

目录

前言	1.1
背景	1.2
常用代码段	1.3
通用逻辑	1.3.1
变量	1.3.2
系统	1.3.3
日期时间	1.3.4
字符和字符串	1.3.5
文件系统	1.3.6
文件	1.3.6.1
文件夹	1.3.6.2
多媒体	1.3.7
图像	1.3.7.1
Pillow	1.3.7.1.1
百度OCR	1.3.7.1.2
音频	1.3.7.2
视频	1.3.7.3
网络相关	1.3.8
BeautifulSoup	1.3.8.1
requests	1.3.8.2
数学	1.3.9
常见语法	1.4
函数参数	1.4.1
dict字典	1.4.2
list列表和set集合	1.4.3
sort排序	1.4.4
enum枚举	1.4.5
collections集合	1.4.6
附录	1.5
参考资料	1.5.1

Python常用代码段

- 最新版本: v1.2
- 更新时间: 20210114

简介

整理出crifan总结的Python各个方面常用的代码段，供需要的参考。包括通用逻辑、变量、系统、日期时间、字符和字符串、文件系统，比如文件和文件夹等、以及第三方库，比如BeautifulSoup、以及多媒体音视频类、包括Pillow、以及网络的Requests等；以及其他常见语法，包括dict字典、list列表、set集合、enum枚举、collections集合等。

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/python_common_code_snippet](#): Python常用代码段

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

- [Python常用代码段 book.crifan.com](#)
- [Python常用代码段 crifan.github.io](#)

离线下载阅读

- [Python常用代码段 PDF](#)
- [Python常用代码段 ePub](#)
- [Python常用代码段 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，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：2021-01-14 20:23:23

背景

多年的技术开发，积累了很多关于 [Python](#) 的常用的一些函数和功能，都已整理到对应的函数库中：

<https://github.com/crifan/crifanLibPython>

且也给出了很多函数的demo演示如何使用：

<https://github.com/crifan/crifanLibPython/tree/master/crifanLib/demo>

但还是解释的不够清楚和全面。

故此处专门把Python的常用的代码段和调用举例，都整理至此，详细解释。

目的：

方便需要时能[快速查阅](#)：直接看代码，一看就懂，无需额外解释。

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： 2020-06-27 20:21:05

常用代码段

下面总结各个方面的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: 2020-07-11 09:46:30

通用逻辑

多次运行一个函数，直到成功运行

执行一个函数（可能有多个可变数量的参数），且尝试多次，直到成功或超出最大此时，最终实现是：

```
def multipleRetry(functionInfoDict, maxRetryNum=5, sleepInterval=1.0, isShowErrWhenFail=False):
    """Do something, retry multiple time if fail

    Args:
        functionInfoDict (dict): function info dict contains:
            maxRetryNum (int): max retry number
            sleepInterval (float): sleep time of each interval
            isShowErrWhenFail (bool): show error when fail if True
    Returns:
        bool
    Raises:
        None

    doSuccess = False
    functionCallback = functionInfoDict["functionCallback"]
    functionParaDict = functionInfoDict.get("functionParaDict", {})

    curRetryNum = maxRetryNum
    while curRetryNum > 0:
        if functionParaDict:
            doSuccess = functionCallback(**functionParaDict)
        else:
            doSuccess = functionCallback()

        if doSuccess:
            break

        time.sleep(sleepInterval)
        curRetryNum -= 1

    if not doSuccess:
        if isShowErrWhenFail:
            functionName = str(functionCallback)
            # '<bound method DevicesMethods.switchToAppStorage of <DevicesMethods object at 0x10c2e0>'
            logging.error("Still fail after %d retry for %s" % (maxRetryNum, functionName))
    return doSuccess
```

说明：

文件

functionCallback 函数类型都要符合：返回值是bool类型才可以

调用举例：

(1) 没有额外参数

```
foundAndClickedWifi = CommonUtils.multipleRetry({"functionC
```

其中：

```
def iOSFromSettingsIntoWifiList(self):  
    ...  
    foundAndClickedWifi = self.findAndClickElement(query=wifiListQuery)  
    return foundAndClickedWifi
```

类似例子：

```
isSwitchOk = self.multipleRetry({"functionCallback": self.isSwitchOk, "functionParaDict": {}})
```

对比之前原始写法：

```
isSwitchOk = self.switchToAppStoreSearchTab()
```

其他类似例子：

```
foundAndClickedDownload = self.multipleRetry({"functionCallback": self.isInputOk, "functionParaDict": {}})
```

详见：

【已解决】AppStore自动安装iOS的app：逻辑优化加等待和多试几次

(2) 有额外参数，参数个数：2个

```
searchInputQuery = {"type": "XCUIElementTypeSearchField", "text": ""}  
isInputOk = CommonUtils.multipleRetry(  
    {  
        "functionCallback": self.wait_element_setText,  
        "functionParaDict": {  
            "locator": searchInputQuery,  
            "text": appName,  
        }  
    }  
)
```

对比之前原始写法：

文件

```
searchInputQuery = {"type": "XCUIElementTypeSearchField", "text": ""}  
isInputOk = self.wait_element_setText(searchInputQuery, appName)
```

其中wait_element_setText的定义是：

```
def wait_element_setText(self, locator, text):
```

对应着之前传入时的：

```
"functionParaDict": {  
    "locator": searchInputQuery,  
    "text": appName,  
}
```

(3) 有额外参数，且加上multipleRetry的额外参数

```
isSwitchOk = CommonUtils.multipleRetry(  
    {"functionCallback": self.switchToAppStoreSearchTab},  
    maxRetryNum = 10,  
    sleepInterval = 0.5,  
)
```

以及类似的：

```
isIntoDetailOk = self.multipleRetry(  
{  
    "functionCallback": self.appStoreSearchResultIntoDetail,  
    "functionParaDict": {  
        "appName": appName,  
    },  
},  
    sleepInterval=0.5  
)
```

之前原始写法：

```
isIntoDetailOk = self.appStoreSearchResultIntoDetail(appName)
```

注：

此处是后来加上的

```
sleepInterval=0.5
```

文件

是因为后来遇到了，即使尝试了5次，依旧没找到，所以增加了没找到的延迟等待时间。

详见：

【已解决】AppStore自动安装iOS的app：逻辑优化加等待和多试几次

(4)

```
isIntoDetailOk = CommonUtils.multipleRetry(  
    {  
        "functionCallback": self.appStoreSearchResultIntoDetail,  
        "functionParaDict": {  
            "appName": appName,  
        }  
    },  
    maxRetryNum = 10,  
    sleepInterval = 0.5,  
)
```

新版：新增参数**isRespFullRetValue**

此处最后更新： 20200925

后续多次优化新增参数：是否返回完整信息

代码：

```

def multipleRetry(functionInfoDict, maxRetryNum=5, sleepInterval=0.1):
    """do something, retry if single call failed, retry until success or max retry number reached.

    Args:
        functionInfoDict (dict): function info dict containing:
            maxRetryNum (int): max retry number
            sleepInterval (float): sleep time (seconds) of each retry
            isShowErrWhenFail (bool): show error when fail if True
            isRespFullRetValue (bool): whether return full return value if True
    Returns:
        isRespFullRetValue=False: bool
        isRespFullRetValue=True: bool / tuple/list/...
    Raises:
        ...
    """
    finalReturnValue = None
    doSuccess = False
    functionCallback = functionInfoDict["functionCallback"]
    functionParaDict = functionInfoDict.get("functionParaDict", {})

    curRetryNum = maxRetryNum
    while curRetryNum > 0:
        if functionParaDict:
            # doSuccess = functionCallback(**functionParaDict)
            respValue = functionCallback(**functionParaDict)
        else:
            # doSuccess = functionCallback()
            respValue = functionCallback()

        doSuccess = False
        if isinstance(respValue, bool):
            doSuccess = bool(respValue)
        elif isinstance(respValue, tuple):
            doSuccess = bool(respValue[0])
        elif isinstance(respValue, list):
            doSuccess = bool(respValue[0])
        else:
            raise Exception("multipleRetry: Not support type of response value: %s" % type(respValue))

        if isRespFullRetValue:
            finalReturnValue = respValue
        else:
            finalReturnValue = doSuccess

        if doSuccess:
            break

        time.sleep(sleepInterval)
        curRetryNum -= 1

    if not doSuccess:
        raise Exception("multipleRetry: Failed after %d retries" % maxRetryNum)

```

```
if isShowErrWhenFail:  
    functionName = str(functionCallback)  
    # '<bound method DevicesMethods.switchToAppSto  
    logging.error("Still fail after %d retry for %s")  
    # return doSuccess  
    return finalReturnValue
```

调用举例：

(1) 默认不返回完整信息，只返回bool值

```
respBoolOrTuple = CommonUtils.multipleRetry(  
    functionInfoDict = {  
        "functionCallback": self.isGotoPayPopupPage,  
        "functionParaDict": {  
            "isRespLocation": False,  
        },  
    },  
)
```

(2) 返回完整信息

```
respBoolOrTuple = CommonUtils.multipleRetry(  
    functionInfoDict = {  
        "functionCallback": self.isGotoPayPopupPage,  
        "functionParaDict": {  
            "isRespLocation": True,  
        },  
    },  
    isRespFullRetValue = True,  
)
```

变量

判断变量类型

优先用 `isinstance`，而不是 `type`

```
>>> isinstance(2, float)
False
>>> isinstance('a', (str, unicode))
True
>>> isinstance((2, 3), (str, list, tuple))
True
```

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: 2020-06-27 18:24:15

系统

此处整理用Python处理系统相关的通用的代码。

系统类型

```
import sys

def osIsWindows():
    return sys.platform == "win32"

def osIsCygwin():
    return sys.platform == "cygwin"

def osIsMacOS():
    return sys.platform == "darwin"

def osIsLinux():
    return sys.platform == "linux"

def osIsAix():
    return sys.platform == "aix"
```

命令行

获取命令行执行命令返回结果

代码:

```
def get_cmd_lines(cmd, text=False):
    # 执行cmd命令，将结果保存为列表
    resultStr = ""
    resultStrList = []
    try:
        consoleOutputByte = subprocess.check_output(cmd, shell=True)
        try:
            resultStr = consoleOutputByte.decode("utf-8")
        except UnicodeDecodeError:
            # TODO: use chardet auto detect encoding
            # consoleOutputStr = consoleOutputByte.decode('gb18030')
            resultStr = consoleOutputByte.decode("gb18030")

        if not text:
            resultStrList = resultStr.splitlines()
    except Exception as err:
        print("err=%s when run cmd=%s" % (err, cmd))

    if text:
        return resultStr
    else:
        return resultStrList
```

硬件信息

获取当前电脑（Win或Mac）的序列号

代码：

```
def getSerialNumber(self):
    """get current computer serial number"""
    # cmd = "wmic bios get serialnumber"
    cmd = ""
    if CommonUtils.osIsWindows():
        # Windows
        cmd = "wmic bios get serialnumber"
    elif CommonUtils.osIsMacOS():
        # macOS
        cmd = "system_profiler SPHardwareDataType | awk '/"
    # TODO: add support other OS
    # AIX: aix
    # Linux: linux
    # Windows/Cygwin: cygwin

    serialNumber = ""
    lines = CommonUtils.get_cmd_lines(cmd)
    if CommonUtils.osIsWindows():
        # Windows
        serialNumber = lines[1]
    elif CommonUtils.osIsMacOS():
        # macOS
        serialNumber = lines[0] # C02Y3N10JHC8

    return serialNumber
```

日期时间

详见：

<https://github.com/crifan/crifanLibPython/blob/master/crifanLib/crifanDatetime.py>

getCurDatetimeStr 生成当前日期时间字符串

```
def getCurDatetimeStr(outputFormat="%Y%m%d_%H%M%S"):  
    """  
    get current datetime then format to string  
  
    eg:  
        20171111_220722  
  
    :param outputFormat: datetime output format  
    :return: current datetime formatted string  
    """  
    curDatetime = datetime.now() # 2017-11-11 22:07:22.705  
    curDatetimeStr = curDatetime.strftime(format=outputFormat)  
    return curDatetimeStr
```

调用举例：

```
curDatetimeStr = getCurDatetimeStr() # '20191219_143400'
```

datetime转时间戳

```
import time

def datetimeToTimestamp(self, datetimeVal, withMilliseconds=False):
    """
        convert datetime value to timestamp
    eg:
        "2006-06-01 00:00:00.123" -> 1149091200
            if with milliseconds -> 1149091200123
    :param datetimeVal:
    :return:
    """

    timetupleValue = datetimeVal.timetuple()
    timestampFloat = time.mktime(timetupleValue) # 1531468736.0
    timestamp10DigitInt = int(timestampFloat) # 1531468736
    timestampInt = timestamp10DigitInt

    if withMilliseconds:
        microsecondInt = datetimeVal.microsecond # 817762
        microsecondFloat = float(microsecondInt)/float(1000000)
        timestampFloat = timestampFloat + microsecondFloat
        timestampFloat = timestampFloat * 1000 # 1531468736.817762
        timestamp13DigitInt = int(timestampFloat) # 1531468736817762
        timestampInt = timestamp13DigitInt

    return timestampInt
```

获取当前时间戳

```
from datetime import datetime

def getCurTimestamp(withMilliseconds=False):
    """
        get current time's timestamp
        (default)not milliseconds -> 10 digits: 1351670162
        with milliseconds -> 13 digits: 1531464292921
    """

    curDatetime = datetime.now()
    return datetimeToTimestamp(curDatetime, withMilliseconds)
```

时间戳精确到毫秒

```
from datetime import datetime, timedelta
timestampStr = datetime.now().strftime("%Y%m%d_%H%M%S_%f")
# 20180712_154134_660436
```

文件

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: 2020-07-11 09:46:56

字符和字符串

详见：

<https://github.com/crifan/crifanLibPython/blob/master/crifanLib/crifanFile.py>

Python3 str转bytes

Python 3中，把字符串转换成字节码，可以有两种写法：

- 方法1：用bytes去转换

```
convertedBytes = bytes(originStr)
```

- 方法2：用字符串str的编码encode

```
encodedBytes = originStr.encode()
```

其中：

都有额外的encoding参数：

- 由于默认都是UTF-8
 - 所以可加可不加
- 也可以根据需要，去加其他你要的编码
 - 如果要加，就是：

```
convertedBytes = bytes(originStr, "UTF-8")
encodedBytes = originStr.encode("UTF-8")
```

字符串格式化为人类易读格式

```
def formatSize(sizeInBytes, decimalNum=1, isUnitWithI=False):
    """
        format size to human readable string

    example:
        3746 -> 3.7KB
        87533 -> 85.5KiB
        98654 -> 96.3 KB
        352 -> 352.0B
        76383285 -> 72.84MB
        763832854988542 -> 694.70TB
        763832854988542665 -> 678.4199PB

    refer:
        https://stackoverflow.com/questions/1094841/reusable-binary-size-formatting
    """

    # https://en.wikipedia.org/wiki/Binary_prefix#Specific_units
    # K=kilo, M=mega, G=giga, T=tera, P=peta, E=exa, Z=zetta,
    sizeUnitList = ['', 'K', 'M', 'G', 'T', 'P', 'E', 'Z']
    largestUnit = 'Y'

    if isUnitWithI:
        sizeUnitListWithI = []
        for curIdx, eachUnit in enumerate(sizeUnitList):
            unitWithI = eachUnit
            if curIdx >= 1:
                unitWithI += 'i'
            sizeUnitListWithI.append(unitWithI)

    # sizeUnitListWithI = ['', 'Ki', 'Mi', 'Gi', 'Ti', 'Pi', 'Ei'
    sizeUnitList = sizeUnitListWithI

    largestUnit += 'i'

    suffix = "B"
    decimalFormat = "," + str(decimalNum) + "f" # ".1f"
    finalFormat = "%" + decimalFormat + sizeUnitSeperator + "%"

    sizeNum = sizeInBytes
    for sizeUnit in sizeUnitList:
        if abs(sizeNum) < 1024.0:
            return finalFormat % (sizeNum, sizeUnit, suffix)
        sizeNum /= 1024.0
    return finalFormat % (sizeNum, largestUnit, suffix)
```

调用:

```
def testKb():
    kbSize = 3746
    kbStr = formatSize(kbSize)
    print("%s -> %s" % (kbSize, kbStr))

def testI():
    iSize = 87533
    iStr = formatSize(iSize, isUnitWithI=True)
    print("%s -> %s" % (iSize, iStr))

def testSeparator():
    seperatorSize = 98654
    seperatorStr = formatSize(seperatorSize, sizeUnitSeparate=True)
    print("%s -> %s" % (seperatorSize, seperatorStr))

def testBytes():
    bytesSize = 352
    bytesStr = formatSize(bytesSize)
    print("%s -> %s" % (bytesSize, bytesStr))

def testMb():
    mbSize = 76383285
    mbStr = formatSize(mbSize, decimalNum=2)
    print("%s -> %s" % (mbSize, mbStr))

def testTb():
    tbSize = 763832854988542
    tbStr = formatSize(tbSize, decimalNum=2)
    print("%s -> %s" % (tbSize, tbStr))

def testPb():
    pbSize = 763832854988542665
    pbStr = formatSize(pbSize, decimalNum=4)
    print("%s -> %s" % (pbSize, pbStr))

def demoFormatSize():
    testKb()
    testI()
    testSeparator()
    testBytes()
    testMb()
    testTb()
    testPb()
```

文件系统

详见：

- <https://github.com/crifan/crifanLibPython/blob/master/crifanLib/crifanFile.py>
- <https://github.com/crifan/crifanLibPython/blob/master/crifanLib/demo/crifanFileDemo.py>

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：2020-06-27 18:56:10

文件

获取文件最后更新时间

```
import os

def getUpdateTime(curFileOrPath):
    """get file/folder latest update time = modify time

    Args:
        curFileOrPath (str): some file or folder path
    Returns:
        int: time stamp int of 13 digit, with milliseconds
    Raises:
        .....
    updateTime = None
    try:
        modifyTime = os.path.getmtime(curFileOrPath) # 159...
        updateTime = int(modifyTime * 1000) # 1593748641327
    except OSError as err:
        errMsg = str(err)
        # print("errMsg=%s" % errMsg)
        pass

    return updateTime
```

详见：

[【已解决】Python中获取文件最后更新时间](#)

提取文件名后缀

```
def extractSuffix(fileNameOrUrl):
    .....
    extract file suffix from name or url
    eg:
    https://cdn2.qupeiyin.cn/2018-09-10/15365514898246.mp4 -> r
        15365514894833.srt -> srt
    .....
    return fileNameOrUrl.split('.')[ -1 ]
```

创建空文件

文件

```
import os

def createEmptyFile(fullFilename):
    """Create a empty file like touch"""
    filePath = os.path.dirname(fullFilename)
    # create folder if not exist
    if not os.path.exists(filePath):
        os.makedirs(filePath)

    with open(fullFilename, 'a'):
        # Note: not use 'w' for maybe conflict for others
        os.utime(fullFilename, None)
```

读取文件二进制数据

```
def readBinDataFromFile(fileToRead):
    """Read binary data from file"""
    binaryData = None
    try:
        readFp = open(fileToRead, "rb")
        binaryData = readFp.read()
        readFp.close()
    except:
        binaryData = None

    return binaryData
```

调用：

```
imgBinData = readBinDataFromFile(imageFullPath)
```

保存二进制数据到文件

```
def saveDataToFile(fullFilename, binaryData):
    """save binary data info file"""
    with open(fullFilename, 'wb') as fp:
        fp.write(binaryData)
        fp.close()
    # print("Complete save file %s" % fullFilename)
```

保存json到文件

```
import json
import codecs

def saveJsonToFile(fullFilename, jsonValue):
    """save json dict into file"""
    with codecs.open(fullFilename, 'w', encoding="utf-8") as jsonFp:
        json.dump(jsonValue, jsonFp, indent=2, ensure_ascii=True)
        # print("Complete save json %s" % fullFilename)
```

从文件中加载出json

```
import json
import codecs

def loadJsonFromFile(fullFilename):
    """load and parse json dict from file"""
    with codecs.open(fullFilename, 'r', encoding="utf-8") as jsonFp:
        jsonDict = json.load(jsonFp)
        # print("Complete load json from %s" % fullFilename)
    return jsonDict
```

