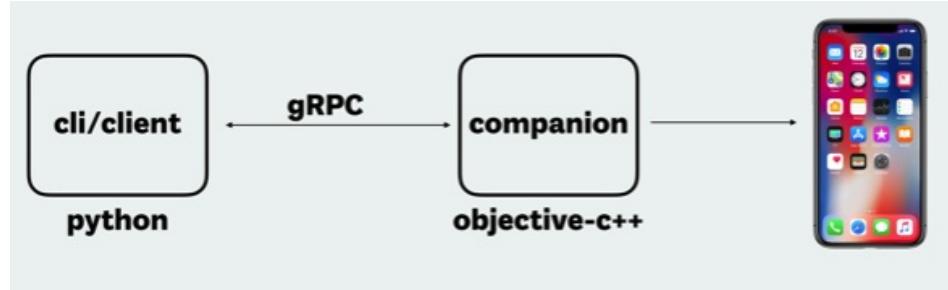
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-08-09 10:22:38

# 常见框架

移动端自动化测试常见框架：

- 多平台支持
  - Appium
    - 主页
      - GitHub
        - [appium/appium: Automation for iOS, Android, and Windows Apps.](#)
      - 官网
        - [Appium: Mobile App Automation Made Awesome.](#)
    - 支持平台
      - iOS
      - Android
      - Windows
      - Mac
  - Airtest
    - 主页
      - GitHub
        - [AirtestProject/Airtest: UI Automation Framework for Games and Apps](#)
      - 支持平台
        - Android
        - Emulator
        - iOS
        - Windows
        - Unity
        - Cocos2dx
        - Egret
        - WeChat
- 单个平台
  - Android
    - `uiautomator2 = u2`
  - iOS
    - `facebook-wda`
    - `idb = iOS Development Bridge`
      - 主页
        - GitHub
          - [facebook/idb: idb is a flexible command line interface for automating iOS simulators and devices](#)
        - 官网
          - [idb · iOS Development Bridge](#)
    - Facebook新出的

- 架构



- 缺点：

- 需要改动被测app的代码才能自动化测试？
  - 想要测试（iOS模拟器或真机）设备，要在被测设备中安装xctest测试用例才可以

u2和facebook-wda都是ATX拆分出来的

最早是：

[NetEaseGame/ATX: Smart phone automation tool. Support iOS, Android, WebApp and game](#)

后来拆分成：

- Android 的 uiautomator2
- iOS 的 facebook-wda

## iOS自动化测试框架发展历史

- iOS底层测试框架
  - iOS 8.0 ~ 9.3 : [UIAutomation](#)
    - 缺点：只能调试单台设备
      - 原因：instruments 限制单台Mac只能对应单台iOS设备
  - iOS 9.3+ : [XCUITest](#)
    - 目的：用以替代旧的 UIAutomation
- 第三方
  - WebDriverAgent
    - 作者：Facebook
    - 核心原理：实现了 WebDriver 的server
      - 通过 server 可以远程控制 iOS 设备
        - 支持各种操作：启动应用、关闭应用、点击、滚动等
        - 通过连接 XCTest.framework 调用苹果的 API 执行动作
    - 优点
      - 能够支持单台 Mac 对应多个iOS设备
        - 支持多个设备同时进行自动化
        - Appium、Macaca 已经集成

