



第四章 判断与断言(上海-悠悠)

第四章 判断与断言(上海-悠悠)

4.1 page页面 expect 断言的几种方式

- 4.1.1 页面断言
- 4.1.2 断言url
- 4.1.3 使用示例

4.2 expect 常用的断言方法

- 4.2.1 expect 使用
- 4.2.2 断言可见与不可见
- 4.2.3 断言是否被选中

4.3 如何判断元素是否存在

- 4.3.1 locator 定位元素
- 4.3.2 query_selector 定位

4.4 判断页面元素状态checkbox和radio

- 4.4.1 常用的元素判断方法
- 4.4.2 locator 定位后判断元素
- 4.4.3 page对象调用的判断方法

4.5 expect 断言输入框

- 4.5.1 断言输入框是否可编辑
- 4.5.2 断言输入框的内容
- 4.5.3 断言输入框的提示语
- 4.5.4 断言 class 属性

4.6 expect 断言打开新页面是否正常

- 4.6.1 expect_page() 断言打开新标签页
- 4.6.2 断言重定向页面

4.7 显示断言 expect_navigation

- 4.7.1 断言重定向页面

4.8 如何对比 2 张图片相似度

- 4.8.1 Ctrl+c/v 借代码
- 4.8.2 cv2.resize() 使用
- 4.8.3 使用示例

4.9 如何断言网页上图片正常显示

- 4.9.1 图片加载示例

▪4.9.2 对比2张图片

▪网易云视频课程

4.1 page页面 expect 断言的几种方式

当打开一个页面的时候，需要断言是否是期望的页面

PageAssertions类提供断言方法，可用于在测试中对页面状态进行断言。

4.1.1 页面断言

主要有四个断言方法

- to_have_title
- not_to_have_title
- to_have_url
- not_to_have_url

to_have_title() 确保页面具有给定的标题。

```
import re
from playwright.sync_api import expect
# 上海悠悠 wx:283340479
# blog:https://www.cnblogs.com/yoyoketang/

# ...
expect(page).to_have_title(re.compile(r".*checkout"))
```

参数

- title_or_reg_exp 预期的标题或正则表达式。
- timeout （可选） 超时时间， 默认5000ms

not_to_have_title 与expect(page).to_have_title()相反。

```
expect(page).not_to_have_title(title_or_reg_exp)
expect(page).not_to_have_title(title_or_reg_exp, **kwargs)
```

4.1.2 断言url

`to_have_url` 确保页面导航到给定的 URL。

```
import re
from playwright.sync_api import expect

# ...
expect(page).to_have_url(re.compile(".*checkout"))
```

`not_to_have_url` 与`expect(page).to_have_url()`相反。

```
expect(page).not_to_have_url(url_or_reg_exp)
expect(page).not_to_have_url(url_or_reg_exp, **kwargs)
```

4.1.3 使用示例

传统断言方式，先获取到值，再判断相等/不相等/包含 ...

这种获取值，再去判断的方式有缺陷：

获取的是当时页面上的状态，页面上状态可能会发生改变，比如点一个按钮，状态由 "未完成" 变成 "已完成"，可能会有延迟情况。

```
from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    context = browser.new_context()
    page = context.new_page()
    page.goto("https://www.baidu.com")

    title = page.title()
    print(title)
    assert title == "百度一下，你就知道"
    print(page.url)
    assert page.url == "https://www.baidu.com/"
```

playwright 推荐的断言方式，使用`expect`

```

from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    context = browser.new_context()
    page = context.new_page()
    page.goto("https://www.baidu.com")

    title = page.title()
    print(title)
    print(page.url)
    expect(page).to_have_title("百度一下，你就知道")
    expect(page).to_have_url("https://www.baidu.com/")

```

expect 断言 超时时间默认是5秒， 在5秒内判断达不到预期结果， 就报超时异常。

4.2 expect 常用的断言方法

playwright 提供了一个 expect方法 用于断言

4.2.1 expect 使用

断言	描述
expect(locator).to_be_checked()	Checkbox is checked
expect(locator).to_be_disabled()	Element is disabled
expect(locator).to_be_editable()	Element is enabled
expect(locator).to_be_empty()	Container is empty
expect(locator).to_be_enabled()	Element is enabled
expect(locator).to_be_focused()	Element is focused
expect(locator).to_be_hidden()	Element is not visible
expect(locator).to_be_visible()	Element is visible
expect(locator).to_contain_text()	Element contains text

