Python (Part-1)

SELENIUM AUTOMATION TESTING

10.Aug.2023

Index

Selenium introduction

- "Selenium" provides a suite of tools and libraries that enable developers and testers to automate browser actions, interact with web elements, and perform functional testing on web applications with support for Java, Python, C#, and more.
- Web Testing: Selenium is commonly used for automating functional and regression testing of web applications.
- Cross-Browser Testing: Selenium supports multiple web browsers, i Firefox, Safari, Edge, and more.
- Automation of Repetitive Tasks: Selenium can automate repetitive t submissions, data extraction, and content validation.

Why python for automation testing

- Ease of Learning and Readability: Python's clean and readable syntax mak it easy for beginners to learn and write code.
- Python has a rich ecosystem of libraries and frameworks that can complement Selenium automation.
- Ease of Integration: Python can easily integrate with other technologies, tools, and databases, making it suitable for end-to-end testing scenarios where different components need to interact.
- Scalability: Python's ability to handle both small-scale and large-scale projects makes it suitable for test automation across different project sizes.

How to install Python

- Go to office website

Windows: https://www.python.org/downloads/windows/

- Download the latest version of Python for Windows. Make sure to choose the appropriate version (Python 3.x is recommended).

- Run the Installer: - Locate the downloaded installer executable file (it should have a .exe extension) and double-click on it to run the installer.
- Add Python to PATH: - On the "Customize installation" screen, ensure that the option "Add Python X.X to PATH" (where "X.X" represents the version number) is checked. This will make it easier to run Python from the Command Prompt.
- Verify Installation:

```
python --version
```

How to install Selenium

- Installing Selenium: Use Below command on PIP to install Selenium Package

pip install selenium

Or

pip3 install selenium (Python 3.x version)

- This command will set up the Selenium WebDriver client library on your machine with all modules and classes.

- pip install -U selenium

  The optional –U flag will upgrade the existing version of the installed package

Data types in Python

- Integer (int): Whole numbers without decimal points.

age = 25

- Floating-Point Number (float): Numbers with decimal points.

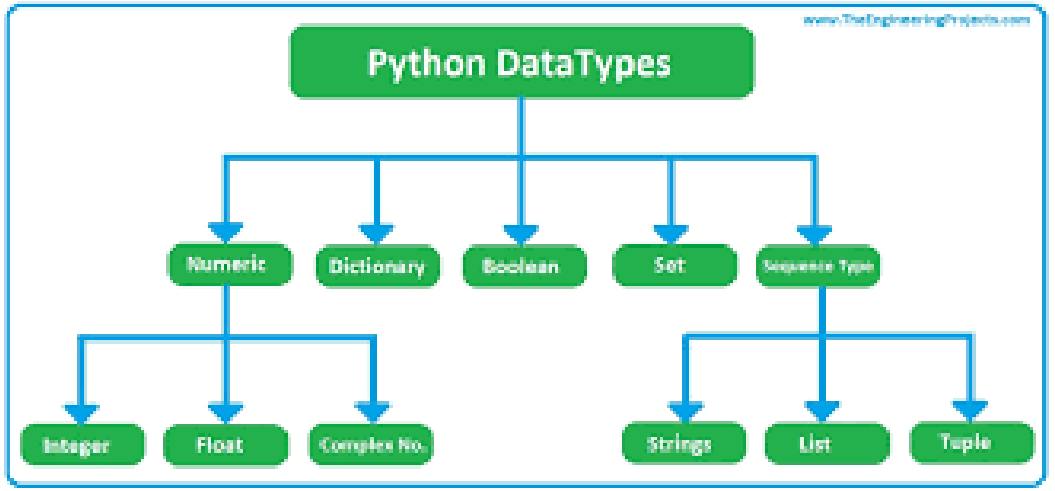
price = 19.99

- String (str): Sequence of characters enclosed in single, double, or triple quotes.

