**大模型提示词提交**

**曹嘉楠-自动化测试脚本-1**

|  |  |
| --- | --- |
| User | 如果你是一名经验丰富的软件测试工程师，要求你根据测试用例编写自动化测试脚本，使用python+seleniumr，你会如何编写 |
| LLM | 作为一名经验丰富的软件测试工程师，编写自动化测试脚本是一项常规工作。使用Python、Selenium来编写自动化测试脚本，通常需要遵循以下步骤：  环境准备：  确保Python环境已安装。  安装Selenium库：可以通过pip install selenium命令安装。  下载与你的Chrome浏览器版本相匹配的ChromeDriver，并将其路径添加到系统环境变量中，或者在代码中指定路径。  编写测试脚本：  导入必要的模块。  设置WebDriver，启动Chrome浏览器。  编写测试用例，通常包括打开网页、定位元素、执行操作（如点击、输入文本等）、验证结果。  关闭浏览器。  下面是一个简单的示例脚本，假设我们要测试一个简单的登录功能：  python  from selenium import webdriver  from selenium.webdriver.common.keys import Keys  from selenium.webdriver.common.by import By  from selenium.webdriver.chrome.service import Service  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  # 指定ChromeDriver的路径  s = Service(executable\_path='path/to/chromedriver')  # 设置WebDriver  driver = webdriver.Chrome(service=s)  try:  # 打开网页  driver.get("http://example.com/login")  # 定位用户名和密码输入框，输入测试数据  username = driver.find\_element(By.NAME, "username")  password = driver.find\_element(By.NAME, "password")    username.send\_keys("testuser")  password.send\_keys("testpass")  # 定位登录按钮并点击  login\_button = driver.find\_element(By.NAME, "login")  login\_button.click()  # 验证登录是否成功，这里假设登录成功后会显示“欢迎回来”  WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.ID, "welcome-message"))  )  assert "Welcome back" in driver.find\_element(By.ID, "welcome-message").text  finally:  # 关闭浏览器  driver.quit()  测试用例管理：  将测试用例组织成测试套件，可以使用unittest框架来管理测试用例。  使用测试数据，可以通过外部数据文件（如CSV、JSON、数据库等）来管理测试数据。  异常处理：  在脚本中添加异常处理逻辑，确保在出现错误时能够优雅地关闭浏览器并记录错误信息。  报告生成：  使用Selenium提供的接口生成测试报告，或者集成第三方报告生成工具如Allure。  持续集成：  将自动化测试脚本集成到持续集成/持续部署(CI/CD)流程中，以便在代码提交时自动运行测试。  请注意，这只是一个基础的示例，实际的自动化测试脚本会更加复杂，需要根据具体的测试需求和项目特点来编写。 |
| User | 现在我将给出图书管理系统的地址为 http://120.26.37.204:8089/library/adminLogin.html ，账号为 librarian1 ，密码为 0123456 ，现在我需要测试第一张图片中的测试用例，第二张图片为借阅图书的原始页面，，如果该书籍借阅失败将如第四张图所示，如果借阅成功将在第四张图片中显示，请给我编写脚本，脚本需基于Python的selenium自动化测试进行编写，脚本都编写在同一个py文件里 |
| LLM | 根据您提供的信息和测试用例，我将为您编写一个Python脚本，使用Selenium进行自动化测试。这个脚本将模拟用户登录到图书管理系统，然后尝试借阅图书，并根据借阅结果进行验证。  首先，请确保您已经安装了Selenium库和相应的WebDriver。以下是脚本的代码：  python  from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.common.keys import Keys  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  # 设置ChromeDriver的路径  driver\_path = 'path/to/chromedriver'  driver = webdriver.Chrome(executable\_path=driver\_path)  try:  # 打开登录页面  driver.get("http://120.26.37.204:8089/library/adminLogin.html")  # 登录系统  username = driver.find\_element(By.NAME, "username")  password = driver.find\_element(By.NAME, "password")  username.send\_keys("librarian1")  password.send\_keys("0123456")  login\_button = driver.find\_element(By.NAME, "login")  login\_button.click()  # 导航到借阅书籍页面  driver.get("http://120.26.37.204:8089/library/borrowBook.html")  # 输入借阅信息  borrow\_id = driver.find\_element(By.NAME, "borrow\_id")  borrow\_id.send\_keys("123456") # 假设的借阅证号  book\_id = driver.find\_element(By.NAME, "book\_id")  book\_id.send\_keys("001") # 假设的图书编号  borrow\_date = driver.find\_element(By.NAME, "borrow\_date")  borrow\_date.send\_keys("2024-10-14") # 假设的借阅日期  # 点击借阅按钮  borrow\_button = driver.find\_element(By.XPATH, "//button[text()='借阅']")  borrow\_button.click()  # 等待结果弹窗出现  WebDriverWait(driver, 10).until(EC.alert\_is\_present())  # 获取弹窗文本并判断结果  alert = driver.switch\_to.alert  message = alert.text  if "借阅成功" in message:  print("借阅成功！")  else:  print("借阅失败：", message)  alert.accept()  finally:  # 关闭浏览器  driver.quit()  注意事项：  ChromeDriver路径：请将driver\_path变量的值替换为您本地ChromeDriver的实际路径。  借阅信息：脚本中使用的借阅证号、图书编号和借阅日期是假设的，您需要根据实际情况进行调整。  页面元素定位：如果页面结构发生变化，可能需要更新元素的定位方式（如By.NAME, By.XPATH等）。  异常处理：脚本中包含了基本的异常处理逻辑，确保在测试结束后浏览器能够正确关闭。  这个脚本将帮助您自动化测试图书借阅功能，并根据结果显示相应的信息。如果有任何问题或需要进一步的帮助，请随时告诉我！ |

**曹嘉楠-自动化测试脚本-2**

|  |  |
| --- | --- |
| User | 现在我将给出图书管理系统的地址为 http://120.26.37.204:8089/library/adminLogin.html ，账号为 librarian1 ，密码为 0123456 ，现在我需要测试第一张图片中的测试用例，第二张图片为借阅日期的原始页面，请给我编写脚本，脚本需基于Python的selenium自动化测试进行编写，脚本都编写在同一个py文件里 |
| LLM | 根据您提供的信息，我将为您编写一个基于Python和Selenium的自动化测试脚本。这个脚本将模拟登录到图书管理系统，并测试第一张图片中的测试用例。请注意，您需要先安装Selenium库以及对应的WebDriver。  以下是一个简单的脚本示例：  python  from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.common.keys import Keys  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  import time  # 设置WebDriver路径  driver\_path = 'path/to/your/webdriver'  # 初始化WebDriver  driver = webdriver.Chrome(executable\_path=driver\_path)  # 打开登录页面  driver.get('http://120.26.37.204:8089/library/adminLogin.html')  # 等待登录页面加载完成  try:  WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.NAME, "username"))  )  except TimeoutException:  print("登录页面加载超时")  # 输入用户名和密码  username = driver.find\_element(By.NAME, "username")  password = driver.find\_element(By.NAME, "password")  username.send\_keys('librarian1')  password.send\_keys('0123456')  # 点击登录按钮  login\_button = driver.find\_element(By.XPATH, '//input[@type="submit"]')  login\_button.click()  # 等待页面跳转  time.sleep(5)  # 测试用例：验证借阅日期是否正确显示  try:  borrow\_date = WebDriverWait(driver, 10).until(  EC.presence\_of\_element\_located((By.XPATH, '//input[@placeholder="请输入借阅日期"]'))  )  print("借阅日期输入框存在，测试通过。")  