



Products

Developers

Live  
for  
Teams

Pricing

# Selenium Python Tutorial (with Example)

[Guide](#)  
[Categories](#)

Press

[Home](#)[Testing on Cloud](#)[Debugging](#)[Best Practices](#)[Tools & Frameworks](#)[Tutorials](#)

[GET A DEMO](#)

[Free Trial](#)

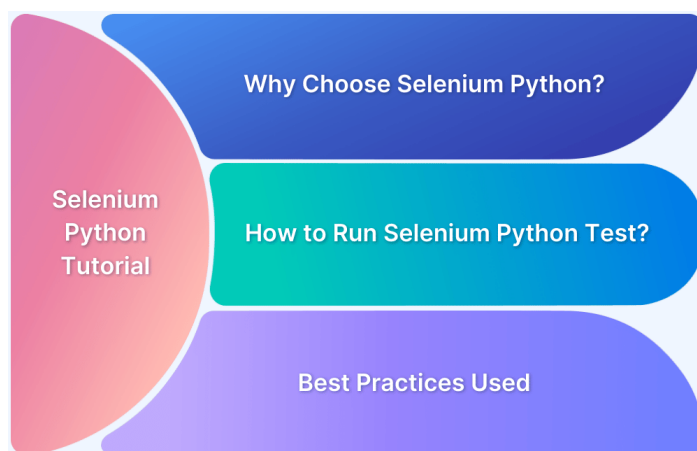
Test Selenium Python on Real Devices and Browsers for accurate test results under real user conditions

September 3, 2024

21 min read

Get Started free

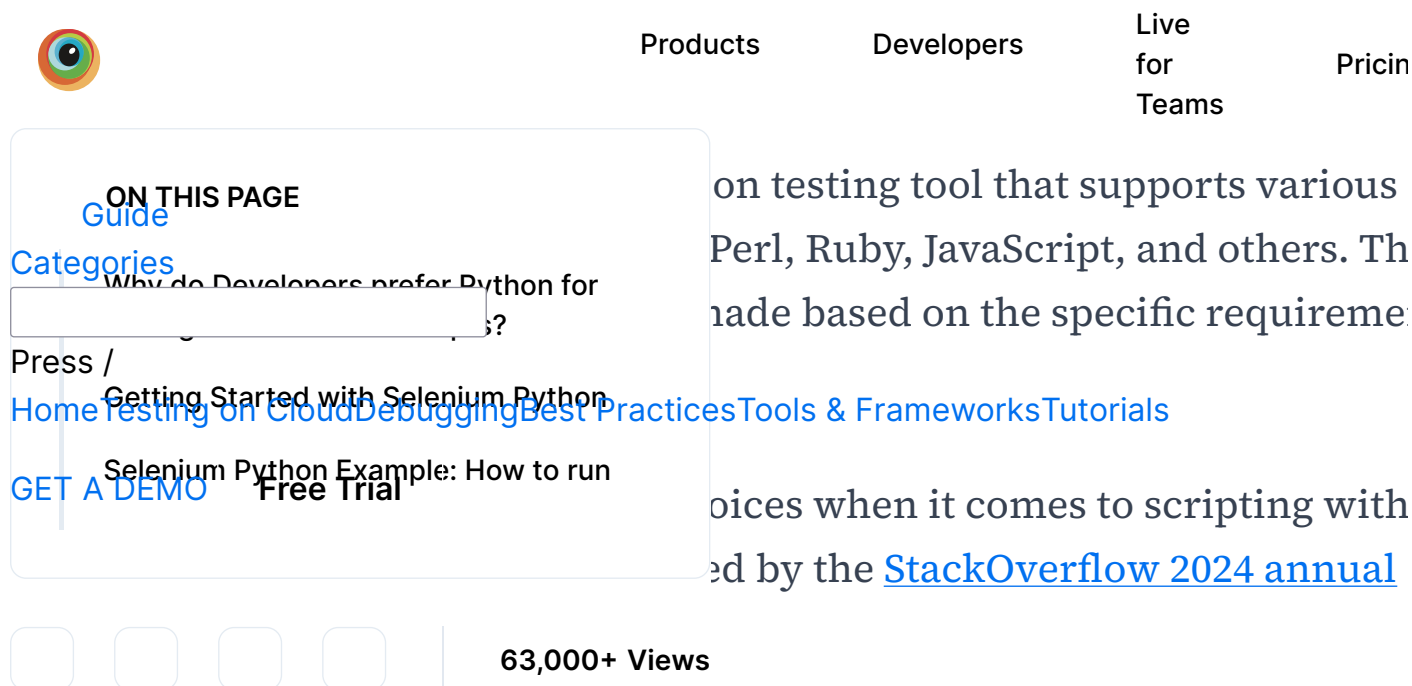
[Contact Sales](#)



[Home](#) > [Guide](#) > [Selenium Python Tutorial \(with Example\)](#)

## Selenium Python Tutorial (with Example

New features are regularly added to web applications to boost user engagement. To ensure these updates work as intended and that the user



## Why do Developers prefer Python for writing Selenium Test Scripts?

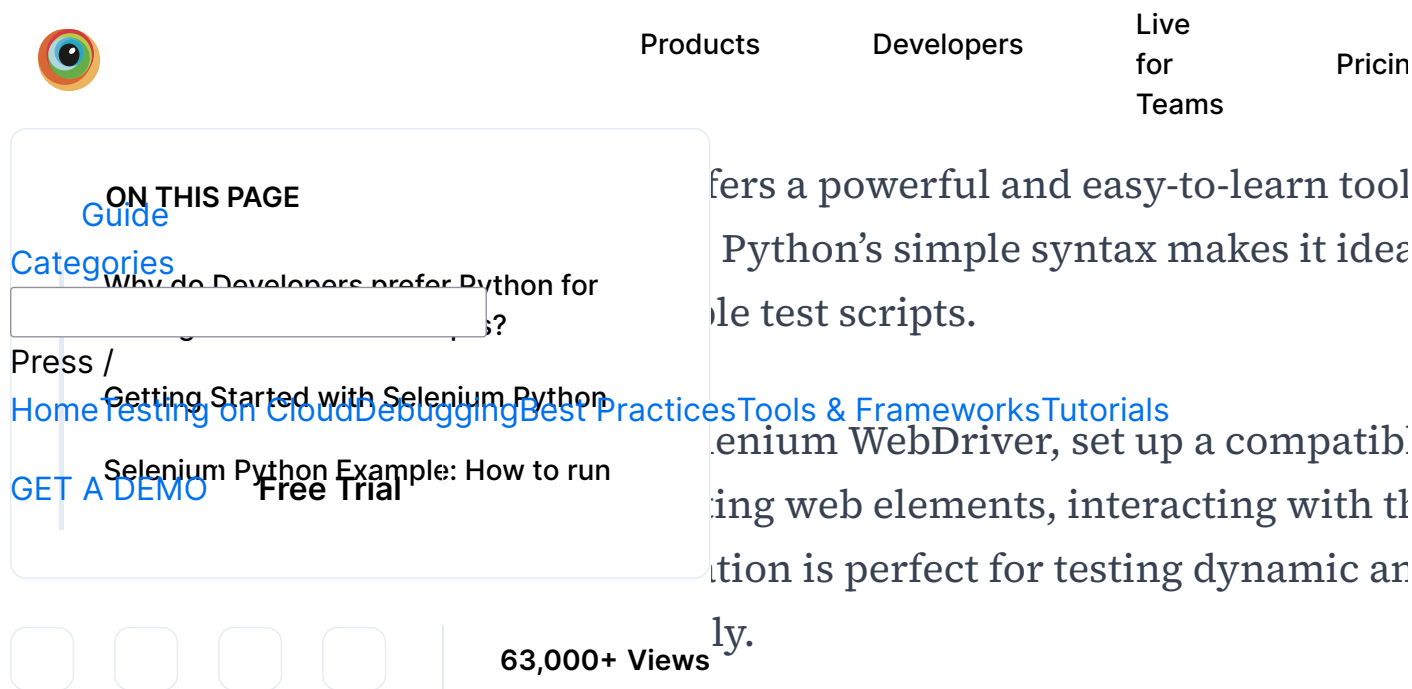
Developers prefer Python for writing Selenium test scripts because of its simplicity, readability, and ease of use. Python's clear and concise syntax allows for faster script development and easier maintenance, which is crucial in testing scenarios.

Additionally, Python has a rich set of libraries and frameworks that complement Selenium, making it easier to handle complex tasks such as data manipulation, reporting, and integration with other tools.

Python's extensive community support and documentation also provide valuable resources for troubleshooting and improving test scripts. These factors make Python a popular choice for Selenium automation.

