Selenium Assignment Questions

Assignment 1

1. Download and launch the "dropdown.html" file.
2. Select date 05-05-2005 from the dropdown and validate the same.
3. Fetch the year from the dropdown and validate the year in Ascending Order.

Dropdown.html

Answer:

```
from selenium import webdriver
import os
import time
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options

# Setup to suppress warnings
options = Options()
options.add_experimental_option('excludeSwitches', ['enable-logging'])

# Open dropdown.html
html_file_path = os.path.abspath("./dropdown.html")
driver = webdriver.Chrome(options=options)
driver.get("file://" + html_file_path)

# Wait for a few seconds so the page stays open
time.sleep(5)

# Close the browser
driver.quit()
```

Assignment 2

1. Download and launch the "Assignment.html" file.
2. Launch the file.
3. Read the table and find the unique rows from the table.

Assignment.html

Answer:

```
from selenium import webdriver

from selenium.webdriver.support.ui import Select

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.chrome.options import Options

import os

import time


# Setup to suppress warnings

options = Options()

options.add_experimental_option('excludeSwitches', ['enable-logging'])


# Open dropdown.html

html_file_path = os.path.abspath("./dropdown.html")

driver = webdriver.Chrome(options=options)

driver.get("file://" + html_file_path)


# Give the browser time to load
```

```
time.sleep(1)


# Step 1: Locate the dropdowns by their name or id (replace with actual values)
day_dropdown = Select(driver.find_element("id", "dob-day"))
month_dropdown = Select(driver.find_element("id", "dob-month"))
year_dropdown = Select(driver.find_element("id", "dob-year"))

# Step 2: Select date 05-05-2005
day_dropdown.select_by_visible_text("05")
month_dropdown.select_by_visible_text("May")
year_dropdown.select_by_visible_text("2005")

# Step 3: Validate selected values
selected_day = day_dropdown.first_selected_option.text
selected_month = month_dropdown.first_selected_option.text
selected_year = year_dropdown.first_selected_option.text

assert selected_day == "05", "Day is not selected correctly"
assert selected_month == "May", "Month is not selected correctly"
assert selected_year == "2005", "Year is not selected correctly"

print(f"Selected date: {selected_day}-{selected_month}-{selected_year} ✅ ")

# Optional wait
time.sleep(2)

# Close browser
driver.quit()
```

Assignment 3

2022 Elections

https://results.eci.gov.in/ResultAcGenMar2022/ConstituencywiseS0510.htm?ac=10

Each question is state wise

1. Output should be name of constituency, candidate name, and vote number/percentage or whatever is the deciding factor, dump all the data in excel with column (all column+state+constituency name).
2. get the candidate which has got the maximum vote in each state with their constituency name.
3. get the candidate which has got the maximum percentage of vote in each state with their constituency name. (percentage)
4. candidate who won with maximum vote difference.
5. candidate who won with maximum vote percentage difference.
6. candidate who won with the minimum vote.
7. candidate who won with minimum vote percentage.
8. total count of candidate who have got less vote than nota.
9. total count of candidates who have gotten greater than 50% vote.
10. name of candidate who has got minimum vote in each state.

Answer: Objective:
- Extract data from state-wise election pages
- Dump into Excel
- Derive metrics

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
import pandas as pd
driver = webdriver.Chrome()
data = []
state_urls = {
  "Goa":
"https://results.eci.gov.in/ResultAcGenMar2022/ConstituencywiseS0510.htm?ac=10",
}
for state, url in state_urls.items():
  driver.get(url)
  constituency = driver.find_element(By.XPATH,
"//span[@id='ctl00_ContentPlaceHolder1_lblACName']").text
  rows = driver.find_elements(By.XPATH, "//table[2]//tr")
```

```
for row in rows[1:]:
    cells = row.find_elements(By.TAG_NAME, "td")
    if len(cells) >= 4:
      data.append({
        "State": state,
        "Constituency": constituency,
        "Candidate": cells[0].text,
        "Party": cells[1].text,
        "Votes": int(cells[2].text.replace(",", "")),
        "Percentage": float(cells[3].text.replace('%', '')) if cells[3].text else 0.0
      })
driver.quit()
df = pd.DataFrame(data)
df.to_excel("election_data.xlsx", index=False)
```

Post-processing:

