

Script Bash para Agilizar a Criação de Projetos Python: Simplicidade e Eficiência



A ideia

Se você já passou pela tarefa repetitiva de configurar um projeto Python do zero, sabe como isso pode consumir tempo: criar pastas, arquivos, configurar o ambiente virtual, instalar dependências e ajustar detalhes como o `.gitignore`.

Mas e se eu te dissesse que você pode automatizar todo esse processo com um único comando? Foi exatamente isso que decidi fazer criando um **script Bash** para **padronizar e agilizar a configuração de novos projetos Python**, apesar da existência de

outras ferramentas como o Cookiecutter uma solução mais robusta para criar projetos baseados em templates.

Por que isso é útil?

Aqui estão algumas razões pelas quais um script automatizado faz sentido para quem trabalha com Python (e outras linguagens):

- **Economia de tempo:** Você não precisa repetir manualmente tarefas básicas para cada projeto.
- **Padronização:** Todo projeto começa com a mesma estrutura e organização, facilitando o entendimento e colaboração.
- **Foco no essencial:** Em vez de gastar tempo configurando, você começa a codificar o que realmente importa.
- **Flexibilidade:** O script pode ser adaptado às suas necessidades, incorporando práticas específicas da sua rotina.

Como funciona?

O script que criei automatiza as seguintes etapas:

1. Criação do diretório do projeto.
2. Configuração do ambiente virtual Python.
3. Instalação de bibliotecas essenciais (como pandas e matplotlib).
4. Geração do arquivo requirements.txt.
5. Criação de pastas organizadas (src, data, tests, etc.).
6. Criação de arquivos base (README.md, main.py, etc.).
7. Inclusão de um .gitignore com as melhores práticas.
8. Um arquivo de orientações (USAGE.md) para facilitar o uso e entendimento do projeto.

Exemplo Prático

Imagine-se trabalhando num projeto de administração financeira, começando um desafio ou desenvolvendo um novo script para análise de dados. Após criar o script Bash, você executa:

```
./setup_project.sh nome_do_projeto
```

E, em segundos, seu projeto está configurado com:

- Um ambiente virtual pronto.
- Dependências instaladas.
- Estrutura de pastas e arquivos organizada.
- Arquivos ignorados pelo Git.
- Tudo pronto para você começar a codificar.

Benefícios na Prática

- **Mais velocidade no desenvolvimento:** Principalmente para quem trabalha em múltiplos projetos pequenos ou está aprendendo Python.
- **Evita erros:** Não se esqueça de criar o `.gitignore` ou instalar dependências essenciais.
- **Organização consistente:** Cada projeto segue o mesmo padrão, facilitando a manutenção.

Disponibilizando o Script

Veja o script abaixo e configure-o de acordo com seus novos projetos em Python. Copie-o e altere-o conforme sua necessidade e comece a usar hoje mesmo:

```
#!/bin/bash

# Verifica se foi fornecido o nome do projeto
if [ -z "$1" ]; then
    echo "Por favor, forneça o nome do projeto. Exemplo:"
    echo "./setup_project.sh nome_do_projeto"
    exit 1
fi

PROJECT_NAME=$1

# Cria o diretório do projeto
mkdir $PROJECT_NAME
cd $PROJECT_NAME

# Cria o ambiente virtual
python -m venv venv

# Ativa o ambiente virtual
source venv/Scripts/activate

# Instala dependências básicas
pip install pandas matplotlib tabulate

# Gera o requirements.txt
pip freeze > requirements.txt

# Cria a estrutura de pastas e arquivos
mkdir data src tests docs images
touch README.md USAGE.md src/simulacao.py src/utis.py
src/configuracoes.py main.py
```

```
# Cria um arquivo .gitignore
cat <<EOL > .gitignore
# Ignorar o ambiente virtual
venv/
```

```
# Ignorar arquivos temporários e cache do Python
__pycache__/
*.py[cod]
*.pyo
*.log
```

```
# Ignorar arquivos de saída
*.csv
*.png
*.jpg
*.jpeg
```

```
# Ignorar arquivos do sistema operacional
.DS_Store
Thumbs.db
EOL
```

```
# Adiciona conteúdo básico ao USAGE.md
cat <<EOL > USAGE.md
# Como Usar o Script de Configuração do Projeto
```