新: 2021-03-17 22:32:25

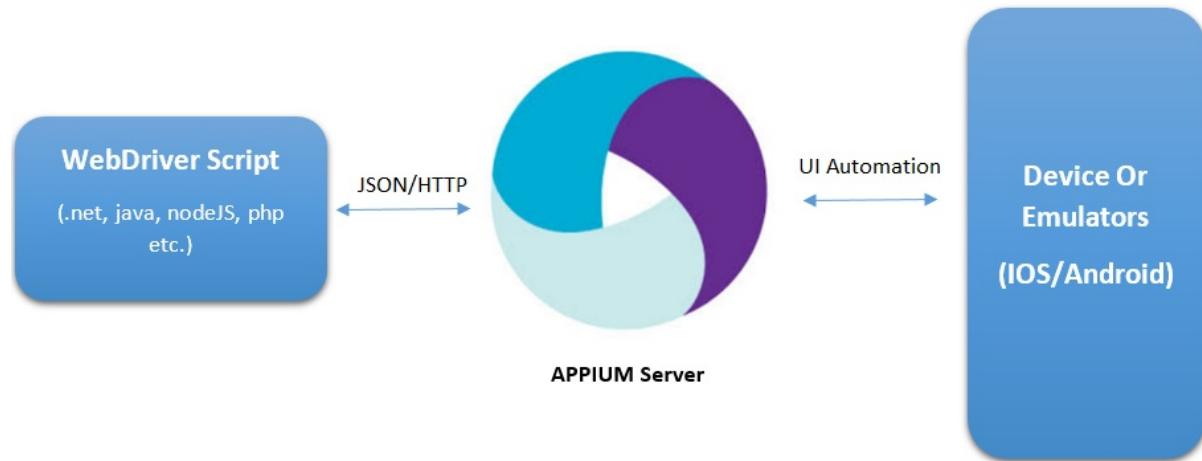
# Appium

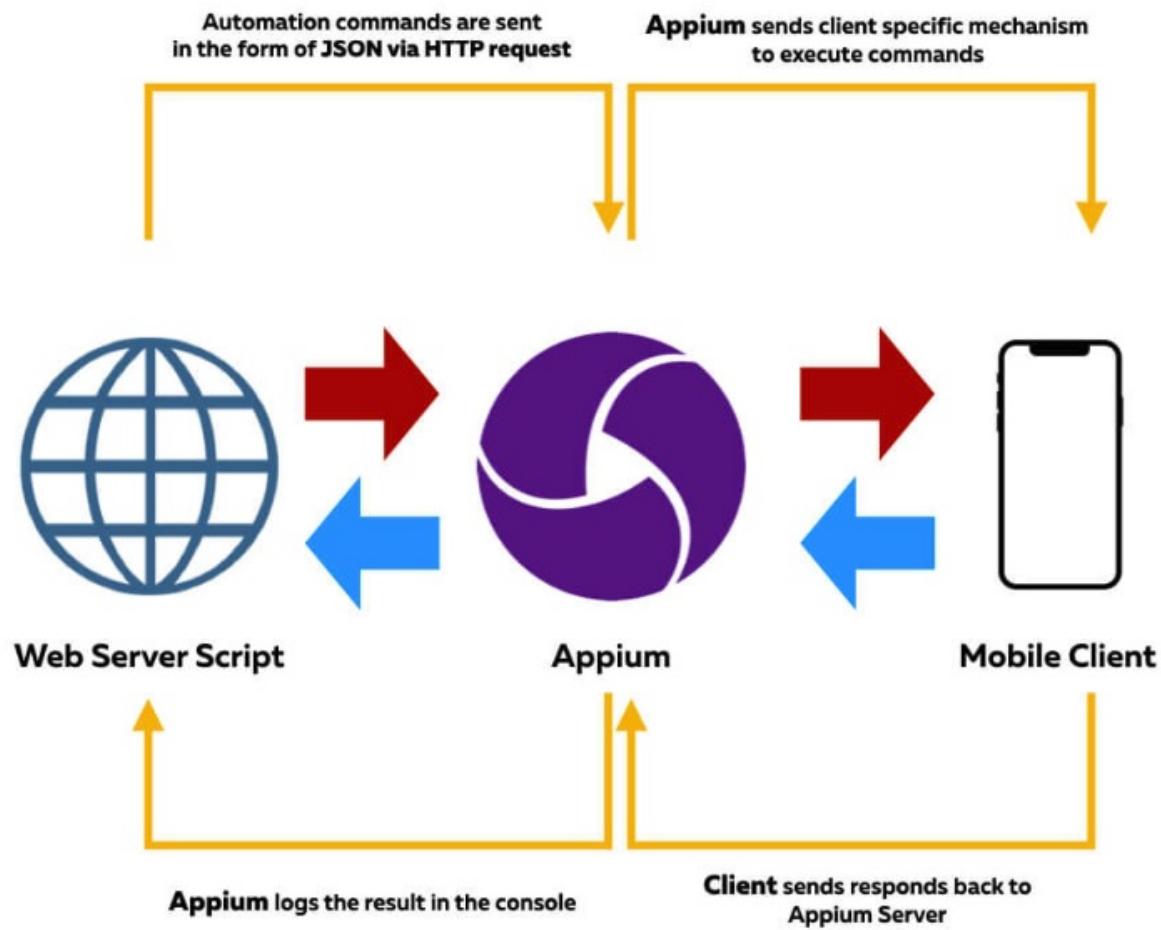
移动端测试框架中，流行度最广的是： Appium

## Appium依赖于底层框架

- iOS
  - Appium 在 iOS > 9.3 后全面采用（底层基于 XCUI Test 的） WebDriverAgent 的方案
- Android >= 4.3 : Google's UiAutomator / UiAutomator2
- Windows : Microsoft's WinAppDriver

## Appium的iOS自动化架构





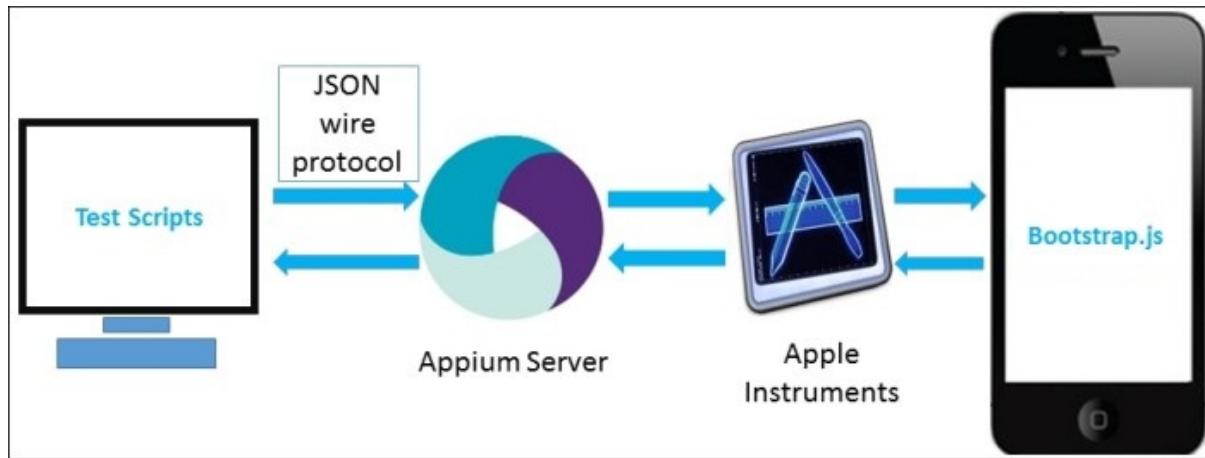
## 用Appium实现iOS自动化

对于iOS的真机，需要预先配置好才可以：

关于Appium的真机的设置：

- 官网 英文
  - [XCUITest Real Devices \(iOS\) - Appium](#)
- readthedocs 中文
  - [Real devices ios - appium](#)
- GitHub
  - [appium-xcuitest-driver/real-device-config.md at master · appium/appium-xcuitest-driver](#)

## iOS真机配置



crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新: 2021-03-17 22:31:01

## uiautomator2

详见另一完整教程：

[安卓自动化测试利器：uiautomator2](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-08-09 10:22:38

## facebook-wda

详见另一完整教程：

[iOS自动化测试利器：facebook-wda](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-08-09 10:22:38