断言	描述
<code>expect(locator).to_have_attribute()</code>	Element has a DOM attribute
<code>expect(locator).to_have_class()</code>	Element has a class property
<code>expect(locator).to_have_count()</code>	List has exact number of children
<code>expect(locator).to_have_css()</code>	Element has CSS property
<code>expect(locator).to_have_id()</code>	Element has an ID
<code>expect(locator).to_have_js_property()</code>	Element has a JavaScript property
<code>expect(locator).to_have_text()</code>	Element matches text
<code>expect(locator).to_have_value()</code>	Input has a value
<code>expect(locator).to_have_values()</code>	Select has options selected
<code>expect(page).to_have_title()</code>	Page has a title
<code>expect(page).to_have_url()</code>	Page has a URL
<code>expect(api_response).to_be_ok()</code>	Response has an OK status

4.2.2 断言可见与不可见

`to_be_visible()`使用示例

```
# 上海悠悠 wx:283340479
# blog:https://www.cnblogs.com/yoyoketang/

from playwright.sync_api import expect

locator = page.locator('.my-element')
expect(locator).to_be_visible()
```

4.2.3 断言是否被选中

`to_be_checked()`使用示例

```
from playwright.sync_api import expect

locator = page.get_by_label("Subscribe to newsletter")
expect(locator).to_be_checked()
```

4.3 如何判断元素是否存在

playwright 如何判断某个元素是否存在？

4.3.1 locator 定位元素

使用 locator 定位元素，不管元素存不存在，都会返回一个locator 对象，可以用到count() 方法统一元素的个数，如果元素个数是 0， 那么元素就不存在

```
"""
判断元素存在
# 上海悠悠 wx:283340479
# blog:https://www.cnblogs.com/yoyoketang/
"""

from playwright.sync_api import sync_playwright

with sync_playwright() as pw:
    browser = pw.chromium.launch()
    page = browser.new_page()

    page.goto("https://www.baidu.com/")

    # 元素存在
    loc1 = page.locator("id=kw")
    print(loc1)
    print(loc1.count())

    # 元素不存在
    loc2 = page.locator('id=yoyo')
    print(loc2)
    print(loc2.count())
```

运行结果

```
<Locator frame=<Frame name= url='https://www.baidu.com/'> selector='id=kw'>
1
<Locator frame=<Frame name= url='https://www.baidu.com/'> selector='id=yoyo'>
0
```

locator 是定位当前页面上的元素，不会自动等待，如果用click等方法结合使用，会自动去等待元素处于可点击状态。

4.3.2 query_selector 定位

ElementHandle 表示页内 DOM 元素。ElementHandles 可以使用page.query_selector()方法创建。

Locator和ElementHandle之间的区别在于后者指向特定元素，而 Locator 捕获如何检索该元素的逻辑。

元素存在返回元素句柄，元素不存在返回None

```
# 元素存在
loc1 = page.query_selector('#kw')
print(loc1) # JSHandle@node

# 元素不存在
loc2 = page.query_selector('#yoyo')
print(loc2) # None
```

也可以用query_selector_all 复数定位方式返回一个list

```
# 元素存在
loc1 = page.query_selector_all('#kw')
print(loc1) # [<JSHandle preview=JSHandle@node>]

# 元素不存在
loc2 = page.query_selector_all('#yoyo')
print(loc2) # []
```

