Candidate Test Solution

QA & Automation Developer Home Assignment

Mohammad H. Bakri

0544508285

The Below Solutions were done a MacBook Pro with Visual Code Studio implementing Python 3.9.6

1. **QA Planning Exercise**

Assume we are developing a code for "replace a string" software utility, activation is from the command line and it is activated as such that **all** occurrences of **string2** within **string1** will be replaced with **string3**:

page2image18112576

./replace.py “string1” “string2” “string3”

*for example:*

./replace.py “dkmjeircuj874357mfdujm3eu934**jjjj**2398” “**jjjj**” “**1984**”

*will result in the following output:*

dkmjeircuj874357mfdujm3eu934**1984**2398

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **TestCase Description** | **String 1** | **String 2** | **String 3** | **Expected Output** |
| 1 | Test the expected behavior | dkmjeircuj874357mfdujm3eu934**jjjj**2398 | **jjjj** | **1984** | dkmjeircuj874357mfdujm3eu934**1984**2398 |
| 2 | Test expected behavior with numbers | dkmjeircuj**8743**57mfdujm3eu934jjjj2398 | 8743 | 1234 | Dkmjeircuj123457mfdujm3eu934**jjjj**2398 |
| 3 | Test expected behavior with alphanumeric | dkmjeirc**uj87**4357mfdujm3eu934jjjj2398 | uj87 | aaaa | dkmjeircaaaa4357mfdujm3eu934**jjjj**2398 |
| 4 | Test expected behavior with symbols | dkmjei**@^ns**874357mfdujm3eu934jjjj2398 | @^ns | #### | dkmjei@^ns874357mfdujm3eu934**jjjj**2398 |
| Negative Testing | | | | | |
| 5 | Test with string of mismatched length |  | **jjj** | aaaa | Should return error of string not of same length |
| 6 | Test with string of mismatched length (vise versa) |  | jjjj | aa | Should return error of string not of same length |
| 7 | Test with empty string |  |  |  | Should return error of empty string |
| 8 | Missing argument |  |  |  | Should return one or more arguments missing |

1. **Q2**

1.Explain what is implicit and explicit wait and when we do use it?

**Implicit** wait is a static wait , where the programmer sets a certain time wait period after which it performs the requested operation , this time period is usually defined by trial and error and is usually used when for example you want to wait for a full webpage to load where elements load in random order so there is no other indication that the page has loaded fully.

**Explicit** wait is a dynamic wait, where the programmer sets the code to wait for a certain condition to happen, it is a more intelligent and an improvement to implicit wait as the wait period is defined by the webpage loading itself and not by the user, it is preferred over implicit as it is more stable and tolerant to changes in webpages or internet connection speeds.

2. Given the following web site: <https://rahulshettyacademy.com/seleniumPractise/#/>

#### Write a code that searches for 3 products, adding them to the cart, proceed to checkout, applies a discount code and verifies that the discount code is accepted.

**Discount code: “rahulshettyacademy”**

The below code was written on a macOS on Visual Studio Code using Python 3.9.6 and selenium 4.7.2

import time

from selenium import webdriver

from selenium.webdriver.chrome.options import Options

from selenium.webdriver.common.by import By

chrome\_options = Options()

chrome\_options.add\_experimental\_option("detach", True)

promoCode = "rahulshettyacademy"

site="https://rahulshettyacademy.com/seleniumPractise/#/"

products = ["Brocolli", "Tomato","Pumpkin "] # list of products to add to cart

driverLocation = '/Users/mohammadba/Downloads/chromedriver' # Local WebDriver Location

driver = webdriver.Chrome(driverLocation,*chrome\_options*=chrome\_options)

driver.get(site)

search = driver.find\_element(By.CLASS\_NAME, "search-keyword")

for element in products:

search.send\_keys(element)

time.sleep(1)

button = driver.find\_element(By.CLASS\_NAME, "products").find\_element(By.TAG\_NAME, "button").click()

time.sleep(0.5)

search.clear() # reset search

CartClick= driver.find\_element(By.CLASS\_NAME, "cart-icon").click()

ProceedToCheckout = driver.find\_element(By.CLASS\_NAME, "action-block").find\_element(By.TAG\_NAME,"button").click()

time.sleep(0.5)

ProceedToCheckout = driver.find\_element(By.CLASS\_NAME,"promoCode")

promoBtn = driver.find\_element(By.CLASS\_NAME,"promoBtn")

time.sleep(1)

ProceedToCheckout.send\_keys(promoCode)

time.sleep(0.2)

promoBtn.click()

time.sleep(5) # Wait for Code Validation

promoBtnColor = driver.find\_element(By.CLASS\_NAME,"promoInfo").get\_attribute('style')

if("green" in promoBtnColor):

print("Promo Code was successful")

else:

print("Promo Code is invalid")

driver.close() # Close WebPage

1. **3 API Automation Exercise**

In this exercise you are required to write automation code for testing the public API IPINFO *https://ipinfo.io/161.185.160.93/geo* - please implement at least 3 different scenarios for testing

1. Please specify the different scenarios you are going to test, for each scenario please write the scenario details and expected behavior.

2. Please implement automation code to test the scenarios you described above, please specify which tools are used to perform the tests

The below code was written on a macOS on Visual Studio Code using Python 3.9.6 with the import of requests and json

import requests

import json

ipaddress = "161.185.160.93"

response = requests.get("https://ipinfo.io/"+ipaddress+"/geo")

|  |  |  |  |
| --- | --- | --- | --- |
| # | Case | Expected behavior | Automation code |
| 1. | check that the api returns an OK response (200) and a valid json | Send api request , expect a json in return and OK in repones header | if(response.status\_code!=200):  print("Web Page did not respond correctly " + response.status\_code)  print(response.status\_code)  try:  json.loads(text)  except ValueError as error:  print(error) |
| 2. | Check that the json contains the same ip address as in the request header | Compare ip from json to ip in the request header | text = json.dumps(response.json()  , *sort\_keys*=True, *indent*=4)  print(text)  if(ipaddress in text):  print("IP address is valid: " + ipaddress)  else:  print("looks like you are behind a firewall and being natted") |
| 3. | check that the json contains at least the following fields [city, timezone, country, loc] | Checks for fields and returns error if any of them doesn’t exist | checkfor= ["city","timezone","country","loc"]  for element in checkfor:  try:  test1 = response.json()[element]==""  test2 = response.json()[element]==None  if(test1 or test2):  print(element + " is empty or does not exist")    except KeyError as e:  print("The Key " + element + " is empty or does not exist") |

1. **4.Bonus Question**

Write a function that takes a list of integers as an argument and returns the max single-digit number.

For example: [-5, 94, 1001, -100, 76, 1, 0, 503]

The function should return 1.

numbers = [-5, 94, 1001, -100, 76, 0,5, 503]

newArray = []

for i in numbers:

if len(str(i)) > 1 :

pass

else:

newArray.append(i)

print(max(newArray))

Thank you