## Getting Started with Selenium Python

Getting started with Selenium using Python involves setting up an



The screenshot shows the top section of a web page. On the left is the Selenium logo. To its right are navigation links: "Products", "Developers", "Live for Teams", and "Pricing". Below these is a horizontal menu with "ON THIS PAGE" and "Guide". A "Categories" dropdown menu is open, showing links to "Why do Developers prefer Python for...", "Press /", "Getting Started with Selenium Python", "Home Testing on Cloud", "Debugging", "Best Practices", "Tools & Frameworks", and "Tutorials". Below the dropdown is a "GET A DEMO" button and a "Free Trial" link. At the bottom of the screenshot, there are four social media icons and a view count of "63,000+ Views".

# Selenium Python Example: How to run your first Test?

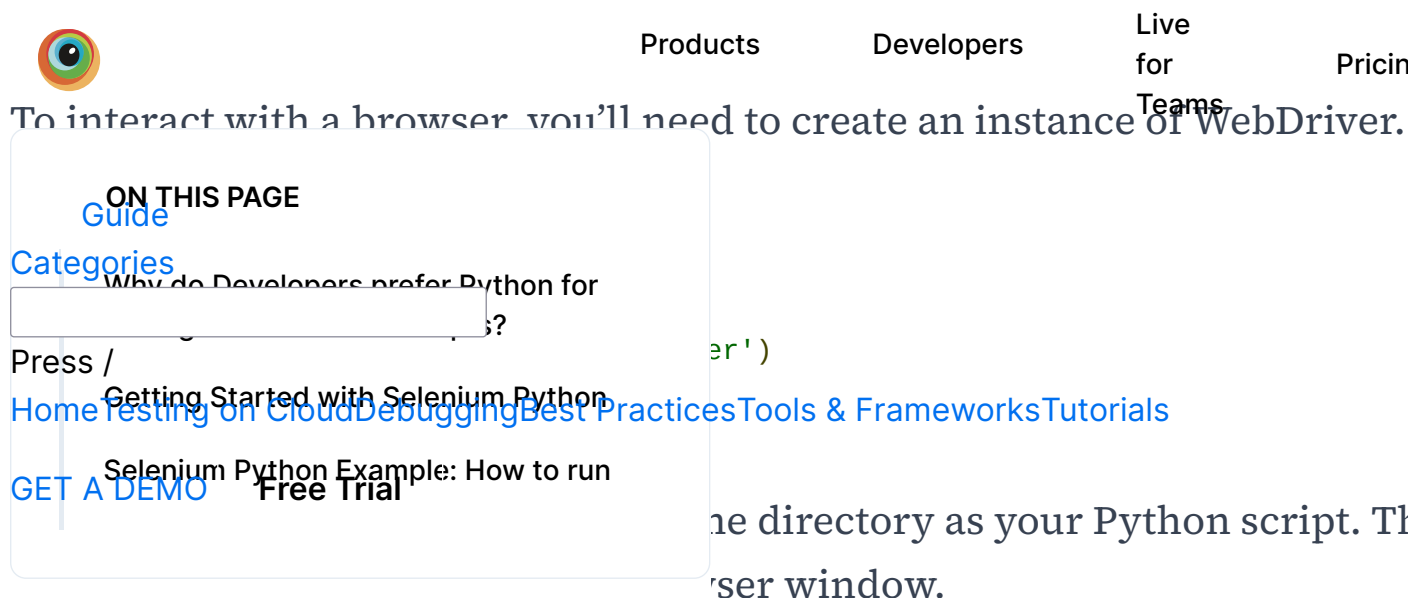
To run Selenium Python Tests here are the steps to follow:

## Step 1. Import the Necessary Classes

First, you'll need to import the WebDriver and Keys classes from Selenium. These classes help you interact with a web browser and emulate keyboard actions.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
```

- **webdriver:** Allows you to control the browser.
- **Keys:** Lets you simulate keyboard key presses.



The screenshot shows the top of the BrowserStack website. At the top left is the BrowserStack logo. To its right are navigation links: "Products", "Developers", "Live for Teams", and "Pricing". Below these is a main heading: "To interact with a browser, you'll need to create an instance of WebDriver." On the left side, there is a sidebar with a "Guide" link, a "Categories" link, and a list of articles including "Why do Developers prefer Python for...", "Getting Started with Selenium Python", and "Selenium Python Example: How to run...". Below the sidebar is a "GET A DEMO" button and a "Free Trial" button. To the right of the sidebar, there is a list of links: "Home", "Testing on Cloud", "Debugging", "Best Practices", "Tools & Frameworks", and "Tutorials".

 63,000+ Views  
Step 3. Load a Website

Use the `.get()` method to navigate to a website. This method waits for the page to load completely:

```
driver.get("https://www.python.org")
```

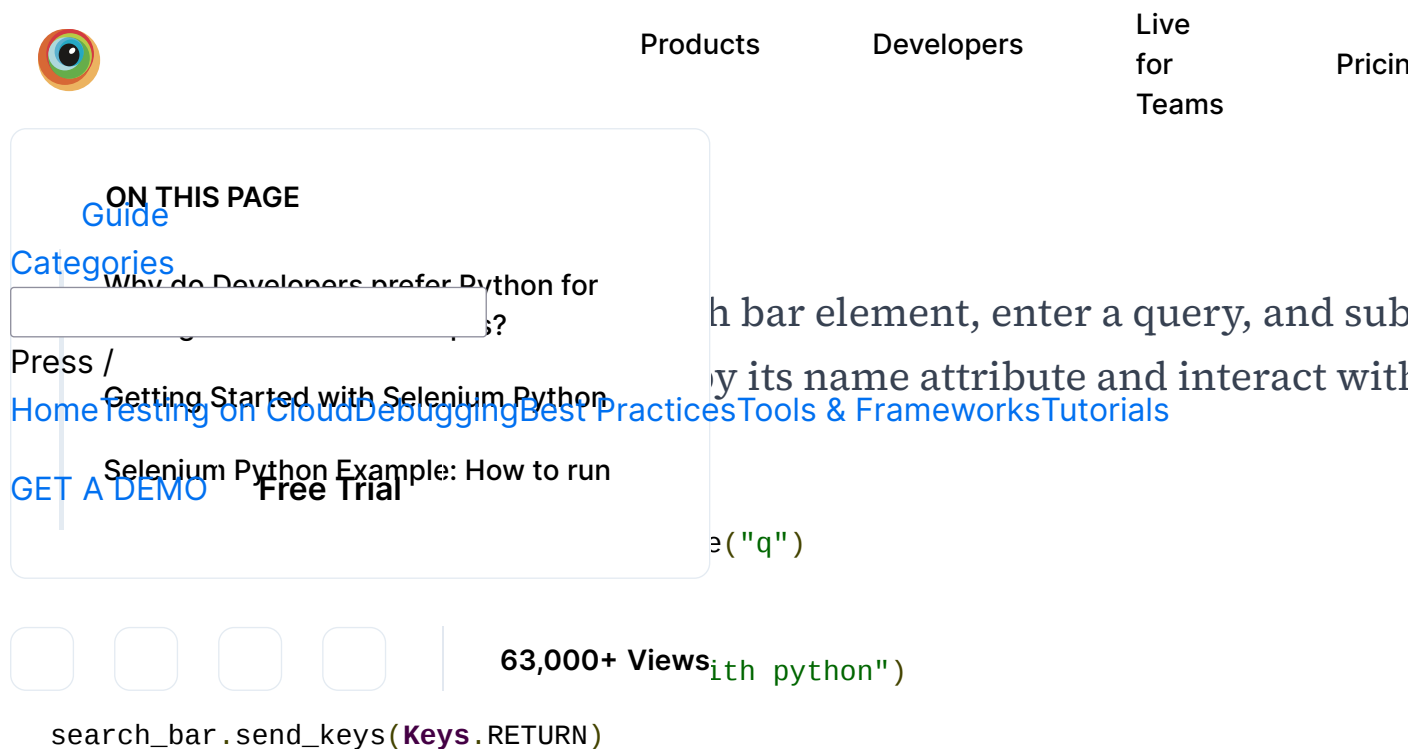
This will open Python's official website in the browser.

#### Step 4. Check the Page Title

Once the page is loaded, you can retrieve and print the page title to verify you're on the right page:

```
print(driver.title)
```

You should see:



The screenshot shows the top navigation bar of the BrowserStack website. It includes the BrowserStack logo on the left, followed by links for 'Products', 'Developers', 'Live for Teams', and 'Pricing'. Below the navigation bar, there is a sidebar menu on the left with the heading 'ON THIS PAGE'. The menu items include 'Guide', 'Categories', 'Why do Developers prefer Python for Selenium?', 'Press /', 'Getting Started with Selenium Python', 'Home Testing on Cloud', 'Debugging', 'Best Practices', 'Tools & Frameworks', and 'Tutorials'. There are also buttons for 'GET A DEMO' and 'Free Trial'. The main content area on the right shows a search bar with the text 'h bar element, enter a query, and sub' and a snippet of code: `search_bar.send_keys(Keys.RETURN)`.

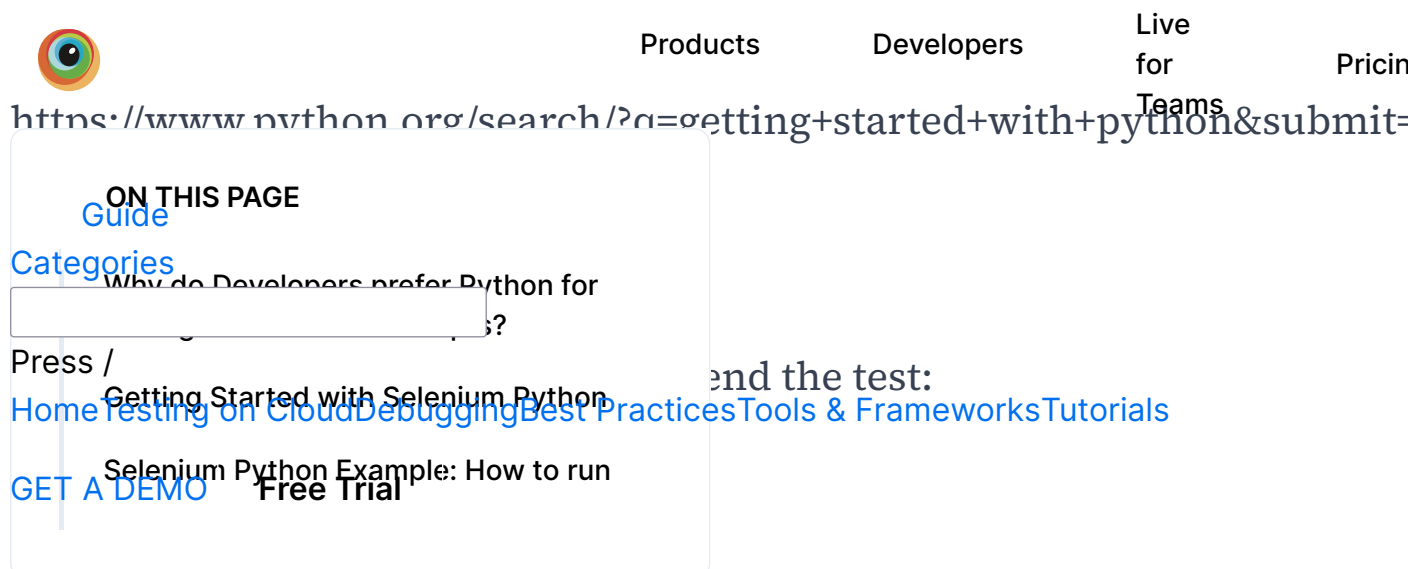
As an explanation :

- `find_element_by_name("q")`: Finds the search bar element.
- `clear()`: Clears any existing text.
- `send_keys("getting started with python")`: Types the query into the search bar.
- `send_keys(Keys.RETURN)`: Simulates pressing the Return (Enter) key.

## Step 6. Verify the Resulting URL

After submitting the search query, you can check the updated URL to confirm the search results page:

```
print(driver.current_url)
```



Summary : 63,000+ Views

Here is the complete script for your first Selenium test in Python. Save this code in a file named selenium\_test.py and run it using python selenium\_test.py:

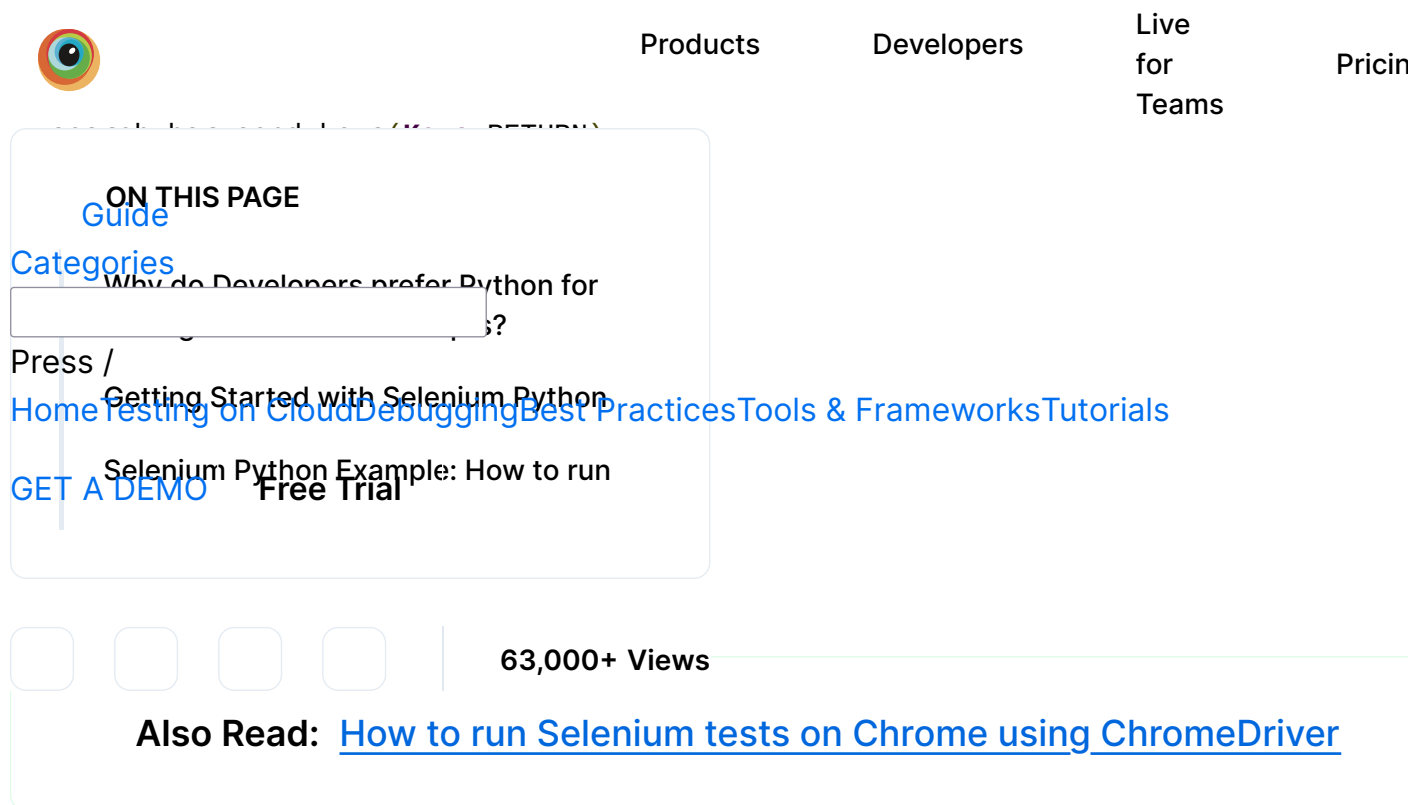
```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