对于用户来说，元素存不存在其实不重要，用户只关注元素的状态是可见还是不可见

4.4 判断页面元素状态checkbox和radio

在操作元素之前，可以先判断元素的状态。判断元素操作状态也可以用于断言。

4.4.1 常用的元素判断方法

page对象调用的判断方法, 传一个selector 定位参数

- `page.is_checked(selector: str)` # checkbox or radio 是否选中
- `page.is_disabled(selector: str)` # 元素是否可以点击或编辑
- `page.is_editable(selector: str)` # 元素是否可以编辑
- `page.is_enabled(selector: str)` # 是否可以操作
- `page.is_hidden(selector: str)` # 是否隐藏
- `page.is_visible(selector: str)` # 是否可见

locator 对象调用的判断方法

- `locator.is_checked()`
- `locator.is_disabled()`
- `locator.is_editable()`
- `locator.is_enabled()`
- `locator.is_hidden()`
- `locator.is_visible()`

元素句柄 的判断方法

- `element_handle.is_checked()`
- `element_handle.is_disabled()`
- `element_handle.is_editable()`
- `element_handle.is_enabled()`
- `element_handle.is_hidden()`
- `element_handle.is_visible()`

元素句柄 (element_handle) 是通过`page.query_selector()`方法调用返回的ElementHandle，这种一般不常用。

关于元素句柄和locator 定位的区别这篇有介绍<https://www.cnblogs.com/yoyoketang/p/17190635.html>

4.4.2 locator 定位后判断元素

locator 对象调用的判断方法

- locator.is_checked()
- locator.is_disabled()
- locator.is_editable()
- locator.is_enabled()
- locator.is_hidden()
- locator.is_visible()

is_checked() 用于判断checkbox or radio 的状态是否被选中

```
<div>
  <label>性别:
    <input type="radio" name="sex" id="man" checked>男
    <input type="radio" name="sex" id="woman">女
    <input type="radio" name="sex" id="no" disabled>人妖
  </label>
</div>
<div>
  <label>标签:
    <input type="checkbox" id="a1"> 旅游
    <input type="checkbox" id="a2">看书
    <input type="checkbox" id="a3" checked >学习
    <input type="checkbox" id="a4" checked disabled>学python
  </label>
</div>
```

性别: ☒ 男 ☐ 女 ☐ 人妖

标签: ☐ 旅游 ☐ 看书 ☒ 学习 ☒ 学python

代码示例

```
print(page.locator('#man').is_checked()) # checked
print(page.locator('#man').is_enabled())
print(page.locator('#no').is_checked())
print(page.locator('#no').is_enabled()) # disabled
```

返回结果

```
True
True
False
False
```

4.4.3 page对象调用的判断方法

page对象调用的判断方法, 传一个selector 定位参数

- `page.is_checked(selector: str)` # checkbox or radio 是否选中
- `page.is_disabled(selector: str)` # 元素是否可以点击或编辑
- `page.is_editable(selector: str)` # 元素是否可以编辑
- `page.is_enabled(selector: str)` # 是否可以操作
- `page.is_hidden(selector: str)` # 是否隐藏
- `page.is_visible(selector: str)` # 是否可见

使用示例

```
# 上海悠悠 wx:283340479
# blog:https://www.cnblogs.com/yoyoketang/
print(page.is_checked('#a3'))
print(page.is_enabled('#a3'))
print(page.is_checked('#a4'))
print(page.is_enabled('#a4'))
```

运行结果

```
True
True
True
False
```

总的来说有2种方式判断元素 `page.is_xx()` 和 `locator.is_xxx()`

4.5 expect 断言输入框

form 表单上输入框常见的一些断言方式

- 1.输入框是否可编辑

- 2.输入框输入内容后，获取输入框内容，判断输入是否正确
- 3.输入框字符长度限制，是否允许为空，最大长度等提示语
- 4.输入框的状态颜色

4.5.1 断言输入框是否可编辑

判断输入框是可编辑状态



代码示例

```
from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    context = browser.new_context()
    page = context.new_page()

    page.goto('http://47.108.155.10/login.html')

    # 输入框
    loc_username = page.get_by_label('用户名:')

    print(loc_username.is_editable())
    # 断言输入框可编辑
    expect(loc_username).to_be_editable()
```