通过二进制生成文件类型对象

(1) Python 3

```
import io

audioBinaryData = audioObj.read()
audioFileLikeObj = io.BytesIO(audioBinaryData)
```

得到对应的文件类型的对象的: `<_io.BytesIO object at 0x115964468>` , 即可去像操作文件一样去操作这个io。

(2) Python 2

```
import StringIO

audioFileLikeObj = StringIO.StringIO()
audioFileLikeObj.write(audioBinaryData)
```

一行代码把字符串写入保存到文件

文件

```
open(fullFilePath, "w").write(fileContentStr).close()
```

举例：

```
open("0408_1600.xml", "w").write(page).close()
```

给文件增加可执行权限：实现 chmod +x 的效果

代码：

```
import os
import stat

curState = os.stat(someFile)
newState = curState.st_mode | stat.S_IEXEC
os.chmod(someFile, newState)
```

再去优化为函数：

```
import os
import stat

def chmodAddX(someFile):
    """add file executable mode, like chmod +x

    Args:
        someFile (str): file full path
    Returns:
        soup
    Raises:
        None
    if os.path.exists(someFile):
        if os.path.isfile(someFile):
            # add executable
            curState = os.stat(someFile)
            newState = curState.st_mode | stat.S_IEXEC
            os.chmod(someFile, newState)
```

调用：

```
chmodAddX(shellFullPath)
```

继续优化：

参考 [How do you do a simple "chmod +x" from within python? - Stack Overflow](#)

如果想要加上，给任何人都有可执行权限，则可以用：

```
def chmodAddX(someFile):
    """add file executable mode, like chmod +x

    Args:
        someFile (str): file full path
    Returns:
        soup
    Raises:
        None

    if os.path.exists(someFile):
        if os.path.isfile(someFile):
            # add executable
            curState = os.stat(someFile)
            # STAT_OWNER_EXECUTABLE = stat.S_IEXEC
            # executableMode = STAT_OWNER_EXECUTABLE
            STAT_EVERYONE_EXECUTABLE = stat.S_IXUSR | stat.S_IXGRP | stat.S_IXOTH
            executableMode = STAT_EVERYONE_EXECUTABLE
            newState = curState.st_mode | executableMode
            os.chmod(someFile, newState)
```

效果：

- 之前：`-rw-r--r--`
- 之后：`-rwxr-xr-x`
 - 给 user group other 都加上 x 的可执行权限

详见：

[【已解决】Python中给Mac中文件加上可执行权限](#)

判断是否是文件对象

文件

```
import sys

def isFileObject(fileObj):
    """check is file like object or not"""
    if sys.version_info[0] == 2:
        return isinstance(fileObj, file)
    else:
        # for python 3:
        # has read() method for:
        # io.IOBase
        # io.BytesIO
        # io.StringIO
        # io.RawIOBase
        return hasattr(fileObj, 'read')
```

计算当前文件名，如果重名，则位数加1

文件

```
import os
import re

def findNextNumberFilename(curFilename):
    """Find the next available filename from current name

    Args:
        curFilename (str): current filename
    Returns:
        next available (not existed) filename
    Raises:
    Examples:
        (1) 'crifanLib/demo/input/image/20201219_172616_dra
            not exist -> 'crifanLib/demo/input/image/20201219_172617_dra
        (2) 'crifanLib/demo/input/image/20191219_172616_dra
            exists -> next until not exist 'crifanLib/demo/input/...
        ....
    newFilename = curFilename

    newPathRootPart, pointSuffix = os.path.splitext(newFilename)
    # 'crifanLib/demo/input/image/20191219_172616_drawRect'
    filenamePrefix = newPathRootPart
    while os.path.exists(newFilename):
        newTailNumberInt = 1
        foundTailNumber = re.search("^(?P<filenamePrefix>..)(?P<tailNumber>[0-9]+)\.(?P<pointSuffix>[.].*)$",
                                    newFilename)
        if foundTailNumber:
            tailNumberStr = foundTailNumber.group("tailNumber")
            tailNumberInt = int(tailNumberStr)
            newTailNumberInt = tailNumberInt + 1 # 2
            filenamePrefix = foundTailNumber.group("filenamePrefix")
            # existed previously saved, change to new name
            newPathRootPart = "%s_%s" % (filenamePrefix, newTailNumberInt)
            # 'crifanLib/demo/input/image/20191219_172616_drawRect'
            newFilename = newPathRootPart + pointSuffix
            # 'crifanLib/demo/input/image/20191219_172616_drawRect'

    return newFilename
```

调用：

```
notExistFile = "crifanLib/demo/input/image/some_not_exist"
nextFilename = findNextNumberFilename(notExistFile)
print("notExistFile=%s -> nextFilename=%s" % (notExistFile, nextFilename))
# notExistFile=crifanLib/demo/input/image/some_not_exist
# nextFilename=crifanLib/demo/input/image/some_not_exist_1

realExistFile = "crifanLib/demo/input/image/20191219_172616_drawRect"
nextUntilNotExistFilename = findNextNumberFilename(realExistFile)
print("realExistFile=%s -> nextUntilNotExistFilename=%s" % (realExistFile, nextUntilNotExistFilename))
# realExistFile=crifanLib/demo/input/image/20191219_172616_drawRect
# nextUntilNotExistFilename=crifanLib/demo/input/image/20191219_172617_drawRect
```

从文件名后缀推断出MIME类型

用库：

- mime
 - GitHub
 - <https://github.com/liluo/mime>

安装mime：

```
pip install mime
```

代码：

```
import mime

fileMimeType = mime.Types.of(curAudioFullFilename)[0].content_type
```

- 输入文件： 'Lots of Hearts.mp3'
- 输出信息： 'audio/mpeg'

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： 2020-07-11 10:18:00

文件夹=文件路径

新建文件夹

对于: python 3.2+

```
import os

def createFolder(folderFullPath):
    """
        create folder, even if already existed
        Note: for Python 3.2+
    """
    os.makedirs(folderFullPath, exist_ok=True)
    # print("Created folder: %s" % folderFullPath)
```

或:

对于: python 3.5+

```
import pathlib
pathlib.Path('/my/directory').mkdir(parents=True, exist_ok=True)
```

批量删除非空文件夹

```
import shutil

if os.path.exists(folderToDelete):
    shutil.rmtree(folderToDelete)
```

注意:

- 删除之前要先用 `os.path.exists` 判断是非存在该目录
 - 如果不存在就去删除, 则会报错: `OSError: [Errno 2] No such file or directory`

os.path 路径处理

```

#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author: Crifan Li
# Update: 20191219
# Function: Demo os.path common used functions

import os

def osPathDemo():
    currentSystemInfo = os.uname()
    print("currentSystemInfo=%s" % (currentSystemInfo, ))
    # currentSystemInfo=posix.uname_result(sysname='Darwin'

    pathSeparatorInCurrentOS = os.path.sep
    print("pathSeparatorInCurrentOS=%s" % pathSeparatorInC
    # pathSeparatorInCurrentOS=/

    fullFilePath = "/Users/limao/dev/crifan/python/notEnoughUnpack/Snip20191212_113.png"
    print("fullFilePath=%s" % fullFilePath)
    # fullFilePath=/Users/limao/dev/crifan/python/notEnoughUnpack/Snip20191212_113.png

    dirname = os.path.dirname(fullFilePath)
    print("dirname=%s" % dirname)
    # dirname=/Users/limao/dev/crifan/python/notEnoughUnpack
    basename = os.path.basename(fullFilePath)
    print("basename=%s" % basename)
    # basename=Snip20191212_113.png
    joinedFullPath = os.path.join(dirname, basename)
    print("joinedFullPath=%s" % joinedFullPath)
    # joinedFullPath=/Users/limao/dev/crifan/python/notEnoughUnpack/Snip20191212_113.png
    isSame = (fullFilePath == joinedFullPath)
    print("isSame=%s" % isSame)
    # isSame=True

    root, pointSuffix = os.path.splitext(fullFilePath)
    print("root=%s, pointSuffix=%s" % (root, pointSuffix))
    # root=/Users/limao/dev/crifan/python/notEnoughUnpack/
    head, tail = os.path.split(fullFilePath)
    print("head=%s, tail=%s" % (head, tail))
    # head=/Users/limao/dev/crifan/python/notEnoughUnpack,
    drive, tail = os.path.splitdrive(fullFilePath)
    print("drive=%s, tail=%s" % (drive, tail))
    # drive=, tail=/Users/limao/dev/crifan/python/notEnoughUnpack/Snip20191212_113.png

    curPath = os.getcwd()
    print("curPath=%s" % curPath)
    # curPath=/Users/limao/dev/crifan/python
    relativePath = os.path.relpath(fullFilePath)
    print("relativePath=%s" % relativePath)
    # relativePath=notEnoughUnpack/Snip20191212_113.png

```

文件

```
isFile = os.path.isfile(fullFilePath)
print("isFile=%s" % isFile)
# isFile=True

isDirectory = os.path.isdir(fullFilePath)
print("isDirectory=%s" % isDirectory)
# isDirectory=False

fileSize = os.path.getsize(fullFilePath)
print("fileSize=%s" % fileSize)
# fileSize=368810

isFileOrFolderRealExist = os.path.exists(fullFilePath)
print("isFileOrFolderRealExist=%s" % isFileOrFolderRealExist)
# isFileOrFolderRealExist=True

if __name__ == "__main__":
    osPathDemo()
```

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: 2020-06-27 18:55:42

多媒体

详见：

- <https://github.com/crifan/crifanLibPython/blob/master/crifanLib/crifanMultimedia.py>
 - <https://github.com/crifan/crifanLibPython/blob/master/crifanLib/demo/crifanMultimediaDemo.py>
-

此处整理多媒体相关的常用Python代码段，主要包含如下内容：

- 图片=图像
- 音频
- 视频

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：2020-07-11 09:24:39

图像

Python中图像处理用的最多是： Pillow

下面图像处理处理的代码，基本上都是用 Pillow 实现的。

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： 2020-07-11 09:23:36

Pillow

- Pillow
 - 继承自： PIL
 - PIL = Python Imaging Library
 - 官网资料：
 - [Image Module — Pillow \(PIL Fork\) 7.0.0 documentation](#)
 - [Image Module — Pillow \(PIL Fork\) 3.1.2 documentation](#)

从二进制生成Image

```
if isinstance(inputImage, bytes):  
    openableImage = io.BytesIO(inputImage)  
    curPillowImage = Image.open(openableImage)
```

pillow变量是：

```
# <PIL.PngImagePlugin.PngImageFile image mode=RGBA size=354x288>  
# <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1000x1000>
```

详见：

- 【已解决】Python如何从二进制数据中生成Pillow的Image
- 【已解决】Python的Pillow如何从二进制数据中读取图像数据

从Pillow的Image获取二进制数据

```
import io  
  
imageIO = io.BytesIO()  
curImg.save(imageIO, curImg.format)  
imgBytes = imageIO.getvalue()
```

详见：

【已解决】Python的Pillow如何返回图像的二进制数据

缩放图片

```

import io
from PIL import Image, ImageDraw

def resizeImage(inputImage,
                newSize,
                resample=Image.BICUBIC, # Image.LANCZOS,
                outputFormat=None,
                outputImageFile=None
                ):
    """
        resize input image
        resize normally means become smaller, reduce size
    :param inputImage: image file object(fp) / filename / bytes
    :param newSize: (width, height)
    :param resample: PIL.Image.NEAREST, PIL.Image.BILINEAR,
                     https://pillow.readthedocs.io/en/stable/reference/
    :param outputFormat: PNG/JPEG/BMP/GIF/TIFF/WebP/..., more
                         https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html#file-formats
                         if input image is filename with suffix, can omit this
    :param outputImageFile: output image file filename
    :return:
        input image file filename: output resized image to
        input image binary bytes: resized image binary bytes
    """

    openableImage = None
    if isinstance(inputImage, str):
        openableImage = inputImage
    elif isinstance(inputImage):
        openableImage = inputImage
    elif isinstance(inputImage, bytes):
        inputImageLen = len(inputImage)
        openableImage = io.BytesIO(inputImage)

    imageFile = Image.open(openableImage) # <PIL.PngImagePlugin.PngImageFile>
    imageFile.thumbnail(newSize, resample)
    if outputImageFile:
        # save to file
        imageFile.save(outputImageFile)
        imageFile.close()
    else:
        # save and return binary byte
        imageOutput = io.BytesIO()
        # imageFile.save(imageOutput)
        outputImageFormat = None
        if outputFormat:
            outputImageFormat = outputFormat
        elif imageFile.format:
            outputImageFormat = imageFile.format
        imageFile.save(imageOutput, outputImageFormat)
        imageFile.close()

```

文件

```
compressedImageBytes = imageOutput.getvalue()
compressedImageLen = len(compressedImageBytes)
compressRatio = float(compressedImageLen)/float(inputImageLen)
print("%s -> %s, resize ratio: %d%%" % (inputImageFile,
                                             outputImageFile, compressRatio))
return compressedImageBytes
```

调用：

```

import sys
import os
curFolder = os.path.abspath(__file__)
parentFolder = os.path.dirname(curFolder)
parentParentFolder = os.path.dirname(parentFolder)
parentParentParentFolder = os.path.dirname(parentParentFolder)
sys.path.append(curFolder)
sys.path.append(parentFolder)
sys.path.append(parentParentFolder)
sys.path.append(parentParentParentFolder)

import datetime
from crifanMultimedia import resizeImage

def testFilename():
    imageFilename = "/Users/crifan/dev/tmp/python/resize_image.jpg"
    outputImageFilename = "/Users/crifan/dev/tmp/python/resize_image_out.jpg"
    print("imageFilename=%s" % imageFilename)
    beforeTime = datetime.datetime.now()
    resizeImage(imageFilename, (300, 300), outputImageFile=outputImageFilename)
    afterTime = datetime.datetime.now()
    print("procesTime: %s" % (afterTime - beforeTime))

    outputImageFilename = "/Users/crifan/dev/tmp/python/resize_image.jpg"
    beforeTime = datetime.datetime.now()
    resizeImage(imageFilename, (800, 800), outputImageFile=outputImageFilename)
    afterTime = datetime.datetime.now()
    print("procesTime: %s" % (afterTime - beforeTime))

def testFileObject():
    imageFilename = "/Users/crifan/dev/tmp/python/resize_image.jpg"
    imageFileObj = open(imageFilename, "rb")
    outputImageFilename = "/Users/crifan/dev/tmp/python/resize_image_out.jpg"
    beforeTime = datetime.datetime.now()
    resizeImage(imageFileObj, (600, 600), outputImageFile=outputImageFilename)
    afterTime = datetime.datetime.now()
    print("procesTime: %s" % (afterTime - beforeTime))

def testBinaryBytes():
    imageFilename = "/Users/crifan/dev/tmp/python/resize_image.jpg"
    imageFileObj = open(imageFilename, "rb")
    imageBytes = imageFileObj.read()
    # return binary bytes
    beforeTime = datetime.datetime.now()
    resizedImageBytes = resizeImage(imageBytes, (800, 800))
    afterTime = datetime.datetime.now()
    print("procesTime: %s" % (afterTime - beforeTime))
    print("len(resizedImageBytes)=%s" % len(resizedImageBytes))

    # save to file

```

文件

```
outputImageFilename = "/Users/crifan/dev/tmp/python/resize_image.py"
beforeTime = datetime.datetime.now()
resizeImage(imageBytes, (750, 750), outputImageFile=outputImageFilename)
afterTime = datetime.datetime.now()
print("procesTime: %s" % (afterTime - beforeTime))

imageFileObj.close()

def demoResizeImage():
    testFilename()
    testFileObject()
    testBinaryBytes()

if __name__ == "__main__":
    demoResizeImage()

# imageFilename=/Users/crifan/dev/tmp/python/resize_image.py
# procesTime: 0:00:00.619377
# procesTime: 0:00:00.745228
# procesTime: 0:00:00.606060
# 1146667 -> 753258, resize ratio: 65%
# procesTime: 0:00:00.773289
# len(resizedImageBytes)=753258
# procesTime: 0:00:00.738237
```

给图片画元素所属区域的边框，且带自动保存加了框后的图片

```

from PIL import Image
from PIL import ImageDraw

def imageDrawRectangle(inputImgOrImgPath,
    rectLocation,
    outlineColor="green",
    outlineWidth=0,
    isShow=False,
    isAutoSave=True,
    saveTail="_drawRect_%wx%h",
    isDrawClickedPosCircle=True,
    clickedPos=None,
):
    """Draw a rectangle for image (and a small circle), and
    save the image file.

    Args:
        inputImgOrImgPath (Image/str): a pillow(PIL) Image
        rectLocation (tuple/list/Rect): the rectangle location
        outlineColor (str): Color name
        outlineWidth (int): rectangle outline width
        isShow (bool): True to call image.show() for debug
        isAutoSave (bool): True to auto save the image file
        saveTail(str): save filename tail part. support for
        clickedPos (tuple): x,y of clicked position; default None
        isDrawClickedPosCircle (bool): draw small circle if
    Returns:
        modified image
    Raises:
    """
    inputImg = inputImgOrImgPath
    if isinstance(inputImgOrImgPath, str):
        inputImg = Image.open(inputImgOrImgPath)
    draw = ImageDraw.Draw(inputImg)

    isRectObj = False
    hasX = hasattr(rectLocation, "x")
    hasY = hasattr(rectLocation, "y")
    hasWidth = hasattr(rectLocation, "width")
    hasHeight = hasattr(rectLocation, "height")
    isRectObj = hasX and hasY and hasWidth and hasHeight
    if isinstance(rectLocation, tuple):
        x, y, w, h = rectLocation
    if isinstance(rectLocation, list):
        x = rectLocation[0]
        y = rectLocation[1]
        w = rectLocation[2]
        h = rectLocation[3]
    elif isRectObj:
        x = rectLocation.x
        y = rectLocation.y

```

文件

```
w = rectLocation.width
h = rectLocation.height

w = int(w)
h = int(h)
x0 = x
y0 = y
x1 = x0 + w
y1 = y0 + h
draw.rectangle(
    [x0, y0, x1, y1],
    # fill="yellow",
    # outline="yellow",
    outline=outlineColor,
    width=outlineWidth,
)

if isDrawClickedPosCircle:
    # radius = 3
    # radius = 2
    radius = 4
    # circleOutline = "yellow"
    circleOutline = "red"
    circleLineWidthInt = 1
    # circleLineWidthInt = 3

    if clickedPos:
        clickedX, clickedY = clickedPos
    else:
        clickedX = x + w/2
        clickedY = y + h/2
    startPointInt = (int(clickedX - radius), int(clickedY - radius))
    endPointInt = (int(clickedX + radius), int(clickedY + radius))
    draw.ellipse([startPointInt, endPointInt], outline=circleOutline)

if isShow:
    inputImg.show()

if isAutoSave:
    saveTail = saveTail.replace("%x", str(x))
    saveTail = saveTail.replace("%y", str(y))
    saveTail = saveTail.replace("%w", str(w))
    saveTail = saveTail.replace("%h", str(h))

inputImgPath = None
if isinstance(inputImgOrImgPath, str):
    inputImgPath = str(inputImgOrImgPath)
elif inputImg.filename:
    inputImgPath = str(inputImg.filename)

if inputImgPath:
```

文件

```
    imgFolderAndName, pointSuffix = os.path.split(imgFolderAndName)
    imgFolderAndName = imgFolderAndName + saveTail
    newImgPath = imgFolderAndName + pointSuffix
    newImgPath = findNextNumberFilename(newImgPath)
else:
    curDatetimeStr = getCurDatetimeStr() # '201912101010'
    suffix = str(inputImg.format).lower() # 'jpeg'
    newImgFilename = "%s%s.%s" % (curDatetimeStr, suffix)
    imgPathRoot = os.getcwd()
    newImgPath = os.path.join(imgPathRoot, newImgFilename)

    inputImg.save(newImgPath)

return inputImg
```

