

Python para Ciência de Dados

Instalação



6 de 2022

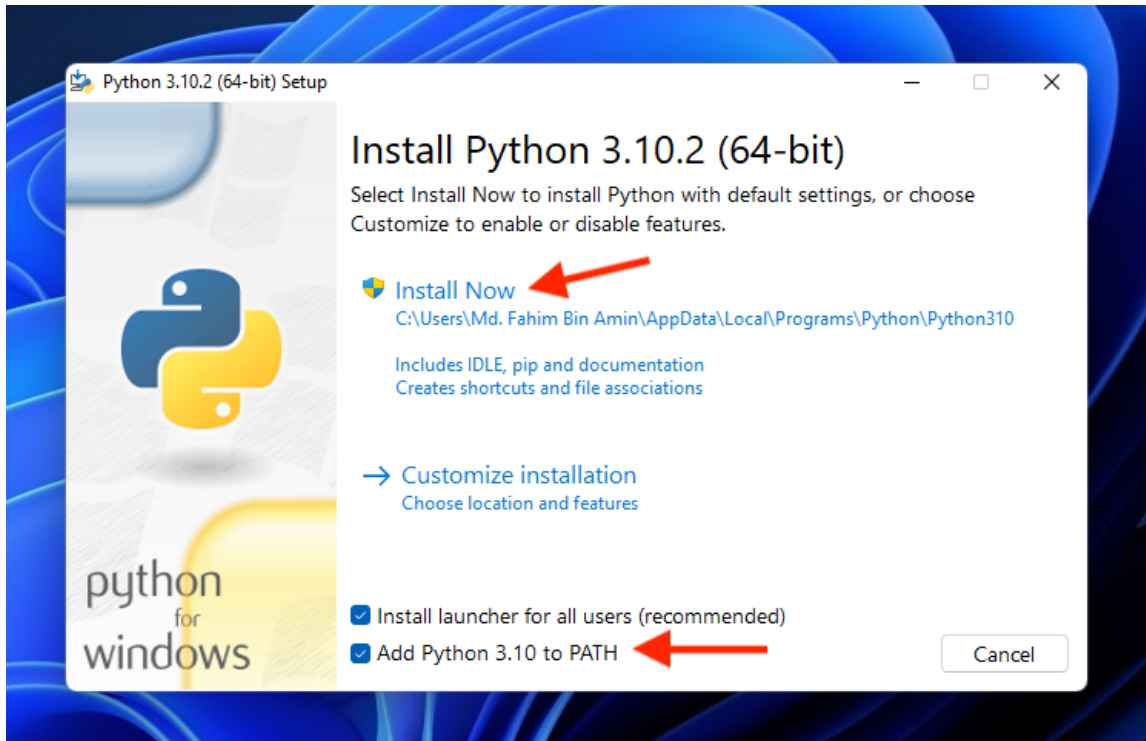
Motivação

- Até agora trabalhamos exclusivamente no Google Colab
 - Poucos recursos de hardware
 - Baixa customização da interface e do ambiente
 - *Minha opinião*: desincentiva reprodutibilidade e controle de versão
- Podemos instalar o Python nas nossas próprias máquinas
 - Problemas com a(s) versão(ões) do Python
 - Problemas com o controle do ambiente
 - Problemas com o espaço necessário
 - Não é tão simples quando parece!