4.5.2 断言输入框的内容

往输入框输入内容：hello

断言输入框的内容是：hello

```
from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    context = browser.new_context()
    page = context.new_page()

    page.goto('http://47.108.155.10/login.html')

    # 输入框
    loc_username = page.get_by_label('用户名:')
    # 1.输入内容
    loc_username.fill('hello')
    print(f'输入框的内容获取: {loc_username.input_value()}')
    # 断言
    expect(loc_username).to_have_value('hello')
    # 2.清空后
    loc_username.clear()
    # 断言
    expect(loc_username).not_to_have_value('hello')
```

4.5.3 断言输入框的提示语

断言元素可见，断言文本值内容'不能为空'

```

from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    context = browser.new_context()
    page = context.new_page()

    page.goto('http://47.108.155.10/login.html')

    # 不输入内容, 直接点提交按钮
    page.get_by_text('立即登录').click()
    loc_tips = page.locator('[data-fv-validator="notEmpty"][data-fv-for="username"]')
    # 断言可见
    expect(loc_tips).to_be_visible()
    # 断言文本内容
    print(loc_tips.text_content())
    expect(loc_tips).to_have_text('不能为空')

```

4.5.4 断言 class 属性

输入框显示红色, 后面有个x, 这种效果是class 属性控制的, 判断class属性即可

```

from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    context = browser.new_context()
    page = context.new_page()

    page.goto('http://47.108.155.10/login.html')

    # 不输入内容, 直接点提交按钮
    page.get_by_text('立即登录').click()
    loc_icon = page.locator('[data-fv-icon-for="username"]')
    # 断言class 属性有 glyphicon-remove

    print(loc_icon.get_attribute('class'))

    expect(loc_icon).to_have_class('form-control-feedback glyphicon glyphicon-remove')

```

4.6 expect 断言打开新页面是否正常

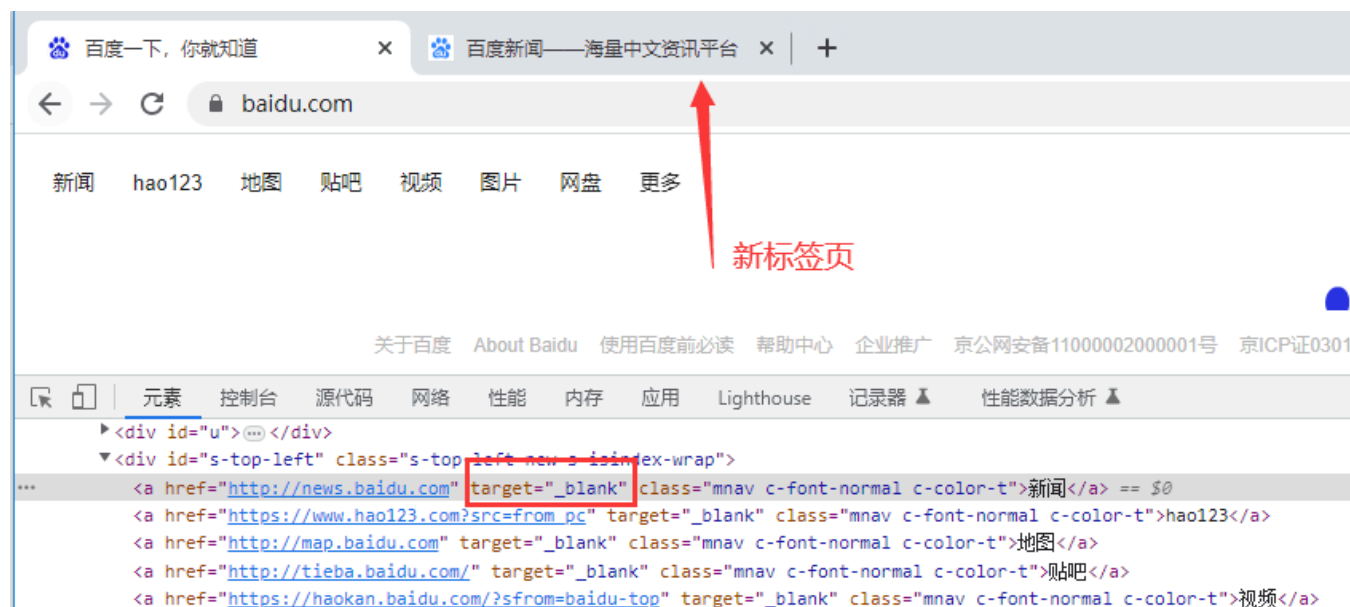
当我们点一个按钮，页面跳转到新的页面，如何判断新页面打开正常？

新页面打开会有2种情况

- 1.a标签带有 `target="_blank"` 属性的链接时，会打开一个新的标签页。
- 2.a标签不带 `target="_blank"` 属性，直接在当前页面刷新。或者点按钮页面重定向到新页面。