## AirTest

另外还有一个，自动化测试工具：网易的 AirTest

- [AirTest](#)
  - 一句话描述：
    - 网易游戏推出的一款跨平台的UI自动化测试框架，适用于游戏和App
  - 官网
    - [主页](#)
    - [Airtest Project](#)
    - 文档
      - [欢迎使用 - Airtest Project Docs](#)
      - [欢迎来到Airtest官方文档！ — airtest 文档](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2021-03-17 22:51:48

## 常见问题

移动端自动化测试会遇到的一些问题，现整理如下。

比如 [Same test cases for Android and iOS automation — pros and cons | by Satyajit Malugu | Medium](#)，其中就提到了：

- 返回按钮的处理等问题
- 界面的源码结构不同
  - iOS：会返回当前页面中所有的元素，包括不可见的（需要滚动后才可见的那些）元素
  - Android：只会当前页面中可见的元素

导致实现移动端的多平台统一测试用例，很不容易

## OCR识别复杂游戏界面中文字，偶尔会误判出错

比如之前折腾：

【已解决】安卓游戏暗黑觉醒自动化：稳定的检测出是首充豪礼首充6元首充98元的首充弹框期间，对于首充豪礼弹框页面：





偶尔可以返回：

```
{'chars': [ {'char': '首', 'location': {'width': 36, 'top': 834, 'left': 938, 'height': 60}}, {'char': '充', 'location': {'width': 36, 'top': 834, 'left': 991, 'height': 60}}, {'char': '6', 'location': {'width': 30, 'top': 834, 'left': 1040, 'height': 60}}, {'char': '元', 'location': {'width': 36, 'top': 834, 'left': 1084, 'height': 60}}, {'char': '首', 'location': {'width': 36, 'top': 834, 'left': 1230, 'height': 60}}, {'char': '充', 'location': {'width': 36, 'top': 834, 'left': 1303, 'height': 60}}, {'char': '9', 'location': {'width': 29, 'top': 834, 'left': 1334, 'height': 60}}, {'char': '8', 'location': {'width': 30, 'top': 834, 'left': 1389, 'height': 60}}, {'char': '元', 'location': {'width': 57, 'top': 834, 'left': 1432, 'height': 60}}], 'location': {'width': 551, 'top': 834, 'left': 938, 'height': 60}, 'words': '首充6元首充98元'}
```



完美解析的效果，即：首充6元首充98元，是图片中准确的文字和充值金额。

由于游戏界面中文字比较复杂，尤其是：

- 特殊的字体
  - 首充豪礼 字体很特别
    - 估计是游戏常用字体
- 额外加了闪光等效果
  - 比如 首充98元
    - 中的 首充 或 98元 外圈闪光

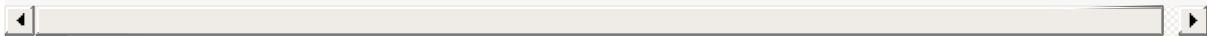
等特殊情况，导致文字检测出来，常常会误判：

比如：

- (1) 首元首充8元

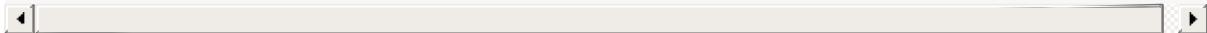
```
{'chars': [ {'char': '首', 'location': {'width': 36, 'top': 836, 'left': 939, 'height': 59}}, {'char': '元', 'location': {'width': 36, 'top': 836, 'left': 1085, 'height': 59}}, {'char': '首', 'location': {'width': 36, 'top': 836, 'left': 1231, 'height': 59}}, {'char': '充', 'location': {'width': 36, 'top': 836, 'left': 1286, 'height': 59}}, {'char': '8', 'location': {}}
```

```
: {'width': 29, 'top': 836, 'left': 1390, 'height': 59}], {'char': '元', 'location': {'width': 55, 'top': 836, 'left': 1432, 'height': 59}}], 'location': {'width': 549, 'top': 836, 'left': 939, 'height': 59}, 'words': '首元首充8元'}
```