except TimeoutException:  print("借阅日期输入框不存在，测试失败。")  # 关闭浏览器  driver.quit()  请将driver\_path替换为您的WebDriver的实际路径。这个脚本首先打开登录页面，然后输入用户名和密码，点击登录按钮。登录后，它会检查借阅日期输入框是否存在，以验证测试用例是否通过。  请注意，这个脚本假设登录表单的元素名称是username和password，并且登录按钮的XPath是//input[@type="submit"]。如果实际的元素名称或XPath不同，您需要相应地修改脚本。  此外，由于网络延迟或其他原因，页面加载可能需要一些时间，因此我使用了time.sleep(5)来等待页面跳转。在实际使用中，您可能需要根据实际情况调整等待时间。使用显式等待（如WebDriverWait）通常是更好的做法，因为它会等待特定条件满足后再继续执行，而不是简单地等待固定的时间。 |

**曹嘉楠-自动化测试脚本-3**

|  |  |
| --- | --- |
| User | 现在我将给出图书管理系统的地址为 http://120.26.37.204:8089/library/adminLogin.html ，账号为 librarian1 ，密码为 0123456 ，现在我需要测试第一张图片中的测试用例，第二张图片为选择多少条每页的原始页面，第三张图片是点击多少条每页后出现的下拉框，假设点击20条/页，如果成功将如图四所示，一个页面将出现二十条借书信息否则将不成功。请给我编写脚本，脚本需基于Python的selenium自动化测试进行编写，脚本都编写在同一个py文件里 |
| LLM | 为了帮助你完成这个任务，我将提供一个基于Python和Selenium的自动化测试脚本。这个脚本将模拟用户登录到图书管理系统，然后测试“每页显示多少条”的功能是否按预期工作。  首先，确保你已经安装了Selenium库和相应的WebDriver。以下是一个示例脚本：  python  from selenium import webdriver  from selenium.webdriver.common.by import By  from selenium.webdriver.common.keys import Keys  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  from selenium.common.exceptions import NoSuchElementException  # 设置WebDriver路径  driver\_path = 'path/to/your/webdriver' # 替换为你的WebDriver路径  # 初始化WebDriver  driver = webdriver.Chrome(executable\_path=driver\_path)  try:  # 打开登录页面  driver.get("http://120.26.37.204:8089/library/adminLogin.html")  # 等待登录表单元素加载  wait = WebDriverWait(driver, 10)  username\_input = wait.until(EC.presence\_of\_element\_located((By.NAME, "username"))) # 替换为实际的用户名输入框的name或id  password\_input = wait.until(EC.presence\_of\_element\_located((By.NAME, "password"))) # 替换为实际的密码输入框的name或id  # 输入用户名和密码  username\_input.send\_keys("librarian1")  password\_input.send\_keys("0123456")  # 提交表单  login\_button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, "//button[@type='submit']"))) # 替换为实际的登录按钮的XPath  login\_button.click()  # 等待页面加载完成  wait.until(EC.presence\_of\_element\_located((By.XPATH, "//table[@class='borrowed-books']"))) # 替换为实际的表格XPath  # 选择每页显示20条  select\_element = wait.until(EC.presence\_of\_element\_located((By.XPATH, "//select[@name='rows']"))) # 替换为实际的选择框XPath  select\_element.click()  option\_20 = wait.until(EC.presence\_of\_element\_located((By.XPATH, "//option[@value='20']"))) # 替换为实际的20条选项XPath  option\_20.click()  # 验证是否显示20条记录  records = driver.find\_elements(By.XPATH, "//table[@class='borrowed-books']/tbody/tr")  assert len(records) == 20, "Expected 20 records, but found {}".format(len(records))  print("Test passed: 20 records are displayed per page.")  except Exception as e:  print("Test failed:", str(e))  finally:  # 关闭浏览器  driver.quit()  注意事项：  WebDriver路径：确保替换driver\_path为你的ChromeDriver或其他WebDriver的实际路径。  