# Criando um executável a partir de um código Python

Nesse presente documento, apresentamos como criar um executável “desafio_unisoma.exe” a partir de um arquivo de código Python “desafio_unisoma.py”.

Para o nosso objetivo, vamos precisar da biblioteca “pyinstaller”.

## Preparação

1. Crie um virtual environment
   Isso ajuda a criar um ambiente limpo e a lidar com as dependências do projeto. Sugerimos usar um “virtualenv” a partir do Pycharm. Instale todos os pacotes necessários para o código Python nesse ambiente virtual, como o Pandas e o Numpy, por exemplo. Lembre-se também de instalar o Pyinstaller no venv
2. Mude o arquivo “pandas-hook”
   Navegue até “*<Project Path>\venv\Lib\site-packages\PyInstaller\hooks”* e encontre o arquivo “hook-pandas.py”. Se ele não existir, crie um. Garanta que esse arquivo contenha o seguinte texto, além de outros que eventualmente já estejam no arquivo:

```
from PyInstaller.utils.hooks import collect_submodules

# Pandas keeps Python extensions loaded with dynamic imports here.
hiddenimports = collect_submodules('pandas._libs')
```

## Criando o arquivo .spec

O objetivo desse arquivo é configurar e preparar a criação do executável. Para criá-lo, rode o seguinte código no terminal do seu projeto:

pyi-makespec —onefile desafio_unisoma.py

Não será necessário mais rodar esse comando. Caso rodá-lo, o arquivo atual será substituído.

## Editando o arquivo .spec

Editaremos o arquivo .spec para adicionar as dependências do projeto, que entrarão no executável.

- Abra o arquivo desafio_unisoma.spec.
- Note o seguinte trecho do arquivo, em que é definida a variável “a”:

```
a = Analysis(
    ['desafio_unisoma.py'],
    pathex=[],
    binaries=[],
    datas=[],
    hiddenimports=[],
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    noarchive=False,
    optimize=0,
)
```

Para adicionar as dependências, iremos editar os valores “hiddenimports” e “datas”.

## Hiddenimports

Nesse atributo, iremos listar os imports implícitos que serão usados pelo executável.
O Pyinstaller não checa se há importações implícitas no código, que são os imports usados no arquivo desafio_unisoma.py. Assim, hiddenimports será uma lista com todos esses imports. Por exemplo, supondo que nosso código use “pandas”, “matplotlib.pyplot”, “os” e “sys”, temos que definir o hiddenimports dentro da variável “a” da seguinte maneira:

```
hiddenimports=['pandas','matplotlib.pyplot','os','sys']
```

## datas

O atributo “datas” é comumente utilizado para a “inserção” de arquivos no executável.
No nosso caso, usaremos o atributo para carregar os arquivos de **todas** as bibliotecas específicas do projeto, que instalamos no virtual environment.
Esse passo é importante para evitar erros de execução.
Para isso, precisamos indicar o caminho em que a biblioteca está instalada e adicionar um objeto específico ao atributo.
Em outras palavras, precisamos adicionar o seguinte código logo abaixo à definição da variável “a” no arquivo .spec:

path_pandas = ‘*Project_Path/venv/Lib/site-packages/Pandas’*
a.datas += Tree(path_pandas, prefix='pandas', excludes=["*.pyc"])

Onde *Project_Path* é o caminho para o seu projeto, e *venv* é o virtual environment criado.

No caso específico do print abaixo, o caminho para o Pandas é 'C:/Users/RaulTeixeira/.virtualenvs/desafio_2024/Lib/site-packages/pandas'

```
a = Analysis(
    ['desafio_unisoma.py'],
    pathex=[],
    binaries=[],
    datas=[],
    hiddenimports=['pandas'],
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    noarchive=False,
    optimize=0,
)

path_pandas = 'C:/Users/RaulTeixeira/.virtualenvs/desafio_2024/Lib/site-packages/pandas'
a.datas += Tree(path_pandas, prefix='pandas', excludes=["*.pyc"])
```

Lembre-se de adicionar as duas linhas de comando ao arquivo .spec para cada biblioteca específica utilizada no projeto.

**Dica**: caso estiver com dificuldades para achar o local em que a biblioteca está instalada no virtual environment, ou caso o comando da próxima seção estiver dando errado, siga os seguintes passos para encontrar o caminho para as bibliotecas para a definição das variáveis “path_”:

- Nos arquivos do projeto, navegar até External Libraries > site-packages
- Botão direito em site-packages > Copy Path/Reference > "Absolute Path"

## Compilando seu programa usando o arquivo .spec

Após salvar as modificações no arquivo .spec, basta rodar o seguinte comando no terminal, no seu projeto:

pyinstaller -y desafio_unisoma.spec

Aguarde o comando rodar. O arquivo exe surgirá dentro de uma nova pasta “dist”, que estará na pasta do projeto.

## Testando o executável

Ao dar um duplo clique em “desafio_unisoma.exe”, o código rodará no prompt de comando, mostrando o log da execução. É interessante adicionar um ponto de parada no código, para evitar que o prompt feche ao final da execução.

# Referência

https://medium.com/@lironsoffer/pyinstaller-with-pandas-problems-solutions-and-workflow-with-code-examples-c72973e1e23f