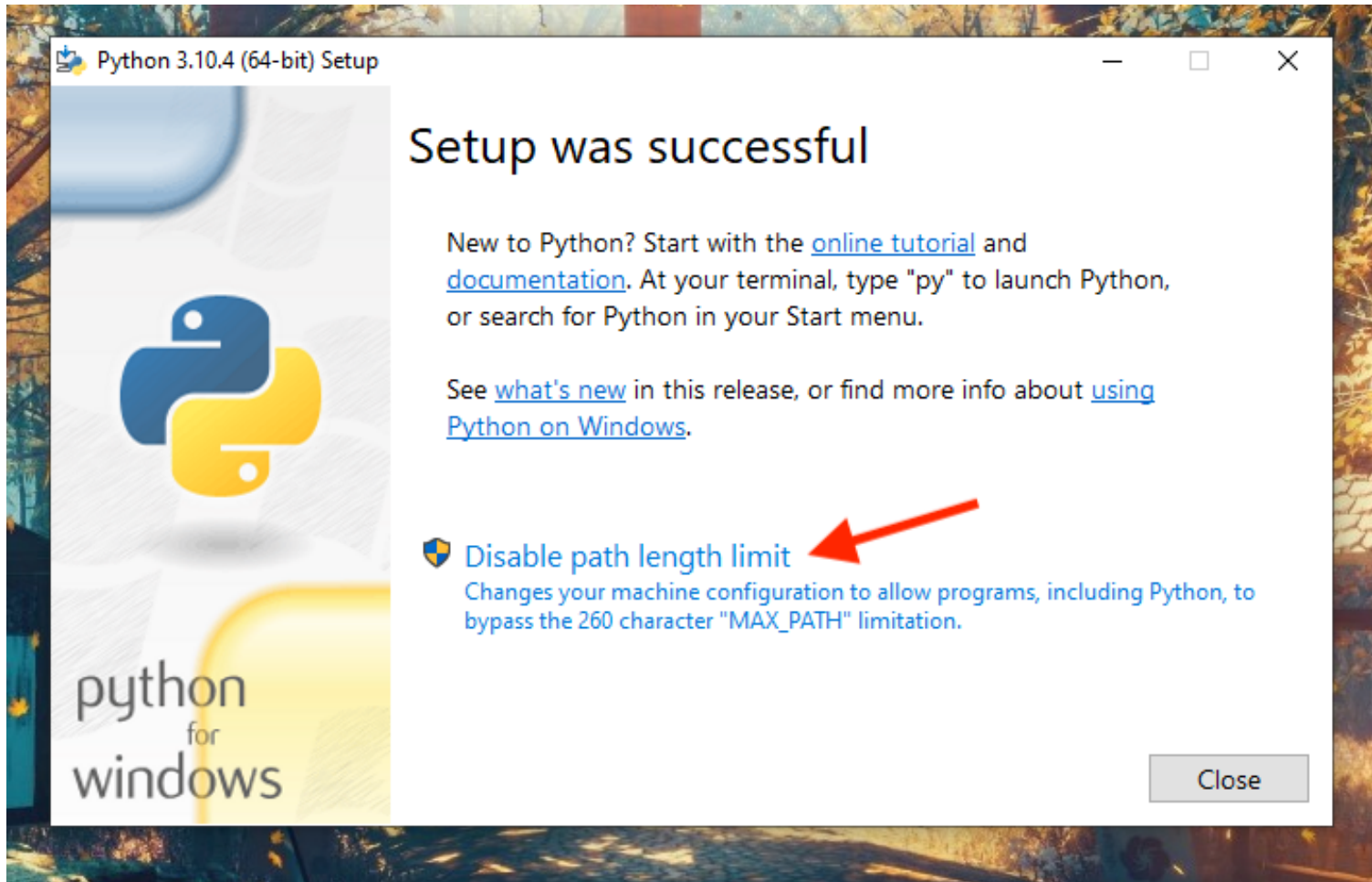
Instalação

Python no Windows

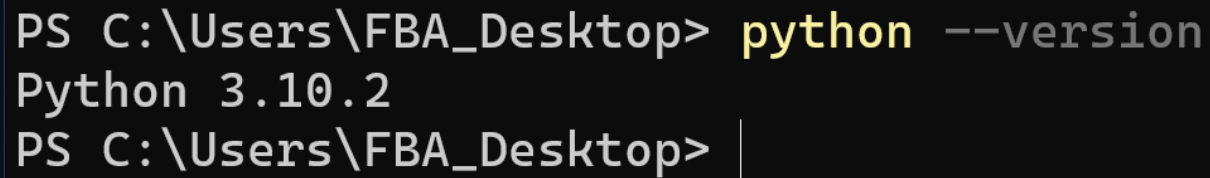
- Garantir que não há um Python já instalado!
- Baixar executável do instalador: <https://www.python.org/downloads/>
- Clicar duas vezes no arquivo



Python no Windows



Python no Windows

A screenshot of a Windows PowerShell terminal window. The window has a dark blue title bar with the text 'PowerShell' and standard window controls (minimize, maximize, close). The terminal content shows a command prompt at 'PS C:\Users\FBA_Desktop>' where the command 'python --version' has been entered. The output 'Python 3.10.2' is displayed on the next line. The prompt is now on a new line, ready for the next command.

```
PS C:\Users\FBA_Desktop> python --version
Python 3.10.2
PS C:\Users\FBA_Desktop> |
```

Python no Linux (Ubuntu)

- Verificar a versão já instalada
- Cada um dos comandos pode retornar uma versão diferente!

```
python --version  
python2 --version  
python3 --version
```

- Atualizar outros pacotes (pode demorar)

```
sudo apt-get update  
sudo apt-get upgrade -y
```

- Caso seja necessário, instalar o Python 3 à força

```
sudo apt-get install -y python3 python3-pip
```

Python no Linux (Ubuntu)

- Configurar as alternativas para o comando python
- Obs.: as linhas eram grande demais, por isso eu quebrei elas!

```
sudo update-alternatives --install  
    /usr/bin/python python /usr/bin/python3.8 1  
  
sudo update-alternatives --install  
    /usr/bin/python python /usr/bin/python3.10 2
```

