Ambientes virtuais em Python

virtual environment" ou simplemente ambiente virtual. É um "espaço" ou "contexto" isolado dentro do qual o Python e seus pacotes podem ser executados. Quando criamos um ambiente virtual é a criada uma pasta que contém uma cópia independente do interpretador Python, juntamente com uma instalação local da biblioteca padrão e outras ferramentas essenciais, como o gerenciador de pacotes pip. Dentro desse ambiente, é possível instalar e gerenciar pacotes sem que isso afete o ambiente global do Python instalado no sistema operacional. Considere que o ambiente virtual é um conceito ou uma abstração que representa um espaço de trabalho isolado para projetos Python.

- As ferramentas como venv e virtualenv são os meios pelos quais esses ambientes virtuais são criados e gerenciados. Elas facilitam a criação desse espaço isolado, onde você pode controlar as versões dos pacotes e do próprio Python, garantindo que seu projeto seja executado em um ambiente controlado e previsível, independente de outros projetos ou configurações globais do sistema.

- A ideia é a de que, quando trabalhamos em múltiplos projetos Python, é comum que cada um deles necessite de versões diferentes de um mesmo pacote. Sem o uso de ambientes virtuais, atualizar um pacote para um projeto pode quebrar outro projeto que dependa de uma versão anterior do mesmo pacote. Ambientes virtuais resolvem este problema permitindo a criação de um ambiente isolado para cada projeto, com suas próprias versões de pacotes instaladas.

Para dar forma ao que estamos estudando, a partir da raiz do disco no seu usuário crie uma pasta chamada PROJ001, depois abra a pasta no VsCode ou seu editor de códigos preferido.

 # Possíveis problemas ao tentar trabalhar com ambientes virtuais ou outros comandos de manipulação de scripts e diretório no Windows:

- Execute no terminal do VsCode o comando:

set-executionpolicy -scope currentuser -executionpolicy remotesigned pip instal

- Este comando normalmente é específico para o PowerShell no Windows e não está diretamente relacionado à criação ou ao uso de ambientes virtuais em Python, mas sim à política de execução de scripts no PowerShell.

- **PowerShell** é uma interface de linha de comando e linguagem de script para sistemas baseados em Windows, oferecem funcionalidades poderosas para automação e gerenciamento do sistema. O *Execution Policy* no PowerShell determina as condições sob as quais scripts PowerShell podem ser executados. Por padrão, o Windows pode ter uma política que restringe a execução de scripts não assinados para proteger contra a execução acidental ou maliciosa de código.

- O **Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned** é um comando usado para alterar a política de execução para o usuário atual, permitindo a execução de scripts PowerShell que foram assinados por um editor confiável, além de scripts não assinados que são executados localmente. Isso é feito para aumentar a segurança, mas ao mesmo tempo permitir ao usuário executar scripts locais e scripts remotos confiáveis.

- A necessidade de alterar a política de execução no PowerShell para trabalhar com Python ou ambientes virtuais geralmente surge quando estamos tentando executar scripts ou comandos (como a instalação de pacotes Python com pip) que são afetados pela política de execução padrão do PowerShell.

- Se tentarmos instalar pacotes Python usando pip dentro de um ambiente virtual e se deparar com restrições devido à política de execução do PowerShell, ajustar essa política pode permitirá prosseguir sem erros.

- Isso é particularmente relevante quando se utiliza ferramentas ou scripts que fazem chamadas para o PowerShell ou quando se está executando o PowerShell diretamente para configurar ou usar ambientes virtuais.

- Se estamos utilizando apenas o Prompt de Comandos do Windows (CMD) ou o terminal integrado do Visual Studio Code (VSCode) e nos deparamos com a necessidade de ajustar a política de execução do PowerShell (Set-ExecutionPolicy), precisaremos ajustar essa política, mesmo que esteja usando esses outros terminais:

- **Integração do PowerShell:** Mesmo que estejamos usando o CMD ou o terminal do VSCode, alguns processos ou ferramentas podem invocar scripts ou comandos do PowerShell internamente. Por exemplo, determinadas extensões do VSCode, ferramentas de desenvolvimento ou scripts de inicialização podem utilizar o PowerShell em segundo plano para realizar suas tarefas.