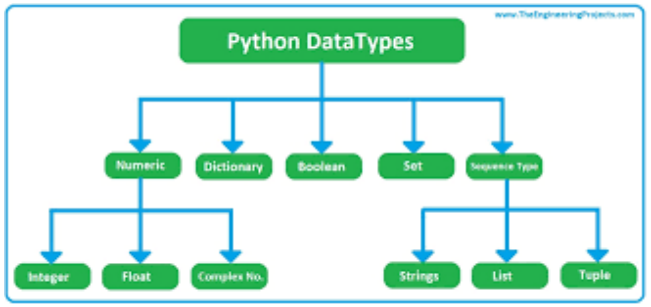
name = "Alice"

- Boolean (bool): Represents either

is_student = True



Data types in Python



- List: Ordered collection of items, mutable.

```
fruits = ['apple', 'banana', 'cherry']
```

- Tuple: Ordered collection of items, immutable.

```
coordinates = (3, 5)
```

- Dictionary: Collection of key-value pairs, keys are unique and immutable.

```
person = {'name': 'John', 'age': 30, 'city': 'New York'}
```

- Set: Collection of unique items, unordered and mutable.

```
unique_numbers = {1, 2, 3, 4, 5}
```

- Range: Represents an immutable sequence of numbers.

```
numbers = range(1, 6) # Represents 1, 2, 3, 4, 5
```

Control structure

- If condition

```
if condition: # Code to execute if the condition is true
else: # Code to execute if the condition is false
```

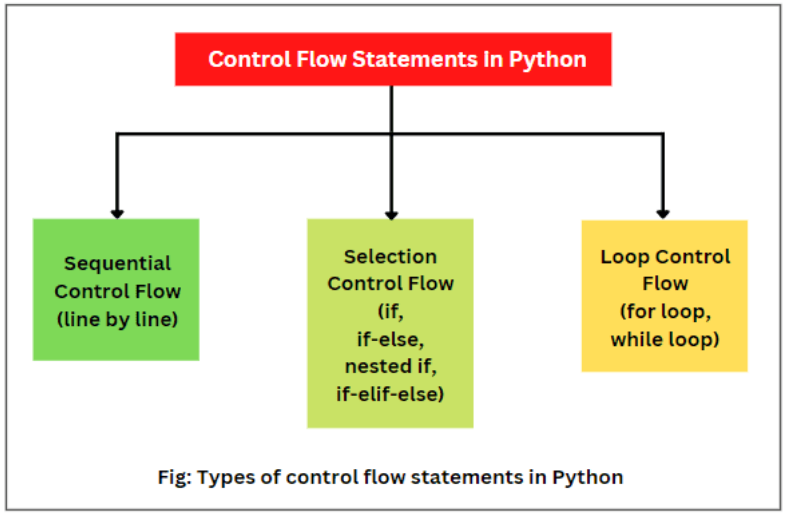


Fig: Types of control flow statements in Python

- For loop

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)
```

- While loop

```
while condition:
    # Code to be executed as long as the condition is true
```

Selenium setup

- Import packages

from selenium import webdriver
from selenium.webdriver.chrome.service import Service

- create a service object

service = Service()

- Create a webdriver instance

driver = webdriver.Chrome(service=service)

- Open a web page

driver.get('<application url>')
timer.sleep(3)

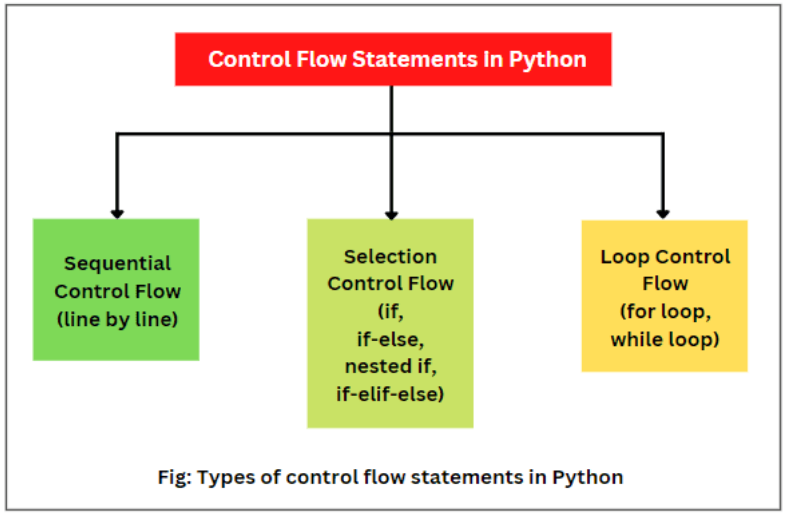


Fig: Types of control flow statements in Python

Selenium setup

- Open a website & maximize the window