- Escolher a versão mais atual

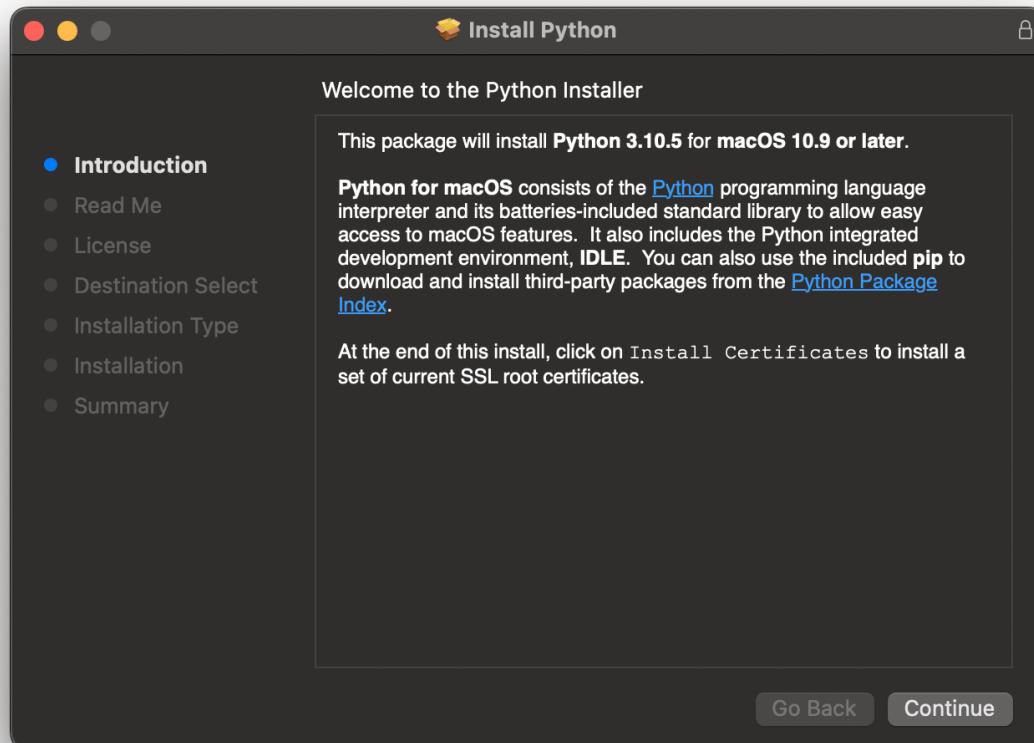
```
sudo update-alternatives --set python /usr/bin/python3.10
```