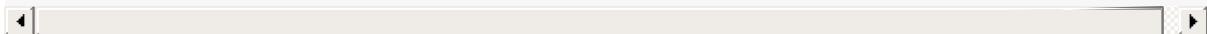
## (2) 首元首8元

```
{'chars': [ {'char': '首', 'location': {'width': 36, 'top': 837, 'left': 938, 'height': 61}}, {'char': '元', 'location': {'width': 75, 'top': 837, 'left': 1087, 'height': 61}}, {'char': '首', 'location': {'width': 37, 'top': 837, 'left': 1236, 'height': 61}}, {'char': '8', 'location': {'width': 30, 'top': 837, 'left': 1381, 'height': 61}}, {'char': '元', 'location': {'width': 54, 'top': 837, 'left': 1424, 'height': 61}}], 'location': {'width': 547, 'top': 837, 'left': 938, 'height': 61}, 'words': '首元首8元'}
```



## (3) 首充元+首充98

```
{'chars': [ {'char': '首', 'location': {'width': 37, 'top': 832, 'left': 934, 'height': 62}}, {'char': '充', 'location': {'width': 38, 'top': 832, 'left': 990, 'height': 62}}, {'char': '元', 'location': {'width': 57, 'top': 832, 'left': 1087, 'height': 62}}, {'char': '+', 'location': {'width': 31, 'top': 832, 'left': 1136, 'height': 62}}, {'char': '首', 'location': {'width': 38, 'top': 832, 'left': 1239, 'height': 62}}, {'char': '充', 'location': {'width': 37, 'top': 832, 'left': 1297, 'height': 62}}, {'char': '9', 'location': {'width': 30, 'top': 832, 'left': 1348, 'height': 62}}, {'char': '8', 'location': {'width': 30, 'top': 832, 'left': 1388, 'height': 62}}], 'location': {'width': 553, 'top': 832, 'left': 934, 'height': 62}, 'words': '首充元+首充98'}
```



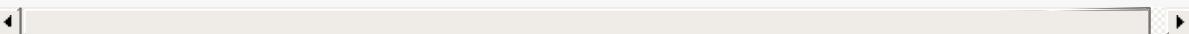
## (4) 首充5元充98元

```
{'chars': [ {'char': '首', 'location': {'width': 58, 'top': 832, 'left': 933, 'height': 62}}, {'char': '充', 'location': {'width': 38, 'top': 832, 'left': 990, 'height': 62}}, {'char': '5', 'location': {'width': 30, 'top': 832, 'left': 1043, 'height': 62}}, {'char': '元', 'location': {'width': 77, 'top': 832, 'left': 1087, 'height': 62}}, {'char': '充', 'location': {'width': 38, 'top': 832, 'left': 1300, 'height': 62}}, {'char': '9', 'location': {'width': 31, 'top': 832, 'left': 1332, 'height': 62}}, {'char': '8', 'location': {'width': 31, 'top': 832, 'left': 1390, 'height': 62}}, {'char': '元', 'location': {'width': 57, 'top': 832, 'left': 1435, 'height': 62}}], 'location': {'width': 558, 'top': 832, 'left': 933, 'height': 62}, 'words': '首充5元充98元'}
```



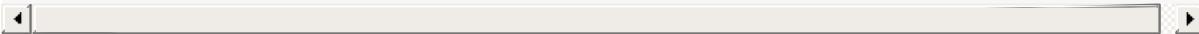
## (5) 首兄元首9元

```
{'chars': [ {'char': '首', 'location': {'width': 36, 'top': 836, 'left': 941, 'height': 61}}, {'char': '兄', 'location': {'width': 37, 'top': 836, 'left': 996, 'height': 61}}, {'char': '元', 'location': {'width': 37, 'top': 836, 'left': 1090, 'height': 61}}, {'char': '首', 'location': {'width': 36, 'top': 836, 'left': 1239, 'height': 61}}, {'char': '9', 'location': {'width': 30, 'top': 836, 'left': 1345, 'height': 61}}, {'char': '元', 'location': {'width': 59, 'top': 836, 'left': 1427, 'height': 61}}], 'location': {'width': 545, 'top': 836, 'left': 941, 'height': 61}, 'words': '首兄元首9元'}
```