说明：

相关函数，详见：findNextNumberFilename，或者干脆去掉这个逻辑即可。

调用：

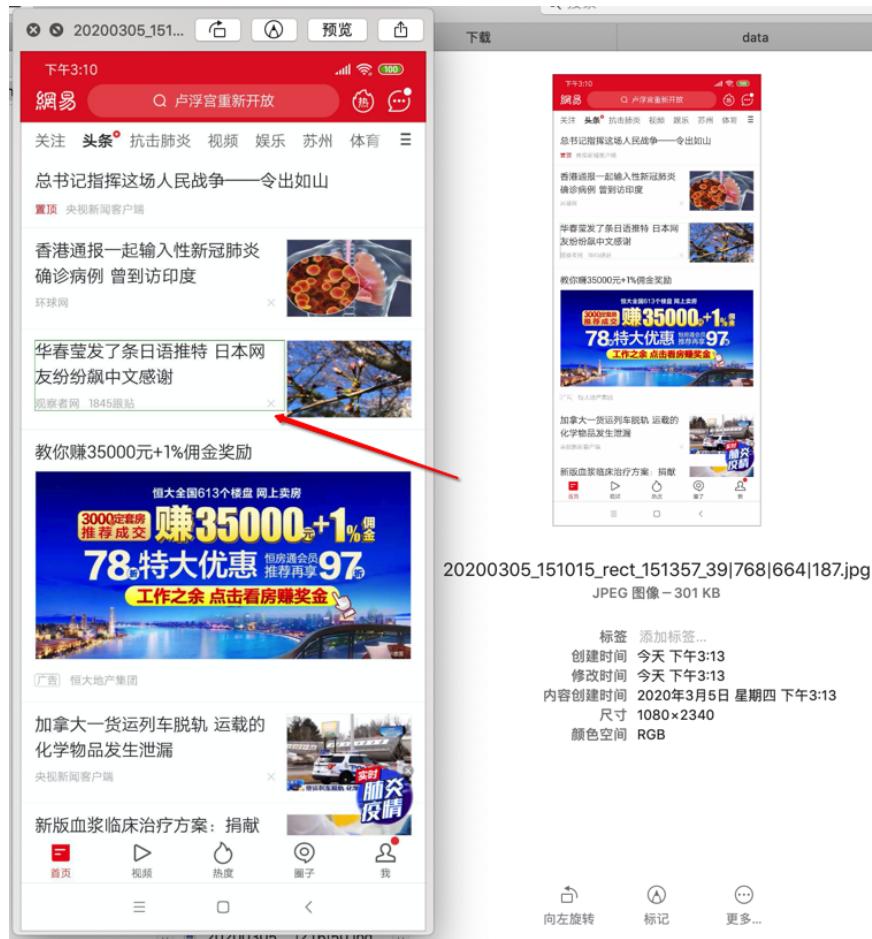
```
curBoundList = self.getElementBounds(eachElement)
curWidth = curBoundList[2] - curBoundList[0]
curHeight = curBoundList[3] - curBoundList[1]
curRect = [curBoundList[0], curBoundList[1], curWidth, curHeight]
curImg = CommonUtils.imageDrawRectangle(curImg, curRect, isRed)
```

或：

```
curTimeStr = CommonUtils.getCurDatetimeStr("%H%M%S")
curSaveTail = "_rect_{0}_{1}{2}|{3}|{4}|{5}{6}.format(curTimeStr)
curImg = CommonUtils.imageDrawRectangle(imgPath, curRect, isRed)
```

效果：

(1) 给原图加上单个元素所属边框



(2) 多次循环后，给同一张图中多个元素加上边框后



20200305_145916_rect_121|
401|232|38_1.jpg
JPEG 图像 – 309 KB

标签 添加标签...
创建时间 今天 下午3:02
修改时间 今天 下午3:02
内容创建时间 2020/3/5
尺寸 1080×2340
颜色空间 RGB

其他调用：

```
imageDrawRectangle(curPillowImg, curLocation)

imageDrawRectangle(curPillowImg, calculatedLocation)

curImg = imageDrawRectangle(imgPath, firstMatchLocation, c)

curImg = imageDrawRectangle(imgPath, firstMatchLocation)
```

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：2020-07-11 09:25:40

百度OCR

详见：

<https://github.com/crifan/crifanLibPython/blob/master/crifanLib/crifanBaiduOcr.py>

在做安卓和iOS的移动端自动化测试期间，会涉及到从图像中提取文字，用的是百度OCR。

其中有些通用的功能，整理出函数，贴出供参考。

百度OCR初始化

```
import os
import re
import base64
import requests
import time
import logging
from collections import OrderedDict
from PIL import Image, ImageDraw

class BaiduOCR():
    # OCR_URL = "https://aip.baidubce.com/rest/2.0/ocr/v1/general"
    # OCR_URL = "https://aip.baidubce.com/rest/2.0/ocr/v1/general_text"
    # OCR_URL = "https://aip.baidubce.com/rest/2.0/ocr/v1/accurate"
    OCR_URL = "https://aip.baidubce.com/rest/2.0/ocr/v1/accurate"

    TOKEN_URL = 'https://aip.baidubce.com/oauth/2.0/token'

    RESP_ERR_CODE_QPS_LIMIT_REACHED = 18
    RESP_ERR_TEXT_QPS_LIMIT_REACHED = "Open api qps request limit reached"

    RESP_ERR_CODE_DAILY_LIMIT_REACHED = 17
    RESP_ERR_TEXT_DAILY_LIMIT_REACHED = "Open api daily request limit reached"

    API_KEY = 'S0xxxxxxxxxxhu'
    SECRET_KEY = 'wLxxxxxxxxxxxxxxxxxxxxpL'

    def initOcr(self):
        self.curToken = self.baiduFetchToken()

    def baiduFetchToken(self):
        """Fetch Baidu token for OCR"""
        params = {
            'grant_type': 'client_credentials',
            'client_id': self.API_KEY,
            'client_secret': self.SECRET_KEY
        }

        resp = requests.get(self.TOKEN_URL, params=params)
        respJson = resp.json()

        respToken = ""
```

文件

```
if ('access_token' in respJson.keys() and 'scope' :  
    if not 'brain_all_scope' in respJson['scope']:  
        logging.error('please ensure has check the  
    else:  
        respToken = respJson['access_token']  
    else:  
        logging.error('please overwrite the correct API')  
  
# '24.8691f3c6dedd0d0d0b30a9dfec604d52.2592000.1578  
return respToken
```

百度OCR图片转文字

```

def baiduImageToWords(self, imageFullPath):
    """Detect text from image using Baidu OCR api"""

    # # Note: if using un-paid = free baidu api, need follow
    # time.sleep(0.15)

    respWordsResutJson = ""

    # 读取图片二进制数据
    imgBinData = readBinDataFromFile(imageFullPath)
    encodedImgData = base64.b64encode(imgBinData)

    paramDict = {
        "access_token": self.curToken
    }

    headerDict = {
        "Content-Type": "application/x-www-form-urlencoded"
    }

    # 参数含义: http://ai.baidu.com/ai-doc/OCR/vk3h7y58v
    dataDict = {
        "image": encodedImgData,
        "recognize_granularity": "small",
        # "vertexes_location": "true",
    }
    resp = requests.post(self.OCR_URL, params=paramDict, headers=headerDict, data=dataDict)
    respJson = resp.json()

    logging.debug("baidu OCR: imgage=%s -> respJson=%s", imageFullPath, respJson)

    if "error_code" in respJson:
        logging.warning("respJson=%s" % respJson)
        errorCode = respJson["error_code"]
        # {'error_code': 17, 'error_msg': 'Open api daily limit reached'}
        # {'error_code': 18, 'error_msg': 'Open api qps reached, the limit count can found from https://ai.baidu.com/ai-doc/OCR/fk3h7xu7h'}
        # 文字识别 - 免费额度 | 百度AI开放平台
        # for "通用文字识别（高精度含位置版）" is "50次/天"
        if errorCode == self.RESP_ERR_CODE_QPS_LIMIT_REACHED:
            # wait sometime and try again
            time.sleep(1.0)
            resp = requests.post(self.OCR_URL, params=paramDict, headers=headerDict, data=dataDict)
            respJson = resp.json()
            logging.debug("baidu OCR: for errorCode=%s, do it again" % errorCode)
        elif errorCode == self.RESP_ERR_CODE_DAILY_LIMIT_REACHED:
            logging.error("Fail to continue using baidu OCR")
            respJson = None

```

```
....  
{  
    "log_id": 6937531796498618000,  
    "words_result_num": 32,  
    "words_result": [  
        {  
            "chars": [  
                ...  
            ....  
            if "words_result" in respJson:  
                respWordsResutJson = respJson  
  
        return respWordsResutJson
```

调用：

```
wordsResultJson = self.baiduImageToWords(imgPath)  
  
respJson = self.baiduImageToWords(screenImgPath)
```

返回结果举例

安卓游戏 暗黑觉醒 首充豪礼

图片：



返回解析后出 json 格式的文字信息：

文件

```
{  
    "log_id": 9009770747370640007,  
    "words_result_num": 12,  
    "words_result": [  
        {  
            "chars": [  
                {  
                    "char": "首",  
                    "location": { "width": 94, "top": 105, "left": 989 },  
                },  
                {  
                    "char": "充",  
                    "location": { "width": 94, "top": 105, "left": 1061 },  
                },  
                {  
                    "char": "豪",  
                    "location": { "width": 95, "top": 105, "left": 1136 },  
                },  
                {  
                    "char": "礼",  
                    "location": { "width": 77, "top": 105, "left": 1213 },  
                }  
            ],  
            "location": { "width": 370, "top": 105, "left": 989 },  
            "words": "首充豪礼"  
        },  
        {  
            "chars": [  
                {  
                    "char": "x",  
                    "location": { "width": 30, "top": 161, "left": 1887 },  
                }  
            ],  
            "location": { "width": 60, "top": 161, "left": 1887 },  
            "words": "x"  
        },  
        {  
            "chars": [  
                {  
                    "char": "充",  
                    "location": { "width": 43, "top": 273, "left": 756 },  
                },  
                {  
                    "char": "值",  
                    "location": { "width": 43, "top": 273, "left": 809 },  
                },  
                {  
                    "char": "元",  
                    "location": { "width": 67, "top": 273, "left": 922 },  
                }  
            ]  
        }  
    ]  
}
```

文件

```
{
  {
    "char": "可",
    "location": { "width": 44, "top": 273, "left": 97 },
  },
  {
    "char": "领",
    "location": { "width": 43, "top": 273, "left": 106 },
  },
  {
    "char": "总",
    "location": { "width": 44, "top": 273, "left": 115 },
  },
  {
    "char": "价",
    "location": { "width": 23, "top": 273, "left": 124 },
  },
  {
    "char": "值",
    "location": { "width": 89, "top": 273, "left": 133 },
  },
  {
    "char": "8",
    "location": { "width": 36, "top": 273, "left": 142 },
  },
  {
    "char": "8",
    "location": { "width": 36, "top": 273, "left": 151 },
  },
  {
    "char": "8",
    "location": { "width": 35, "top": 273, "left": 160 },
  },
  {
    "char": "钻",
    "location": { "width": 43, "top": 273, "left": 169 },
  },
  {
    "char": "豪",
    "location": { "width": 44, "top": 273, "left": 178 },
  },
  {
    "char": "华",
    "location": { "width": 43, "top": 273, "left": 187 },
  },
  {
    "char": "大",
    "location": { "width": 43, "top": 273, "left": 196 },
  },
  {
    "char": "礼",
    "location": { "width": 27, "top": 273, "left": 205 }
```

文件

```
        },
    ],
    "location": { "width": 911, "top": 273, "left": 758,
    "words": "充值元可领总价值888钻豪华大礼"
},
{
    "chars": [
        {
            "char": "送",
            "location": { "width": 65, "top": 369, "left": 821 },
        }
    ],
    "location": { "width": 107, "top": 369, "left": 832,
    "words": "送"
},
{
    "chars": [
        {
            "char": "绝",
            "location": { "width": 38, "top": 390, "left": 971 },
        },
        {
            "char": "版",
            "location": { "width": 38, "top": 390, "left": 1009 },
        },
        {
            "char": "萌",
            "location": { "width": 38, "top": 390, "left": 1047 },
        },
        {
            "char": "宠",
            "location": { "width": 38, "top": 390, "left": 1085 },
        },
        {
            "char": "\u00b7",
            "location": { "width": 31, "top": 390, "left": 1123 },
        },
        {
            "char": "\u00b2",
            "location": { "width": 39, "top": 390, "left": 1161 },
        },
        {
            "char": "属",
            "location": { "width": 38, "top": 390, "left": 1199 },
        },
        {
            "char": "神",
            "location": { "width": 38, "top": 390, "left": 1237 },
        },
        {
            "char": "兵",
            "location": { "width": 38, "top": 390, "left": 1275 },
        }
    ]
}
```

```
        "location": { "width": 39, "top": 390, "left": 140 },
    ],
    "location": { "width": 524, "top": 390, "left": 934,
        "words": "绝版萌宠、专属神兵"
    },
    {
        "chars": [
            {
                "char": "绝",
                "location": { "width": 20, "top": 515, "left": 378 },
            },
            "location": { "width": 33, "top": 515, "left": 378,
                "words": "绝"
            },
            {
                "chars": [
                    {
                        "char": "\u20e3",
                        "location": { "width": 33, "top": 516, "left": 1992 },
                    },
                    "location": { "width": 33, "top": 516, "left": 1992,
                        "words": "\u20e3"
                    },
                    {
                        "chars": [
                            {
                                "char": "版",
                                "location": { "width": 20, "top": 545, "left": 379 },
                            },
                            "location": { "width": 31, "top": 545, "left": 379,
                                "words": "版"
                            },
                            {
                                "chars": [
                                    {
                                        "char": "\u20e3",
                                        "location": { "width": 26, "top": 776, "left": 125 },
                                    },
                                    {
                                        "char": "外",
                                        "location": { "width": 26, "top": 776, "left": 125 },
                                    },
                                    {
                                        "char": "\u20e3",
                                        "location": { "width": 27, "top": 776, "left": 125 },
                                    },
                                    {
                                        "char": "\u20e3",
                                        "location": { "width": 27, "top": 776, "left": 125 },
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        ]
    }
}
```

文件

```
        "char": "包",
        "location": { "width": 27, "top": 776, "left": 1225 },
    ],
    "location": { "width": 125, "top": 776, "left": 1225 },
    "words": "额外礼包"
},
{
    "chars": [
        {
            "char": "首",
            "location": { "width": 38, "top": 830, "left": 935 },
        },
        {
            "char": "充",
            "location": { "width": 38, "top": 830, "left": 935 },
        },
        {
            "char": "元",
            "location": { "width": 38, "top": 830, "left": 1065 },
        },
        {
            "char": "充",
            "location": { "width": 38, "top": 830, "left": 1225 },
        },
        {
            "char": "9",
            "location": { "width": 31, "top": 830, "left": 1225 },
        },
        {
            "char": "8",
            "location": { "width": 31, "top": 830, "left": 1225 },
        },
        {
            "char": "元",
            "location": { "width": 38, "top": 830, "left": 1455 },
        }
    ],
    "location": { "width": 549, "top": 830, "left": 935 },
    "words": "首充元充98元"
},
{
    "chars": [
        {
            "char": "战",
            "location": { "width": 42, "top": 970, "left": 375 },
        },
        {
            "char": "斗",
            "location": { "width": 42, "top": 970, "left": 425 },
        },
    ],
}
```

文件

```
{
  "char": "1",
  "location": { "width": 35, "top": 970, "left": 52 },
},
{
  "char": "5",
  "location": { "width": 35, "top": 970, "left": 52 },
},
{
  "char": "0",
  "location": { "width": 34, "top": 970, "left": 52 },
},
{
  "char": "0",
  "location": { "width": 35, "top": 970, "left": 62 },
},
{
  "char": "0",
  "location": { "width": 34, "top": 970, "left": 62 },
}
],
"location": { "width": 327, "top": 970, "left": 373,
"words": "战斗15000"
},
{
  "chars": [
    {
      "char": "战",
      "location": { "width": 43, "top": 969, "left": 106 },
    },
    {
      "char": "斗",
      "location": { "width": 43, "top": 969, "left": 110 },
    },
    {
      "char": "1",
      "location": { "width": 36, "top": 969, "left": 114 },
    },
    {
      "char": "6",
      "location": { "width": 36, "top": 969, "left": 118 },
    },
    {
      "char": "0",
      "location": { "width": 35, "top": 969, "left": 122 },
    },
    {
      "char": "0",
      "location": { "width": 36, "top": 969, "left": 126 },
    },
    {
      "char": "0",
      "location": { "width": 36, "top": 969, "left": 130 },
    }
  ]
}
```

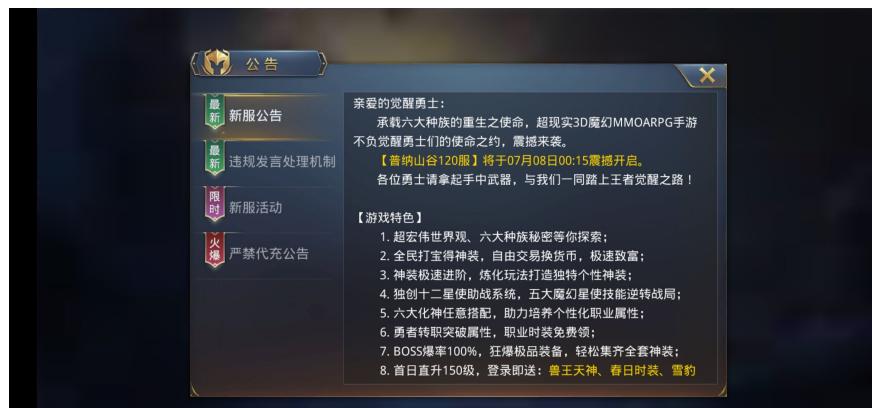
```
        "char": "0",
        "location": { "width": 29, "top": 969, "left": 151 },
    },
    "location": { "width": 330, "top": 969, "left": 1648,
    "words": "战斗16000"
}
]
```

其中：

- 首充豪礼
 - 都能完整检测出来：已经是效果很不错了
 - 当然偶尔也会有失误，比如偶尔
 - 只解析出部分内容：首充豪
 - 或个别字错了：首充豪机
 - 本身图片上 礼 也的确很像 机
 - 作为OCR犯此错误，完全可以理解

安卓游戏 暗黑觉醒 公告弹框

图片：



返回结果json：

```
{  
    "log_id": 2793391773289550472,  
    "words_result_num": 23,  
    "words_result": [  
        {  
            "chars": [  
                {  
                    "char": "公",  
                    "location": { "width": 28, "top": 125, "left": 634 },  
                },  
                {  
                    "char": "告",  
                    "location": { "width": 29, "top": 125, "left": 663 },  
                }  
            ],  
            "location": { "width": 92, "top": 125, "left": 634 },  
            "words": "公告"  
        },  
        {  
            "chars": [  
                {  
                    "char": "最",  
                    "location": { "width": 21, "top": 240, "left": 535 },  
                }  
            ],  
            "location": { "width": 33, "top": 240, "left": 535 },  
            "words": "最"  
        },  
        {  
            "chars": [  
                {  
                    "char": "豪",  
                    "location": { "width": 26, "top": 233, "left": 962 },  
                },  
                {  
                    "char": "爱",  
                    "location": { "width": 26, "top": 233, "left": 962 },  
                },  
                {  
                    "char": "的",  
                    "location": { "width": 26, "top": 233, "left": 962 },  
                },  
                {  
                    "char": "觉",  
                    "location": { "width": 26, "top": 233, "left": 1081 },  
                },  
                {  
                    "char": "醒",  
                    "location": { "width": 26, "top": 233, "left": 1081 },  
                }  
            ]  
        }  
    ]  
}
```

```
{
    "char": "勇",
    "location": { "width": 26, "top": 233, "left": 10 },
},
{
    "char": "士",
    "location": { "width": 25, "top": 233, "left": 10 },
},
{
    "char": ";",
    "location": { "width": 21, "top": 233, "left": 10 },
}
],
"location": { "width": 253, "top": 233, "left": 922,
"words": "亲爱的觉醒勇士:" },
},
{
    "chars": [
        {
            "char": "新",
            "location": { "width": 26, "top": 266, "left": 52 },
        },
        {
            "char": "新",
            "location": { "width": 27, "top": 266, "left": 58 },
        },
        {
            "char": "服",
            "location": { "width": 26, "top": 266, "left": 64 },
        },
        {
            "char": "公",
            "location": { "width": 27, "top": 266, "left": 66 },
        },
        {
            "char": "告",
            "location": { "width": 26, "top": 266, "left": 70 },
        }
    ],
"location": { "width": 201, "top": 266, "left": 535,
"words": "新客服公告" },
},
{
    "chars": [
        {
            "char": "承",
            "location": { "width": 26, "top": 282, "left": 96 },
        },
        {
            "char": "載",
            "location": { "width": 26, "top": 282, "left": 102 },
        }
    ]
}
```