- Verificar o resultado

```
python --version
```

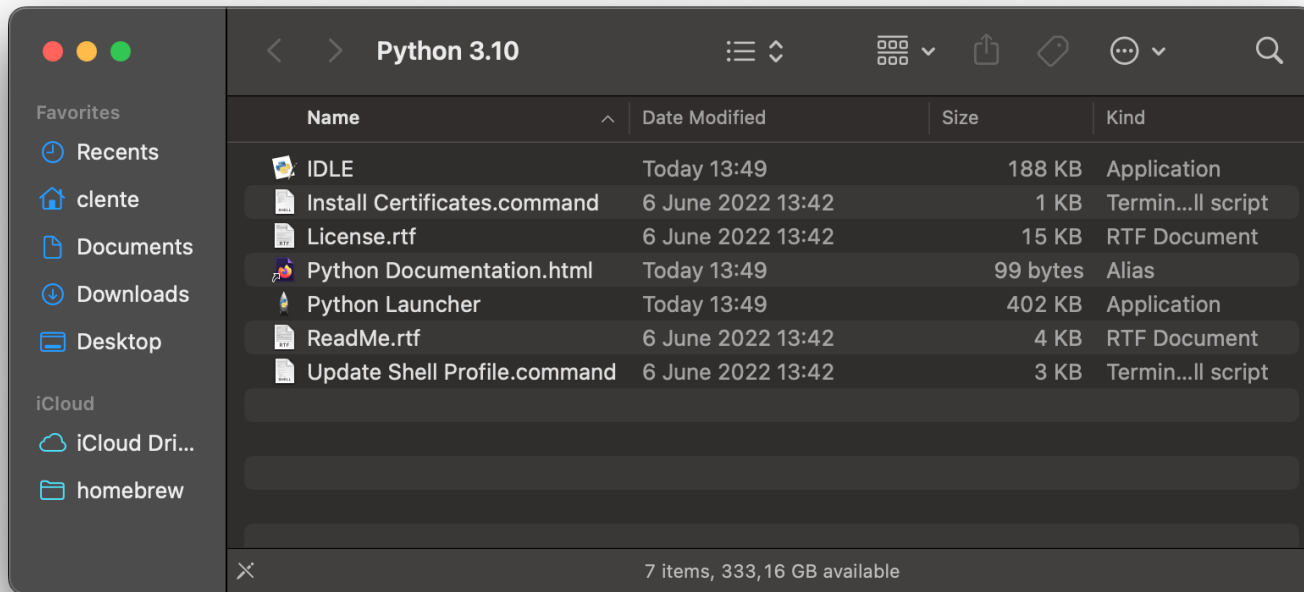

Python no Mac

- Baixar executável do instalador: <https://www.python.org/downloads/>
- Clicar duas vezes no arquivo



Python no Mac

- Examinar a pasta resultante



Python no Mac

- Selecionar a versão mais recente como sendo a padrão

```
unlink /usr/local/bin/python  
ln -s /usr/local/bin/python3.10 /usr/local/bin/python
```

- Verificar o resultado

```
python --version
```

Pip

- Pip é o instalador de pacotes do Python, muito importante
- Vale a pena confirmar se o pip também foi instalado

```
pip --version
```

- Se for necessário instalar ou atualizar:

```
python -m ensurepip --upgrade
```

Ambientes Virtuais

Conda

- Conda é um gerenciador de pacotes, dependências e ambientes para Python (e outras linguagens como R)
 - Anaconda é uma distribuição otimizada de Python que inclui diversos pacotes úteis para ciência de dados
 - Miniconda é uma versão é um instalador mínimo do Anaconda que não vem com todos os pacotes
- Instalar o Miniconda: <https://docs.conda.io/en/latest/miniconda.html>
 - No Windows, escolher o executável de 64-bits
 - No Mac, escolher o pkg Intel ou o pkg M1 a depender do chip
 - No Linux, escolher o script de 64-bits e executar o comando:

```
chmod +x Miniconda3-latest-Linux-x86_64.sh  
./Miniconda3-latest-Linux-x86_64.sh
```

Conda

- Verificar se a instalação funcionou

```
conda --version
```

- Note que o Conda é baseado em uma distribuição própria
 - Os pacotes instalados via Conda não vêm do PyPI, esqueça o Pip
 - O Python do Conda também não vem do repositório oficial

```
# Criar env com uma versão do Python
```

```
conda create -n meuenv python=3.8
```

```
# Instalar um pacote do novo env
```

```
conda activate meuenv
```

```
conda install numpy
```

```
# Desativar env
```

```
conda deactivate
```

Virtualenv

- Virtualenv é uma ferramenta para isolamento de ambientes
 - Parte do Virtualenv já vem embutido via o pacote venv
- A instalação é mais simples com o PipX

```
# Windows e Linux
```

```
python -m pip install --user pipx
```

```
# Mac (requer o Homebrew: https://brew.sh/)
```

```
brew install pipx
```

```
pipx ensurepath
```

- Depois é só usar o PipX para instalar o Virtualenv

```
pipx install virtualenv
```

```
virtualenv --help
```


Virtualenv

- Diferentemente do Conda, o Virtualenv cria pastas com envs

```
virtualenv meuenv
```

- Para ativar um env, é necessário carregar os conteúdos da pasta

```
# Windows
```

```
Set-ExecutionPolicy Unrestricted -Scope Process  
meuenv\Scripts\activate.ps1
```

```
# Linux e Mac
```

```
source meuenv/bin/activate
```