元素定位：根据实际的HTML结构，可能需要调整元素的定位方式（如By.NAME, By.XPATH等）。  错误处理：脚本中包含了基本的错误处理，确保在元素未找到或其他异常情况下能够给出提示。  这个脚本将帮助你自动化测试图书管理系统中的“每页显示多少条”功能。如果有任何问题或需要进一步的定制，请随时告诉我！ |
| User | # 验证是否显示20条记录  records = driver.find\_elements(By.XPATH, "//table[@class='borrowed-books']/tbody/tr") 这里的xpath是定位哪里的 |
| LLM | 在提供的脚本中，XPath "//table[@class='borrowed-books']/tbody/tr" 用于定位表格中的每一行记录。这个XPath表达式的意思是：  //table[@class='borrowed-books']：查找页面上所有具有类名 borrowed-books 的 <table> 元素。  /tbody/tr：在找到的表格元素中，进一步定位到 <tbody> 标签下的所有 <tr> 元素，这些 <tr> 元素代表表格中的行。  因此，这个XPath表达式最终会返回页面上所有属于 borrowed-books 表格的行元素，通常这些行包含了表格的记录数据。  如果你的表格结构不同，或者类名不是 borrowed-books，你需要根据实际的HTML结构调整这个XPath。例如，如果你的表格没有类名，或者行元素直接位于 <table> 标签下而不是 <tbody>，你可能需要修改XPath来适应这些差异。  如果你不确定表格的具体结构，可以使用浏览器的开发者工具（通常可以通过按F12键打开）来检查表格的HTML代码，并据此构造正确的XPath。在开发者工具中，你可以右键点击表格或行元素，选择“检查”或“审查元素”，然后在出现的HTML代码中找到你想要定位的元素的路径。 |

**曹嘉楠-自动化测试脚本-4**

|  |  |
| --- | --- |
| User | 现在我将给出图书管理系统的地址为 http://120.26.37.204:8089/library/adminLogin.html ，账号为 librarian1 ，密码为 0123456 ，现在我需要测试第一张图片中的测试用例，第二张图片为输入页码的页面，如果输入非数字页码时，输入内容将被清空，点击确定也无响应如测试成功，否则失败。请给我编写脚本，脚本需基于Python的selenium自动化测试进行编写，脚本都编写在同一个py文件里 |
| LLM | 根据您的要求，我将提供一个基于Python和Selenium的自动化测试脚本。这个脚本将模拟登录过程，输入非数字页码，并验证系统的反应是否符合预期。  首先，确保您已经安装了Selenium库和相应的WebDriver。以下是一个简单的测试脚本示例：  python  from selenium import webdriver  from selenium.webdriver.common.keys import Keys  from selenium.webdriver.common.by import By  import time  # 初始化WebDriver  driver = webdriver.Chrome()  try:  # 打开登录页面  driver.get("http://120.26.37.204:8089/library/adminLogin.html")    # 等待页面加载  time.sleep(2)    # 找到用户名和密码输入框并输入信息  username\_input = driver.find\_element(By.NAME, "username") # 假设用户名输入框的name属性为"username"  password\_input = driver.find\_element(By.NAME, "password") # 假设密码输入框的name属性为"password"    username\_input.send\_keys("librarian1")  password\_input.send\_keys("0123456")    # 找到登录按钮并点击  login\_button = driver.find\_element(By.XPATH, "//button[@type='submit']") # 假设登录按钮的xpath为"//button[@type='submit']"  login\_button.click()    # 等待登录完成  time.sleep(2)    # 找到页码输入框并输入非数字页码  page\_input = driver.find\_element(By.NAME, "page") # 假设页码输入框的name属性为"page"  page\_input.send\_keys("abc")    # 模拟按下回车键  page\_input.send\_keys(Keys.RETURN)    # 等待系统反应  time.sleep(2)    # 验证输入内容是否被清空  assert page\_input.get\_attribute("value") == ""    # 验证点击确定后是否有响应  # 这里需要根据实际页面元素和逻辑来编写验证代码，例如检查某个元素是否出现或消失  # 例如：  # assert driver.find\_element(By.ID, "expected\_element\_id").is\_displayed()    print("测试成功！")    