# Create a new instance of the Chrome driver
driver = webdriver.Chrome('./chromedriver')

# Open the Python website
driver.get("https://www.python.org")

# Print the page title
print(driver.title)

# Find the search bar using its name attribute
search_bar = driver.find_element_by_name("q")
```



The screenshot shows the top navigation bar of the Selenium Python Tutorial page. It includes the Selenium logo, a search bar, and navigation links for Products, Developers, Live for Teams, and Pricing. A sidebar on the left contains a 'ON THIS PAGE' section with links to Guide, Categories, and various tutorial topics like 'Why do Developers prefer Python for Selenium?', 'Getting Started with Selenium Python', and 'Selenium Python Example: How to run Selenium tests on Chrome using ChromeDriver'. Below the sidebar, there are four social media icons and a view count of 63,000+. An 'Also Read' section recommends the article 'How to run Selenium tests on Chrome using ChromeDriver'.

## Interacting with Common Elements in Selenium

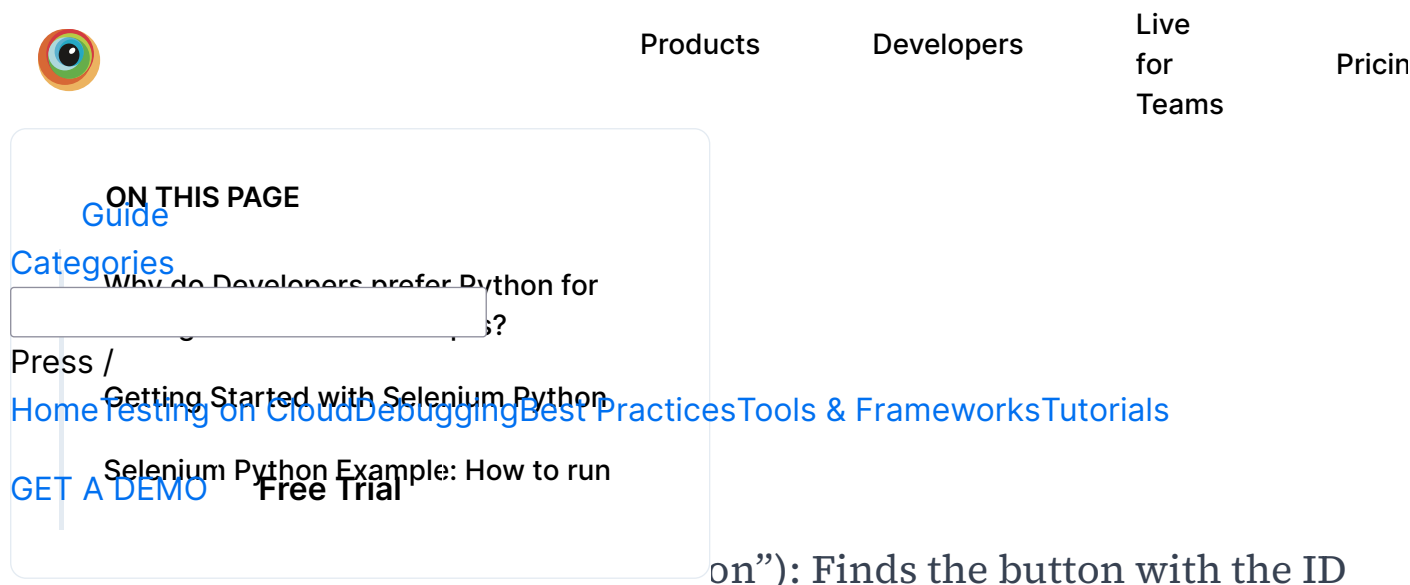
Selenium allows you to perform a variety of actions on web elements. You already touched upon entering input, here's how to interact with buttons, dropdowns:

Assuming you want to click a button with the ID “submit-button” after entering the input in the search bar :

```
# Locate the button by its ID attribute
button = driver.find_element_by_id("submit-button")

# Click the button
button.click()
```

If you need to click a link by its text:



The screenshot shows the top section of the BrowserStack website. On the left is the BrowserStack logo. To its right are navigation links: "Products", "Developers", "Live for Teams", and "Pricing". Below these is a large white box containing a "ON THIS PAGE" section. This section lists several links: "Guide", "Categories", "Why do Developers prefer Python for Selenium?", "Press / Getting Started with Selenium Python", "Home Testing on Cloud Debugging Best Practices Tools & Frameworks Tutorials", and "Selenium Python Example: How to run Selenium on BrowserStack". At the bottom of this box are two buttons: "GET A DEMO" and "Free Trial".



- `find_element_by_link_text("Click Here")`: Finds a link with the text "Click Here".
- `click()`: Simulates a mouse click on the element.

**Read More:** [Selenium Testing with Python: Automated Testing of a Signup Form](#)

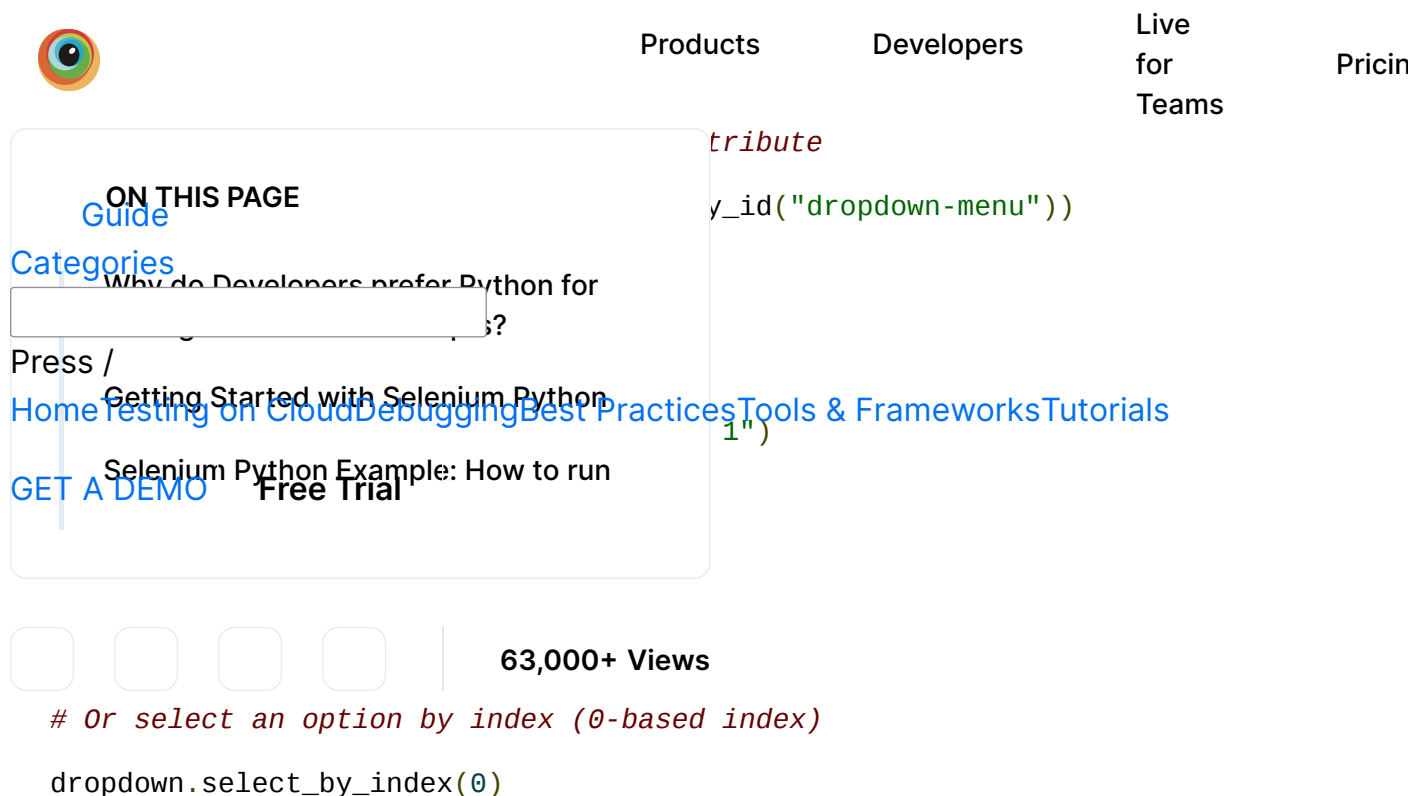
Though dropdowns are not present on this site, they are quite common for application testing

For dropdown menus, Selenium provides the `Select` class to handle options within `<select>` elements.

**Example: Selecting an Option from a Dropdown**

Assuming you have a dropdown menu with the ID "dropdown-menu":





The screenshot shows a web page layout. At the top, there is a navigation bar with links: "Products", "Developers", "Live for Teams", and "Pricing". Below this, a sidebar on the left contains a search bar, a "Categories" section with a list of items, and a "GET A DEMO" button. The main content area on the right features a large heading "ON THIS PAGE" followed by a list of links: "Guide", "Categories", "Why do Developers prefer Python for...", "Press /", "Getting Started with Selenium Python", "Home Testing on CloudDebuggingBest PracticesTools & FrameworksTutorials", and "Selenium Python Example: How to run". Below the sidebar, there are four empty square boxes and a text "63,000+ Views". At the bottom, there is a code snippet: 