文件

```
},
{
  "char": "六",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "大",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "种",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "族",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "的",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "重",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "生",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "之",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "使",
  "location": { "width": 27, "top": 282, "left": 10 },
},
{
  "char": "命",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": ",",
  "location": { "width": 21, "top": 282, "left": 10 },
},
{
  "char": "超",
  "location": { "width": 26, "top": 282, "left": 10 },
},
{
  "char": "现",
```

文件

```
        "location": { "width": 26, "top": 282, "left": 14 },
},
{
    "char": "实",
    "location": { "width": 26, "top": 282, "left": 14 },
},
{
    "char": "3",
    "location": { "width": 22, "top": 282, "left": 14 },
},
{
    "char": "D",
    "location": { "width": 21, "top": 282, "left": 14 },
},
{
    "char": "魔",
    "location": { "width": 26, "top": 282, "left": 14 },
},
{
    "char": "M",
    "location": { "width": 61, "top": 284, "left": 14 },
},
{
    "char": "M",
    "location": { "width": 35, "top": 284, "left": 14 },
},
{
    "char": "0",
    "location": { "width": 31, "top": 284, "left": 14 },
},
{
    "char": "A",
    "location": { "width": 25, "top": 284, "left": 14 },
},
{
    "char": "R",
    "location": { "width": 25, "top": 284, "left": 14 },
},
{
    "char": "P",
    "location": { "width": 25, "top": 284, "left": 14 },
},
{
    "char": "G",
    "location": { "width": 25, "top": 284, "left": 14 },
},
{
    "char": "幻",
    "location": { "width": 26, "top": 282, "left": 14 },
},
```

文件

```
        "char": "手",
        "location": { "width": 26, "top": 282, "left": 10 },
    },
    {
        "char": "游",
        "location": { "width": 26, "top": 282, "left": 16 },
    }
],
"location": { "width": 867, "top": 282, "left": 984,
"words": "承载六大种族的重生之使命，超现实3D魔 MMOARPG幻手游"
},
{
    "chars": [
        {
            "char": "不",
            "location": { "width": 37, "top": 337, "left": 984 },
        },
        {
            "char": "负",
            "location": { "width": 23, "top": 337, "left": 984 },
        },
        {
            "char": "觉",
            "location": { "width": 25, "top": 337, "left": 984 },
        },
        {
            "char": "體",
            "location": { "width": 23, "top": 337, "left": 1051 },
        },
        {
            "char": "勇",
            "location": { "width": 23, "top": 337, "left": 1051 },
        },
        {
            "char": "士",
            "location": { "width": 25, "top": 337, "left": 1051 },
        },
        {
            "char": "们",
            "location": { "width": 25, "top": 337, "left": 1051 },
        },
        {
            "char": "的",
            "location": { "width": 25, "top": 337, "left": 1051 },
        },
        {
            "char": "使",
            "location": { "width": 25, "top": 337, "left": 1051 },
        },
        {
            "char": "命",
            "location": { "width": 25, "top": 337, "left": 1051 },
        }
    ]
}
```

```
        "location": { "width": 25, "top": 337, "left": 12 },
    },
    {
        "char": "之",
        "location": { "width": 25, "top": 337, "left": 12 },
    },
    {
        "char": "约",
        "location": { "width": 25, "top": 337, "left": 12 },
    },
    {
        "char": ",",
        "location": { "width": 20, "top": 337, "left": 12 },
    },
    {
        "char": "震",
        "location": { "width": 25, "top": 337, "left": 12 },
    },
    {
        "char": "撼",
        "location": { "width": 25, "top": 337, "left": 12 },
    },
    {
        "char": "来",
        "location": { "width": 23, "top": 337, "left": 12 },
    },
    {
        "char": "袭",
        "location": { "width": 25, "top": 337, "left": 12 },
    },
    {
        "char": "\u2022",
        "location": { "width": 18, "top": 337, "left": 12 }
    }
],
"location": { "width": 580, "top": 337, "left": 923,
"words": "不负觉醒勇士们的使命之约,震撼来袭。"
},
{
    "chars": [
    {
        "char": "最",
        "location": { "width": 20, "top": 364, "left": 535 },
    }
],
"location": { "width": 33, "top": 364, "left": 535,
"words": "最"
},
{
    "chars": [
    {
```

文件

```
        "char": "新",
        "location": { "width": 26, "top": 389, "left": 535 },
    },
    {
        "char": "违",
        "location": { "width": 26, "top": 389, "left": 561 },
    },
    {
        "char": "规",
        "location": { "width": 25, "top": 389, "left": 621 },
    },
    {
        "char": "发",
        "location": { "width": 26, "top": 389, "left": 667 },
    },
    {
        "char": "言",
        "location": { "width": 26, "top": 389, "left": 703 },
    },
    {
        "char": "处",
        "location": { "width": 26, "top": 389, "left": 740 },
    },
    {
        "char": "理",
        "location": { "width": 26, "top": 389, "left": 776 },
    },
    {
        "char": "机",
        "location": { "width": 26, "top": 389, "left": 812 },
    },
    {
        "char": "制",
        "location": { "width": 26, "top": 389, "left": 848 },
    }
],
"location": { "width": 343, "top": 389, "left": 535,
"words": "新规发言处理机制"
},
{
    "chars": [
        {
            "char": "[",
            "location": { "width": 21, "top": 387, "left": 95 },
        },
        {
            "char": "普",
            "location": { "width": 25, "top": 387, "left": 106 },
        },
        {
            "char": "纳",
            "location": { "width": 25, "top": 387, "left": 132 },
        }
    ]
}
```

```
        "location": { "width": 26, "top": 387, "left": 10 },
},
{
    "char": "山",
    "location": { "width": 25, "top": 387, "left": 10 },
},
{
    "char": "谷",
    "location": { "width": 26, "top": 387, "left": 10 },
},
{
    "char": "1",
    "location": { "width": 21, "top": 387, "left": 10 },
},
{
    "char": "2",
    "location": { "width": 21, "top": 387, "left": 10 },
},
{
    "char": "0",
    "location": { "width": 21, "top": 387, "left": 10 },
},
{
    "char": "服",
    "location": { "width": 26, "top": 387, "left": 10 },
},
{
    "char": "]",
    "location": { "width": 21, "top": 387, "left": 10 },
},
{
    "char": "将",
    "location": { "width": 25, "top": 387, "left": 10 },
},
{
    "char": "于",
    "location": { "width": 26, "top": 387, "left": 10 },
},
{
    "char": "0",
    "location": { "width": 20, "top": 387, "left": 10 },
},
{
    "char": "7",
    "location": { "width": 21, "top": 387, "left": 10 },
},
{
    "char": "月",
    "location": { "width": 25, "top": 387, "left": 10 },
},
```

文件

```
        "char": "0",
        "location": { "width": 21, "top": 387, "left": 14 },
    },
    {
        "char": "8",
        "location": { "width": 21, "top": 387, "left": 14 },
    },
    {
        "char": "\u20e3",
        "location": { "width": 25, "top": 387, "left": 14 },
    },
    {
        "char": "0",
        "location": { "width": 21, "top": 387, "left": 14 },
    },
    {
        "char": "0",
        "location": { "width": 21, "top": 387, "left": 14 },
    },
    {
        "char": "\u20e3",
        "location": { "width": 21, "top": 387, "left": 14 },
    },
    {
        "char": "1",
        "location": { "width": 21, "top": 387, "left": 14 },
    },
    {
        "char": "5",
        "location": { "width": 21, "top": 387, "left": 14 },
    },
    {
        "char": "\u20e3",
        "location": { "width": 25, "top": 387, "left": 14 },
    },
    {
        "char": "\u20e3",
        "location": { "width": 26, "top": 387, "left": 14 },
    },
    {
        "char": "\u20e3",
        "location": { "width": 25, "top": 387, "left": 14 },
    },
    {
        "char": "\u20e3",
        "location": { "width": 26, "top": 387, "left": 14 },
    },
    {
        "char": "\u20e3",
        "location": { "width": 21, "top": 387, "left": 14 }
    }
```

文件

```
        ],
        "location": { "width": 711, "top": 387, "left": 997,
        "words": "【普纳山谷120服】将于07月08日00:15震撼开启。"
    },
    {
        "chars": [
            {
                "char": "各",
                "location": { "width": 23, "top": 438, "left": 967 },
            },
            {
                "char": "位",
                "location": { "width": 25, "top": 438, "left": 1000 },
            },
            {
                "char": "勇",
                "location": { "width": 25, "top": 438, "left": 1033 },
            },
            {
                "char": "士",
                "location": { "width": 25, "top": 438, "left": 1066 },
            },
            {
                "char": "请",
                "location": { "width": 25, "top": 438, "left": 1100 },
            },
            {
                "char": "拿",
                "location": { "width": 23, "top": 438, "left": 1133 },
            },
            {
                "char": "起",
                "location": { "width": 25, "top": 438, "left": 1166 },
            },
            {
                "char": "手",
                "location": { "width": 23, "top": 438, "left": 1200 },
            },
            {
                "char": "中",
                "location": { "width": 23, "top": 438, "left": 1233 },
            },
            {
                "char": "武",
                "location": { "width": 25, "top": 438, "left": 1266 },
            },
            {
                "char": "器",
                "location": { "width": 23, "top": 438, "left": 1300 },
            },
        ]
    }
}
```

文件

```
        "char": "、",
        "location": { "width": 20, "top": 438, "left": 16 },
    },
    {
        "char": "亅",
        "location": { "width": 25, "top": 438, "left": 16 },
    },
    {
        "char": "我",
        "location": { "width": 23, "top": 438, "left": 16 },
    },
    {
        "char": "们",
        "location": { "width": 25, "top": 438, "left": 16 },
    },
    {
        "char": "—",
        "location": { "width": 23, "top": 438, "left": 16 },
    },
    {
        "char": "同",
        "location": { "width": 23, "top": 438, "left": 16 },
    },
    {
        "char": "踏",
        "location": { "width": 25, "top": 438, "left": 16 },
    },
    {
        "char": "上",
        "location": { "width": 23, "top": 438, "left": 16 },
    },
    {
        "char": "王",
        "location": { "width": 25, "top": 438, "left": 16 },
    },
    {
        "char": "者",
        "location": { "width": 25, "top": 438, "left": 16 },
    },
    {
        "char": "覺",
        "location": { "width": 23, "top": 438, "left": 16 },
    },
    {
        "char": "體",
        "location": { "width": 25, "top": 438, "left": 16 },
    },
    {
        "char": "之",
        "location": { "width": 23, "top": 438, "left": 16 },
    },
},
```

文件

```
{
    "char": "路",
    "location": { "width": 25, "top": 438, "left": 17 },
},
{
    "char": "!",
    "location": { "width": 20, "top": 438, "left": 18 },
}
],
"location": { "width": 853, "top": 438, "left": 987,
"words": "各位勇士请拿起手中武器,与我们一同踏上王者觉醒之路!" },
{
    "chars": [
        {
            "char": "限",
            "location": { "width": 21, "top": 486, "left": 535 },
        },
        {
            "location": { "width": 35, "top": 486, "left": 535,
"words": "限"
        },
        {
            "chars": [
                {
                    "char": "时",
                    "location": { "width": 26, "top": 512, "left": 535 },
                },
                {
                    "char": "新",
                    "location": { "width": 26, "top": 512, "left": 535 },
                },
                {
                    "char": "服",
                    "location": { "width": 26, "top": 512, "left": 601 },
                },
                {
                    "char": "活",
                    "location": { "width": 25, "top": 512, "left": 601 },
                },
                {
                    "char": "动",
                    "location": { "width": 26, "top": 512, "left": 601 },
                }
            ],
"location": { "width": 202, "top": 512, "left": 534,
"words": "时新服活动"
        },
        {
            "chars": [
                {
                    "char": "!"
                }
            ]
        }
    ]
}
```

文件

```
        "char": "【",
        "location": { "width": 22, "top": 538, "left": 927 },
    },
    {
        "char": "游",
        "location": { "width": 26, "top": 538, "left": 953 },
    },
    {
        "char": "戏",
        "location": { "width": 26, "top": 538, "left": 989 },
    },
    {
        "char": "特",
        "location": { "width": 26, "top": 538, "left": 1025 },
    },
    {
        "char": "鱼",
        "location": { "width": 27, "top": 538, "left": 1051 },
    },
    {
        "char": "】",
        "location": { "width": 19, "top": 538, "left": 1077 },
    }
],
"location": { "width": 187, "top": 538, "left": 927,
"words": "【游戏特色】"
},
{
    "chars": [
        {
            "char": "火",
            "location": { "width": 20, "top": 612, "left": 537 },
        }
    ],
"location": { "width": 33, "top": 612, "left": 537,
"words": "火"
},
{
    "chars": [
        {
            "char": "1",
            "location": { "width": 21, "top": 588, "left": 927 },
        },
        {
            "char": ".",
            "location": { "width": 21, "top": 588, "left": 1053 },
        },
        {
            "char": "超",
            "location": { "width": 26, "top": 588, "left": 1079 },
        }
    ],
"location": { "width": 100, "top": 588, "left": 927,
"words": "超1."
}
]
```

文件

```
{
  "char": "宏",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "伟",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "世",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "界",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "观",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "\u20e5",
  "location": { "width": 21, "top": 588, "left": 10 },
},
{
  "char": "六",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "大",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "种",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "族",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "秘",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "密",
  "location": { "width": 26, "top": 588, "left": 10 },
},
{
  "char": "等",
  "location": { "width": 26, "top": 588, "left": 10 }
```

```
},
{
  "char": "你",
  "location": { "width": 26, "top": 588, "left": 14 },
},
{
  "char": "探",
  "location": { "width": 26, "top": 588, "left": 14 },
},
{
  "char": "索",
  "location": { "width": 26, "top": 588, "left": 14 },
},
{
  "char": ";",
  "location": { "width": 18, "top": 588, "left": 14 }
},
],
"location": { "width": 642, "top": 588, "left": 971,
"words": "1.超宏伟世界观、六大种族秘密等你探索;" },
},
{
  "chars": [
    {
      "char": "爆",
      "location": { "width": 27, "top": 635, "left": 55 },
    },
    {
      "char": "",
      "location": { "width": 27, "top": 635, "left": 55 },
    },
    {
      "char": "禁",
      "location": { "width": 26, "top": 635, "left": 62 },
    },
    {
      "char": "代",
      "location": { "width": 27, "top": 635, "left": 66 },
    },
    {
      "char": "充",
      "location": { "width": 27, "top": 635, "left": 70 },
    },
    {
      "char": "公",
      "location": { "width": 26, "top": 635, "left": 72 },
    },
    {
      "char": "告",
      "location": { "width": 27, "top": 635, "left": 72 }
    }
  ]
}
```

```
        ],
        "location": { "width": 273, "top": 635, "left": 534,
        "words": "爆严禁代充公告"
    },
    {
        "chars": [
            {
                "char": "2",
                "location": { "width": 22, "top": 642, "left": 954 }
            },
            {
                "char": ".",
                "location": { "width": 21, "top": 642, "left": 1000 }
            },
            {
                "char": "全",
                "location": { "width": 26, "top": 642, "left": 1026 }
            },
            {
                "char": "民",
                "location": { "width": 26, "top": 642, "left": 1052 }
            },
            {
                "char": "打",
                "location": { "width": 26, "top": 642, "left": 1078 }
            },
            {
                "char": "宝",
                "location": { "width": 27, "top": 642, "left": 1104 }
            },
            {
                "char": "得",
                "location": { "width": 26, "top": 642, "left": 1130 }
            },
            {
                "char": "神",
                "location": { "width": 27, "top": 642, "left": 1156 }
            },
            {
                "char": "装",
                "location": { "width": 26, "top": 642, "left": 1182 }
            },
            {
                "char": ",",
                "location": { "width": 21, "top": 642, "left": 1208 }
            },
            {
                "char": "自",
                "location": { "width": 26, "top": 642, "left": 1234 }
            },
            {
                "char": "由",
                "location": { "width": 26, "top": 642, "left": 1260 }
            }
        ]
    }
}
```

```
        "char": "由",
        "location": { "width": 26, "top": 642, "left": 10 },
    },
    {
        "char": "交",
        "location": { "width": 26, "top": 642, "left": 10 },
    },
    {
        "char": "易",
        "location": { "width": 27, "top": 642, "left": 10 },
    },
    {
        "char": "换",
        "location": { "width": 27, "top": 642, "left": 10 },
    },
    {
        "char": "货",
        "location": { "width": 27, "top": 642, "left": 10 },
    },
    {
        "char": "而",
        "location": { "width": 26, "top": 642, "left": 10 },
    },
    {
        "char": ",",
        "location": { "width": 21, "top": 642, "left": 10 },
    },
    {
        "char": "极",
        "location": { "width": 26, "top": 642, "left": 10 },
    },
    {
        "char": "速",
        "location": { "width": 26, "top": 642, "left": 10 },
    },
    {
        "char": "致",
        "location": { "width": 26, "top": 642, "left": 10 },
    },
    {
        "char": "富",
        "location": { "width": 27, "top": 642, "left": 10 },
    },
    {
        "char": ";",
        "location": { "width": 22, "top": 642, "left": 10 }
    }
],
"location": { "width": 719, "top": 642, "left": 992,
"words": "2.全民打宝得神装,自由交易换货币,极速致富;" },
},
```

文件

```
{  
  "chars": [  
    {  
      "char": "3",  
      "location": { "width": 21, "top": 693, "left": 96 }  
    },  
    {  
      "char": ".",  
      "location": { "width": 21, "top": 693, "left": 108 }  
    },  
    {  
      "char": "神",  
      "location": { "width": 25, "top": 693, "left": 110 }  
    },  
    {  
      "char": "裝",  
      "location": { "width": 25, "top": 693, "left": 110 }  
    },  
    {  
      "char": "极",  
      "location": { "width": 26, "top": 693, "left": 110 }  
    },  
    {  
      "char": "速",  
      "location": { "width": 25, "top": 693, "left": 110 }  
    },  
    {  
      "char": "进",  
      "location": { "width": 26, "top": 693, "left": 110 }  
    },  
    {  
      "char": "阶",  
      "location": { "width": 26, "top": 693, "left": 110 }  
    },  
    {  
      "char": ",",  
      "location": { "width": 20, "top": 693, "left": 110 }  
    },  
    {  
      "char": "炼",  
      "location": { "width": 25, "top": 693, "left": 110 }  
    },  
    {  
      "char": "化",  
      "location": { "width": 26, "top": 693, "left": 110 }  
    },  
    {  
      "char": "玩",  
      "location": { "width": 25, "top": 693, "left": 110 }  
    },  
    {  
      "char": "","  
      "location": { "width": 25, "top": 693, "left": 110 }  
    }  
  ]  
}
```

文件

```
        "char": "法",
        "location": { "width": 25, "top": 693, "left": 10 },
    },
    {
        "char": "打",
        "location": { "width": 26, "top": 693, "left": 10 },
    },
    {
        "char": "造",
        "location": { "width": 25, "top": 693, "left": 10 },
    },
    {
        "char": "独",
        "location": { "width": 25, "top": 693, "left": 10 },
    },
    {
        "char": "特",
        "location": { "width": 26, "top": 693, "left": 10 },
    },
    {
        "char": "↑",
        "location": { "width": 25, "top": 693, "left": 10 },
    },
    {
        "char": "性",
        "location": { "width": 25, "top": 693, "left": 10 },
    },
    {
        "char": "神",
        "location": { "width": 26, "top": 693, "left": 10 },
    },
    {
        "char": "装",
        "location": { "width": 26, "top": 693, "left": 10 },
    },
    {
        "char": ";",
        "location": { "width": 21, "top": 693, "left": 10 }
    }
],
"location": { "width": 686, "top": 693, "left": 994,
"words": "3. 神装极速进阶, 炼化玩法打造独特个性神装;" },
},
{
    "chars": [
    {
        "char": "4",
        "location": { "width": 19, "top": 746, "left": 994 },
    },
    {
        "char": ".",
        "location": { "width": 19, "top": 746, "left": 994 },
    }
]
```