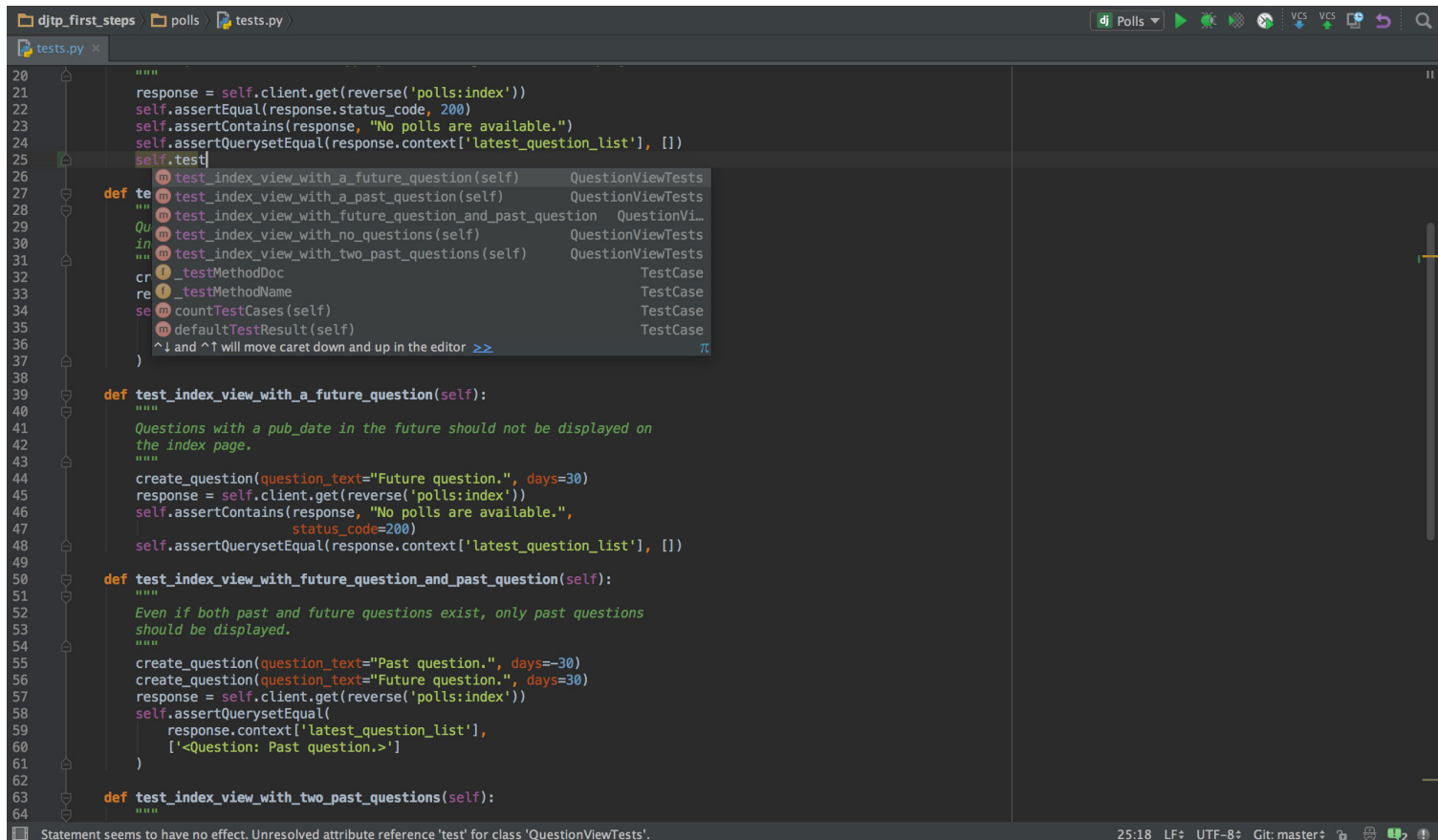
- Por padrão, a versão do Python é a global, mas podemos mudar
 - É necessário ter essa versão instalada!

```
virtualenv --python=/usr/local/bin/python3.8 meuenv
```