4.6.1 expect_page() 断言打开新标签页

使用示例，打开百度页面的-新闻链接，会出现一个新标签页



```

from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False, slow_mo=1000)
    context = browser.new_context() # 创建上下文, 浏览器实例

    page = context.new_page() # 打开标签页
    page.goto("https://www.baidu.com/")
    print(page.title())

    # 显示断言打开了新 tab 标签页
    with context.expect_page() as new_page_info:
        page.click('text=新闻') # Opens a new tab
    new_page = new_page_info.value

    print(new_page.title())
    print(new_page.url)
    # 断言新页面 title 和 url
    expect(new_page).to_have_title('百度新闻—海量中文资讯平台')
    expect(new_page).to_have_url('https://news.baidu.com/')

```

4.6.2 断言重定向页面

当点按钮时候, 出现重定向到其它页面, 比如点登录, 登录成功后重定向到index首页。
添加项目时, 重定向到列表页, 这些都是直接在当前tab页上刷新

第一种直接断言新页面

```

from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False, slow_mo=1000)
    context = browser.new_context() # 创建上下文, 浏览器实例

    page = context.new_page() # 打开标签页
    page.goto("http://47.108.155.10/login.html")
    page.get_by_placeholder("请输入用户名").fill("yoyo")
    page.get_by_placeholder("请输入密码").fill("aa123456")
    page.get_by_role("button", name="立即登录 >").click()

    # 断言新页面 title 和 url
    expect(page).to_have_title('首页')
    expect(page).to_have_url('**/index.html')

```

第二种可以使用 `page.expect_navigation()` 显示断言

```
from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False, slow_mo=1000)
    context = browser.new_context() # 创建上下文, 浏览器实例

    page = context.new_page() # 打开标签页
    page.goto("http://47.108.155.10/login.html")
    page.get_by_placeholder("请输入用户名").fill("yoyo")
    page.get_by_placeholder("请输入密码").fill("aa123456")
    page.get_by_role("button", name="立即登录 >").click()

    # 显示断言重定向
    with page.expect_navigation(url='**/index.html'):
        page.get_by_role("button", name="立即登录 >").click()
```

4.7 显示断言 `expect_navigation`

当我们点击某个按钮, 页面重定向请求后重新导航到其它页面, 有2种断言方式

- 1.直接断言page对象的title和url
- 2.`page.expect_navigation()` 显示断言

4.7.1 断言重定向页面

当点按钮时候, 出现重定向到其它页面, 比如点登录, 登录成功后重定向到index首页。
添加项目时, 重定向到列表页, 这些都是直接在当前tab页上刷新

第一种直接断言新页面


```

from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False, slow_mo=1000)
    context = browser.new_context() # 创建上下文, 浏览器实例

    page = context.new_page() # 打开标签页
    page.goto("http://47.108.155.10/login.html")
    page.get_by_placeholder("请输入用户名").fill("yoyo")
    page.get_by_placeholder("请输入密码").fill("aa123456")
    page.get_by_role("button", name="立即登录 >").click()

    # 断言新页面 title 和 url
    expect(page).to_have_title('首页')
    expect(page).to_have_url('http://47.108.155.10/index.html')
    # re 表达式
    expect(page).to_have_url(re.compile(".*index.html"))