文件

```
        "location": { "width": 19, "top": 746, "left": 10 },
},
{
    "char": "独",
    "location": { "width": 22, "top": 746, "left": 10 },
},
{
    "char": "创",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": "十",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": "二",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": "星",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": "使",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": "助",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": "战",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": "系",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": "统",
    "location": { "width": 23, "top": 746, "left": 10 },
},
{
    "char": ",",
    "location": { "width": 19, "top": 746, "left": 10 },
},
{
    "char": "五",
    "location": { "width": 22, "top": 746, "left": 10 },
},
```

```
        "char": "大",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "魔",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "幻",
        "location": { "width": 22, "top": 746, "left": 14 },
    },
    {
        "char": "星",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "使",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "技",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "能",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "逆",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "转",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "战",
        "location": { "width": 23, "top": 746, "left": 14 },
    },
    {
        "char": "局",
        "location": { "width": 22, "top": 746, "left": 14 },
    },
    {
        "char": ";",
        "location": { "width": 15, "top": 746, "left": 14 }
    }
],
"location": { "width": 821, "top": 746, "left": 989,
"words": "4.独创十二星使助战系统,五大魔幻星使技能逆转战局;" },
},
```

文件

文件

```
        "char": "力",
        "location": { "width": 25, "top": 795, "left": 10 },
    },
    {
        "char": "培",
        "location": { "width": 25, "top": 795, "left": 10 },
    },
    {
        "char": "养",
        "location": { "width": 25, "top": 795, "left": 10 },
    },
    {
        "char": "个",
        "location": { "width": 25, "top": 795, "left": 10 },
    },
    {
        "char": "性",
        "location": { "width": 23, "top": 795, "left": 10 },
    },
    {
        "char": "化",
        "location": { "width": 25, "top": 795, "left": 10 },
    },
    {
        "char": "职",
        "location": { "width": 25, "top": 795, "left": 10 },
    },
    {
        "char": "业",
        "location": { "width": 25, "top": 795, "left": 10 },
    },
    {
        "char": "属",
        "location": { "width": 23, "top": 795, "left": 10 },
    },
    {
        "char": "性",
        "location": { "width": 25, "top": 795, "left": 10 },
    },
    {
        "char": ";",
        "location": { "width": 18, "top": 795, "left": 10 }
    }
],
"location": { "width": 719, "top": 795, "left": 992,
"words": "5.六大化神任意搭配,助力培养个性化职业属性;" },
{
    "chars": [
        {
            "char": "6",
            "location": { "width": 25, "top": 795, "left": 10 }
        }
    ]
}
```

文件

```
        "location": { "width": 20, "top": 844, "left": 98 },
},
{
    "char": ".",
    "location": { "width": 20, "top": 844, "left": 108 },
},
{
    "char": "勇",
    "location": { "width": 25, "top": 844, "left": 118 },
},
{
    "char": "者",
    "location": { "width": 26, "top": 844, "left": 118 },
},
{
    "char": "轉",
    "location": { "width": 25, "top": 844, "left": 118 },
},
{
    "char": "職",
    "location": { "width": 25, "top": 844, "left": 118 },
},
{
    "char": "突",
    "location": { "width": 26, "top": 844, "left": 118 },
},
{
    "char": "破",
    "location": { "width": 26, "top": 844, "left": 118 },
},
{
    "char": "屬",
    "location": { "width": 25, "top": 844, "left": 118 },
},
{
    "char": "性",
    "location": { "width": 25, "top": 844, "left": 118 },
},
{
    "char": ",",
    "location": { "width": 21, "top": 844, "left": 118 },
},
{
    "char": "職",
    "location": { "width": 25, "top": 844, "left": 118 },
},
{
    "char": "業",
    "location": { "width": 25, "top": 844, "left": 118 },
},
```

文件

```
        "char": "时",
        "location": { "width": 25, "top": 844, "left": 12 },
    },
    {
        "char": "装",
        "location": { "width": 25, "top": 844, "left": 14 },
    },
    {
        "char": "免",
        "location": { "width": 38, "top": 844, "left": 14 },
    },
    {
        "char": "费",
        "location": { "width": 25, "top": 844, "left": 14 },
    },
    {
        "char": "领",
        "location": { "width": 25, "top": 844, "left": 14 },
    },
    {
        "char": ";",
        "location": { "width": 21, "top": 844, "left": 14 },
    }
],
"location": { "width": 586, "top": 844, "left": 994,
"words": "6. 勇者转职突破属性, 职业时装免费领;" },
{
    "chars": [
        {
            "char": "7",
            "location": { "width": 23, "top": 893, "left": 994 },
        },
        {
            "char": ".",
            "location": { "width": 22, "top": 893, "left": 1016 },
        },
        {
            "char": "B",
            "location": { "width": 23, "top": 893, "left": 1038 },
        },
        {
            "char": "0",
            "location": { "width": 22, "top": 893, "left": 1060 },
        },
        {
            "char": "S",
            "location": { "width": 23, "top": 893, "left": 1082 },
        },
        {
            "char": "S",
            "location": { "width": 23, "top": 893, "left": 1104 },
        }
    ]
}
```



```
        "char": "备",
        "location": { "width": 27, "top": 893, "left": 14 },
    },
    {
        "char": ",",
        "location": { "width": 22, "top": 893, "left": 14 },
    },
    {
        "char": "轻",
        "location": { "width": 28, "top": 893, "left": 14 },
    },
    {
        "char": "松",
        "location": { "width": 28, "top": 893, "left": 14 },
    },
    {
        "char": "集",
        "location": { "width": 28, "top": 893, "left": 14 },
    },
    {
        "char": "齐",
        "location": { "width": 28, "top": 893, "left": 14 },
    },
    {
        "char": "全",
        "location": { "width": 28, "top": 893, "left": 14 },
    },
    {
        "char": "套",
        "location": { "width": 27, "top": 893, "left": 14 },
    },
    {
        "char": "神",
        "location": { "width": 28, "top": 893, "left": 14 },
    },
    {
        "char": "装",
        "location": { "width": 28, "top": 893, "left": 14 },
    },
    {
        "char": ";",
        "location": { "width": 23, "top": 893, "left": 14 }
    }
],
"location": { "width": 813, "top": 893, "left": 992,
"words": "7.BOSS爆率100%,狂爆极品装备,轻松集齐全套神装;" },
{
    "chars": [
        {
            "char": "8",
            "location": { "width": 28, "top": 893, "left": 14 }
        }
    ]
}
```

文件

```
        "location": { "width": 25, "top": 945, "left": 96 },
},
{
    "char": ".",
    "location": { "width": 23, "top": 945, "left": 101 },
},
{
    "char": "首",
    "location": { "width": 44, "top": 945, "left": 101 },
},
{
    "char": "日",
    "location": { "width": 29, "top": 945, "left": 101 },
},
{
    "char": "直",
    "location": { "width": 29, "top": 945, "left": 101 },
},
{
    "char": "升",
    "location": { "width": 29, "top": 945, "left": 101 },
},
{
    "char": "1",
    "location": { "width": 23, "top": 945, "left": 101 },
},
{
    "char": "5",
    "location": { "width": 25, "top": 945, "left": 101 },
},
{
    "char": "0",
    "location": { "width": 25, "top": 945, "left": 101 },
},
{
    "char": "级",
    "location": { "width": 29, "top": 945, "left": 101 },
},
{
    "char": ",",
    "location": { "width": 25, "top": 945, "left": 101 },
},
{
    "char": "登",
    "location": { "width": 30, "top": 945, "left": 101 },
},
{
    "char": "录",
    "location": { "width": 29, "top": 945, "left": 101 },
},
```

文件

```
        "char": "即",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "送",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "丨",
        "location": { "width": 23, "top": 945, "left": 16 },
    },
    {
        "char": "兽",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "王",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "天",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "神",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "丶",
        "location": { "width": 25, "top": 945, "left": 16 },
    },
    {
        "char": "春",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "日",
        "location": { "width": 30, "top": 945, "left": 16 },
    },
    {
        "char": "时",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "装",
        "location": { "width": 29, "top": 945, "left": 16 },
    },
    {
        "char": "丶",
        "location": { "width": 25, "top": 945, "left": 16 },
    },
```

```
{  
    "char": "雪",  
    "location": { "width": 29, "top": 945, "left": 17 }  
},  
{  
    "char": "豹",  
    "location": { "width": 28, "top": 945, "left": 16 }  
}  
,  
{"location": { "width": 879, "top": 945, "left": 968,  
    "words": "8.首日直升150级,登录即送:兽王天神、春日时装、雪豹"  
}  
}  
]
```

计算文字的位置

背景：百度OCR返回的文字都是json字典，希望能从对应匹配到的单词，找到对应的位置坐标信息

代码：

```
def calcWordsLocation(self, wordStr, curWordsResult):
    """Calculate words location from result

    Args:
        wordStr (str): the words to check
        curWordsResult (dict): the baidu OCR result of current words
    Returns:
        location, a tuple (x, y, width, height)
    Raises:
    Examples
        wordStr="首充"
        curWordsResult= {
            "chars": [
                {
                    "char": "寻",
                    "location": {
                        "width": 15,
                        "top": 51,
                        "left": 725,
                        "height": 24
                    }
                },
                ...
                {
                    "char": "首",
                    "location": {
                        "width": 15,
                        "top": 51,
                        "left": 971,
                        "height": 24
                    }
                },
                {
                    "char": "充",
                    "location": {
                        "width": 15,
                        "top": 51,
                        "left": 986,
                        "height": 24
                    }
                }
            ],
            "location": {
                "width": 280,
                "top": 51,
                "left": 725,
                "height": 24
            },
            "words": "寻宝福利大厅商城首充"
        }
```

```

        }
        -> (971, 51, 30, 24)
    ....
    (x, y, width, height) = (0, 0, 0, 0)
    matchedStr = curWordsResult["words"]
    # Note: for special, contain multilple words, here only
    foundWords = re.search(wordStr, matchedStr)
    if foundWords:
        logging.debug("foundWords=%s" % foundWords)

    firstMatchedPos = foundWords.start()
    lastMatchedPos = foundWords.end() - 1

    matchedStrLen = len(matchedStr)
    charResultList = curWordsResult["chars"]
    charResultListLen = len(charResultList)

    firstCharResult = None
    lastCharResult = None
    if matchedStrLen == charResultListLen:
        firstCharResult = charResultList[firstMatchedPos]
        lastCharResult = charResultList[lastMatchedPos]
    else:
        # Special: for 'Loading' matched ' Loading', but
        # so using find the corresponding char, then go
        # Note: following method not work for regex str

        firtToMatchChar = wordStr[0]
        lastToMatchChar = wordStr[-1]

        for eachCharResult in charResultList:
            if firstCharResult and lastCharResult:
                break

            eachChar = eachCharResult["char"]
            if firtToMatchChar == eachChar:
                firstCharResult = eachCharResult
            elif lastToMatchChar == eachChar:
                lastCharResult = eachCharResult

    # Note: follow no need check words, to support input
    # firstLocation = None
    # lastLocation = None
    # if firstCharResult["char"] == firtToMatchChar:

```

文件

```
#     firstLocation = firstCharResult["location"]
# if lastCharResult["char"] == lastToMatchChar:
#     lastLocation = lastCharResult["location"]
firstLocation = firstCharResult["location"]
lastLocation = lastCharResult["location"]

# if firstLocation and lastLocation:

# support both horizontal and vertical words
firstLeft = firstLocation["left"]
lastLeft = lastLocation["left"]
minLeft = min(firstLeft, lastLeft)
x = minLeft

firstHorizontalEnd = firstLeft + firstLocation["width"]
lastHorizontalEnd = lastLeft + lastLocation["width"]
maxHorizontalEnd = max(firstHorizontalEnd, lastHorizontalEnd)
width = maxHorizontalEnd - x

lastTop = lastLocation["top"]
minTop = min(firstLocation["top"], lastTop)
y = minTop

lastVerticalEnd = lastTop + lastLocation["height"]
height = lastVerticalEnd - y

return x, y, width, height
```

调用：

```
calculatedLocation = self.calcWordsLocation(eachInputWords)
```

把坐标位置转成中间坐标值

作用：用于后续点击按钮中间坐标值

代码：

文件

```
def locationToCenterPos(self, wordslocation):
    """Convert location of normal button to center position

    Args:
        wordslocation (tuple): words location, (x, y, width, height)
            Example: (267, 567, 140, 39)
    Returns:
        tuple, (x, y), the location's center position, norm
            Example: (337.0, 586.5)
    Raises:
        .....
    """
    x, y, width, height = wordslocation
    centerX = x + width/2
    centerY = y + height/2
    centerPosition = (centerX, centerY)
    return centerPosition
```

调用：

```
curCenterX, curCenterY = self.locationToCenterPos(eachLocat
```

检测文字是否在结果中

代码：

```
def isWordsInResult(self, respJson, wordsOrWordsList, isMatchMultiple):
    """Check words is in result or not

    Args:
        respJson (dict): Baidu OCR responded json
        wordsOrWordsList (str/list): single input str or str list
        isMatchMultiple (bool): for each single str, to match multiple
    Returns:
        dict, matched result
    Raises:
        ...
    # Note: use OrderedDict instead dict to keep order, for orderedDict
    orderedMatchedResultDict = OrderedDict()

    inputWordsList = wordsOrWordsList
    if isinstance(wordsOrWordsList, str):
        inputWords = str(wordsOrWordsList)
        inputWordsList = [inputWords]

    wordsResultList = respJson["words_result"]
    for curInputWords in inputWordsList:
        curMatchedResultList = []
        for eachWordsResult in wordsResultList:
            eachWords = eachWordsResult["words"]
            foundCurWords = re.search(curInputWords, eachWords)
            if foundCurWords:
                curMatchedResultList.append(eachWordsResult)
                if not isMatchMultiple:
                    break

    orderedMatchedResultDict[curInputWords] = curMatchedResultList
    return orderedMatchedResultDict
```

调用：

```
matchedResultDict = self.isWordsInResult(wordsResultJson, True)
```

检测当前屏幕中是否包含对应文字

```

def isWordsInCurScreen(self, wordsOrWordsList, imgPath=None):
    """Found words in current screen

    Args:
        wordsOrWordsList (str/list): single input str or str/list
        imgPath (str): current screen image file path; default None
        isMatchMultiple (bool): for each single str, to match multiple
        isRespShortInfo (bool): return simple=short=normal=full
    Returns:
        matched result, type=bool/list[bool]/dict/tuple, depends on args
    Raises:
    .....
    returnValue = None

    if not imgPath:
        # do screenshot
        imgPath = self.getCurScreenshot()

    wordsResultJson = self.baiduImageToWords(imgPath)

    isMultipleInput = False
    inputWords = None
    inputWordsList = []

    if isinstance(wordsOrWordsList, list):
        isMultipleInput = True
        inputWordsList = list(wordsOrWordsList)
    elif isinstance(wordsOrWordsList, str):
        isMultipleInput = False
        inputWords = str(wordsOrWordsList)
        inputWordsList = [inputWords]

    matchedResultDict = self.isWordsInResult(wordsResultJson,
                                              isMultipleInput,
                                              inputWordsList)

    # add calculated location and words
    # Note: use OrderedDict instead dict to keep order, for better performance
    processedResultDict = OrderedDict()
    for eachInputWords in inputWordsList:
        isCurFound = False
        # curLocatoinList = []
        # curWordsList = []
        curResultList = []

```

```

curWordsMatchedResultList = matchedResultDict[each]
if curWordsMatchedResultList:
    isCurFound = True
    for curIdx, eachWordsMatchedResult in enumerate(curWordsMatchedResultList):
        curMatchedWords = eachWordsMatchedResult["words"]
        calculatedLocation = self.calcWordsLocation(curMatchedWords)
        # curLocatoinList.append(calculatedLocation)
        # curWordsList.append(curMatchedWords)
        curResult = (curMatchedWords, calculatedLocation)
        curResultList.append(curResult)

        # processedResultDict[eachInputWords] = (isCurFound, curResultList)
        processedResultDict[eachInputWords] = (isCurFound, curResultList)
        logging.debug("For %s, matchedResult=%s from imgPath=%s" % (each, curResultList, imgPath))

if isMultipleInput:
    if isRespShortInfo:
        isFoundList = []
        for eachInputWords in processedResultDict.keys():
            isCurFound, noUse = processedResultDict[eachInputWords]
            isFoundList.append(isCurFound)
        # Note: no mattter isMatchMultiple, both only one tuple
        retBoolList = isFoundList
        retValue = retBoolList
    else:
        if isMatchMultiple:
            retTuple = processedResultDict, imgPath
            retValue = retTuple
        else:
            # Note: use OrderedDict instead dict to keep order
            respResultDict = OrderedDict()
            for eachInputWords in processedResultDict.keys():
                # isCurFound, curLocatoinList, curWordsList
                isCurFound, curResultList = processedResultDict[eachInputWords]
                # singleLocation = None
                # singleWords = None
                singleResult = (None, None)
                if isCurFound:
                    # singleLocation = curLocatoinList
                    # singleWords = curWordsList[0]
                    singleResult = curResultList[0]
                # respResultDict[eachInputWords] = (isCurFound, curResultList)
                respResultDict[eachInputWords] = (isCurFound, curResultList)
            retTuple = respResultDict, imgPath
            retValue = retTuple
    else:
        singleInputResult = processedResultDict[inputWords]
        # isCurFound, curLocatoinList, curWordsList = singleInputResult

```

文件

```
isCurFound, curResultList = singleInputResult
if isRespShortInfo:
    # Note: no matter isMatchMultiple, both only
    retBool = isCurFound
    retValue = retBool
else:
    if isMatchMultiple:
        # retTuple = isCurFound, curLocatoinList, curWordsList
        retTuple = isCurFound, curResultList, imgPath
        retValue = retTuple
    else:
        singleResult = (None, None)
        # singleLocation = None
        # singleWords = None
        if isCurFound:
            # singleLocation = curLocatoinList[0]
            # singleWords = curWordsList[0]
            singleResult = curResultList[0]
        # retTuple = isCurFound, singleLocation, singleWords
        retTuple = isCurFound, singleResult, imgPath
        retValue = retTuple

logging.debug("Input: %s, output=%s", wordsOrWordsList,
return retValue
```

调用：

```
allResultDict, _ = self.isWordsInCurScreen(allStrList, imgPath)
```

获取当前屏幕中的文字

```
def getWordsInCurScreen(self):
    """get words in current screenshot"""
    screenImgPath = self.getCurScreenshot()
    wordsResultJson = self.baiduImageToWords(screenImgPath)
    return wordsResultJson
```

调用：

```
curScreenWords = self.getWordsInCurScreen()
```

检测当前屏幕中是否存在某些信息

```

def checkExistInScreen(self,
                      imgPath=None,
                      mandatoryStrList=[],
                      mandatoryMinMatchCount=0,
                      optionalStrList=[],
                      # optionalMinMatchCount=2,
                      optionalMinMatchCount=1,
                      isRespFullInfo=False
                      ):
    """
    Check whether mandatory and optional str list in current screen
    """

    Args:
        imgPath (str): current screen image file path; default None
        mandatoryStrList (list): mandatory str, at least one str
        mandatoryMinMatchCount (int): minimal match count
        optionalStrList (list): optional str, some may match
        optionalMinMatchCount (int): for `optionalStrList`, if match
        isRespFullInfo (bool): return full info or not, full info
    Returns:
        matched result, type=bool/tuple, depends on `isRespFullInfo`
    Raises:
        None

    if not imgPath:
        imgPath = self.getCurScreenshot()
        logging.debug("imgPath=%s", imgPath)

    isExist = False
    # Note: use OrderedDict instead dict to keep order, for
    respMatchLocation = OrderedDict()

    isMandatoryMatch = True
    isMandatoryShouldMatchAll = (mandatoryMinMatchCount <=
                                  len(mandatoryStrList))
    isOptionalMatch = True

    allStrList = []
    allStrList.extend(mandatoryStrList)
    allStrList.extend(optionalStrList)

    optionalMatchCount = 0
    mandatoryMatchCount = 0
    allResultDict, _ = self.isWordsInCurScreen(allStrList,
                                              for eachStr, (isFoundCur, curResultList) in allResultD:
                                                  if eachStr in mandatoryStrList:
                                                      if isFoundCur:
                                                          mandatoryMatchCount += 1
    """