```
# Or select an option by index (0-based index)
dropdown.select_by_index(0)
```

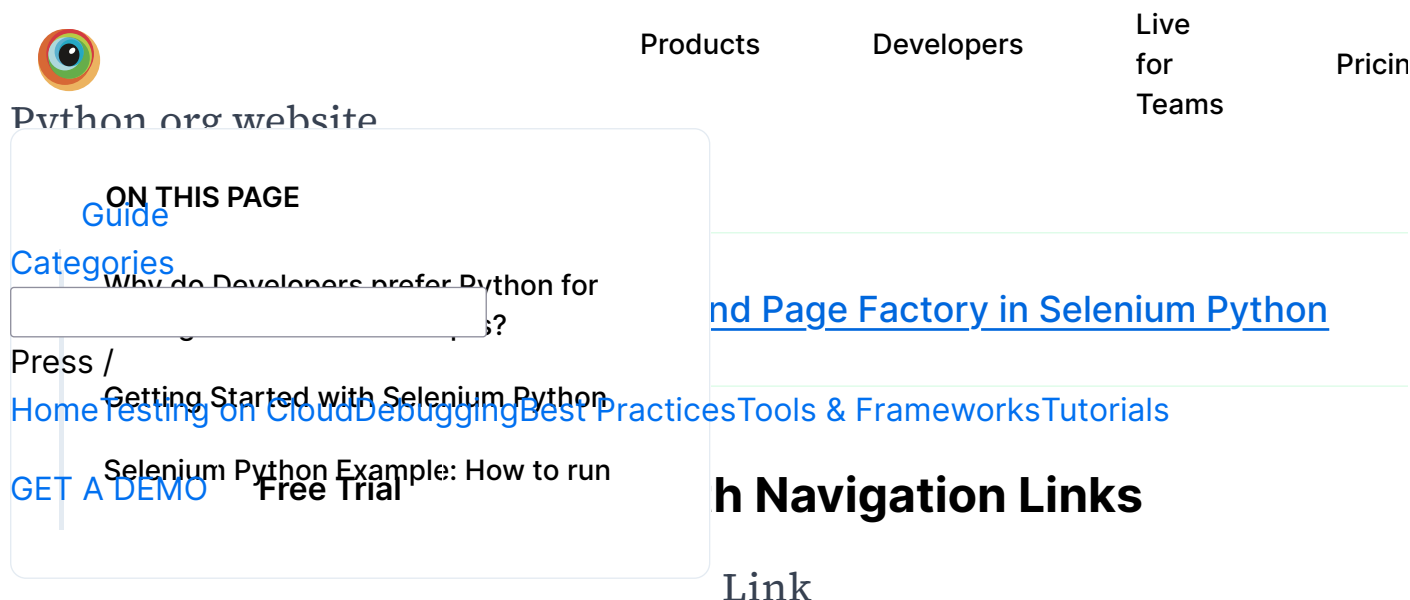
## Explanation:

- `Select(driver.find_element_by_id("dropdown-menu"))`: Creates a `Select` object for the dropdown menu.
- `select_by_visible_text("Option 1")`: Selects an option by its visible text.
- `select_by_value("option1")`: Selects an option by its value attribute.
- `select_by_index(0)`: Selects an option by its index in the dropdown.

## Navigate through HTML DOM Elements

The HTML Document Object Model (DOM) represents the structure of a web page as a tree of objects. Selenium allows you to interact with these elements using various locator strategies.

In our first test script, we have already used some of the methods used to



The image shows the top section of the Python.org website. At the top left is the Python logo. To its right are navigation links: "Products", "Developers", "Live for Teams", and "Pricing". Below the logo, the text "Python.org website" is visible. A large, light blue box highlights a section titled "ON THIS PAGE" which lists several links: "Guide", "Categories", "Why do Developers prefer Python for...", "Press /", "Getting Started with Selenium Python", "Home Testing on Cloud Debugging Best Practices Tools & Frameworks Tutorials", and "Selenium Python Example: How to run". Below this box, there are four small square icons and the text "63,000+ Views". To the right of the box, the text "Link" is visible.

To click the “Downloads” link, you can use the `.find_element_by_link_text()` method, but here’s how to use other locators to achieve the same, example using `find_element_by_xpath()`:

```
from selenium import webdriver

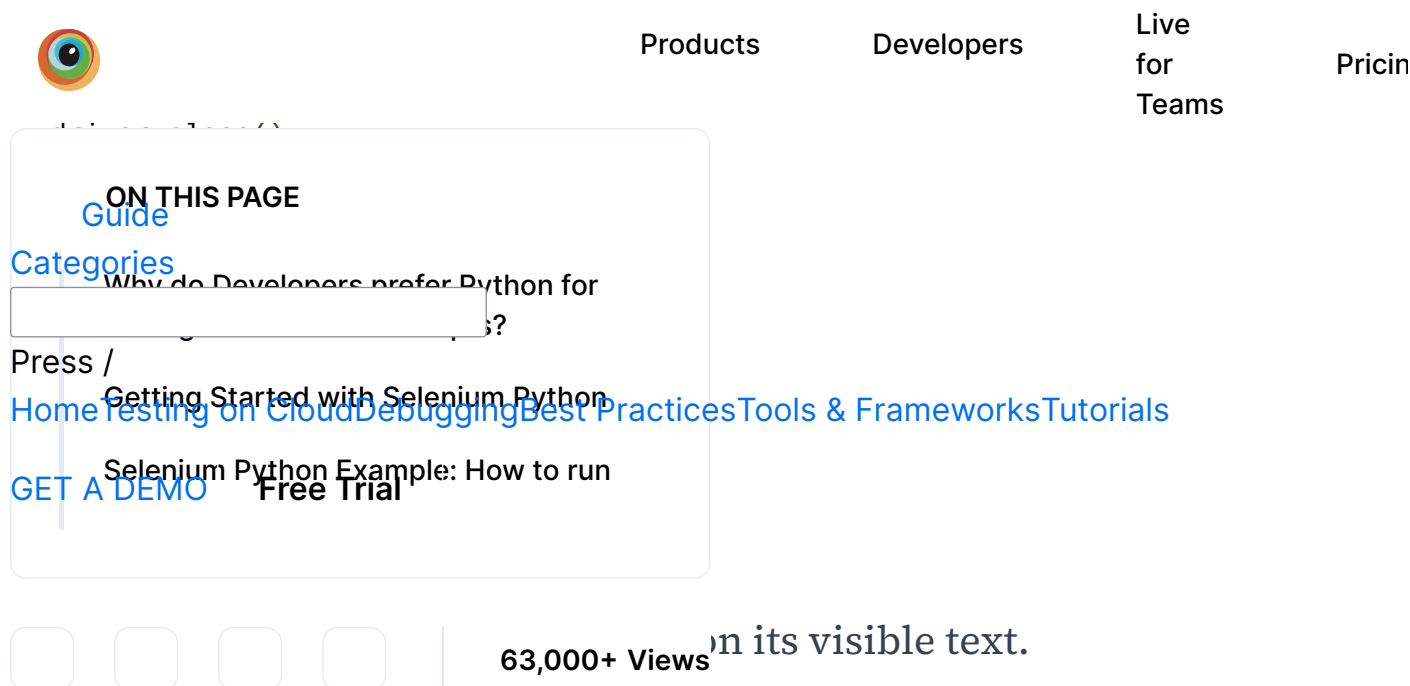
# Set up the WebDriver
driver = webdriver.Chrome('./chromedriver')