```

第二种可以使用 page.expect_navigation() 显示断言

```

from playwright.sync_api import sync_playwright, expect

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False, slow_mo=1000)
    context = browser.new_context() # 创建上下文, 浏览器实例

    page = context.new_page() # 打开标签页
    page.goto("http://47.108.155.10/login.html")
    page.get_by_placeholder("请输入用户名").fill("yoyo")
    page.get_by_placeholder("请输入密码").fill("aa123456")
    page.get_by_role("button", name="立即登录 >").click()

    # 显示断言重定向
    with page.expect_navigation(url='**/index.html'):
        page.get_by_role("button", name="立即登录 >").click()

```

4.8 如何对比 2 张图片相似度

在做 web 自动化的时候, 有些场景需要去判断页面上的图片与预期的图片是否一样, 或者判断图片有没正确的加载出来, 需用到图片对比。

如果你之前接触过 airtest, 那么你应该知道它是专业搞图片对比的, 所以我们应该去那借点代

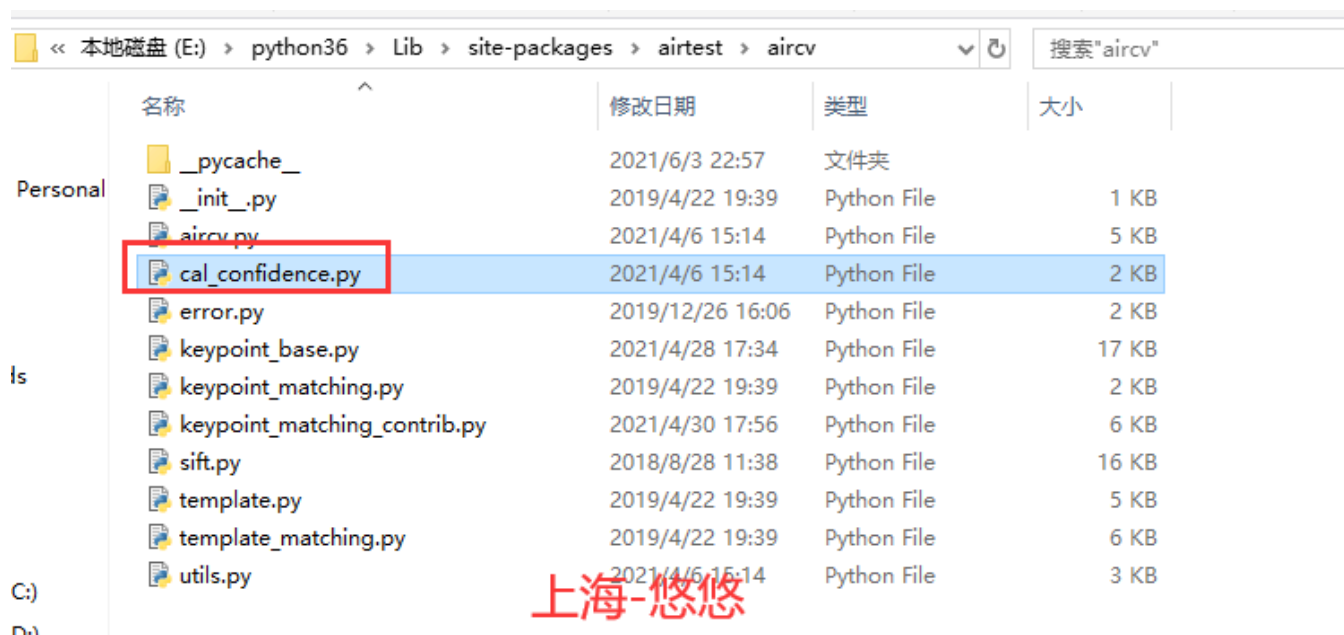
码过来！

环境准备：

```
pip install opencv-python
pip install numpy
```

4.8.1 Ctrl+c/v 借代码

找到 `Lib\site-packages\airtest\aircv` 目录下的 `cal_confidence.py` 文件，就是我们要借的代码了



复制后的完整代码

```

"""These functions calculate the similarity of two images of the same size."""

import cv2
import numpy as np

def img_mat_rgb_2_gray(img_mat):
    """
    Turn img_mat into gray_scale, so that template match can figure the img data.    print(type(im
    """
    # "input must be instance of np.ndarray"
    assert isinstance(img_mat[0][0], np.ndarray),
    return cv2.cvtColor(img_mat, cv2.COLOR_BGR2GRAY)

def cal_ccoeff_confidence(im_source, im_search):
    """求取两张图片的可信度，使用TM_CCOEFF_NORMED方法."""
    # 扩展置信度计算区域
    im_search = cv2.copyMakeBorder(im_search, 10, 10, 10, 10, cv2.BORDER_REPLICATE)

    im_source, im_search = img_mat_rgb_2_gray(im_source), img_mat_rgb_2_gray(im_search)
    res = cv2.matchTemplate(im_source, im_search, cv2.TM_CCOEFF_NORMED)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

    return max_val

def cal_rgb_confidence(img_src_rgb, img_sch_rgb):
    """同大小彩图计算相似度."""
    # 扩展置信度计算区域
    img_sch_rgb = cv2.copyMakeBorder(img_sch_rgb, 10, 10, 10, 10, cv2.BORDER_REPLICATE)