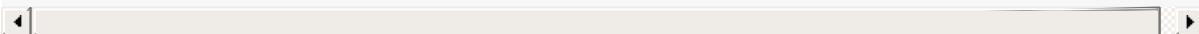
(6) 首元98元

```
'chars': [ {'char': '首', 'location': {'width': 36, 'top': 837, 'left': 940, 'height': 61}}, {'char': '元', 'location': {'width': 37, 'top': 837, 'left': 1088, 'height': 61}}, {'char': '9', 'location': {'width': 29, 'top': 837, 'left': 1344, 'height': 61}}, {'char': '8', 'location': {'width': 30, 'top': 837, 'left': 1381, 'height': 61}}, {'char': '元', 'location': {'width': 61, 'top': 837, 'left': 1424, 'height': 61}}], 'location': {'width': 546, 'top': 837, 'left': 940, 'height': 61}, 'words': '首元98元'}
```



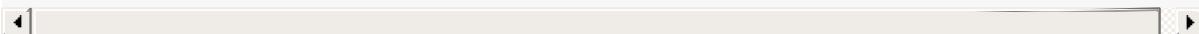
(7) 首元首元

```
'chars': [ {'char': '首', 'location': {'width': 34, 'top': 837, 'left': 938, 'height': 56}}, {'char': '元', 'location': {'width': 34, 'top': 837, 'left': 1092, 'height': 56}}, {'char': '首', 'location': {'width': 34, 'top': 837, 'left': 1246, 'height': 56}}, {'char': '元', 'location': {'width': 50, 'top': 837, 'left': 1420, 'height': 56}}], 'location': {'width': 547, 'top': 837, 'left': 938, 'height': 56}, 'words': '首元首元'}
```



(8) 首元道充98元

```
'chars': [ {'char': '首', 'location': {'width': 35, 'top': 835, 'left': 940, 'height': 60}}, {'char': '元', 'location': {'width': 35, 'top': 835, 'left': 1085, 'height': 60}}, {'char': '道', 'location': {'width': 35, 'top': 835, 'left': 1230, 'height': 60}}, {'char': '充', 'location': {'width': 36, 'top': 835, 'left': 1302, 'height': 60}}, {'char': '9', 'location': {'width': 30, 'top': 835, 'left': 1333, 'height': 60}}, {'char': '8', 'location': {'width': 30, 'top': 835, 'left': 1387, 'height': 60}}, {'char': '元', 'location': {'width': 35, 'top': 835, 'left': 1430, 'height': 60}}], 'location': {'width': 531, 'top': 835, 'left': 940, 'height': 60}, 'words': '首元道充98元'}
```



(9) 首充元首充98元

```
'chars': [ {'char': '首', 'location': {'width': 61, 'top': 828, 'left': 934, 'height': 67}}, {'char': '充', 'location': {'width': 41, 'top': 829, 'left': 995, 'height': 66}}, {'char': '元', 'location': {'width': 41, 'top': 830, 'left': 1095, 'height': 67}}, {'char': '首', 'location': {'width': 61, 'top': 833, 'left': 1237, 'height': 67}}, {'char': '充', 'location': {'width': 39, 'top': 833, 'left': 1297, 'height': 67}}, {'char': '9', 'location': {'width': 33, 'top': 834, 'left': 1331, 'height': 66}}, {'char': '8', 'location': {'width': 34, 'top': 834, 'left': 1391, 'height': 67}}, {"char": "元", "location": {"width": 39, "top": 835, "left": 1439, "height": 67}}], 'location': {'width': 554, 'top': 828, 'left': 934, 'height': 74}, 'words': '首充元首充98元'}
```



