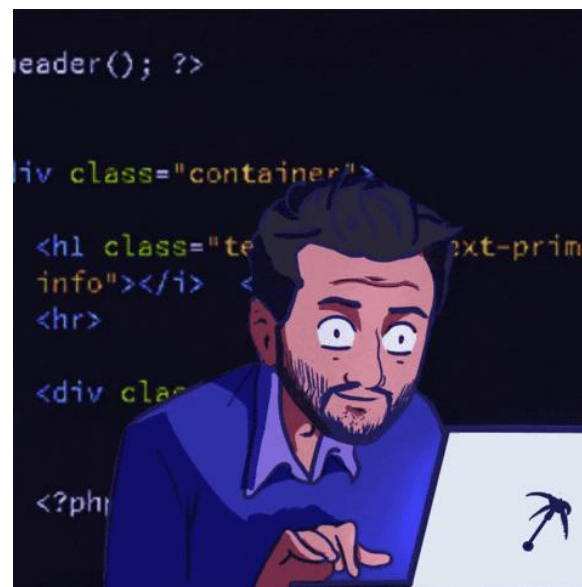
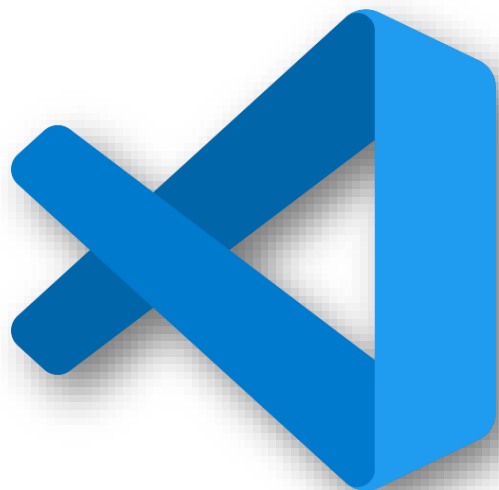




LÓGICA DE PROGRAMAÇÃO

VSCODE



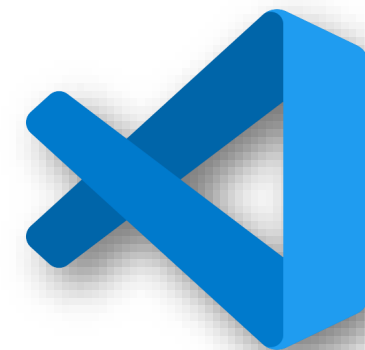
LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

VsCode

O Visual Studio Code é um editor de código-fonte desenvolvido pela Microsoft, lançado em abril de 2015. Ele se tornou rapidamente uma das principais escolhas entre os desenvolvedores, independentemente da plataforma ou linguagem de programação que utilizem.

Essa popularidade pode ser atribuída a vários fatores que destacam o VS Code em relação a outros editores de código disponíveis no mercado.



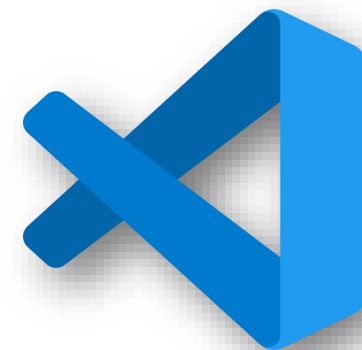
LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

VsCode

Principais atrativos:

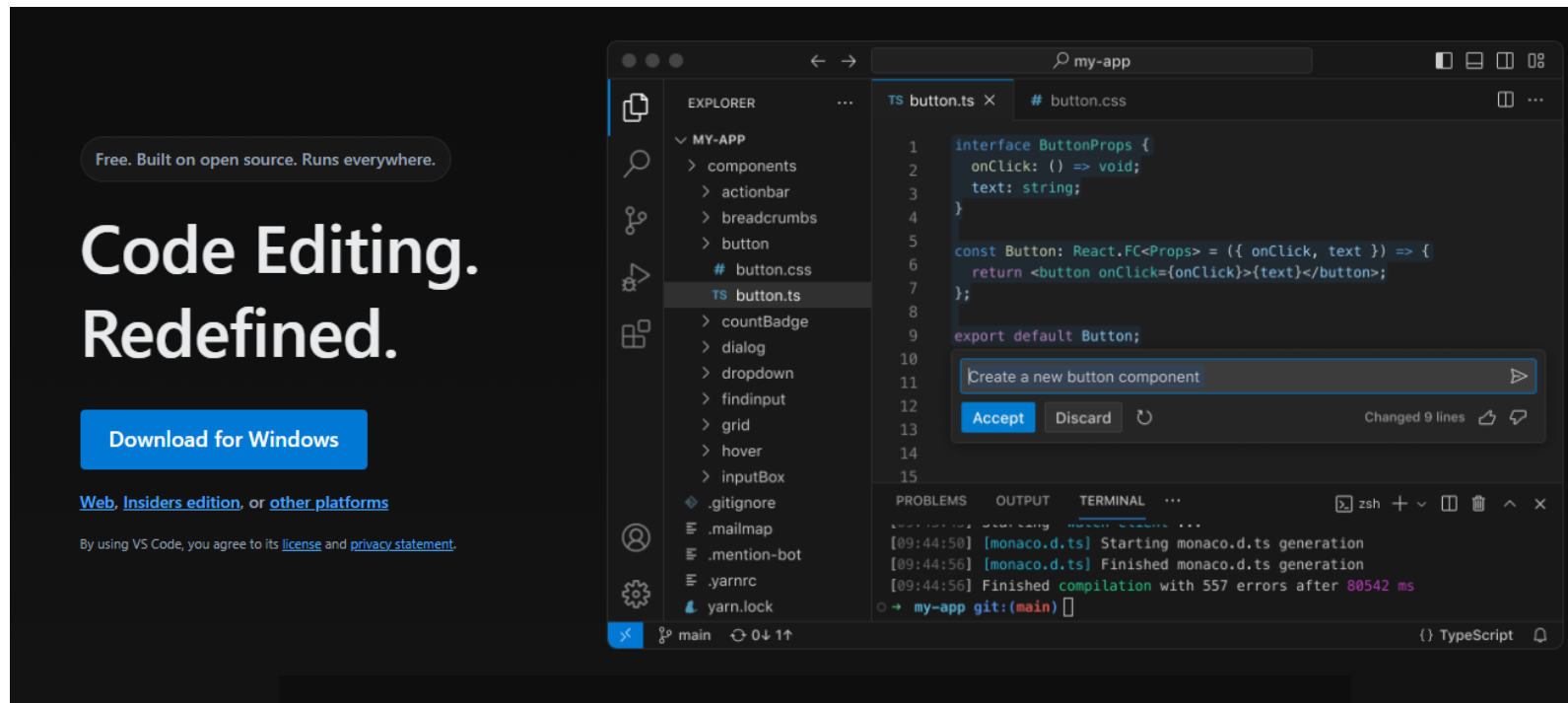
- Interface intuitiva e personalizável;
- Suporte a uma ampla variedade de linguagens de programação;
- Diversas extensões para melhorar as funcionalidades da ferramenta;
- Integração com sistemas de controle de versão (git);
- Comunidade e suporte ativos;



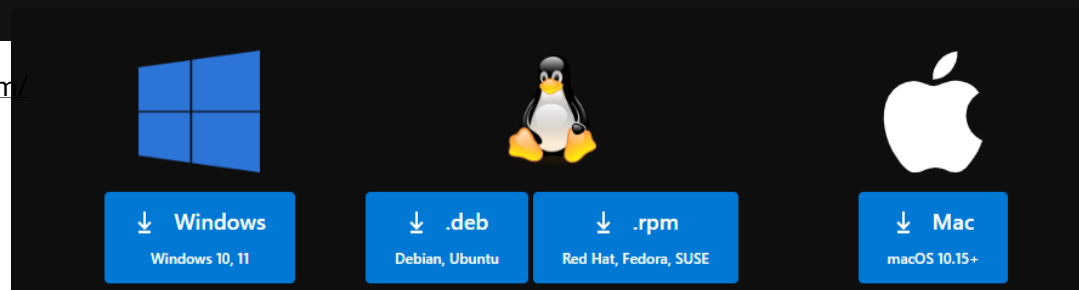
LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

VsCode



<https://code.visualstudio.com/>



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Manipulação de arquivos em Python

A habilidade de ler e manipular arquivos é uma competência essencial no desenvolvimento de software. Seja para realizar a análise de dados, configurar sistemas ou manipular informações, a capacidade de interagir com diferentes tipos de arquivos é indispensável.