Scripts de Ativação de Ambiente Virtual: Quando criamos um ambiente virtual em Python no Windows, são gerados scripts de ativação específicos para diferentes shells, incluindo o PowerShell (Activate.ps1). Se tentarmos ativar um ambiente virtual usando este script específico do PowerShell a partir do terminal do VSCode ou do CMD (invocando explicitamente o PowerShell), poderemos nos deparar com restrições de política de execução.

Caso realmente seja necessário ajustar a política de execução para certas operações no PowerShell, fazer isso com cautela e entender as implicações de segurança é importante.

Lembre-se, essa necessidade pode surgir devido a interações específicas entre ferramentas de desenvolvimento, scripts e a configuração de segurança do sistema operacional, não sendo um requisito padrão para o uso de ambientes virtuais em Python em todos os cenários. Fonte: https://docs.python.org/3/library/venv.html

Para ajustar a política de execução no PowerShell e criar um ambiente virtual chamado virtualenv a partir do diretório `C:\proj001` no Windows, deveremos seguir os seguintes passos:

Ajustar a Política de Execução no PowerShell

- Abra o PowerShell como administrador, pois esta ação requer privilégios elevados.
- Pressione Windows + X e selecione “Windows PowerShell (Admin)” ou “Terminal do Windows (Admin)”.
- No PowerShell, execute o seguinte comando:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

- Esse comando solicitará sua confirmação. Responda Y para confirmar a alteração.

Criar um ambiente virtual no C:\proj001

- Navegue até o diretório usando o CMD, PowerShell, ou o terminal integrado do VSCode. Se estiver usando o PowerShell ou o terminal do VSCode, o comando é: `cd C:\proj001`

- Em seguida, execute o comando para criar o ambiente virtual chamado venv: `python -m venv venv`. No PowerShell ou CMD, usamos o módulo venv do Python, que é a ferramenta padrão para criar ambientes virtuais.

- Esse comando cria um novo diretório chamado venv dentro de C:\proj001, contendo o ambiente virtual.

Ativar um ambiente virtual no C:\proj001

- Para começar a usar o ambiente virtual, precisaremos ativá-lo. A ativação do ambiente virtual varia ligeiramente entre o PowerShell e o CMD.

- No PowerShell, `.\venv\Scripts\activate.ps1`

- No CMD, execute: `.vlenv\Scripts\activate.bat`

- Após a ativação de um ambiente virtual em Python, o prompt de comando (seja no CMD, PowerShell, ou terminal integrado do Visual Studio Code) exibe o nome do ambiente virtual entre parênteses no início da linha do prompt.

- Exemplo no CMD: `(venv) C:\proj001>`

- Exemplo no No PowerShell: `(venv) PS C:\proj001>`

Lembre-se: a presença de (venv) no início do prompt indica que qualquer comando Python ou pip que você executar nesse terminal afetará apenas o ambiente virtual venv, sem interferir com outros projetos Python ou com a instalação global do Python no seu sistema. Isso ajuda a manter as dependências de projetos diferentes isoladas umas das outras.

Se temos o diretório base chamado proj001, e uma virtual env chamada venv, devemos sempre criar os sub-diretórios a partir do diretório base?

Quando se trata de organizar estruturas de diretório em projetos Python, especialmente com a presença de ambientes virtuais como venv, a prática recomendada é manter seu código-fonte, incluindo diretórios de organização lógica como mvc (Model-View-Controller), fora do diretório do ambiente virtual (venv). As razões são:

- **Separação de Responsabilidades:** Um ambiente virtual (venv), destina-se exclusivamente a gerenciar as dependências do seu projeto, incluindo o interpretador Python, bibliotecas e scripts. Ele cria um ambiente isolado para seu projeto, evitando conflitos entre dependências de diferentes projetos.

- **Código do Projeto** (proj001): Contém o código-fonte do projeto, incluindo aplicativos, módulos, pacotes, testes e quaisquer outros recursos necessários (como diretórios mvc para a arquitetura Model-View-Controller).