(10) 首6元首充8元

```
'chars': [ {'char': '首', 'location': {'width': 41, 'top': 835, 'left': 942, 'height': 66}},
```

```
{'char': '6', 'location': {'width': 34, 'top': 835, 'left': 1036, 'height': 66}}, {'char': '元', 'location': {'width': 82, 'top': 835, 'left': 1084, 'height': 66}}, {'char': '首', 'location': {'width': 41, 'top': 835, 'left': 1247, 'height': 66}}, {'char': '充', 'location': {'width': 39, 'top': 835, 'left': 1288, 'height': 66}}, {'char': '8', 'location': {'width': 33, 'top': 835, 'left': 1382, 'height': 66}}, {'char': '元', 'location': {'width': 82, 'top': 835, 'left': 1431, 'height': 66}}], 'location': {'width': 587, 'top': 835, 'left': 942, 'height': 66}, 'words': '首6元首充8元'}
```

等等情况。

所以如果用如下逻辑（正则）

```
"首充((\d+元)|(元)|(\d+))", # 首充元 / 首充xxx元 / 首充xxx
```

的代码

```
def isGotoPayPopupPage(self, isRespLocation=False):
    """Check if goto payment popup page or not"""
    gotoPayStrList = [
        "^前往充值$", # 剑玲珑
        "^立即充值$", # 至尊屠龙
        "^充值$", # 剑玲珑, 首充之后, 手动点击 每日充值 后
        "^充点小钱$", # 御剑仙缘
        # "^首充豪礼?", # 暗黑觉醒: 首充豪礼 但有时候无法识别出 礼 -> 只能用于识别, 无法用于
        # 点击, 所以换成下面:
        # 暗黑觉醒: 首充6元 和 首充98元 ->
        # "首充\d+元", # (1) 识别成 首元首充8元
        # "首充?\d+元", # (2) 首元首8元
        "首充((\d+元)|(元)|(\d+))", # 首充元 / 首充xxx元 / 首充xxx

        # 剑玲珑, 首充 之后, 但是点击 领取 并不能进入下一页
        # "^领取$",
        # "^领取奖励$", # 偶尔会由于屏幕弹框 领域奖励 而误判进入了 前往充值 页面
    ]
    respBoolOrTuple = self.isExistAnyStr(gotoPayStrList, isRespFullInfo=isRespLocation)

    logging.info("GotoPay: respBoolOrTuple=%s", respBoolOrTuple)
    return respBoolOrTuple
```

注： `isExistAnyStr` 的具体实现，可参考：[图像 · Python常用代码段](#)

去单次调用，往往会造成误判，而无法识别出我们希望的值

所以，为了稳定的检测出是否是首充豪礼的弹框充值野蛮，后来改用逻辑：

多次尝试调用，直到检测成功为止

具体代码是：

```
def isGotoPayPopupPage_multipleRetry(self, isRespLocation=False):
    """鉴于 暗黑觉醒 的 首充弹框 的检测很不稳定，所以 增加此函数 多次尝试 以确保 当是首充页面时，的确能稳定的检测出的确是"""
    respBoolOrTuple = CommonUtils.multipleRetry(
        functionInfoDict = {
            "functionCallback": self.isGotoPayPopupPage,
            "functionParaDict": {
                "isRespLocation": isRespLocation,
            },
        },
        isRespFullRetValue = isRespLocation,
    )
    return respBoolOrTuple
```

注：其中 `multipleRetry` 详见：[通用逻辑 · Python常用代码段](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2021-03-17 23:01:28

## 附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-08-09 10:22:38

## 参考资料

- iOS自动化测试利器: [facebook-wda](#)
- 安卓自动化测试利器: [uiautomator2](#)
- [Introduction - Appium](#)
- [Supported Platforms - Appium](#)
- [XCUITest](#)
- [UIAutomation](#)
- [UiAutomator / UiAutomator2](#)
- [WinAppDriver](#)
- [XCUITest Real Devices \(iOS\) - Appium](#)
- [Real devices ios - appium](#)
- [appium-xcuitest-driver/real-device-config.md at master · appium/appium-xcuitest-driver](#)
- [Go Back - Appium](#)
- [...](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2021-03-17 22:46:44