Python oferece uma variedade de ferramentas e bibliotecas nativas que tornam simples o manuseio de arquivos, sejam eles de texto, CSV, JSON, entre outros formatos.



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Manipulação de arquivos em Python

Quando manipulamos arquivos em Python, os tipos de permissão são um aspecto crucial a ser considerado.

'r' (Leitura): Abre o arquivo apenas para leitura. É o modo padrão. Lança um erro se o arquivo não existir.

'w' (Escrita): Abre o arquivo para escrita. Cria um novo arquivo se não existir. Sobrescreve o conteúdo se o arquivo já existir.



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Manipulação de arquivos em Python

'a' (Anexar): Abre o arquivo para anexar conteúdo. Cria um novo arquivo se não existir. Adiciona conteúdo ao final do arquivo.

'r+' (Leitura e Escrita): Abre o arquivo para leitura e escrita. Lança um erro se o arquivo não existir.



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Manipulação de arquivos em Python

'w+' (Escrita e Leitura): Abre o arquivo para escrita e leitura. Cria um novo arquivo se não existir. Sobrescreve o conteúdo se o arquivo já existir.

'a+' (Anexar e Ler): Abre o arquivo para anexar e ler. Cria um novo arquivo se não existir.

'x' (Criação Exclusiva): Cria um novo arquivo. Lança um erro se o arquivo já existir.



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Manipulação de arquivos em Python

A instrução **with** é utilizada para garantir a finalização de recursos adquiridos.

Por exemplo: quando um arquivo é aberto, podemos utilizar **try e finally**, para fazer a execução da nossa lógica e depois fechar o arquivo

Porém alguns erros de código poderiam resultar em uma não execução da instrução de fechar o arquivo, fazendo com que os recursos ainda ficassem alocados a ele.

Por isso, foi providenciada a instrução **with**, que realiza esta mesma operação de forma simples, que garante o encerramento dos recursos alocados



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Manipulação de arquivos em Python

A instrução **with**:

```
# com try e finally
try:
    f = open("teste.txt", "w")
    f.write("Testando escrita")
finally:
    f.close()

# com with
with open("teste2.txt", "w") as f:
    f.write("Testando escrita com with")
```



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Games utilizando PyGame.

Pygame é uma biblioteca em Python destinado à criação de games.

Ele oferece recursos e capacidades que simplificam o desenvolvimento e o controle de elementos visuais, áudios, detecções de impacto, movimentações e outros aspectos comuns em jogos.

Utilizando o Pygame, é viável produzir games 2D de maneira acessível e lógica.



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

PyGame – Principais Características.

Facilidade de uso: Ideal para iniciantes em programação de jogos.

Compatibilidade: Funciona em vários sistemas operacionais (Windows, macOS, Linux).

Gráficos e animações: Suporte a manipulação de imagens e animações.

Sons e músicas: Permite adicionar trilhas sonoras e efeitos sonoros.

Eventos e controles: Gerencia entradas de teclado, mouse e joysticks.

Game loop: Controla a lógica, atualizações e eventos do jogo.

