# Organizando seu projeto Python

1. Organização de pastas:

* Criar repositório no Git, definir nome, descrição e licença do projeto, e utilizar o comando: git clone URL\_REPOSITÓRIO
* Criar uma pasta com o nome do projeto e dentro dela incluir a estrutura básica de três pastas obrigatórias: **docs, tests, código fonte** (com nome do projeto) e uma pasta opcional de manipulação geral (pode se chamar Scripts ou Tools)
* Incluir o arquivo de configuração do ‘gitignore’ com o comando: type nul > .gitignore
* Incluir quais arquivos serão ignorados pelo git, utilizar o site: <https://www.toptal.com/developers/gitignore/>

1. Definição de versão do Python:

Utilizar a Lib pyenv para definir qual será a versão do Python utilizada no projeto:

Se não estiver com o pyenv instalado e configurado nas variáveis de ambiente, seguir:

* + - Executar o comando no CMD: git clone https://github.com/pyenv-win/pyenv-win.git "%USERPROFILE%\.pyenv"
    - Configure a variável de ambiente:

[System.Environment]::SetEnvironmentVariable('PYENV',$env:USERPROFILE + "\.pyenv\pyenv-win\","User")

[System.Environment]::SetEnvironmentVariable('PYENV\_ROOT',$env:USERPROFILE + "\.pyenv\pyenv-win\","User")

[System.Environment]::SetEnvironmentVariable('PYENV\_HOME',$env:USERPROFILE + "\.pyenv\pyenv-win\","User")

[System.Environment]::SetEnvironmentVariable('path', $env:USERPROFILE + "\.pyenv\pyenv-win\bin;" + $env:USERPROFILE + "\.pyenv\pyenv-win\shims;" + [System.Environment]::GetEnvironmentVariable('path', "User"),"User")

Abrir o CMD na pasta raiz do projeto e utilizar os comandos:

* + - Para instalar uma versão do Python → pyenv install NÚMERO\_DA\_VERSÃO
    - Para definir qual versão do Python usar → pyenv local NÚMERO\_DA\_VERSÃO
    - Para consultar as versões disponíveis → pyenv versions

O comando gera um arquivo chamado ‘.python-version’.

(sempre utilizar a última versão possível do Python para seu projeto, que dependerá das Libs que você irá utilizar).

1. Configuração geral sobre o projeto:
   * + Se não estiver com o poetry instalado, executar o comando: pip install poetry
     + Definir as configurações sobre o projeto com o comando na pasta raiz do projeto: poetry init -n
     + Abrir o arquivo criado ‘pyproject.toml’ e definir nome do projeto, versão, descrição e versão mínima do Python requerida para execução.
2. Configuração do ambiente virtual:

Para configurar o ambiente virtual na própria pasta raiz do projeto com o poetry, se nunca executou essa configuração antes, utiliza o comando: poetry config **virtualenvs.in-project** true

Criar um ambiente virtual dentro da pasta raiz do projeto com o comando: poetry install --no-root

(deverá criar uma pasta com nome ‘.venv’)

Para ativar o ambiente virtual utilizar o comando: poetry shell

Caso precise sair do ambiente virtual utilizar o comando: deactivate

1. Incluir formatador de código (<https://pep8.org/>):

Adicionar o formatador dentro do **ambiente virtual**:

poetry add **--dev** blue

poetry add **--dev** isort

(clausura **--dev** denota que apenas o desenvolvedor irá utilizar essa Lib, e não os códigos do projeto, caso queira remover utilizar poetry remove)

Para formatar o código de todos os arquivos utilizar o comando: blue .

Para organizar os imports de maneira correta utilizar o comando: isort .

1. Adicionar **TESTES**:

Incluir a Lib de testes pytest com o comando: poetry add --dev pytest

Dentro da pasta ‘tests’ incluir o arquivo ‘\_\_init\_\_.py’ , e todos os arquivos de testes que serão criados devem iniciar como ‘test\_’

Para rodar todos os testes criados utilizar o comando: pytest -v

1. Adicionar **DOCUMENTAÇÃO:**

Incluir a Lib de documentação mkdocs com o comando: poetry add --dev mkdocs

Incluir documentação com o comando: mkdocs new .

Esse comando gera o arquivo ‘mkdocs.yml’ , a pasta docs e o ‘index.md’ incluir o nome do projeto nos arquivos e atualizar a documentação com descrição, exemplos de uso e demais informações sobre o projeto.

1. Ferramenta de análise estática (checagem ou linters)

Incluir a Lib prospector com o comando: poetry add --dev prospector

Para executar utilizar o comando: prospector

Para executar validando doc strings utilizar o comando: prospector –with-tool pep257

1. Segurança de bibliotecas

Para verificar a segurança das Libs utilizadas, instalar a Lib pip-audit com o comando: poetry add --dev pip-audit

Realizar a checagem das Libs com o comando: pip-audit

1. **Automação de comandos**

Para facilitar, utilizar GNU Make e configurar todos os comandos que serão utilizados ao longo do projeto.

Criar um arquivo com nome ‘Makefile’ (sem extensão) na raiz do projeto e incluir os comandos que serão utilizados.

Para não fazer buscas por comandos no sistema operacional inserir na primeira linha do arquivo: .PHONY: install format lint test sec

Incluir outras instruções automáticas:

install:

@poetry install --no-root

format:

@isort .

@blue .

lint:

@blue . --check

@isort . --check

@prospector --with-tool pep257 --doc-warning

test:

@pytest -v

sec:

@pip-audit