

Curso de Automação de Testes de Frontend Web com Python e Selenium

...

Aula 02

Paulo Oliveira

Agenda


- Revisão da Instalação do Python + VS Code
- Instalação Extensão Python
- Criação de Ambientes
- Instalação do Selenium
- Primeiro Script
- Locators

Revisão da Instalação do Python + VS Code

- Guia Passo a Passo
 - <https://github.com/paulocoliveira/alunos-curso-automacao-frontend>


Instalação da Extensão Python

Extension: Python



Python

v2824.2.1

Microsoft  microsoft.com | 115,802,115 | 5 (583)

Python language support with extension access points for IntelliSense (Pylance), Debugging (Python Debugger), linting, formatting, refactoring, unit tests, and more.

[Install](#) [Uninstall](#) [Switch to Pre-Release Version](#)

This extension is enabled globally.

DETAILS

FEATURES

CHANGELOG

EXTENSION PACK

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: >=3.7), including features such as IntelliSense (Pylance), linting, debugging (Python Debugger), code navigation, code formatting, refactoring, variable explorer, test explorer, and more!

Support for vscode.dev

The Python extension does offer some support when running on `vscode.dev` (which includes `github.dev`). This includes partial IntelliSense for open files in the editor.

Installed extensions

The Python extension will automatically install the following extensions by default to provide the best Python development experience in VS Code:

- Pylance** - to provide performant Python language support
- Python Debugger** - to provide a seamless debug experience with debugpy


These extensions are optional dependencies, meaning the Python extension will remain fully functional if they fail to be installed. Any or all of these extensions can be disabled or **uninstalled** at the expense of some features. Extensions installed through the marketplace are subject to the Marketplace Terms of Use.

Quick start

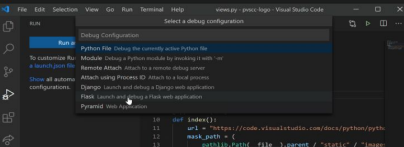
- Step 1.** Install a supported version of Python on your system (note: that the system install of Python on macOS is not supported).
- Step 2.** Install the Python extension for Visual Studio Code.
- Step 3.** Open or create a Python file and start coding!

Set up your environment

- Select your Python interpreter by clicking on the status bar



Configure the debugger through the Debug Activity Bar



Categories

Programming Languages | Debuggers | **Linters**

Formatters | Other | Data Science | Machine Learning

Resources

Marketplace

Issues

Repository

License

Microsoft

More Info

Published	2016-01-18, 15:53:11
Last released	2024-03-14, 10:16:50
Last updated	2024-03-13, 15:25:32
Identifier	ms-python.python

Criação de Ambientes Virtuais

- Objetivo
 - Ambientes virtuais são usados para isolar dependências de projetos Python, permitindo que diferentes projetos usem diferentes versões de bibliotecas e evitando conflitos entre elas.
- Passo a Passo
 - Navegue para o diretório do seu projeto
 - `cd path\to\your\project`
 - Crie um ambiente virtual
 - `python -m venv myenv`
 - Ative o ambiente virtual
 - Windows: `.\myenv\Scripts\Activate`
 - Mac: `source myenv/bin/activate`
 - Use o ambiente virtual
 - Desative o ambiente virtual
 - `deactivate`
- Referências
 - <https://docs.python.org/pt-br/3/library/venv.html>

Instalação do Selenium

- Comando para instalação do Selenium
 - `pip install selenium`
 - `pip3 install selenium`

Primeiro Script

- Cenário
 - Abrir o site www.uol.com.br
 - Maximizar o browser
 - Obter o título da página
 - Imprimir no console o título da página
 - Fechar o browser

Locators

- Os localizadores do Selenium são como o bloco de construção de um script de teste que ajuda os testadores a interagir com os elementos no Modelo de Objeto de Documento (DOM).

Locators

Locator	Descrição	Sintaxe em Python
id	Identifica o WebElement usando o atributo ID.	<code>driver.find_element(By.ID, "IdValue")</code>
name	Identifica o WebElement usando o atributo Nome	<code>driver.find_element(By.NAME, "nameValue")</code>
className	Usa o atributo Classe para identificar o objeto	<code>driver.find_element(By.CLASS_NAME, "classValue")</code>
linkText	Usa o texto em hiperlinks para localizar o WebElement	<code>driver.find_element(By.LINK_TEXT, "textofLink")</code>
partialLinkText	Usa uma parte do texto em hiperlinks para localizar o WebElement	<code>driver.find_element(By.PARTIAL_LINK_TEXT, "PartialTextofLink")</code>
tagName	Usa o tagName para localizar o WebElement desejado	<code>driver.find_element(By.TAG_NAME, "htmlTag")</code>

Locators

- Sites usados nos exemplos:
 - <https://ecommerce-playground.lambdatest.io/index.php?route=account/login>
 - <https://www.lambdatest.com/selenium-playground/ajax-form-submit-demo>
 - <https://www.lambdatest.com/selenium-playground/>
 - <https://ecommerce-playground.lambdatest.io/index.php?route=account/register>

Perguntas?

- Momento aberto para dúvidas



Próximos Passos

- O que esperar da próxima aula
 - Locators - parte 2
 - Interagindo com os elementos
- Prática de Casa
 - Escolher um ou mais sites quaisquer
 - Criar um script com os seguintes passos:
 - Abra o site
 - Maximize a janela do browser
 - Localize 3 elementos na página
 - Obtenha o título da página
 - Imprima o título da página no console
 - Feche o browser
 - Criar 6 scripts, um para cada tipo de locator (id, name, className, linkText, partialLinkText, tagName)
- Ganhe visibilidade todos os dias

Curso de Automação de Testes de Frontend Web com Python e Selenium

...

Aula 02

Paulo Oliveira