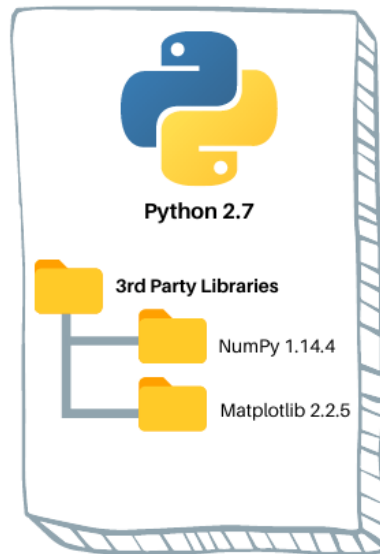


LÓGICA DE PROGRAMAÇÃO

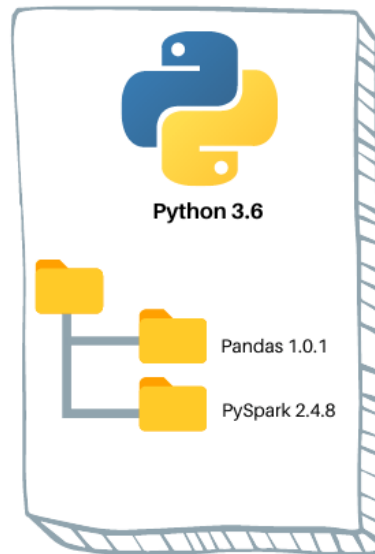
LINGUAGEM DE PROGRAMAÇÃO

Ambiente Virtual

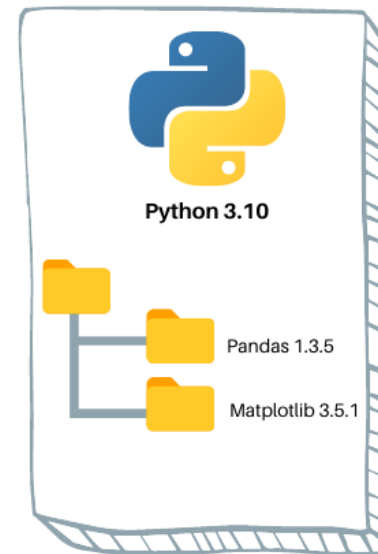
Virtual Environment 1



Virtual Environment 2



Virtual Environment 3



AMBIENTE DE DESENVOLVIMENTO

GERENCIAMENTO DE DEPENDÊNCIAS

Ambientes virtuais e Gerenciamento de dependências

Quando estamos desenvolvendo diversos projetos em Python, é comum utilizarmos diferentes versões de uma mesma biblioteca entre projetos. Por exemplo, imagine que estamos desenvolvendo o **projeto x** com a **biblioteca xyz em sua versão 1.0** e o **projeto y** com a **mesma biblioteca, porém na versão 2.0**. Como faríamos para **gerenciar estas dependências** e o sistema operacional saber qual **versão correta executar** quando estivermos em **cada projeto**?

Esta é uma tarefa um tanto quanto complexa para o SO gerenciar, podendo acarretar em **conflitos entre as versões** e causar muita dor de cabeça. Sendo assim, o mais comum é utilizarmos **diferentes ambientes virtuais**, chamados de **virtualenvs**, um para cada projeto.

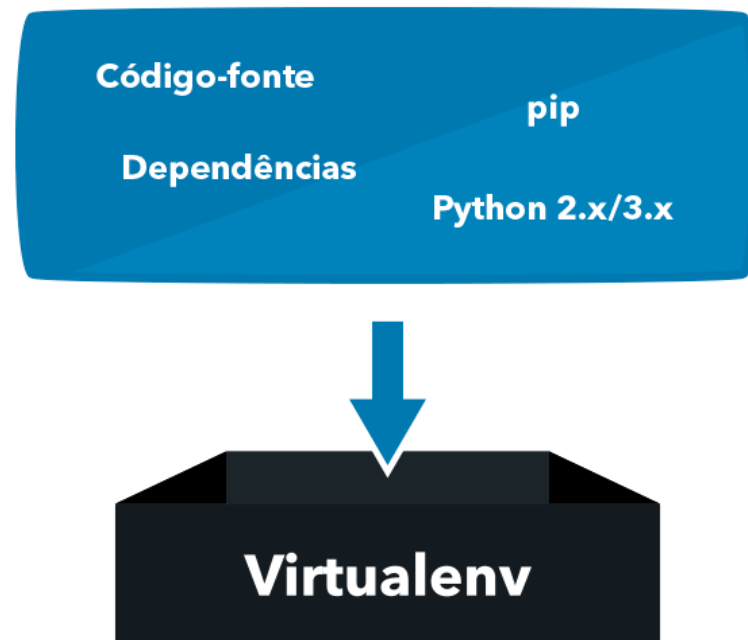


AMBIENTE DE DESENVOLVIMENTO

GERENCIAMENTO DE DEPENDÊNCIAS

O que é um virtualenv?

Basicamente, um ambiente virtual empacota todas as dependências que um projeto precisa e armazena em um diretório, fazendo com que nenhum pacote seja instalado diretamente no sistema operacional. Sendo assim, cada projeto pode possuir seu próprio ambiente e, conseqüentemente, suas bibliotecas em versões específicas.