```
with webdriver.Chrome(service=service) as
driver:
    # open website using driver
    driver.get("<URL>")
    # maximize the browser window
    driver.maximize_window()
    # print the title of the page
    print(driver.title)
    # wait for 2 seconds
    time.sleep(2)
```

Selenium setup

- Implicit wait & page load

```
# implement implicit wait
driver.implicitly_wait(20)

# implement page load timeout
driver.set_page_load_timeout(20)
```

- It is recommended to locate web elements using ID where ever possible instead of using relative XPATH.

- This will avoid any impact of a UI design change with selenium scripts.

Selenium automation basic

- Find the element by name

```
# enter user name & password
username_input = driver.find_element(By.NAME, 'username')
username_input.send_keys('<user-name>')
# enter password
password_input = driver.find_element(By.NAME, 'password')
password_input.send_keys('<password>')
```

- click on login button

```
login_button = driver.find_element(By.XPATH, '//button[text()="Log
In"]')
login_button.click()
```

- click on left navigation menu

```
public_online_booking = driver.find_element(By.XPATH,
'//span[text()="<label>"]')
public_online_booking.click()
```

Selenium automation basic

- Enter a text into a text field

```
        search_by_dropdown = driver.find_element(By.XPATH,
"//input[@placeholder='Search by Code']")
        # enter text in dynamic dropdown 'PUB100' & select 'PUB100003'
        search_by_dropdown.click()
        search_by_dropdown.send_keys('PUB100')
```

- Select a dropdown item

```
        religion_dropdown = driver.find_element(By.XPATH,
"//select[@name='religion']")
        # Create a Select object to work with the select element
        select_religion = Select(religion_dropdown)
        # Select an option by its text (e.g., "HINDU")
        select_religion.select_by_visible_text('HINDU')
```

Selenium automation basic

- Select a calendar

```
        date_of_birth = driver.find_element(By.XPATH,
"//input[@id='maxDOB']")
        date_of_birth.send_keys('01/01/1980')
```

- Select item from a dynamic dropdown

```
        country_input = driver.find_element(By.ID, "country")
       country_input.send_keys("ind")
```

- Explicit wait for data loading

```
        wait = WebDriverWait(driver, 10)
        wait.until(EC.presence_of_element_located((By.LINK_TEXT,
"India")))
        # click on <a xpath="1">India</a>
        driver.find_element(By.LINK_TEXT, "India").click()
```

Selenium automation basic

- Using CSS_SELECTOR – syntax "<element>.<class-name>"

```
driver.find_element(By.CSS_SELECTOR,'li.o_m2o_dropdown_option').click()

driver.find_element(By.CSS_SELECTOR, "input.btn.btn-success.btn-
lg").click()
```

- Locate an element with matching label

```
cart_items = driver.find_element(By.XPATH, "//a[contains(text(),
'Checkout')]")
```

References

- What is selenium: http://surl.li/kcxzt
- Selenium with Python: http://surl.li/kcyab
- Python tutorials: http://surl.li/kcyak
- Selenium with Python full tutorial: https://youtu.be/2DD-ynCIZ4w

Thank you