    # 转HSV强化颜色的影响
    img_src_rgb = cv2.cvtColor(img_src_rgb, cv2.COLOR_BGR2HSV)
    img_sch_rgb = cv2.cvtColor(img_sch_rgb, cv2.COLOR_BGR2HSV)
    src_bgr, sch_bgr = cv2.split(img_src_rgb), cv2.split(img_sch_rgb)

    # 计算BGR三通道的confidence，存入bgr_confidence:
    bgr_confidence = [0, 0, 0]
    for i in range(3):
        res_temp = cv2.matchTemplate(src_bgr[i], sch_bgr[i], cv2.TM_CCOEFF_NORMED)
        min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res_temp)
        bgr_confidence[i] = max_val

    return min(bgr_confidence)

```

主要用到2个方法

- `cal_ccoeff_confidence` 灰度对比，先把2张图片弄成黑白色，再去对比
- `cal_rgb_confidence` 彩色对比，这个对彩色的颜色要求更精准。

正常情况下，如果是对比相似度，用`cal_ccoeff_confidence`就可以了。

如果你要判断2个图片完全一样，那就用`cal_rgb_confidence` 彩色对比。

4.8.2 cv2.resize() 使用

使用语法：`cv2.resize(src, dsize, dst=None, fx=None, fy=None, interpolation=None)`

- `src` 源图像
- `dsize` 输出图像的大小
- `fx` width方向的缩放比例
- `fy` height方向的缩放比例
- `interpolation` 这个是指定插值的方式

因为对比2张图片先要把图片大小设置为一致，使用`cv2.resize()`方法处理原始的图片，用于对比。

4.8.3 使用示例

准备3张图片用于测试



```

if __name__ == '__main__':
    # 对比2个图片
    img1 = cv2.resize(cv2.imread('images/liu1.png'), (100, 100))
    img2 = cv2.resize(cv2.imread('images/liu2.png'), (100, 100))
    res1 = cal_ccoeff_confidence(img1, img2)
    print(res1) # 0.32
    img3 = cv2.resize(cv2.imread('images/liu3.png'), (100, 100))
    res2 = cal_ccoeff_confidence(img2, img3)
    print(res2) # 0.75

```

4.9 如何断言网页上图片正常显示

有些网页上会有一些图片，那么做web网页测试的时候，如何判断页面图片正常加载了，并且是期望的图片正常显示了？

4.9.1 图片加载示例

以个人头像为例

首页 > 个人中心 > 个人资料


个人资料

我的课程

我的收藏

我的消息

个人信息



修改头像

昵 称:

日 期:

生 日:

性 别: ☒ 男 ☐ 女

浏览器开发者工具截图显示HTML结构：

```

= 0.5 # 断言相似度
    page.pause()

```

网易云视频课程

网易云视频完整课程地址<https://study.163.com/course/>

[courseMain.htm?courseId=1213382811&share=2&shareId=480000002230338](https://study.163.com/course/courseMain.htm?courseId=1213382811&share=2&shareId=480000002230338)

Playwright + Python

Web 自动化测试

上海-悠悠