```

文件

```
        respMatchLocation[eachStr] = curResultList
    else:
        if isMandatoryShouldMatchAll:
            isMandatoryMatch = False
            break
    elif eachStr in optionalStrList:
        if isFoundCur:
            optionalMatchCount += 1
            respMatchLocation[eachStr] = curResultList

    if mandatoryStrList:
        if not isMandatoryShouldMatchAll:
            if mandatoryMatchCount >= mandatoryMinMatchCount:
                isMandatoryMatch = True
        else:
            isMandatoryMatch = False

    if optionalStrList:
        if optionalMatchCount >= optionalMinMatchCount:
            isOptionalMatch = True
        else:
            isOptionalMatch = False

    isExist = isMandatoryMatch and isOptionalMatch
    logging.debug("isMandatoryMatch=%s, isOptionalMatch=%s"

    if isRespFullInfo:
        logging.debug("mandatoryStrList=%s, optionalStrList=%s"
                      "mandatoryStrList, optionalStrList, isExist, respMatchLocation, imgPath)
        return (isExist, respMatchLocation, imgPath)
    else:
        logging.debug("mandatoryStrList=%s, optionalStrList=%s"
                      "mandatoryStrList, optionalStrList, isExist)
        return isExist
```

调用：

```
checkResult = self.checkExistInScreen(
    imgPath=imgPath,
    optionalStrList=strList,
    optionalMinMatchCount=1,
    isRespFullInfo=isRespFullInfo,
)
```

和：

文件

```
minOptionalMatchCount = 2

mandatoryList = [
    # "公告",
    "^\u5e02\u53d6$",
    '^strongly_recommended$', # 王城英雄，不规则弹框
    ...
]

possibleTitleList = [
    "^\u53d6\u5316\u53d6\u5316$",
]
optionalList = []
otherOptionalList = [
    "new(\u5e02\u53d6)?",
    "game\u53d6\u5316",
    "(\u53d6\u5316)?update(\u5e02\u53d6)?",
    ...
    "log\u53d6\u5316",
]
optionalList.extend(possibleTitleList)
optionalList.extend(otherOptionalList)

checkResult = self.checkExistInScreen(
    imgPath=imgPath,
    mandatoryStrList=mandatoryList,
    mandatoryMinMatchCount=1,
    optionalStrList=optionalList,
    optionalMinMatchCount=minOptionalMatchCount,
    isRespFullInfo=isRespFullInfo,
)
```

和：

文件

```
mandatoryList = [
    "^购买$", # 造梦西游: '购买'
    "^¥\d+元?", # 剑玲珑, 至尊屠龙
    "\d+元", # 造梦西游: '6元券3元券3'
    "^充值$", # 剑玲珑, 至尊屠龙
]
optionalList = [
    # common
    "元宝",
    "月卡",
    ...
    # 剑玲珑
    "赠",
]
isRealPay, matchResult, imgPath = self.checkExistInScreen(
    mandatoryStrList=mandatoryList,
    mandatoryMinMatchCount=2,
    optionalStrList=optionalList,
    optionalMinMatchCount=2,
    isRespFullInfo=True,
)
```

和:

```
mandatoryList = [
    "^((继续)|(结束))$", # 暗黑觉醒: '继续' or '结束'
    "\d秒$", # 暗黑觉醒: '5秒', '1秒'
]
isAutoConverstion, matchResult, imgPath = self.checkExistInScreen(
    imgPath=imgPath,
    mandatoryStrList=mandatoryList,
    mandatoryMinMatchCount=2,
    isRespFullInfo=True,
)
```

和:

文件

```
mandatoryList = [
    "充值", # 造梦西游: "充值战神榜邮件各但,挑战竞技好友"
    "首充$", # 剑玲珑, 至尊屠龙
    ...
]
optionalList = [
    # common
    "组队", # 至尊屠龙, 剑玲珑
    "任务", # 至尊屠龙, 剑玲珑

    # "商城", # 剑玲珑
    "寻宝", # 剑玲珑
    "福利大厅", # 剑玲珑
    ...
    "背包",
]

isHome, matchResult, imgPath = self.checkExistInScreen(
    mandatoryStrList=mandatoryList,
    mandatoryMinMatchCount=1,
    optionalStrList=optionalList,
    isRespFullInfo=True,
)
```

和:

```
mandatoryList = [
    "立即登录", # 剑玲珑, 至尊屠龙
    ...
    "登录$", # 暗黑觉醒
]
optionalList = [
    # common
    "一键((注册)|(试玩))",
    "忘记密码", # 至尊屠龙, 青云诀
    "((用户)|(账号)|(手机))登录", # 剑玲珑, 至尊屠龙,
    "点击选服", # 青云诀, 暗黑觉醒
    ...
    # 造梦西游
    "游客登录", #更新: 不能用游客登录, 否则后续无法弹出支付页面
]

respUserLogin = self.checkExistInScreen(
    mandatoryStrList=mandatoryList,
    mandatoryMinMatchCount=1,
    optionalStrList=optionalList,
    isRespFullInfo=isRespFullInfo
)
```

是否存在任意一个词组

```
def isExistAnyStr(self, strList, imgPath=None, isRespFullInfo=False):
    """Is any str exist or not

    Args:
        strList (list): str list to check exist or not
        imgPath (str): current screen image file path; default None
        isRespFullInfo (bool): return full info or not, full info includes
    Returns:
        matched result, type=bool/tuple, depends on `isRespFullInfo`
    Raises:
        ...
    if not imgPath:
        imgPath = self.getCurScreenshot()

    checkResult = self.checkExistInScreen(
        imgPath=imgPath,
        optionalStrList=strList,
        optionalMinMatchCount=1,
        isRespFullInfo=isRespFullInfo,
    )
    if isRespFullInfo:
        isExistAny, matchResult, imgPath = checkResult
        logging.debug("isExistAny=%s, matchResult=%s, imgPath=%s", isExistAny, matchResult, imgPath)
        return (isExistAny, matchResult, imgPath)
    else:
        isExistAny = checkResult
        logging.debug("isExistAny=%s, for %s", isExistAny, strList)
        return isExistAny
```

调用：

```
isExist, matchResult, imgPath = self.isExistAnyStr(buttonList)
```

和：

```
mandatoryList = [
    # 御剑仙缘
    "^(请)?点击\s?[^"]+[^"]+", ]{1,6}[^"]+?", # '请点击'"]
]

respResult = self.isExistAnyStr(mandatoryList, imgPath=imgPath)
```

和：

```
requireManualOperationList = [
    "完成指定操作", # speical: 造梦西游 的 '(请完成指定操作)梦口'
]
isRequireManual, _, imgPath = self._existAnyStr(requireMa
```

和:

```
lanuchStrList = [
    "^4399手机游戏$", # 剑玲珑
    "^西瓜游戏$", # 青云诀
]
isLaunch, _, imgPath = self._existAnyStr(lanuchStrList, is
```

和:

```
loadingStrList = [
    # 登录类
    "正在登录", # 正在登录
    "logging",
    ...
    "游戏资源", # 青云诀:本地游戏资源已是最新
    ...
]
isLoadingSth, _, imgPath = self._existAnyStr(loadingSt
```

和:

```
gotoPayStrList = [
    "^前往充值$", # 剑玲珑
    "^立即充值$", # 至尊屠龙
    ...
]
respBoolOrTuple = self._existAnyStr(gotoPayStrList, isRes
```

判断是否所有的字符都存在

```
def isExistAllStr(self, strList, imgPath=None, isRespFullInfo=False):
    """Is all str exist or not

    Args:
        strList (list): str list to check exist or not
        imgPath (str): current screen image file path; default None
        isRespFullInfo (bool): return full info or not, full info means return matched result, type=bool/tuple, depends on `isRespFullInfo` parameter
    Returns:
        matched result, type=bool/tuple, depends on `isRespFullInfo` parameter
    Raises:
        None
    if not imgPath:
        imgPath = self.getCurScreenshot()
    checkResult = self.checkExistInScreen(imgPath=imgPath, strList=strList)
    if isRespFullInfo:
        isExistAll, matchResult, imgPath = checkResult
        logging.debug("isExistAll=%s, matchResult=%s, imgPath=%s", isExistAll, matchResult, imgPath)
        return (isExistAll, matchResult, imgPath)
    else:
        isExistAll = checkResult
        logging.debug("isExistAll=%s, for %s", isExistAll, strList)
        return isExistAll
```

调用：

```
realPayStrList = [
    "^\u20ac\d+\u20ac?", # 剑玲珑，至尊屠龙
    "^\u20ac充值$", # 剑玲珑，至尊屠龙
]
return self.isExistAllStr(realPayStrList, isRespFullInfo=isExistAll)
```

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：2020-07-11 10:17:30

音频

播放音频

树莓派中用python播放音频

前提：

树莓派中，先去安装vlc：

```
sudo apt-get install vlc
```

代码：

```
import vlc
instance = vlc.Instance('--aout=alsa')
p = instance.media_player_new()
m = instance.media_new('/home/pi/Music/lizhongsheng_massif.mp3')
p.set_media(m)
p.play()
```

即可播放音频。

实现设置音量，暂停，继续播放等操作的代码是：

```
p.pause()
vlc.libvlc_audio_set_volume(p, 40)
p.play()
vlc.libvlc_audio_set_volume(p, 90)
```

Mac中调用mpv播放音频

播放音频：

```
cmdPlayer = "mpv"
cmdParaFilePath = tmpAudioFileFullPath
cmdArgList = [cmdPlayer, cmdParaFilePath]

if gCurSubProcess:
    gCurSubProcess.terminate()

gCurSubProcess = subprocess.Popen(cmdArgList)
log.debug("gCurSubProcess=%s", gCurSubProcess)
```

文件

停止播放：

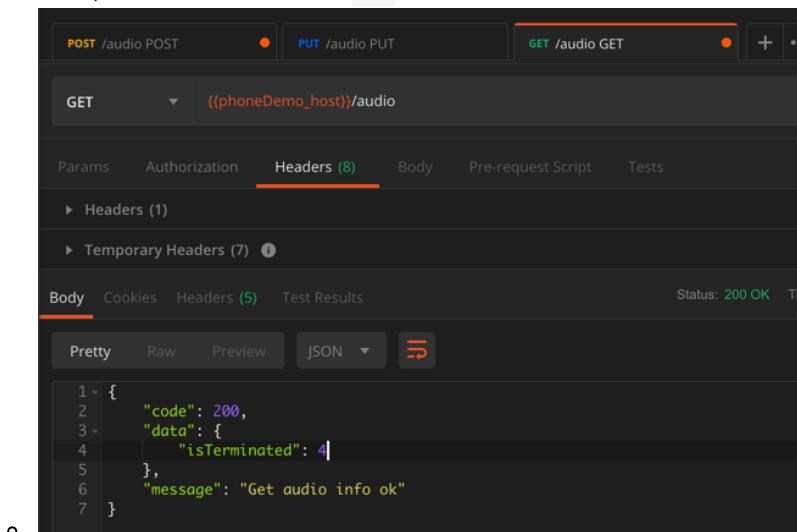
```
if audioControl == "stop":  
    if gCurSubProcess:  
        gCurSubProcess.terminate()  
  
    respData = {  
        "audioControl": "ok"  
    }  
  
else:  
    respData = {  
        "audioControl": "Unsupport command"  
    }
```

获取播放效果：

```
if gCurSubProcess:  
    isTerminated = gCurSubProcess.poll() # None  
    # stdout_data, stderr_data = gCurSubProcess.communicate()  
    # stdoutStr = str(stdout_data)  
    # stderrStr = str(stderr_data)  
    respData = {  
        "isTerminated": isTerminated,  
        # "stdout_data": stdoutStr,  
        # "stderr_data": stderrStr,  
    }
```

返回结果：

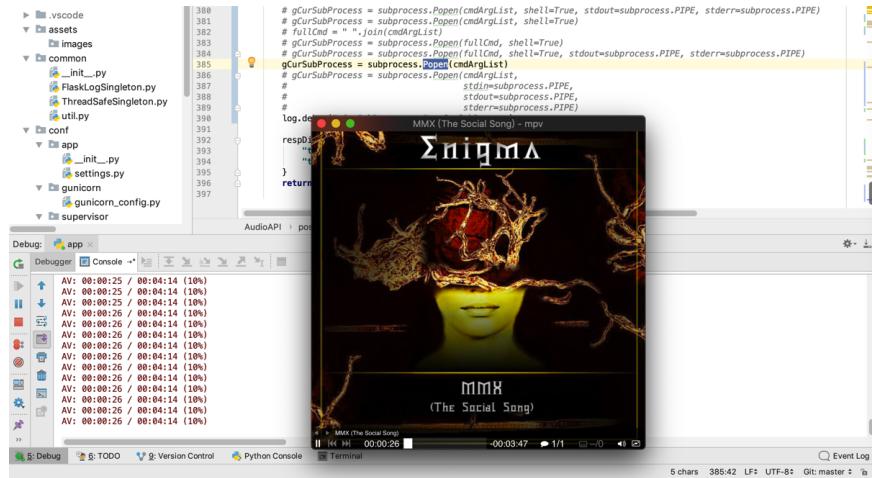
- 正在播放：返回isTerminated为 null
- 被终止后，返回isTerminated为 4



播放的效果：

mac系统中播放音频

PyCharm的console中输出当前播放的信息 -》 如果是mac的terminal中，
则是覆盖式的，不会这么多行 同时弹框GUI窗口



mp3

解析mp3等音频文件得到时长信息

用库：

- audioread
 - GitHub
 - beetbox/audioread: cross-library (GStreamer + Core Audio + MAD + FFmpeg) audio decoding for Python

```
import audioread

try:
    audioFullPath = "/your/input/audio/file.mp3"

    with audioread.audio_open(audioFullPath) as audioFp:
        audioInfo["duration"] = audioFp.duration
        audioInfo["channels"] = audioFp.channels
        audioInfo["sampleRate"] = audioFp.samplerate

except OSError as osErr:
    logging.error("OSError when open %s error %s", audioFullPath)
except EOFError as eofErr:
    logging.error("EOFError when open %s error %s", audioFullPath)
except audioread.DecodeError as decodeErr:
    logging.error("Decode audio %s error %s", audioFullPath)
```

后经整理成函数：

文件

```
import audioread

def detectAudioMetaInfo(audioFullPath):
    """
        detect audio meta info: duration, channels, sampleRate
    """
    isOk = False
    errMsg = ""
    audioMetaInfo = {
        "duration": 0,
        "channels": 0,
        "sampleRate": 0,
    }

    try:
        with audioread.audio_open(audioFullPath) as audioFp:
            audioMetaInfo["duration"] = audioFp.duration
            audioMetaInfo["channels"] = audioFp.channels
            audioMetaInfo["sampleRate"] = audioFp.samplerate

            isOk = True
    except OSError as osErr:
        errMsg = "detect audio info error: %s" % str(osErr)
    except EOFError as eofErr:
        errMsg = "detect audio info error: %s" % str(eofErr)
    except audioread.DecodeError as decodeErr:
        errMsg = "detect audio info error: %s" % str(decodeErr)

    if isOk:
        return isOk, audioMetaInfo
    else:
        return isOk, errMsg
```

调用:

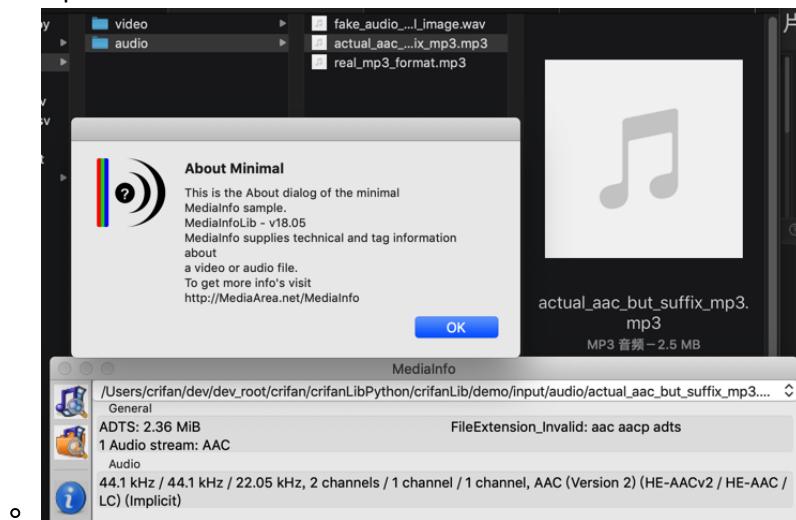
```
def demoDetectAudioMeta():
    curPath = os.path.dirname(__file__)
    inputAudioList = [
        "input/audio/actual_aac_but_suffix_mp3.mp3",
        "input/audio/real_mp3_format.mp3",
        "not_exist_audio.wav",
        "input/audio/fake_audio_actual_image.wav",
    ]

    for eachAudioPath in inputAudioList:
        eachAudioFullPath = os.path.join(curPath, eachAudioPath)
       isOk, errOrInfo = detectAudioMetaInfo(eachAudioFullPath)
        print("isOk=%s, errOrInfo=%s" % (isOk, errOrInfo))

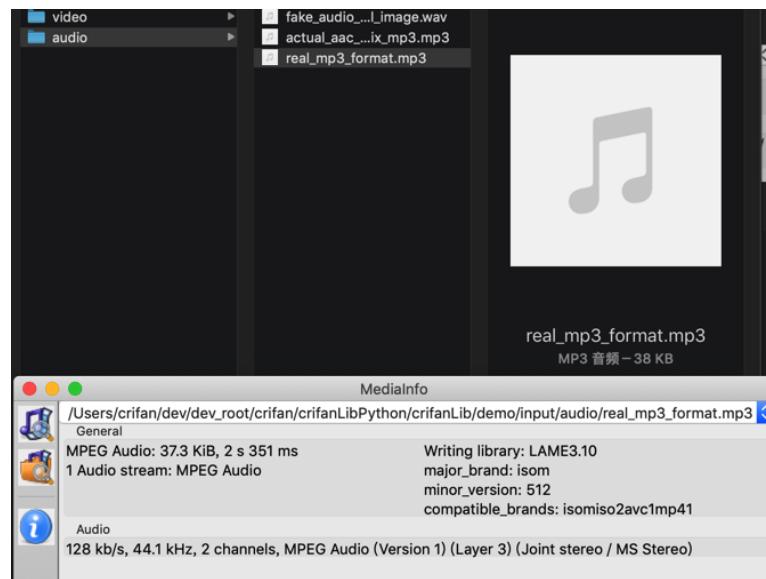
if __name__ == "__main__":
    demoDetectAudioMeta()
```

对应的音频文件，用MediaInfo检测出的信息：

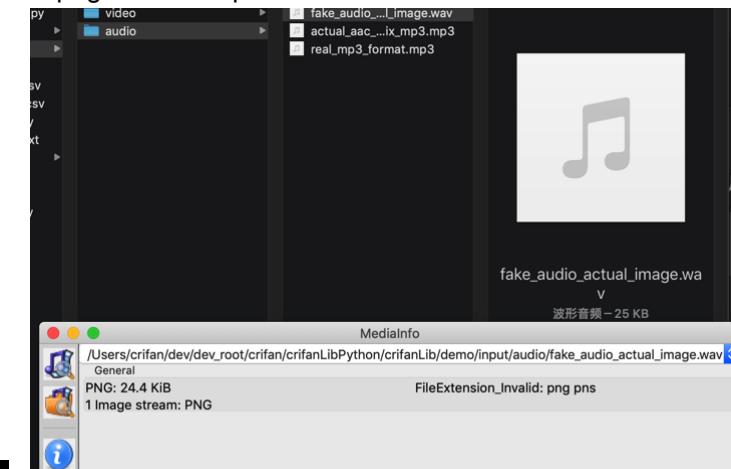
- 正常mp3



文件



- 异常mp3：
 - 故意把png图片改成mp3



输出：

```
# isOk=True, errOrInfo={'duration': 637.8, 'channels': 2,
# isOk=True, errOrInfo={'duration': 2.3510204081632655, 'ch
# isOk=False, errOrInfo=detect audio info error: [Errno 2]
# isOk=False, errOrInfo=detect audio info error:
```