# Open the Python website
driver.get("https://www.python.org/")

# Locate the "Downloads" link using XPath
downloads_link = driver.find_element_by_xpath("//a[text()='Downloads']")

# Click the "Downloads" link
downloads_link.click()

# Optionally, print the current URL to confirm navigation
print(driver.current_url)
```



## Step 2. Access and Interact with Header Sections

Example: Accessing the Main Header

To access the main header text, you can use different locators to find the header element.

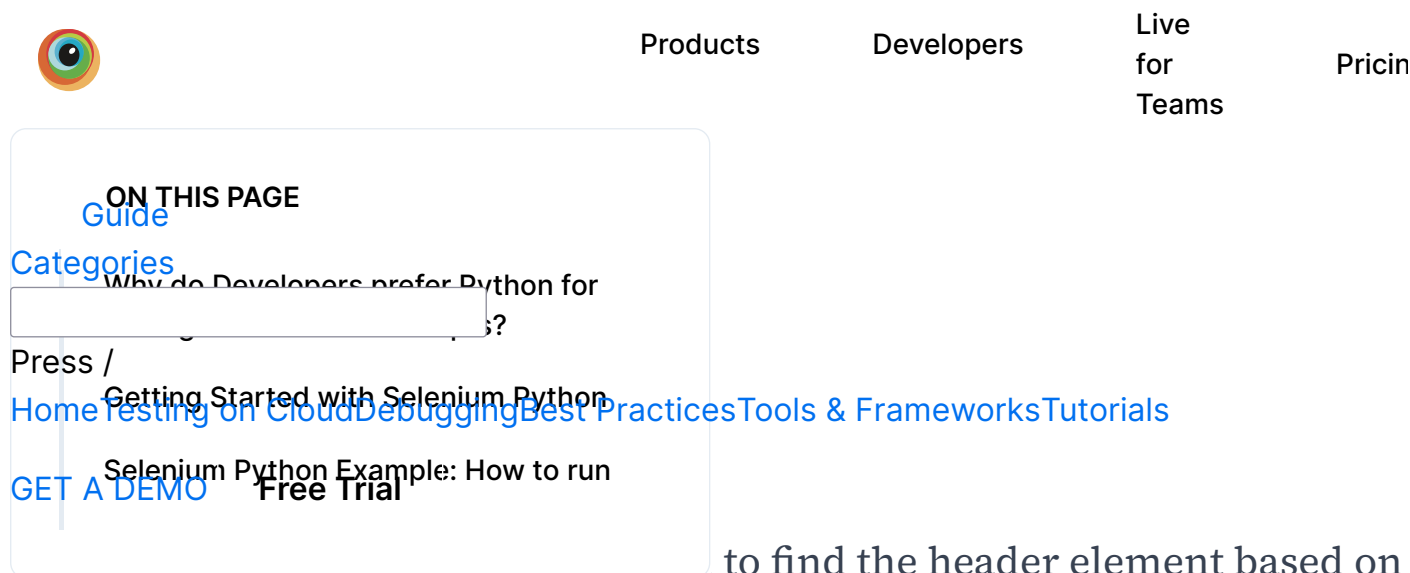
Using `find_element_by_class_name`:

```
from selenium import webdriver

# Set up the WebDriver
driver = webdriver.Chrome('./chromedriver')

# Open the Python website
driver.get("https://www.python.org/")

# Locate the header element using its class name
header = driver.find_element_by_class_name("introduction")
```



63,000+ Views

## Step 3. Interact with Forms and Input Fields

Example: Filling Out and Submitting the Search Form

To interact with the search form, you can use the `.find_element_by_name()` method to locate the input field.

Using `find_element_by_name`:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

# Set up the WebDriver
driver = webdriver.Chrome('./chromedriver')

# Open the Python website
driver.get("https://www.python.org/")

# Locate the search bar using its name attribute
search_bar = driver.find_element_by_name("q")
```



## Explanation:

- Name Attribute: `find_element_by_name("q")` locates the search input field by its name attribute.

# Navigate through Windows and Frames

When working with multiple browser windows or tabs, or dealing with iframes (frames), you may need to switch contexts to interact with different elements.

## Step 1. Handling Multiple Browser Windows or Tabs

### Example: Switching Between Windows

To handle multiple browser windows or tabs:

```
from selenium import webdriver
```



Products

Developers

Live  
for  
Teams

Pricing

## ON THIS PAGE

[Guide](#)

[Categories](#)

Why do Developers prefer Python for Selenium?

Press /

Getting Started with Selenium Python

[Home](#) [Testing on Cloud](#) [Debugging](#) [Best Practices](#) [Tools & Frameworks](#) [Tutorials](#)

Selenium Python Example: How to run

[GET A DEMO](#) [Free Trial](#)

`js://www.google.com', '_blank');")`

63,000+ Views

*# Switch to the new tab*

```
driver.switch_to.window(driver.window_handles[1])
```

*# Perform actions in the new tab (e.g., search for 'Selenium')*

```
search_bar = driver.find_element_by_name("q")
```

```
search_bar.clear()
```

```
search_bar.send_keys("Selenium")
```

```
search_bar.send_keys(Keys.RETURN)
```

*# Switch back to the original tab*

```
driver.switch_to.window(driver.window_handles[0])
```

*# Close the browser*

```
driver.quit()
```

## Explanation:

- window\_handles:** Retrieves a list of window handles. Switch to a specific window using `switch_to.window()`.



Products

Developers

Live  
for  
Teams

Pricing

## Step 2: Switching Between Frames

ON THIS PAGE

[Guide](#)[Categories](#)[Why do Developers prefer Python for Selenium?](#)

Press /

[Getting Started with Selenium Python](#)[Home](#) [Testing on Cloud](#) [Debugging](#) [Best Practices](#) [Tools & Frameworks](#) [Tutorials](#)[GET A DEMO](#) [Free Trial](#) [Selenium Python Example: How to run Selenium](#)

its within an iframe:

Short Keys

63,000+ Views

# Set up the WebDriver

`driver = webdriver.Chrome('./chromedriver')`

# Open the Python website

`driver.get("https://www.python.org/")`

# Example site with iframe (replace with an actual URL that contains iframes)

`driver.get("https://www.w3schools.com/html/html_iframe.asp")`

# Switch to the iframe using its name or ID

`driver.switch_to.frame("iframeResult")`

# Perform actions within the iframe

`print(driver.find_element_by_tag_name("h1").text)`

# Switch back to the default content

`driver.switch_to.default_content()`

# Close the browser



Products

Developers

Live  
for  
Teams

Pricing

## ON THIS PAGE

[Guide](#)[Categories](#)[Why do Developers prefer Python for Selenium?](#)[Press /](#)[Getting Started with Selenium Python](#)[Home](#) [Testing on Cloud](#) [Debugging](#) [Best Practices](#) [Tools & Frameworks](#) [Tutorials](#)[Selenium Python Example: How to run](#)[GET A DEMO](#) [Free Trial](#)

specific iframe.

comes back to the main page.

 | **63,000+ Views**

elements are present before interacting with them.

## Step 1. Implicit Waits

### Example: Using Implicit Waits

```
from selenium import webdriver


# Set up the WebDriver
driver = webdriver.Chrome('./chromedriver')

# Set implicit wait
driver.implicitly_wait(10) # seconds


# Open the Python website
driver.get("https://www.python.org/")

# Locate an element with implicit wait
search_bar = driver.find_element_by_name("q")
search_bar.send_keys("Python")
```





[Products](#)
[Developers](#)
[Live for Teams](#)
[Pricing](#)



driver.quit()

ON THIS PAGE


[Guide](#)
[Categories](#)

Why do Developers prefer Python for Selenium?

Press /

[Getting Started with Selenium Python](#)
[Home Testing on Cloud](#)
[Debugging Best Practices](#)
[Tools & Frameworks](#)
[Tutorials](#)

[GET A DEMO](#)
[Free Trial](#)



63,000+ Views

it time for finding elements. If an element is not found, the WebDriver will wait up to the specified time.

```

from selenium import webdriver

from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Set up the WebDriver
driver = webdriver.Chrome('./chromedriver')

# Open the Python website
driver.get("https://www.python.org/")

# Define WebDriverWait with a maximum wait time of 10 seconds
wait = WebDriverWait(driver, 10)

# Wait for the search bar to be present in the DOM
search_bar = wait.until(EC.presence_of_element_located((By.NAME, "q")))

# Perform actions on the search bar
search_bar.send_keys("Python")
    
```



Products

Developers

Live  
for  
Teams

Pricing

## ON THIS PAGE

[Guide](#)[Categories](#)[Why do Developers prefer Python for Selenium?](#)[Press /](#)[Getting Started with Selenium Python](#)[Home](#) [Testing on Cloud](#) [Debugging](#) [Best Practices](#) [Tools & Frameworks](#) [Tutorials](#)[GET A DEMO](#) [Free Trial](#) [Selenium Python Example: How to run](#)

is an instance of WebDriverWait, of 10 seconds.

itLocated(By.NAME, "q")): Pauses if the element is found by its name attribute. If the element is not found within the specified timeout, a TimeoutException will be raised.



63,000+ Views

**Must Read:** [How to use Wait commands in Selenium WebDriver](#)

## Assertions and Validations

To ensure that the application behaves as expected, you can use assertions and validations.

### Verifying Expected Conditions Using Assertions

#### Example: Verifying Page Title and Search Results

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Set up the WebDriver
```

## Guide

## Categories

rch bar to be present

## Getting Started with Selenium Python

Getting Started with Selenium Python

Selenium Python Example: How to run

```
element_located((By.NAME, "q"))
```

GET A DEMO

## Free Trial

```
search_bar.send_keys(Keys.RETURN)
```

```
assert "Python" in driver.title
```

```
results = wait.until(EC.presence_of_element_located((By.CSS_SELECTOR, "ul.list-re
```

```
assert "Python" in results.text
```

```
print(driver.title)
```

```
print(results.text)
```

```
driver.quit()
```

- **Assertions:** Used to check if the conditions are met. For example, check the title or text of elements matches expected values.



Products

Developers

Live  
for  
Teams

Pricing

## ON THIS PAGE

[Guide](#)[Categories](#)[Why do Developers prefer Python for Selenium?](#)[Press /](#)[Getting Started with Selenium Python](#)[Home](#) [Testing on Cloud](#) [Debugging](#) [Best Practices](#) [Tools & Frameworks](#) [Tutorials](#)[Selenium Python Example: How to run](#)[GET A DEMO](#)[Free Trial](#)

## Pop-ups

JavaScript alerts, confirmation dialogs, or prompts are used to display messages to the user. This article provides ways to handle these pop-ups.



63,000+ Views

JavaScript alerts are simple pop-up messages that require user interaction to dismiss. Selenium allows you to interact with these alerts using the `switch_to.alert()` method.