- **Facilidade de Manutenção:** Manter o ambiente virtual separado do código-fonte simplifica o gerenciamento de dependências e a manutenção do código. Podemos facilmente excluir, recriar ou atualizar o ambiente virtual sem afetar o código-fonte do seu projeto.

- **Controle de Versão:** um ambiente virtual não deve ser incluído no controle de versão (como Git), pois contém arquivos específicos do sistema que podem não ser relevantes ou compatíveis em diferentes ambientes ou sistemas operacionais. O código-fonte e os recursos lógicos do projeto, por outro lado, devem ser versionados.

- **Estrutura de Diretório Recomendada:** considerando essas boas práticas, uma estrutura de diretório recomendada para o seu projeto proj001 com um ambiente virtual venv poderia ser:

```
proj001/
|
├── venv/              # Ambiente virtual
|
├── mvc/               # Diretório MVC para seu projeto
|   ├── models/
|   ├── views/
|   └── controllers/
|
├── tests/             # Testes
|
├── .gitignore         # Git ignore file
|
├── requirements.txt   # Lista de dependências
|
└── app.py             # Aplicativo principal ou ponto de entrada
```

Sistema de diretórios: entender como manipular diretórios lista-los utilizando o Python Menezes (2019).

Um diretório corrente em Python é considerado um ***path*** (caminho).

Trabalhar diretamente com arquivos no mesmo diretório quando o projeto é pequeno pode ser uma comodidade. Porém, normalmente um projeto começa

com uma pequena ideia e vai aumentando de tamanho conforme o cliente vai percebendo o quanto o programa pode ajudá-lo, por isso é importante estar atendo ao fator escalabilidade.

- A partir da raiz do disco, crie um diretório chamado `proj001`

- Abra o *Visual Studio Code,* abra o path criado e após crie o arquivo principal.py.

No Linux ou IOS

- Crie o ambiente virtual chamado venv: `python -m venv env`
- Ativar o ambiente virtual no Linux ou IOS: `source venv/bin/activate`

No Windows - considerando que temos o diretório base proj001, e preciso ativar a política do segurança, os comandos para criar um ambiente virtual chamado venv serão:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
python -m venv venv
```

- Ativar o Ambiente virtual no Windows com PowerShell:
  `.\venv\Scripts\Activate.ps1`
- Ativar o Ambiente virtual no Windows com bash:
  `source venv/Scripts/activate`

- Desativar o ambiente virtual no Windows com Powershell ou bash:
  `.\venv\Scripts\Activate.ps1`
- Ativar o Ambiente virtual no Windows com Powershell ou bash:
  `deactivate`

Verifica o path – módulo os função **getcwd**() no Linux ou IOS

```
import os
root_dir = os.getcwd()
print(f'\nO Path é: {root_dir} \n')
```

Referências

F. V. C. Santos, Rafael. Python: Guia prático do básico ao avançado (Cientista de dados Livro 2) (p. 180). rafaelfvcs. Edição do Kindle.

Mueller, John Paul. **Programação Funcional Para Leigos**. Alta Books. Edição do Kindle, 2020.

MENEZES, Nilo Ney Coutinho. **Introdução à programação com Python**. 3ª Edição. 2019. Editora Novatec - São Paulo - SP - Brasil.

Postar todos os códigos no ambiente virtual.ifro.edu.br na “Tarefa de 26-09-2022 - até 03-10-2022 às 18:30 - vale até 2 pontos”.

Atenção: Para essa atividade o número de tentativas de envio é um (1)

Comandos em Python:

- Criar ambiente virtual em python linux e ios: python -m venv env
- Ativar o ambiente virtual em python linux e ios: source env/bin/activate/users/projts/env

set-executionpolicy -scope currentuser -executionpolicy remotesigned pip install windows-curses

https://www.geradorcpf.com – gerador de cpf para testes de programação pip install pytest pytest-benchmark

pip install flask

pip install locust

pylint --generate-rcfile > .pylintrc