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：2020-07-11 09:45:02

视频

从视频中提取出音频mp3文件

```

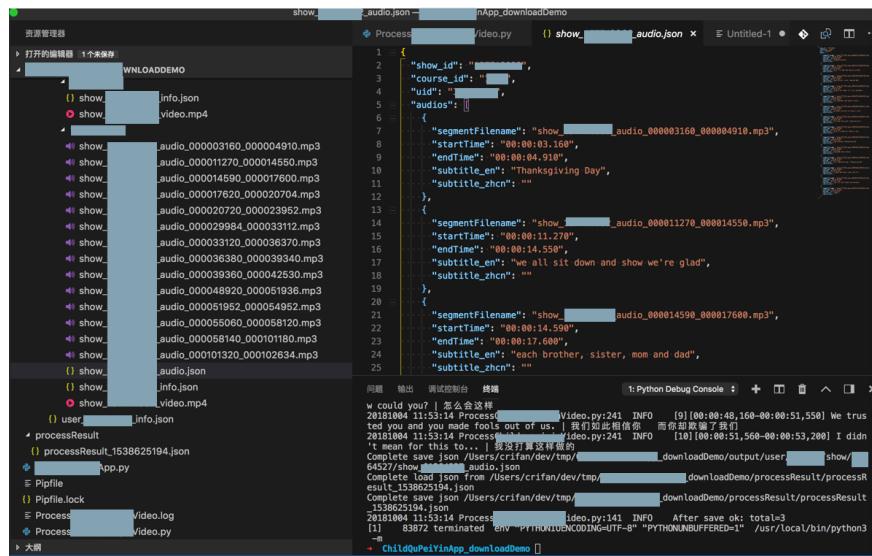
import os
import logging
import subprocess

videoFullPath = "show_157712932_video.mp4"
startTimeStr = "00:00:11.270"
# startTimeStr = "%02d:%02d:%02d.%03d" % (startTime.hours,
endTimeStr = "00:00:14.550"
# endTimeStr = "%02d:%02d:%02d.%03d" % (endTime.hours, end
outputAudioFullPath = "show_157712932_audio_000011270_000011270.mp3"

# extract audio segment from video
# ffmpeg -i show_157712932_video.mp4 -ss 00:00:11.270 -to 00:00:14.550 -c:a mp3 -b:a 128k -f mp3 > show_157712932_audio_000011270_000011270.mp3
if not os.path.exists(outputAudioFullPath):
    ffmpegCmd = "ffmpeg -i %s -ss %s -to %s -b:a 128k %s" % (videoFullPath, startTimeStr, endTimeStr, outputAudioFullPath)
    subprocess.call(ffmpegCmd, shell=True)
    logging.info("Complete use ffmpeg extract audio: %s", outputAudioFullPath)

```

可以从mp4中提取出mp3音频：



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：2020-07-11 09:45:32

网络相关

此处整理和网络相关的一些常用代码段。

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: 2020-07-11 09:47:42

BeautifulSoup

用网络库下载到页面源码后，就是去解析（HTML等）内容了。

Python中最常用的HTML（和XML）解析库之一就是： BeautifulSoup

从xml转换出soup

背景：

iOS自动化期间，常会涉及到，获取到当前页面源码，是xml字符串，需要转换为soup，才能后续操作

所以整理出通用转换逻辑

```
def xmlToSoup(xmlStr):
    """convert to xml string to soup
       Note: xml is tag case sensitive -> retain tag upper

    Args:
        xmlStr (str): xml str, normally page source
    Returns:
        soup
    Raises:
        .....
    # HtmlParser = 'html.parser'
    # XmlParser = 'xml'
    XmlParser = 'lxml-xml'
    curParser = XmlParser

    soup = BeautifulSoup(xmlStr, curParser)
    return soup
```

举例：

(1)

```
curPageXml = self.get_page_source()
soup = CommonUtils.xmlToSoup(curPageXml)
```

获取到xml字符串后，去转换为soup

是否包含符合特定条件的soup节点

```
def isContainSpecificSoup(soupList, elementName, isSizeValidCallback, matchNum):
    """判断BeautifulSoup的soup的list中，是否包含符合条件的soup
    只匹配指定个数的元素才视为找到了
    元素名相同
    面积大小是否符合条件

    Args:
        soupList (list): list of BeautifulSoup objects
        elementName (str): element name
        isSizeValidCallback (function): callback function to check if size is valid
        matchNum (int): number of matched specific number

    Returns:
        bool
    Raises:
        None

    isFound = False

    matchedSoupList = []

    for eachSoup in soupList:
        # if hasattr(eachSoup, "tag"):
        if hasattr(eachSoup, "name"):
            # curSoupTag = eachSoup.tag
            curSoupTag = eachSoup.name
            if curSoupTag == elementName:
                if hasattr(eachSoup, "attrs"):
                    soupAttr = eachSoup.attrs
                    soupWidth = int(soupAttr["width"])
                    soupHeight = int(soupAttr["height"])
                    curSoupSize = soupWidth * soupHeight
                    isSizeValid = isSizeValidCallback(curSoupSize)
                    if isSizeValid:
                        matchedSoupList.append(eachSoup)

    matchedSoupNum = len(matchedSoupList)
    if matchNum == 0:
        isFound = True
    else:
        if matchedSoupNum == matchNum:
            isFound = True

    return isFound
```

说明：

判断soup内，是否有符合特定条件的soup

举例：

(1) iOS的弹框，有上角带关闭按钮时，去判断一个弹框，是否符合对应条件，以便于判断是否可能是弹框

```
nextSiblingeSoupGenerator = possibleCloseSoup.next_sibling:  
nextSiblingeSoupList = list(nextSiblingeSoupGenerator)  
  
hasLargeImage = CommonUtils.isContainSpecificSoup(nextSibl:  
isPossibleClose = hasLargeImage
```

相关函数

```
def isPopupWindowSize(self, curSize):  
    """判断一个soup的宽高大小是否是弹框类窗口(Image, Other等) 的大/  
    # global FullScreenSize  
    FullScreenSize = self.X * self.totalY  
    curSizeRatio = curSize / FullScreenSize # 0.289  
    PopupWindowSizeMinRatio = 0.25  
    # PopupWindowSizeMaxRatio = 0.9  
    PopupWindowSizeMaxRatio = 0.8  
    # isSizeValid = curSizeRatio >= MinPopupWindowSizeRatio  
    # is popup like window, size should large enough, but :  
    isSizeValid = PopupWindowSizeMinRatio <= curSizeRatio <  
    return isSizeValid
```

(2)

```
hasNormalButton = CommonUtils.isContainSpecificSoup(nextSibl:
```

相关函数：

```
def isNormalButtonSize(self, curSize):  
    """判断一个soup的宽高大小是否是普通的按钮大小"""  
    NormalButtonSizeMin = 30*30  
    NormalButtonSizeMax = 100*100  
    isNormalSize = NormalButtonSizeMin <= curSize <= NormalBu  
    return isNormalSize
```

查找元素，限定条件是符合对应的几级的父元素的条件

背景：

很多时候，需要对于iOS的app的页面的源码，即xml中，查找符合特定情况的的元素

文件

这些特定情况，往往是parent或者前几层级的parent中，元素符合一定条件，往往是type，以及宽度是屏幕宽度，高度是屏幕高度等等

所以提取出公共函数，用于bs的find查找元素

```
def bsChainFind(curLevelSoup, queryChainList):
    """BeautifulSoup find with query chain

Args:
    curLevelSoup (soup): BeautifulSoup
    queryChainList (list): str list of all level q
Returns:
    soup
Raises:
Examples:
    input:
        [
            {
                "tag": "XCUIElementTypeWindow",
                "attrs": {"visible": "true", "enable"
            },
            {
                "tag": "XCUIElementTypeButton",
                "attrs": {"visible": "true", "enable"
            },
            {
                "tag": "XCUIElementTypeStaticText",
                "attrs": {"visible": "true", "enable"
            },
        ]
    output:
        soup node of
            <XCUIElementTypeStaticText type="XCUIE
            in :
            <XCUIElementTypeWindow type="XCUIEleme
                <XCUIElementTypeOther type="XCUIEl
                    <XCUIElementTypeOther type="XC
                        <XCUIElementTypeOther type="XC
                            <XCUIElementTypeButton
                                <XCUIElementTypeSta
                                    <XCUIElementTypeSta
                                    <XCUIElementTypeSta
                                    <XCUIElementTypeSta
                                </XCUIElementTypeButton
                            </XCUIElementTypeOther>
                        </XCUIElementTypeOther>
                    </XCUIElementTypeOther>
                </XCUIElementTypeWindow>
            ....
foundSoup = None
if queryChainList:
    chainListLen = len(queryChainList)

    if chainListLen == 1:
        # last one
```

```
curLevelFindDict = queryChainList[0]
curTag = curLevelFindDict["tag"]
curAttrs = curLevelFindDict["attrs"]
foundSoup = curLevelSoup.find(curTag, attrs=curAttrs)
else:
    highestLevelFindDict = queryChainList[0]
    curTag = highestLevelFindDict["tag"]
    curAttrs = highestLevelFindDict["attrs"]
    foundSoupList = curLevelSoup.find_all(curTag, attrs=curAttrs)
    if foundSoupList:
        childrenChainList = queryChainList[1:]
        for eachSoup in foundSoupList:
            eachSoupResult = CommonUtils.bsChainedParse(eachSoup)
            if eachSoupResult:
                foundSoup = eachSoupResult
                break

return foundSoup
```

举例：

(1)

```
.....
    微信-小程序 弹框 警告 尚未进行授权:
    <XCUIElementTypeButton type="XCUIElementTypeButton"
        <XCUIElementTypeOther type="XCUIElementTypeOther"
            <XCUIElementTypeImage type="XCUIElementTypeImage"/>
            <XCUIElementTypeStaticText type="XCUIElementTypeStaticText"/>
            <XCUIElementTypeTextView type="XCUIElementTypeTextView"/>
            <XCUIElementTypeButton type="XCUIElementTypeButton"/>
            <XCUIElementTypeButton type="XCUIElementTypeButton"/>
        </XCUIElementTypeOther>
    </XCUIElementTypeButton>
.....
warningChainList = [
{
    "tag": "XCUIElementTypeButton",
    "attrs": {"visible": "true", "enabled": "true", "width": 100, "height": 40}
},
{
    "tag": "XCUIElementTypeOther",
    "attrs": {"visible": "true", "enabled": "true"}
},
{
    "tag": "XCUIElementTypeStaticText",
    "attrs": {"visible": "true", "enabled": "true", "value": "尚未进行授权，无法使用该功能。"}
},
]
warningSoup = CommonUtils.bsChainFind(soup, warningChainList)
```

相关:

找到元素后，再去点击：

```
if warningSoup:
    parentOtherSoup = warningSoup.parent
    confirmSoup = parentOtherSoup.find(
        "XCUIElementTypeButton",
        attrs={"visible": "true", "enabled": "true", "name": "确认"})
    if confirmSoup:
        self.clickElementCenterPosition(confirmSoup)
        foundAndProcessedPopup = True
```

(2)

```
.....
    系统弹框 拍照或录像:
    <XCUIElementTypeOther type="XCUIElementTypeOther" >
        <XCUIElementTypeOther type="XCUIElementTypeOther" >
            <XCUIElementTypeOther type="XCUIElementTypeOther" >
                <XCUIElementTypeOther type="XCUIElementTypeOther" >
                    <XCUIElementTypeOther type="XCUIElementTypeOther" >
                        <XCUIElementTypeOther type="XCUIElementTypeOther" >
                            <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                    <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                        <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                            <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                    <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                        <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                            <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                    <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                        <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                            <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                                <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                                    <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                                        <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                                            <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                                                <XCUIElementTypeOther type="XCUIElementTypeOther" >
                                                                                                 .....
    photoCameraChainList = [
        {
            "tag": "XCUIElementTypeOther",
            "attrs": {"enabled":"true", "visible":"true"}
        },
    ]
```

```
{  
    "tag": "XCUIElementTypeTable",  
    "attrs": {"enabled":"true", "visible":"true", "x":  
},  
{  
    "tag": "XCUIElementTypeStaticText",  
    "attrs": {"enabled":"true", "visible":"true", "val  
},  
]  
photoCameraSoup = CommonUtils.bsChainFind(soup, photoCamera
```

(3) iOS 设置 无线局域网 列表页 找 当前已连接的WiFi, 特征是带蓝色
 的:

```
....  
    设置 无线局域网 列表页:  
    <XCUIElementTypeTable type="XCUIElementTypeTable" >  
        . . .  
        <XCUIElementTypeCell type="XCUIElementTypeCell"  
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="WLAN" />  
            <XCUIElementTypeOther type="XCUIElementTypeImage" />  
            <XCUIElementTypeOther type="XCUIElementTypeImage" />  
            <XCUIElementTypeImage type="XCUIElementTypeImage" />  
        </XCUIElementTypeCell>  
        <XCUIElementTypeCell type="XCUIElementTypeCell"  
            <XCUIElementTypeStaticText type="XCUIElementTypeText" value="WPS" />  
            <XCUIElementTypeImage type="XCUIElementTypeImage" />  
            <XCUIElementTypeImage type="XCUIElementTypeImage" />  
            <XCUIElementTypeImage type="XCUIElementTypeImage" />  
        </XCUIElementTypeCell>  
....  
  
curPageXml = self.get_page_source()  
soup = CommonUtils.xmlToSoup(curPageXml)  
blueCheckChainList = [  
    {  
        "tag": "XCUIElementTypeCell",  
        "attrs": {"enabled": "true", "visible": "true", "x": "100", "y": "100"}  
    },  
    {  
        "tag": "XCUIElementTypeOther",  
        "attrs": {"enabled": "true", "visible": "true"}  
    },  
    {  
        "tag": "XCUIElementTypeImage",  
        "# attrs": {"enabled": "true", "visible": "true", "name": "UIPreferenceIcon"}  
        "attrs": {"enabled": "true", "name": "UIPreferenceIcon"}  
    },  
]  
blueCheckSoup = CommonUtils.bsChainFind(soup, blueCheckChainList)  
if blueCheckSoup:
```

requests

下载图片，保存为二进制文件

```
import requests
resp = requests.get(pictureUrl)
with open(saveFullPath, 'wb') as saveFp:
    saveFp.write(resp.content)
```

详见：

[【已解决】Python的requests中如何下载二进制数据保存为图片文件](#)

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：2020-07-07 17:11:49

数学

详见：

<https://github.com/crifan/crifanLibPython/blob/master/crifanLib/crifanMath.py>

md5

md5计算

- md5
 - Python 3 中已改名 hashlib
 - 且update参数只允许 bytes
 - 不允许 str

的md5代码：

```
from hashlib import md5 # only for python 3.x

def generateMd5(strToMd5) :
    """
    generate md5 string from input string
    eg:
        xxxxxxxx -> af0230c7fcc75b34cbb268b9bf64da79
    :param strToMd5: input string
    :return: md5 string of 32 chars
    """

    encrptedMd5 = ""
    md5Instance = md5()
    # print("type(md5Instance)=%s" % type(md5Instance)) # 1
    # print("type(strToMd5)=%s" % type(strToMd5)) # type(str)
    bytesToMd5 = bytes(strToMd5, "UTF-8")
    # print("type(bytesToMd5)=%s" % type(bytesToMd5)) # type(bytes)
    md5Instance.update(bytesToMd5)
    encrptedMd5 = md5Instance.hexdigest()
    # print("type(encrptedMd5)=%s" % type(encrptedMd5)) # 1
    # print("encrptedMd5=%s" % encrptedMd5) # encrptedMd5=af0230c7...
    return encrptedMd5
```

之前旧版本的 Python 2 (<= 2.7) 版本：

- md5还是个独立模块
 - 还没有并入 hashlib

- 注：
 - 好像 python 2.7 中已将md5并入 hashlib
 - 但是 update 参数还允许 str (而不是 bytes)
- update参数允许str

的md5代码：

```
try:  
    import md5  
except ImportError:  
    from hashlib import md5  
  
def generateMd5(strToMd5) :  
    encryptedMd5 = ""  
    md5Instance = md5.new()  
    #md5Instance=<md5 HASH object @ 0x1062af738>  
    md5Instance.update(strToMd5)  
    encryptedMd5 = md5Instance.hexdigest()  
    #encryptedMd5=af0230c7fcc75b34ccb268b9bf64da79  
    return encryptedMd5
```

常见语法

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: 2020-06-26 11:41:52

函数参数

可变参数

之前一个用到了可变参数的函数是：

```
def multipleRetry(self, functionInfoDict, maxRetryNum=5, sleepInterval=1):
    """
    do something, retry mutiple time if fail
    """

    Args:
        functionInfoDict (dict): function info dict containing
        maxRetryNum (int): max retry number
        sleepInterval (float): sleep time of each interval

    Returns:
        bool
    Raises:
        .....

    doSuccess = False
    functionCallback = functionInfoDict["functionCallback"]
    functionParaDict = functionInfoDict.get("functionParaDict")

    curRetryNum = maxRetryNum
    while curRetryNum > 0:
        if functionParaDict:
            doSuccess = functionCallback(**functionParaDict)
        else:
            doSuccess = functionCallback()

        if doSuccess:
            break

        time.sleep(sleepInterval)
        curRetryNum -= 1

    if not doSuccess:
        functionName = str(functionCallback)
        # '<bound method DevicesMethods.switchToAppStoreSession of <DevicesMethods object at 0x10c2d0>'

        logging.error("Still fail after %d retry for %s", curRetryNum, functionName)
        return doSuccess
```

其中的：

```
functionCallback(**functionParaDict)
```

中的：

```
**functionParaDict
```

表示，dict类型的参数，内部包含多个key和value，用**去展开后，传入真正要执行的函数

几种调用中带参数的例子是：

```
searchInputQuery = {"type": "XCUIElementTypeSearchField", "isInputOk": self.multipleRetry(  
    {  
        "functionCallback": self.wait_element_setText,  
        "functionParaDict": {  
            "locator": searchInputQuery,  
            "text": appName,  
        }  
    }  
)}
```

之前原始写法：

```
searchInputQuery = {"type": "XCUIElementTypeSearchField", "isInputOk": self.wait_element_setText(searchInputQuery, app)}
```

其中wait_element_setText的定义是：

```
def wait_element_setText(self, locator, text):
```

对应着之前传入时的：

```
"functionParaDict": {  
    "locator": searchInputQuery,  
    "text": appName,  
}
```

即可，给出上述细节，便于理解，传入的参数是如何用 ** 展开的。

详见：

[【已解决】Python中如何实现函数调用时多个可变数量的参数传递](#)

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：2020-06-26 11:44:46

dict字典

删除dict中某个键（和值）

- 常见写法：

```
del yourDict["keyToDelete"]
```

- 更加Pythonic的写法：

```
yourDict.pop("keyToDelete")
```

注意：

为了防止出现 `KeyError`，注意确保要删除的key都是存在的，否则就要先判断存在，再去删除。

OrderedDict

想要获取OrderedDict的最后一个item（的key和value）

```
next(reversed(someOrderedDict.items()))
```

另外，只需要获取最后一个元素的key，则可以：

```
next(reversed(someOrderedDict.keys()))
```

或：

```
next(reversed(someOrderedDict))
```

详见：

【已解决】Python中获取OrderedDict中最后一个元素

合并2个dict的值

(1) 如果无需保留原有（第一个dict）的值，则用`update`即可：

```
firstDict.update(secondDict)
```

支持: Python >=3.5

(2) 如果要保留之前的dict的值, 则用**展开

```
thirdDict = (**firstDict, **secondDict)
```

支持: Python 2 和 Python <=3.4

详见:

【已解决】Python中如何合并2个dict字典变量的值

有序字典OrderedDict的初始化

```
from collections import OrderedDict

orderedDict = OrderedDict()
```