```

from selenium import webdriver

from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

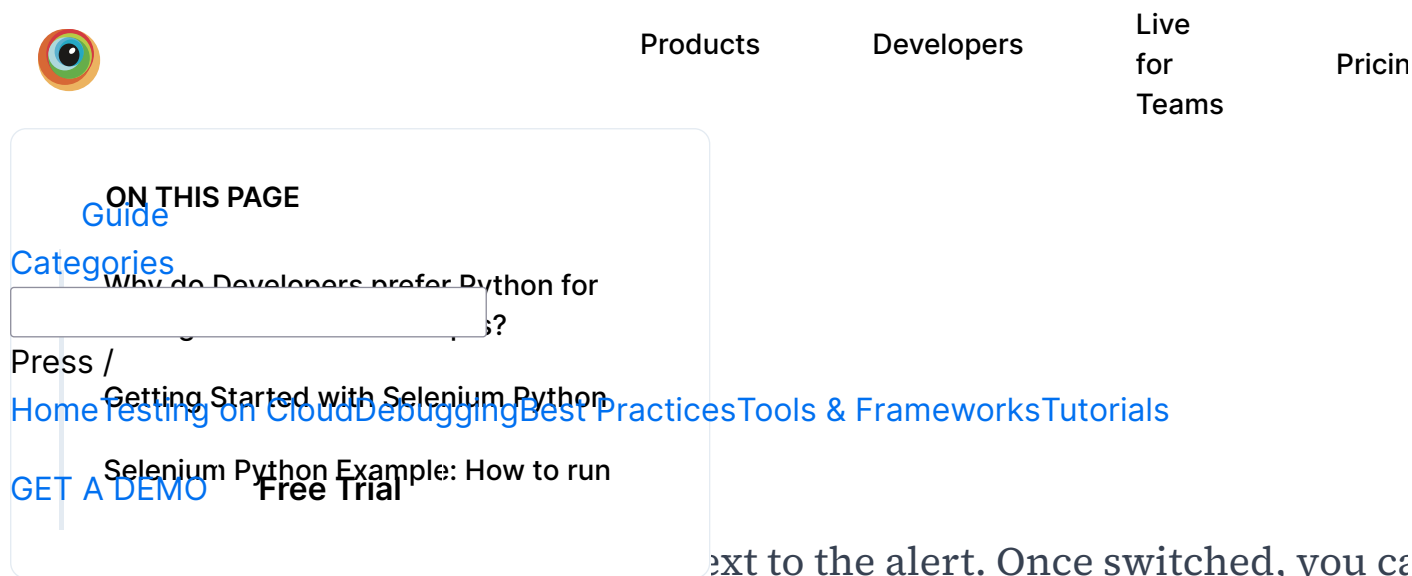
# Set up the WebDriver
driver = webdriver.Chrome('./chromedriver')

# Open a website that triggers an alert (example URL)
driver.get("https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/A

# Click a button that triggers an alert
trigger_alert_button = driver.find_element(By.ID, "alertButton") # Adjust locator
trigger_alert_button.click()

# Switch to the alert and accept it
alert = driver.switch_to.alert

```



- `alert.accept()`: Accepts the alert, which is equivalent to clicking “OK” on alert.

## Other Alert Actions:

- `alert.dismiss()`: Clicks “Cancel” on a confirmation dialog.
- `alert.send_keys(“text”)`: Sends text to a prompt dialog (if applicable).


# Cleanup and Teardown

Properly closing the browser session is crucial for releasing resources and ensuring that your automation script runs cleanly.

## Properly Closing the Browser Session

### Example: Closing the Browser

```
from selenium import webdriver
```



[Products](#)
[Developers](#)

[Live for Teams](#)
[Pricing](#)

ON THIS PAGE

[Guide](#)
[Categories](#)

[Why do Developers prefer Python for Selenium?](#)

[Press /](#)
[Getting Started with Selenium Python](#)
[Home Testing on Cloud](#)
[Debugging](#)
[Best Practices](#)
[Tools & Frameworks](#)
[Tutorials](#)

[Selenium Python Example: How to run](#)

[GET A DEMO](#)
[Free Trial](#)

63,000+ Views

```
driver.quit()
```

## Explanation:

- `driver.quit()`: Closes all browser windows and ends the WebDriver session. This is the preferred method for cleanup as it ensures the browser process is terminated and resources are freed.

## Alternative Methods:

- `driver.close()`: Closes the current window. If it's the only window open, it will end the session. Use `driver.quit()` for complete cleanup.

Read More: [How to Create and Use Action Class in Selenium Python](#)

# Testing Framework Integration



Products

Developers

Live  
for  
Teams

Pricing

methods

## ON THIS PAGE

[Guide](#)

[Categories](#)

[Why do Developers prefer Python for Selenium?](#)

[Press /](#)

[Getting Started with Selenium Python](#)

[Home Testing on Cloud Debugging Best Practices Tools & Frameworks Tutorials](#)

[Selenium Python Example: How to run](#)

[GET A DEMO](#)

[Free Trial](#)

## Framework

A framework that provides a structured approach to writing tests, including test case management, test execution, and reporting. Using Selenium with unittest allows for writing test cases using unittest's own methods, and detailed test reports to maintain automated tests.



63,000+ Views

## Example: Basic Test with unittest

```
import unittest


from selenium import webdriver

from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

class PythonOrgSearchTest(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        cls.driver = webdriver.Chrome('./chromedriver')
        cls.driver.get("https://www.python.org/")