```python
max_vote_per_state = df.loc[df.groupby('State')['Votes'].idxmax()]
max_pct_per_state = df.loc[df.groupby('State')['Percentage'].idxmax()]
```

Assignment 4

Please do the following assignment for cucumber framework -

1. Install Cucumber

2. Create a Cucumber project

3. Use the attached feature file and implement the stepDefinitions for all the scenarios in the feature file. (You can use dummy code in the stepDefinition methods)

Login.feature

4. Execute TestRunner.

5. Assign tags to specific scenarios in the feature file and execute TestRunner for those particular tags.

Answer: **TestRunner.java**

```
package Runners;

import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/Features",
    glue = {"StepDefinations"},
    plugin = {"pretty", "html:target/cucumber.html"}
)
public class TestRunner {
}
```

LoginStepDefinitions.java

```
package StepDefinations;

import io.cucumber.java.en.*;

public class LoginStepDefinations {

  @Given("User is on NetBanking landing page")
  public void user_is_on_net_banking_landing_page() {
    System.out.println("User navigated to NetBanking landing page");
  }

  @When("User login into application with username and password")
  public void user_login_with_username_and_password() {
    System.out.println("Logging in with default credentials");
  }

  @Then("Home page is populated")
  public void home_page_is_populated() {
    System.out.println("Home page displayed successfully");
  }
```

```
  @And("Cards are displayed")
  public void cards_are_displayed() {
    System.out.println("Cards are visible");
  }

  @When("User login into application with {string} and password {string}")
  public void user_login_with_username_and_password_param(String username, String
password) {
    System.out.println("Logging in with user: " + username + ", password: " + password);
  }

  @And("Cards displayed are {string}")
  public void cards_displayed_are(String status) {
    System.out.println("Cards displayed status: " + status);
  }

  @When("^User login in to application with (.+) and password (.+)$")
  public void user_login_in_to_application_with_and_password(String username, String
password) {
    System.out.println("Scenario Outline - Username: " + username + ", Password: " +
password);
  }


}
```

login.feature

```
Feature: Application Login

Scenario: Home page default login
Given User is on NetBanking landing page
When User login into application with username and password
Then Home page is populated
And Cards are displayed
```

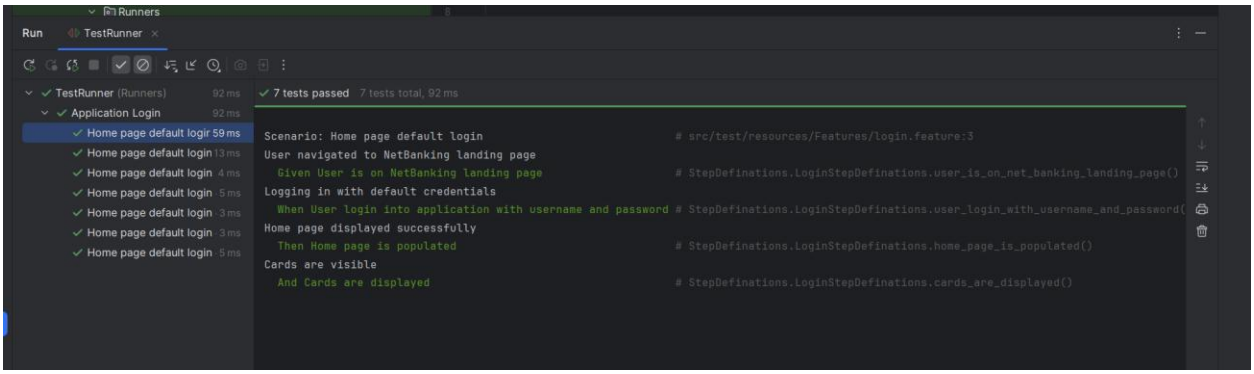
Scenario: Home page default login
Given User is on NetBanking landing page
When User login into application with "jin" and password "1234"
Then Home page is populated
And Cards displayed are "true"

Scenario: Home page default login
Given User is on NetBanking landing page
When User login into application with "john" and password "4321"
Then Home page is populated
And Cards displayed are "false"

Scenario Outline: Home page default login
Given User is on NetBanking landing page
When User login in to application with <Username> and password <password>
Then Home page is populated
And Cards displayed are "true"

Examples:
Username	password
user1	pass1
user2	pass2
user3	pass3
user4	pass4

OUTPUT:



Assignment 5

Please do the following assignment for TestNG framework -

1. Install TestNG

2. Create a TestNG Project

3. Create 2 test classes (with 3 test cases each).

4. Keep the 2 test classes in 2 different <test> tags in testng.xml

5. Execute the tests above using testng.xml

6. Assign a group to a few test cases and update testng.xml to run test cases belonging to the group.

7. Assign priority to the test cases.