except Exception as e:  print("测试失败：", e)  finally:  # 关闭浏览器  driver.quit()  请注意，这个脚本中的元素定位（如By.NAME, By.XPATH等）需要根据实际的HTML结构进行调整。您需要检查登录页面和输入页码页面的HTML代码，找到正确的元素定位方法。  此外，最后的验证步骤需要根据实际的页面逻辑来编写。例如，如果点击确定后应该出现一个提示框，那么您需要检查这个提示框是否出现。如果页面上某个元素应该消失或出现，那么您需要检查这个元素的状态。  请根据您的实际需求调整和完善这个脚本。 |
| User | 无法通过value值判断该怎么编写代码 |
| LLM | 如果无法通过value值来判断输入内容是否被清空，我们可以考虑其他方法来验证输入框是否为空。例如，我们可以通过检查输入框的textContent或者innerText属性来判断。但是，这些属性通常不适用于输入框，因为它们主要用于显示文本的元素。  在这种情况下，我们可以使用Selenium的get\_attribute方法来获取输入框的value属性，即使这个属性在DOM中不可见。如果输入框被清空，那么这个属性的值应该是空字符串。  另外，如果点击确定后页面没有响应，我们可以通过检查页面的URL是否改变，或者页面上某个特定元素是否出现或消失来判断。如果页面没有跳转，且没有新元素出现，那么我们可以认为点击确定后页面没有响应。  下面是一个修改后的脚本示例：  python  from selenium import webdriver  from selenium.webdriver.common.keys import Keys  from selenium.webdriver.common.by import By  import time  # 初始化WebDriver  driver = webdriver.Chrome()  try:  # 打开登录页面  driver.get("http://120.26.37.204:8089/library/adminLogin.html")    # 等待页面加载  time.sleep(2)    # 找到用户名和密码输入框并输入信息  username\_input = driver.find\_element(By.NAME, "username") # 假设用户名输入框的name属性为"username"  password\_input = driver.find\_element(By.NAME, "password") # 假设密码输入框的name属性为"password"    username\_input.send\_keys("librarian1")  password\_input.send\_keys("0123456")    # 找到登录按钮并点击  login\_button = driver.find\_element(By.XPATH, "//button[@type='submit']") # 假设登录按钮的xpath为"//button[@type='submit']"  login\_button.click()    # 等待登录完成  time.sleep(2)    # 找到页码输入框并输入非数字页码  page\_input = driver.find\_element(By.NAME, "page") # 假设页码输入框的name属性为"page"  page\_input.send\_keys("abc")    # 模拟按下回车键  page\_input.send\_keys(Keys.RETURN)    # 等待系统反应  time.sleep(2)    # 验证输入内容是否被清空  assert page\_input.get\_attribute("value") == ""    # 验证点击确定后是否有响应  # 这里需要根据实际页面元素和逻辑来编写验证代码  # 例如，检查页面URL是否改变，或者某个元素是否出现或消失  current\_url = driver.current\_url  time.sleep(1) # 等待可能的页面响应  new\_url = driver.current\_url  assert current\_url == new\_url, "页面URL发生了变化，表明有响应"    print("测试成功！")    except Exception as e:  print("测试失败：", e)  finally:  # 关闭浏览器  driver.quit()  请注意，这个脚本中的元素定位（如By.NAME, By.XPATH等）需要根据实际的HTML结构进行调整。您需要检查登录页面和输入页码页面的HTML代码，找到正确的元素定位方法。  此外，最后的验证步骤需要根据实际的页面逻辑来编写。例如，如果点击确定后应该出现一个提示框，那么您需要检查这个提示框是否出现。如果页面上某个元素应该消失或出现，那么您需要检查这个元素的状态。  请根据您的实际需求调整和完善这个脚本。 |