    def test_search_python(self):
        search_bar = self.driver.find_element(By.NAME, "q")
        search_bar.send_keys("Python")
        search_bar.send_keys(Keys.RETURN)
        self.assertIn("Python", self.driver.title)
```



ProductsDevelopersLive for TeamsPricing

ON THIS PAGE

Guide

Categories

Why do Developers prefer Python for Selenium?

Press /

Getting Started with Selenium Python

HomeTesting on CloudDebuggingBest PracticesTools & FrameworksTutorials

GET A DEMO

Free Trial

Selenium Python Example: How to run

63,000+ Views

se class. Each method within the clas

- `setUpClass()`: Initializes resources needed for the tests. Runs once before test methods are executed.
- `tearDownClass()`: Cleans up resources. Runs once after all test methods completed.
- `unittest.main()`: Runs the tests and provides output in the console.

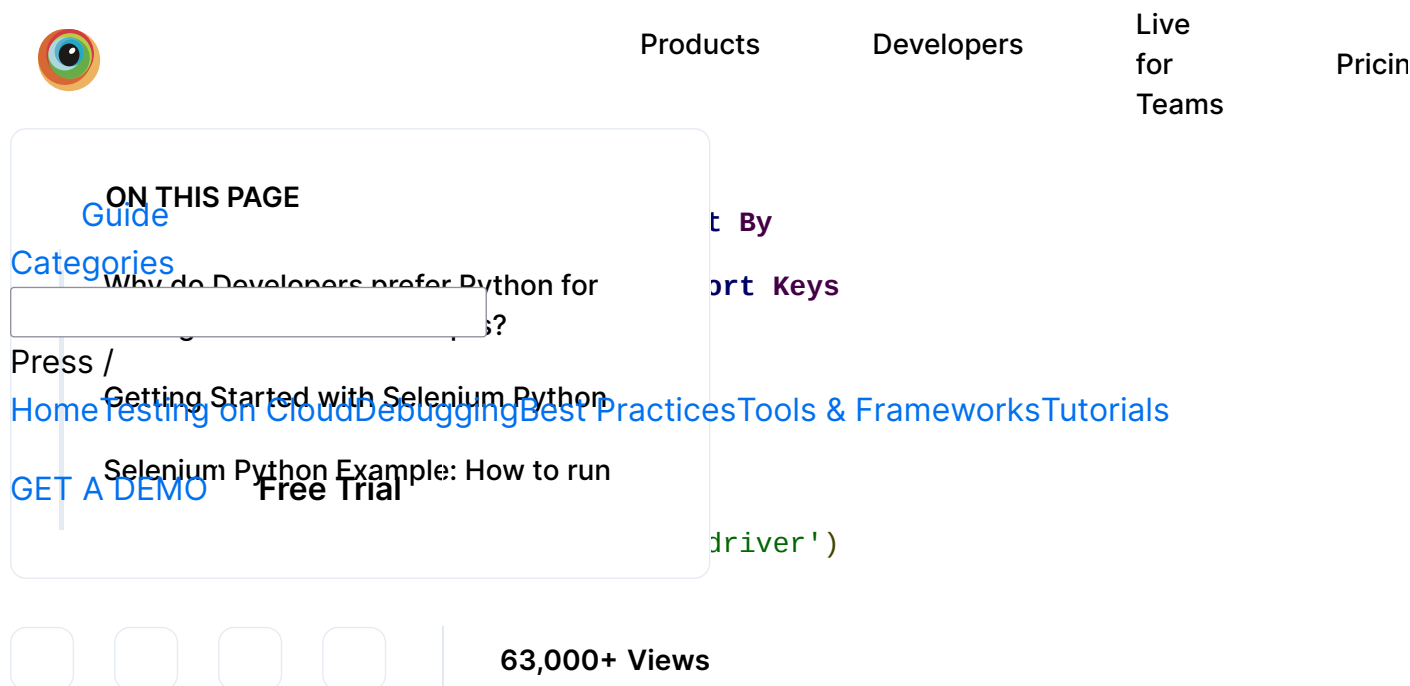
Read More: [Top 8 Python Testing Frameworks in 2024](#)

## 2. Integrate with pytest Framework

pytest is a powerful and flexible Python testing framework that simplifies writing tests with its rich feature set, including fixtures, parameterized tests and detailed assertions. Integrating Selenium with pytest enhances test organization, facilitates advanced setup/teardown functionality, and generates comprehensive test reports, improving test reliability and clarity.

Example: Basic Test with pytest





The screenshot shows the top navigation bar of the BrowserStack website with links for Products, Developers, Live for Teams, and Pricing. The main content area features a sidebar with 'ON THIS PAGE' links: Guide, Categories, GET A DEMO, and Free Trial. The article title is 'Selenium Python Example: How to run Python tests on a real device'. The author is 'By' and the date is '10/10/2023'. The article has '63,000+ Views'. The code snippet shown is a Python test function using Selenium WebDriver to search for 'Python' on the Python.org website and assert that the page title contains 'Python'.

ON THIS PAGE

Guide

Categories

GET A DEMO

Free Trial

Selenium Python Example: How to run Python tests on a real device

By

10/10/2023


63,000+ Views

```
def test_search_python(driver):  
    driver.get("https://www.python.org/")  
    search_bar = driver.find_element(By.NAME, "q")  
    search_bar.send_keys("Python")  
    search_bar.send_keys(Keys.RETURN)  
    assert "Python" in driver.title
```

## Explanation:

- `pytest.fixture()`: Defines a fixture that sets up and tears down resources. `scope="module"` ensures the fixture is run once per module.
- `yield`: Provides the driver instance to the test function and performs cleanup after the test completes.
- `assert`: Checks that the condition is met. pytest will report the assertion failure if the

## Run Selenium Python Tests on Real Devices



[Products](#)
[Developers](#)
[Live for Teams](#)
[Pricing](#)

- [Selenium Python Tutorial \(with Example\)](#)

ON THIS PAGE

[Guide](#)

[Categories](#)

Why do Developers prefer Python for Selenium?

Press /

Getting Started with Selenium Python

Selenium Python Example: How to run Selenium Python

GET A DEMO Free Trial





[y in Selenium Python](#)

[s in Selenium](#)

[p\(\) in Selenium](#)

[Selenium Python](#)

[Selenium Python?](#)

63,000+ Views

[How to perform web scraping using Selenium and Python](#)

- [How to Create and Use Action Class in Selenium Python](#)
- [How to download a file using Selenium and Python](#)
- [How to Press Enter without Element in Selenium Python?](#)
- [UI Automation using Python and Selenium: Tutorial](#)
- [Get Current URL in Selenium using Python: Tutorial](#)

## Best Practices using Selenium WebDriver with Python

Here are five best practices for using Selenium WebDriver with Python:

1. Use Explicit Waits: Prefer explicit waits over implicit waits to handle dynamic content. Explicit waits ensure that your script interacts with elements only when they are ready, reducing the chances of encountering timing issues.
2. Organize Tests with Frameworks: Integrate Selenium tests with testing



Products

Developers

Live  
for  
Teams

Pricing

#### ON THIS PAGE

[Guide](#)

[Categories](#)

[Why do Developers prefer Python for Selenium?](#)

[Press /](#)

[Getting Started with Selenium Python](#)

[Home](#) [Testing on Cloud](#) [Debugging](#) [Best Practices](#) [Tools & Frameworks](#) [Tutorials](#)

[Selenium Python Example: How to run](#)

[GET A DEMO](#)

[Free Trial](#)

plement the Page Object Model to  
fic code. This design pattern promote  
nd easier updates.


lement error handling and logging to  
h as element not found or timeout er  
les insights into test failures.

run tests in headless mode or use bro  
and reduce resource consumption. A

ensure that browser drivers are up-to-date for compatibility and  
performance improvements.



63,000+ Views



[Products](#)
[Developers](#)

[Live for Teams](#)
[Pricing](#)

ON THIS PAGE





[Guide](#)
[Categories](#)

[Why do Developers prefer Python for Selenium?](#)

[Press /](#)
[Getting Started with Selenium Python](#)

[Home](#)
[Testing on Cloud](#)
[Debugging](#)
[Best Practices](#)
[Tools & Frameworks](#)
[Tutorials](#)

[GET A DEMO](#)
[Free Trial](#)

63,000+ Views

Cross-platform testing on BrowserStack ensures consistent application performance across different environments. Additionally, it offers real-time debugging features such as live logs, screenshots, and video recordings, which aid in quick troubleshooting.

Seamless integration with CI/CD pipelines further automates the testing process, enabling tests to run on every code change and providing immediate feedback on application quality. Overall, BrowserStack Automate enables comprehensive, efficient, and reliable testing, fostering continuous development and deployment.

**Talk to an Expert**

#### TAGS

Python

Selenium



Products

Developers

Live  
for  
Teams

Pricing

## Related Guides

Guide

Categories

Press /

[Home](#)[Testing on Cloud](#)[Debugging](#)[Best Practices](#)[Tools & Frameworks](#)[Tutorials](#)

[GET A DEMO](#)

Free Trial

### How to Create and Use Action Class in Selenium Python

Actions class is an ability provided b...

December 19, 2022  
7 min read

### How to Double Click on an Element in Selenium Python?

Learn more about double click in...

March 1, 2023  
8 min read

### How to Press Enter without Element in Selenium Python?

Learn how to press enter in Selenium...

June 3, 2024  
7 min read

[View all guides](#)

## Test Automation on Real Devices & Browsers

Try BrowserStack Automate for Automation Testing for websites on 3500+ real Devices & Browser. Seamlessly Integrate with Frameworks to run parallel tests and get reports on custom dashboards



Products

Developers

Live  
for  
Teams

Pricin

#### ON THIS PAGE

[Guide](#)

[Categories](#)

Why do Developers prefer Python for

Press /

Getting Started with Selenium Python

[Home](#) [Testing on Cloud](#) [Debugging](#) [Best Practices](#) [Tools & Frameworks](#) [Tutorials](#)

Selenium Python Example: How to run

[GET A DEMO](#)

[Free Trial](#)



63,000+ Views



PRODUCTS

WHY

RESOURCES

COMPANY

Live for Teams

Pricing

ON THIS PAGE

[Guide](#)

[Categories](#)

[Why do Developers prefer Python for Selenium?](#)

[Press / Getting Started with Selenium Python](#)

[Home Testing on CloudDebuggingBest PracticesTools & FrameworksTutorials](#)

[Selenium Python Example: How to run Selenium](#)

[GET A DEMO](#)

[Free Trial](#)

63,000+ Views

[Find a partner](#)

[Trust Center](#)

[Test University \(Beta\)](#)

Test Observability

Accessibility Testing

Accessibility

Automation

App Accessibility Testing

Low Code Automation

Bug Capture

More Resources

Test On Devices

Tools

Cross Browser Testing

Test on iPhone • Test on iPad • Test on Galaxy • Test on IE • Test on Android • Test on iOS • Test on Right Devices •

SpeedLab • Screenshots • Responsive • Nightwatch.js

Selenium • Test Management • Emulators vs Real Device •

Mobile App Testing

Mobile Emulators

Contact Us