Ambientes de Desenvolvimento

PyCharm

- IDE clássica, criada em 1991 especificamente para o Python

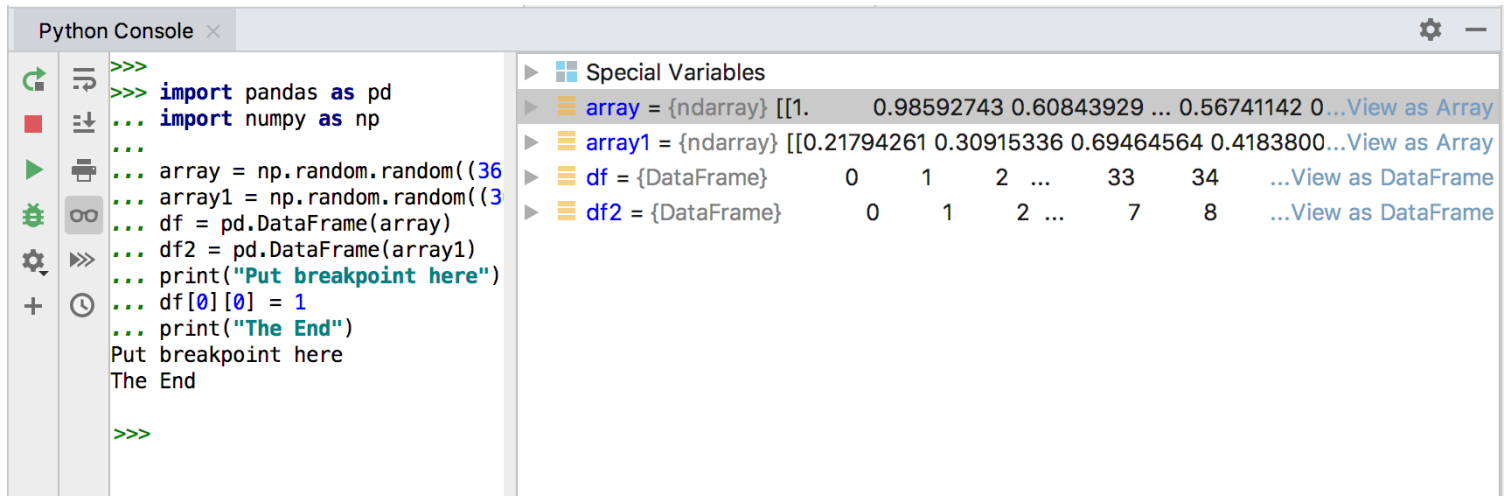


```
20
21
22 response = self.client.get(reverse('polls:index'))
23 self.assertEqual(response.status_code, 200)
24 self.assertContains(response, "No polls are available.")
25 self.assertQuerysetEqual(response.context['latest_question_list'], [])
26 self.test
27
28 def test_index_view_with_a_future_question(self):
29     """
30     Questions with a pub_date in the future should not be displayed on
31     the index page.
32     """
33     create_question(question_text="Future question.", days=30)
34     response = self.client.get(reverse('polls:index'))
35     self.assertContains(response, "No polls are available.",
36                         status_code=200)
37     self.assertQuerysetEqual(response.context['latest_question_list'], [])
38
39 def test_index_view_with_future_question_and_past_question(self):
40     """
41     Even if both past and future questions exist, only past questions
42     should be displayed.
43     """
44     create_question(question_text="Past question.", days=-30)
45     create_question(question_text="Future question.", days=30)
46     response = self.client.get(reverse('polls:index'))
47     self.assertQuerysetEqual(
48         response.context['latest_question_list'],
49         ['<Question: Past question.>']
50     )
51
52 def test_index_view_with_two_past_questions(self):
53     """
54     """
55     create_question(question_text="Past question.", days=-30)
56     create_question(question_text="Past question.", days=-30)
57     response = self.client.get(reverse('polls:index'))
58     self.assertQuerysetEqual(
59         response.context['latest_question_list'],
60         ['<Question: Past question.>']
61     )
62
63 def test_index_view_with_no_questions(self):
64     """
65     """
66     response = self.client.get(reverse('polls:index'))
67     self.assertContains(response, "No polls are available.")
68     self.assertQuerysetEqual(response.context['latest_question_list'], [])
```

Statement seems to have no effect. Unresolved attribute reference 'test' for class 'QuestionViewTests'.

PyCharm

- Cheia de funcionalidades e ferramentas embutidas



The screenshot displays the PyCharm interface with the Python Console and the Special Variables tool window. The console shows a series of Python commands for importing pandas and numpy, generating random arrays, and creating DataFrames. The Special Variables window shows the state of these variables, including their types and values.

```
>>>
>>> import pandas as pd
... import numpy as np
...
... array = np.random.random((36
... array1 = np.random.random((3
... df = pd.DataFrame(array)
... df2 = pd.DataFrame(array1)
... print("Put breakpoint here")
... df[0][0] = 1
... print("The End")
Put breakpoint here
The End
>>>
```

Special Variables

- `array` = {ndarray} [[1. 0.98592743 0.60843929 ... 0.56741142 0...[View as Array](#)
- `array1` = {ndarray} [[0.21794261 0.30915336 0.69464564 0.4183800...[View as Array](#)
- `df` = {DataFrame}