AMBIENTE DE DESENVOLVIMENTO

GERENCIAMENTO DE DEPENDÊNCIAS

Como funciona uma virtualenv?

O funcionamento de uma **virtualenv** é bem simples. Basicamente, uma cópia dos diretórios necessários para que um programa Python seja executado é criada, incluindo:

- ✓ PIP (Gerenciador de pacotes)
- ✓ A versão do Python utilizada no projeto (2.x ou 3.x)
- ✓ Dependências instaladas com o pip (armazenadas no diretório site-packages)
- ✓ Seu código fonte
- ✓ Bibliotecas comuns do Python.

Isso faz com que as dependências sejam sempre instaladas em uma virtualenv específica, não mais no Sistema Operacional. A partir daí, cada projeto executa seu código-fonte utilizando sua virtualenv própria.



AMBIENTE DE DESENVOLVIMENTO

GERENCIAMENTO DE DEPENDÊNCIAS

Instalando a virtualenv

A instalação de uma **virtualenv** é feita utilizando o **pip**, gerenciador de pacotes do Python. Após instalar o **pip**, utilizamos o comando abaixo para instalar o pacote **virtualenv** em nosso computador:

```
pip install virtualenv
```




AMBIENTE DE DESENVOLVIMENTO

GERENCIAMENTO DE DEPENDÊNCIAS

Criando uma nova virtualenv

O processo de criação de uma **virtualenv** é bastante simples e pode ser feito utilizando um único comando, como podemos ver abaixo:



```
python -m virtualenv venv  
ou  
python -m venv venv
```



O mais comum é criar a **virtualenv** na raiz do projeto que ela irá pertencer. Isso permite uma organização maior das virtualenvs que possuímos em nosso computador.



AMBIENTE DE DESENVOLVIMENTO

GERENCIAMENTO DE DEPENDÊNCIAS

Ativando uma virtualenv

Após criar uma **virtualenv**, precisamos ativá-la para que possamos instalar os pacotes necessários do projeto. Para isso, utilizamos o seguinte comando:



```
source nome_da_virtualenv/bin/activate # Linux ou MacOS
```

```
nome_da_virtual_env\Script\Activate # Windows
```



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Framework?

Um **framework** é uma estrutura de software predefinida que fornece um conjunto de ferramentas, bibliotecas e diretrizes para facilitar o desenvolvimento de aplicações.

Ele oferece uma base sobre a qual os desenvolvedores podem construir seu código, seguindo padrões e boas práticas, permitindo que foquem nas funcionalidades específicas de sua aplicação, em vez de se preocupar com detalhes de baixo nível ou infraestrutura.



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Flask.

Flask é um microframework web para Python. Ele é leve, flexível e fácil de usar, projetado para criar aplicações web rapidamente.

Flask fornece o essencial para desenvolvimento web, como roteamento de URLs, manipulação de requisições e respostas, e suporte a templates, mas permite que você escolha e integre outras bibliotecas conforme necessário.



Flask



LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Vantagens do Flask.

Leve e Simples: É um **microframework**, com estrutura mínima, ideal para projetos pequenos e rápidos.

Flexível e Modular: Permite adicionar apenas as funcionalidades que você precisa, sem sobrecarregar o projeto.

Extensível: Pode ser facilmente expandido com diversas bibliotecas e extensões.

Facilidade de Uso: Simples de aprender, com código claro e organizado.

Comunidade Ativa: Tem uma grande comunidade e excelente documentação, facilitando o suporte e o desenvolvimento.



AMBIENTE DE DESENVOLVIMENTO

GERENCIAMENTO DE DEPENDÊNCIAS

Instalando o Flask

A instalação do **Flask** é feita utilizando o **pip**, gerenciador de pacotes do Python. Utilizamos o comando abaixo para instalar o **Flask** em nosso computador:

```
pip install flask
```



AMBIENTE DE DESENVOLVIMENTO

GERENCIAMENTO DE DEPENDÊNCIAS

Estrutura base Flask

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def homepage():
    return "Meu site no Flask"

if __name__ == "__main__":
    app.run()
```



VAMOS PRATICAR.