1. Introdução

Este projeto foi configurado usando o script
`setup_project.sh`, que cria automaticamente:

- Estrutura básica de pastas e arquivos.
- Ambiente virtual configurado.
- Dependências essenciais instaladas.

2. Estrutura do Projeto

- `src/`: Código-fonte principal.
- `data/`: Dados para simulações ou testes.
- `tests/`: Scripts para testes unitários.
- `docs/`: Documentação adicional.
- `images/`: Imagens coletadas ou geradas no desenvolvimento.
- `venv/`: Ambiente virtual (ignorado pelo Git).

3. Como usar o script

1. Salve o script como `setup_project.sh`.
2. Dê permissão de execução ao script:

```
chmod +x setup_project.sh
```

3. Execute o script passando o nome do projeto como argumento:

```
./setup_project.sh nome_do_projeto
```

4. Como Executar o Projeto

1. Ative o ambiente virtual:

- **Windows**: Terminal VSCode Git Bash

```
source venv/Scripts/activate
```

- **Linux/Mac**:

```
source venv/bin/activate
```

2. Execute o arquivo principal:

```
python main.py
```

5. Como Adicionar Dependências

1. Instale o pacote:

```
pip install nome_do_pacote
```

2. Atualize o `\`requirements.txt\``:

```
pip freeze > requirements.txt
```

6. OBSERVAÇÃO

Você pode usar o `\`pipreqs\`` para gerar um arquivo `requirements.txt` com apenas as bibliotecas que estão sendo utilizadas diretamente no seu projeto. Isso ajuda a evitar dependências desnecessárias, resultando em um arquivo mais enxuto e específico.

Como usar o pipreqs

1. Instale o pipreqs: Certifique-se de que o ambiente virtual está ativo e instale o pacote:

```
pip install pipreqs
```

2. Gere o arquivo `requirements.txt`: Execute o comando abaixo no diretório raiz do seu projeto (substitua `./` pelo caminho correto, se necessário):

```
pipreqs ./ --force
```

`./`: Representa o diretório atual.

`--force`: Sobrescreve o arquivo `requirements.txt` existente.

3. O que o `pipreqs` faz?

- Ele escaneia os arquivos Python no diretório especificado.
- Identifica as bibliotecas importadas diretamente no código.
- Gera um `requirements.txt` contendo apenas essas bibliotecas e suas versões.

7. Git Hub

Certifique-se de manter o `\.gitignore\`` atualizado para evitar versionar arquivos desnecessários.

8. Resultado

Ao executar o script, você terá:

- Um diretório com o nome do projeto.
- Ambiente virtual configurado.
- Estrutura organizada de pastas e arquivos.
- Arquivo `.gitignore` para evitar versionar arquivos desnecessários.
- Um `USAGE.md` com instruções claras de uso.

EOL

```
# Mensagem de sucesso
```

```
echo "Projeto '$PROJECT_NAME' configurado com sucesso!"
```

Conclusão

Automatizar a configuração de projetos em Python não é apenas uma questão de praticidade, mas também de produtividade e organização.

Um script simples como este pode economizar horas de trabalho ao longo do tempo e tornar seus projetos mais consistentes.

Se você gostou da ideia ou tem sugestões de melhoria, comente aqui! Vamos trocar experiências e aprender juntos.

Siga-me no LinkedIn: www.linkedin.com/comm/mynetwork/discovery-see-all?usecase=PEOPLE_FOLLOWS&followMember=izairton-oliveira-de-vasconcelos-a1916351