0	1	2	...	33	34	...
---	---	---	-----	----	----	-----

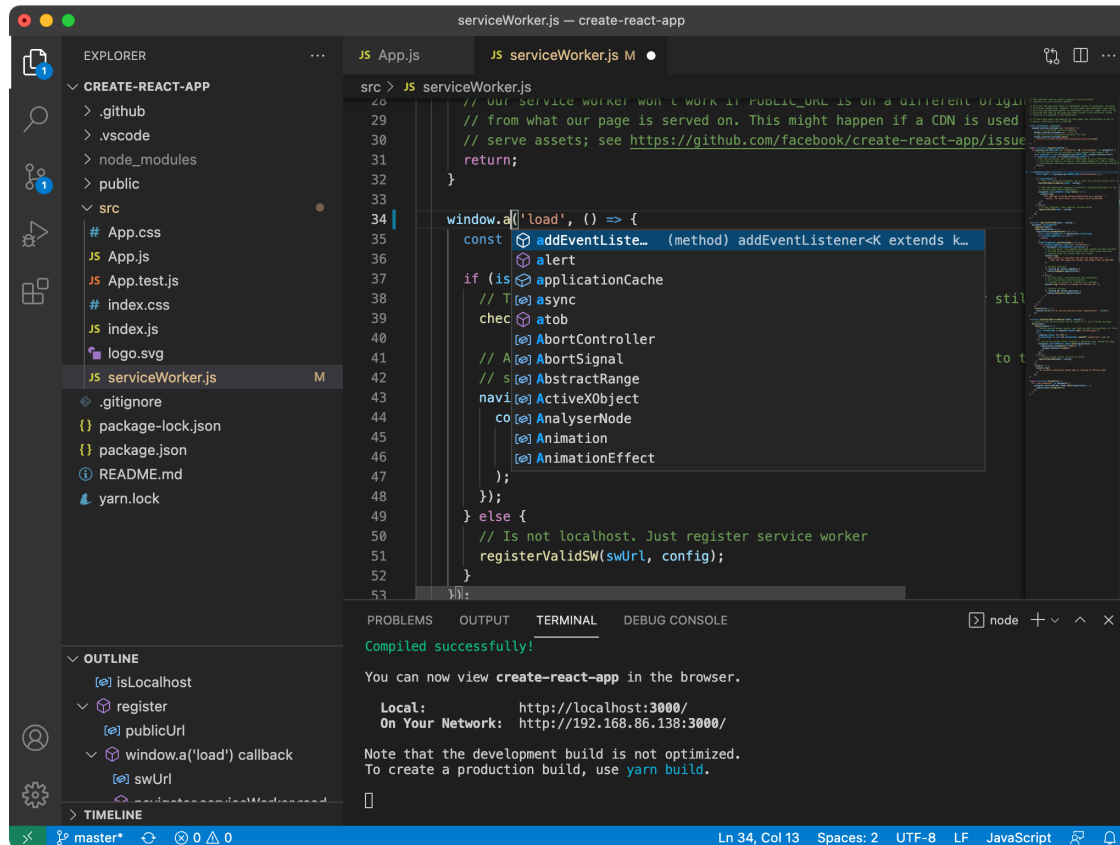
[View as DataFrame](#)
- `df2` = {DataFrame}

0	1	2	...	7	8	...
---	---	---	-----	---	---	-----

[View as DataFrame](#)