后续正常作为普通dict使用

```
>>> from collections import OrderedDict
>>> orderedDict = OrderedDict()
>>> orderedDict["key2"] = "value2"
>>> orderedDict["key1"] = "value1"
>>> orderedDict["key3"] = "value3"

>>> orderedDict
OrderedDict([('key2', 'value2'), ('key1', 'value1'), ('key3', 'value3')])
```

dict的递归的合并更新

```
def recursiveMergeDict(aDict, bDict):
    """
    Recursively merge dict a to b, return merged dict b
    Note: Sub dict and sub list's won't be overwritten but
          added.

    example:
    (1) input and output example:
    input:
    {
        "keyStr": "strValueA",
        "keyInt": 1,
        "keyBool": true,
        "keyList": [
            {
                "index0Item1": "index0Item1",
                "index0Item2": "index0Item2"
            },
            {
                "index1Item1": "index1Item1"
            },
            {
                "index2Item1": "index2Item1"
            }
        ]
    }

    and

    {
        "keyStr": "strValueB",
        "keyInt": 2,
        "keyList": [
            {
                "index0Item1": "index0Item1_b"
            },
            {
                "index1Item1": "index1Item1_b"
            }
        ]
    }

    output:

    {
        "keyStr": "strValueB",
        "keyBool": true,
        "keyInt": 2,
        "keyList": [
            {

```

```

        "index0Item1": "index0Item1_b",
        "index0Item2": "index0Item2"
    },
    {
        "index1Item1": "index1Item1_b"
    },
    {
        "index2Item1": "index2Item1"
    }
]
}

(2) code usage example:
import copy
cDict = recursiveMergeDict(aDict, copy.deepcopy(bDict))

Note:
bDict should use deepcopy, otherwise will be altered after

.....
aDictItems = None
if (sys.version_info[0] == 2): # is python 2
    aDictItems = aDict.iteritems()
else: # is python 3
    aDictItems = aDict.items()

for aKey, aValue in aDictItems:
    # print("----- [%s]=%s" % (aKey, aValue))
    if aKey not in bDict:
        bDict[aKey] = aValue
    else:
        bValue = bDict[aKey]
        # print("aValue=%s" % aValue)
        # print("bValue=%s" % bValue)
        if isinstance(aValue, dict):
            recursiveMergeDict(aValue, bValue)
        elif isinstance(aValue, list):
            aValueListLen = len(aValue)
            bValueListLen = len(bValue)
            bValueListMaxIdx = bValueListLen - 1
            for aListIdx in range(aValueListLen):
                # print("---[%d]" % aListIdx)
                aListItem = aValue[aListIdx]
                # print("aListItem=%s" % aListItem)
                if aListIdx <= bValueListMaxIdx:
                    bListItem = bValue[aListIdx]
                    # print("bListItem=%s" % bListItem)
                    recursiveMergeDict(aListItem, bListItem)
                else:
                    # print("bDict=%s" % bDict)
                    # print("aKey=%s" % aKey)

```

文件

```
# print("aListItem=%s" % aListItem)
bDict[aKey].append(aListItem)
```

```
return bDict
```

调用举例：

```
templateJson = {
    "author": "Crifan Li <admin@crifan.com>",
    "description": "gitbook书的描述",
    "gitbook": "3.2.3",
    "language": "zh-hans",
    "links": { "sidebar": { "主页": "http://www.crifan.com" } },
    "plugins": [
        "theme-comscore",
        "anchors",
        "-lunr",
        "-search",
        "search-plus",
        "disqus",
        "-highlight",
        "prism",
        "prism-themes",
        "github-buttons",
        "splitter",
        "-sharing",
        "sharing-plus",
        "tbfed-pagefooter",
        "expandable-chapters-small",
        "ga",
        "donate",
        "sitemap-general",
        "copy-code-button",
        "callouts",
        "toolbar-button"
    ],
    "pluginsConfig": {
        "callouts": { "showTypeInHeader": false },
        "disqus": { "shortName": "crifan" },
        "donate": {
            "alipay": "https://www.crifan.com/files/res/crifan_cc_alipay.html",
            "alipayText": "支付宝打赏给Crifan",
            "button": "打赏",
            "title": "",
            "wechat": "https://www.crifan.com/files/res/crifan_cc_wechat.html",
            "wechatText": "微信打赏给Crifan"
        },
        "ga": { "token": "UA-28297199-1" },
        "github-buttons": {
            "buttons": [
                {
                    "count": true,
                    "repo": "gitbook_name",
                    "size": "small",
                    "type": "star",
                    "user": "crifan"
                }
            ]
        }
    }
}
```

```
},
{
  "count": false,
  "size": "small",
  "type": "follow",
  "user": "crifan",
  "width": "120"
}
]
},
"prism": { "css": ["prism-themes/themes/prism-atom-dark.css"] },
"sharing": {
  "all": [
    "douban",
    "facebook",
    "google",
    "instapaper",
    "line",
    "linkedin",
    "messenger",
    "pocket",
    "qq",
    "qzone",
    "stumbleupon",
    "twitter",
    "viber",
    "vk",
    "weibo",
    "whatsapp"
  ],
  "douban": false,
  "facebook": true,
  "google": false,
  "hatenaBookmark": false,
  "instapaper": false,
  "line": false,
  "linkedin": false,
  "messenger": false,
  "pocket": false,
  "qq": true,
  "qzone": false,
  "stumbleupon": false,
  "twitter": true,
  "viber": false,
  "vk": false,
  "weibo": true,
  "whatsapp": false
},
"sitemap-general": {
  "prefix": "https://book.crifan.com/gitbook/gitbook_ng.html"
},
```

```
"tbfed-pagefooter": {
    "copyright": "crifan.com, 使用CC BY 4.0协议发布 all right reserved,
    "modify_format": "YYYY-MM-DD HH:mm:ss",
    "modify_label": "最后更新: "
},
"theme-default": { "showLevel": true },
"toolbar-button": {
    "icon": "fa-file-pdf-o",
    "label": "下载PDF",
    "url": "http://book.crifan.com/books/gitbook_name/pdf"
}
},
"root": "./src",
"title": "Gitbook的书名"
}

currentJson = {
    "description": "crifan整理的Python各个方面常用的代码段，供需要的读者参考和学习",
    "pluginsConfig": {
        "github-buttons": { "buttons": [{ "repo": "python_common_code" }]},
        "sitemap-general": {
            "prefix": "https://book.crifan.com/gitbook/python_common_code"
        },
        "toolbar-button": {
            "url": "http://book.crifan.com/books/python_common_code/index.html"
        }
    },
    "title": "Python常用代码段"
}

bookJson = recursiveMergeDict(templateJson, copy.deepcopy(currentJson))
```

crifan.com, 使用[CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)协议发布 all right reserved,
powered by Gitbook 最后更新: 2020-07-11 10:37:49

list列表和set集合

list vs set

- set
 - 适用于检测某元素是否在集合内、对集合进行一定的数学操作
 - 不支持indexing, slicing
- list
 - 普通的数组
 - 支持indexing, slicing

把list换成set

```
someSet = set([])
for eachItem in someList:
    someSet.add(eachItem)
```

set集合转换为字符串

```
someSetStr = ", ".join(someSet)
```

把列表转为python正则中的group中可能出现的选项

```
def listToPatternGroup(curList):
    """Convert list to pattern group"""
    patternGroupList = list(map(lambda curType: "(%s)" % curType,
                                curList))
    groupP = "|".join(patternGroupList) # '(aaa)|(bbb)|(ccc)'
    return groupP
```

调用：

```
ValidPlatformTypeList = ["iOS", "Android"]
ValidPlatformRule = listToPatternGroup(ValidPlatformTypeList)
```

目的是用于后续的正则判断

```
TaskFilenamePattern = "(?P<taskDate>\d+)_(?P<businessType>%s)
```

文件

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: 2020-12-31 17:47:33

sort排序

详见：

- <https://github.com/crifan/crifanLibPython/blob/master/crifanLib/crifanDict.py>
- <https://github.com/crifan/crifanLibPython/blob/master/crifanLib/demo/crifanDictDemo.py>

对字典根据key去排序

```
from collections import OrderedDict

def sortDictByKey(originDict):
    """
        Sort dict by key
    """
    originItems = originDict.items()
    sortedOriginItems = sorted(originItems)
    sortedOrderedDict = OrderedDict(sortedOriginItems)
    return sortedOrderedDict
```

调用：

```
def demoSortDictByKey():
    originDict = {
        "c": "abc",
        "a": 1,
        "b": 22
    }
    print("originDict=%s" % originDict)
    # originDict={'c': 'abc', 'a': 1, 'b': 22}
    sortedOrderedDict = sortDictByKey(originDict)
    print("sortedOrderedDict=%s" % sortedOrderedDict)
    # sortedOrderedDict=OrderedDict([('a', 1), ('b', 22), ('c', 'abc')])
```

sort和sorted

```
# Function: Demo sorted
#   mainly refer official doc:
#       排序指南 - Python 3.8.2 文档
#       https://docs.python.org/zh-cn/3/howto/sorting.html
# Author: Crifan Li
# Update: 20200304

from operator import itemgetter, attrgetter

print("%s %s %s" % ('='*40, "sort", '='*40))

originIntList = [5, 2, 3, 1, 4]
originIntList.sort()
sortedSelfIntList = originIntList
print("sortedSelfIntList=%s" % sortedSelfIntList)
# sortedSelfIntList=[1, 2, 3, 4, 5]

print("%s %s %s" % ('='*40, "sorted", '='*40))

intList = [5, 2, 3, 1, 4]
sortedIntList = sorted(intList)
print("sortedIntList=%s" % sortedIntList)
# sortedIntList=[1, 2, 3, 4, 5]

reversedSortIntList = sorted(intList, reverse=True)
print("reversedSortIntList=%s" % reversedSortIntList)
# reversedSortIntList=[5, 4, 3, 2, 1]

intStrDict = {5: 'A', 1: 'D', 2: 'B', 4: 'E', 3: 'B'}
dictSortedIntList = sorted(intStrDict)
print("dictSortedIntList=%s" % dictSortedIntList)
# dictSortedIntList=[1, 2, 3, 4, 5]

normalStr = "Crifan Li best love language is Python"
strList = normalStr.split()
print("strList=%s" % strList)
sortedStrList = sorted(strList, key=str.lower)
print("sortedStrList=%s" % sortedStrList)
# strList=['Crifan', 'Li', 'best', 'love', 'language', 'is'
# sortedStrList=['best', 'Crifan', 'is', 'language', 'Li',
```

```

studentTupleList = [
    # name, grade, age
    ('Cindy', 'A', 15),
    ('Crifan', 'B', 12),
    ('Tony', 'B', 10),
]
sortedTupleList_lambda = sorted(studentTupleList, key=lambda student: student[2])
print("sortedTupleList_lambda=%s" % sortedTupleList_lambda)
# sortedTupleList_lambda=[('Tony', 'B', 10), ('Crifan', 'B', 12), ('Cindy', 'A', 15)]

# same as single function:
def getStudentAge(curStudentTuple):
    return curStudentTuple[2] # [2] is age
sortedTupleList_singleFunction = sorted(studentTupleList, key=getStudentAge)
print("sortedTupleList_singleFunction=%s" % sortedTupleList_singleFunction)
# sortedTupleList_singleFunction=[('Tony', 'B', 10), ('Crifan', 'B', 12), ('Cindy', 'A', 15)]

# same as operator itemgetter:
sortedTupleList_operator = sorted(studentTupleList, key=itemgetter(2))
print("sortedTupleList_operator=%s" % sortedTupleList_operator)
# sortedTupleList_operator=[('Tony', 'B', 10), ('Crifan', 'B', 12), ('Cindy', 'A', 15)]

class Student:
    def __init__(self, name, grade, age):
        self.name = name
        self.grade = grade
        self.age = age
    def __repr__(self):
        return repr((self.name, self.grade, self.age))

studentObjectList = [
    Student('john', 'A', 15),
    Student('jane', 'A', 15),
    Student('dave', 'A', 15),
]
sortedObjectList = sorted(studentObjectList, key=lambda student: student.age)
print("sortedObjectList=%s" % sortedObjectList)
# sortedObjectList=[('john', 'A', 15), ('jane', 'A', 15), ('dave', 'A', 15)]

# same as operator attrgetter:
sortedObjectList_operator = sorted(sortedObjectList, key=attrgetter('age'))
print("sortedObjectList_operator=%s" % sortedObjectList_operator)
# sortedObjectList_operator=[('john', 'A', 15), ('jane', 'A', 15), ('dave', 'A', 15)]

```

文件

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: 2020-06-27 13:17:24

enum枚举

枚举基本用法

枚举定义

举例1：

```
from enum import Enum

class BatteryState(Enum):
    Unknown = 0
    Unplugged = 1
    Charging = 2
    Full = 3
```

举例2：

```
import enum

class ScreenshotQuality(enum.Enum):
    Original = 0
    Medium = 1
    Low = 2
```

举例3：

```
class SentenceInvalidReason(Enum):
    NONE = "none"
    UNKNOWN = "unknown"
    EMPTY = "empty"
    TOO_SHORT = "too short"
    TOO_LONG = "too long"
    TOO_MANY_INVALID_WORD = "contain too many invalid words"
```

初始化创建枚举值

直接传入对应的（此处是int）值即可：

```
batteryStateInt = 2
curBatttryStateEnum = BatteryState(batteryStateInt)
```

log输出是：

```
curBattrystateEnum=BatteryState.Charging
```

获取枚举的名称

```
curBattrystateName = curBattrystateEnum.name
```

输出: 'Charging'

获取枚举的值

```
curBattrystateValue = curBattrystateEnum.value
```

输出: 2

类似, 直接从定义中获取值:

```
gScreenQuality = ScreenshotQuality.Low.value # 2
```

枚举高级用法

给枚举中添加函数

```
class TipType(enum.Enum):
    NoTip = "NoTip"
    TenPercent = "TenPercent"
    FifthPercent = "FifthPercent"
    TwentyPercent = "TwentyPercent"

    # @property
    def getTipPercent(self):
        tipPercent = 0.0
        if self == TipType.NoTip:
            tipPercent = 0.0
        elif self == TipType.TenPercent:
            tipPercent = 0.10
        elif self == TipType.FifthPercent:
            tipPercent = 0.15
        elif self == TipType.TwentyPercent:
            tipPercent = 0.20
        gLog.debug("self=%s -> tipPercent=%s", self, tipPercent)
        return tipPercent
```

调用:

```
tipPercent = initiatorTipType.getTipPercent()  
# tipPercent=0.1
```

注意事项

字符串枚举定义最后不要加逗号

enum 定义期间不要加（多余的）逗号：

```
class ScreenshotQuality(enum.Enum):  
    Original = 0,  
    Medium = 1,  
    Low = 2,
```

否则 value 就是 tuple 元祖了：

```
gScreenQuality = ScreenshotQuality.Low.value # 实际上是 (2,)  
print("gScreenQuality=%s" % gScreenQuality) # gScreenQualit  
print("type(gScreenQuality)=%s" % type(gScreenQuality)) # 1
```

```
38  
39     class ScreenshotQuality(enum.Enum):  
40         ... Original = 0, # XCTImageQualityOriginal Original image quality, represented as a lossless PNG  
41         ... Medium = 1, # * XCTImageQualityMedium Medium image quality, represented as a high-quality lossy  
42         ... Low = 2, # XCTImageQualityLow Low image quality  
43  
44     gScreenQuality = ScreenshotQuality.Low.value # 2  
45     print("gScreenQuality=%s" % gScreenQuality)  
46     print("type(gScreenQuality)=%s" % type(gScreenQuality))
```

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： 2020-06-27 12:56:53

collections集合

根据官网：

[collections --- 容器数据类型 — Python 3.8.1 文档](#)

介绍，集合有很多种，列出供了解：

- `namedtuple()` : 创建命名元组子类的工厂函数
- `deque` : 类似列表(list)的容器，实现了在两端快速添加(append)和弹出(pop)
- `ChainMap` : 类似字典(dict)的容器类，将多个映射集合到一个视图里面
- `Counter` : 字典的子类，提供了可哈希对象的计数功能
- `OrderedDict` : 字典的子类，保存了他们被添加的顺序
- `defaultdict` : 字典的子类，提供了一个工厂函数，为字典查询提供一个默认值
- `UserDict` : 封装了字典对象，简化了字典子类化
- `UserList` : 封装了列表对象，简化了列表子类化
- `UserString` : 封装了字符串对象，简化了字符串子类化

待以后用到了，再详细总结。

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：2020-06-27 13:07:42

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: 2020-03-17 09:11:34

参考资料

- 【已解决】AppStore自动安装iOS的app：逻辑优化加等待和多试几次
- 【已解决】Python中如何实现函数调用时多个可变数量的参数传递
- 【已解决】Python中如何合并2个dict字典变量的值
- 【已解决】Python中给Mac中文件加上可执行权限
- 【已解决】Python如何从二进制数据中生成Pillow的Image
- 【已解决】Python的Pillow如何从二进制数据中读取图像数据
- 【已解决】Python的requests中如何下载二进制数据保存为图片文件
- 【已解决】Python中用Pillow去缩小分辨率以及保持画质同时最大程度压缩图片
- 【已解决】Python中实现二进制数据的图片的压缩
- 【已解决】Python中如何解析mp3等音频文件得到时长信息
- 【已解决】用Python代码从视频中提取出音频mp3文件
- 【已解决】Python 3中通过二进制生成文件类型对象
- 【已解决】用ffmpeg从mp4视频中提取出整个mp3以及根据时间段去分割mp3
- 【已解决】python中从文件名后缀推断出MIME类型
- 【已解决】Python中从int值生成Enum枚举和获取枚举值的字符串或名字
- 【未解决】python的wda中调整appium的settings参数实现功能优化
- 【已解决】Python中实现类似touch创建一个空文件
- 【已解决】Python中根据key去对字典排序
- 【已解决】Python 3中如何把字符串str转换成字节码bytes
- 【已解决】Python的md5运行出错：发生异常AttributeError
builtin_function_or_method object has no attribute new
- 【已解决】Python 3中md5报错：Unicode-objects must be encoded before hashing
- 【已解决】Python 3中判断变量类型
- 【已解决】Python中删除字典dict中的键值
- 【已解决】Python中获取文件最后更新时间
- 【已解决】Python中获取OrderedDict中最后一个元素
-
- 【已解决】python中判断单个或多个单词是否是全部小写或首字母小写
- 【已解决】Python中如何让Enum的字符串输出字段的值而不带类型的前缀 – 在路上
- 【已解决】Python中给枚举添加内置函数或属性
- 【基本解决】Python中把wma、wav等格式音频转换为mp3 – 在路上
- 【已解决】Python中如何格式化大小为人类易读的效果
- 【已解决】Python中获取带毫秒的时间戳

- [【已解决】Python中实现dict的递归的合并更新](#)
- [【已解决】Python中把list换成set](#)
- [【整理】python中一次性创建多级文件夹，判断一个文件夹是否已经存在 – 在路上](#)
- [【已解决】Python中如何递归的删除整个非空文件夹 – 在路上](#)
-
- [python - Correct way to write line to file? - Stack Overflow](#)
- [How do you do a simple "chmod +x" from within python? - Stack Overflow](#)
- [io.BytesIO.getvalue](#)
- [排序指南 — Python 3.8.2 文档](#)
- [Python 常用指引 — Python 3.8.2 文档](#)
- [Built-in Functions — Python 3.8.2 documentation](#)
- [operator — Standard operators as functions — Python 3.8.2 documentation](#)
- [Python 常用指引 — Python 3.8.2 文档](#)
- [3.8.2 Documentation](#)
- [编程常见问题 — Python 3.8.2 文档](#)
- [术语对照表 — Python 3.8.2 文档](#)
- [enum — Support for enumerations — Python 3.8.2 documentation](#)
- [enum --- 对枚举的支持 — Python 3.9.0a4 文档](#)
- [collections --- 容器数据类型 — Python 3.8.1 文档](#)
- [md5 not support in python 3.6 and django 1.10 - Stack Overflow](#)
- [list - Python: how to join entries in a set into one string? - Stack Overflow](#)
-

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: 2020-07-11 10:37:08