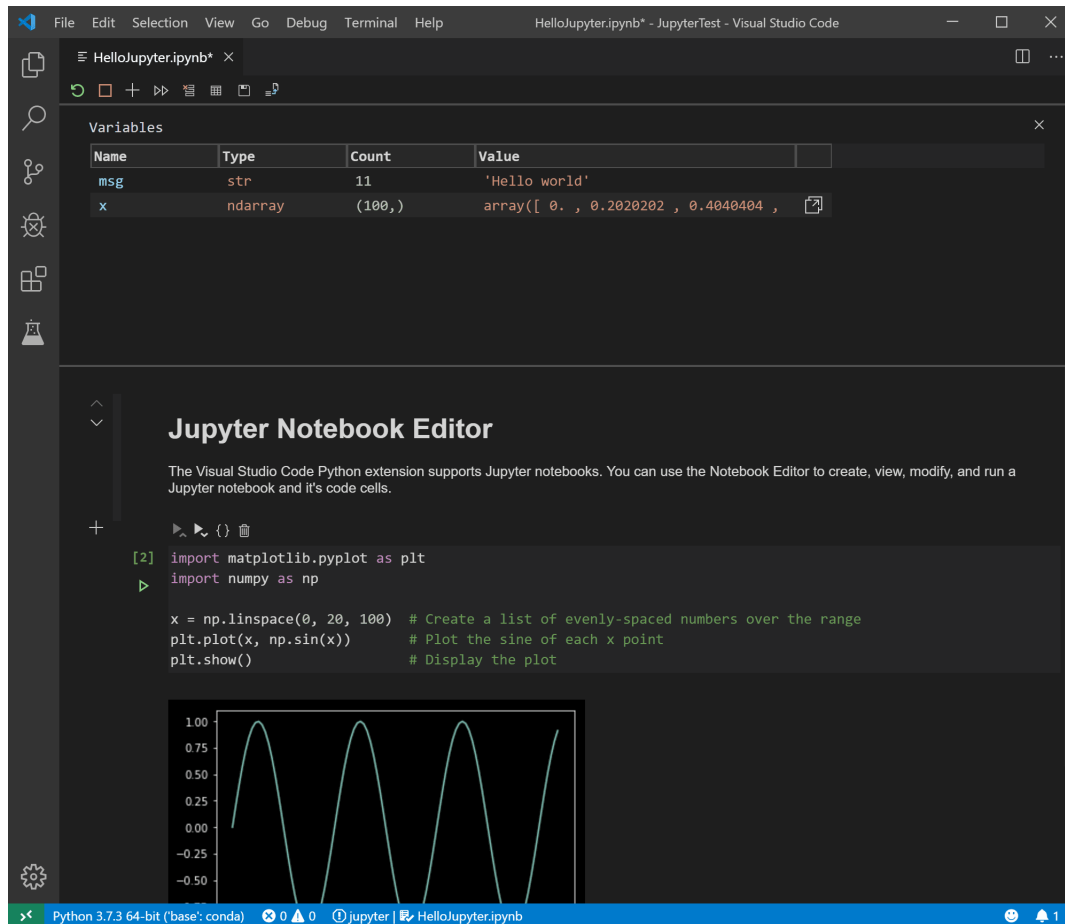
VS Code

- Atualmente o editor de código mais usado do mundo



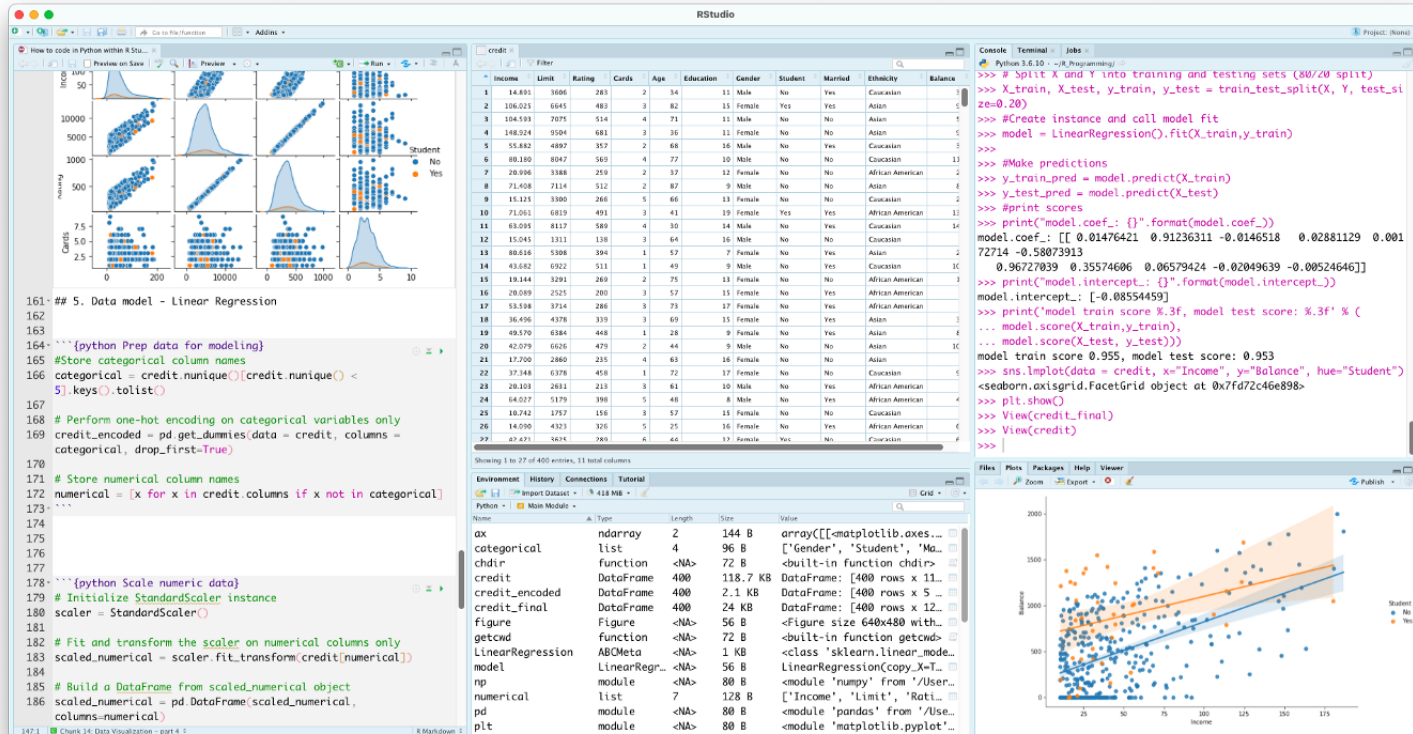
VS Code

- Simples, rápido e flexível (suporte para qualquer linguagem)



RStudio

- Agora é possível programar Python no RStudio!



RStudio

- Vamos falar mais sobre a integração entre o R e o Python no futuro
 - Com o pacote `reticulate` é possível rodar Python e R juntos
 - Funciona com Virtualenv ou Conda
 - O próprio R gerencia os ambientes e as instalações
- No RStudio a integração fica ainda melhor
 - É possível criar um RMarkdown com R e Python misturados
 - As duas linguagens conseguem compartilhar objetos!
 - Todas as funcionalidades familiares do RStudio com Python