Objetivos da Aula:

- Fundamentos sobre a linguagem python
- Variáveis e tipos de dados

Instalação do python e Configuração de Variáveis de Ambiente

Instalação do **python** no Windows

Visite o site oficial do **python** (**https://www.python.org/downloads/**).

Abrir o prompt de comando e digitar:

```
python -V
```

se aparecer a versão está tudo instalado

O propósito inicial do Python

Criado como uma linguagem de propósito geral.

Popularizado em **automação, ciência de dados, machine learning** e desenvolvimento de software.

Python precisou de frameworks e servidores adicionais para rodar aplicações web.

Como Python é usado no desenvolvimento web

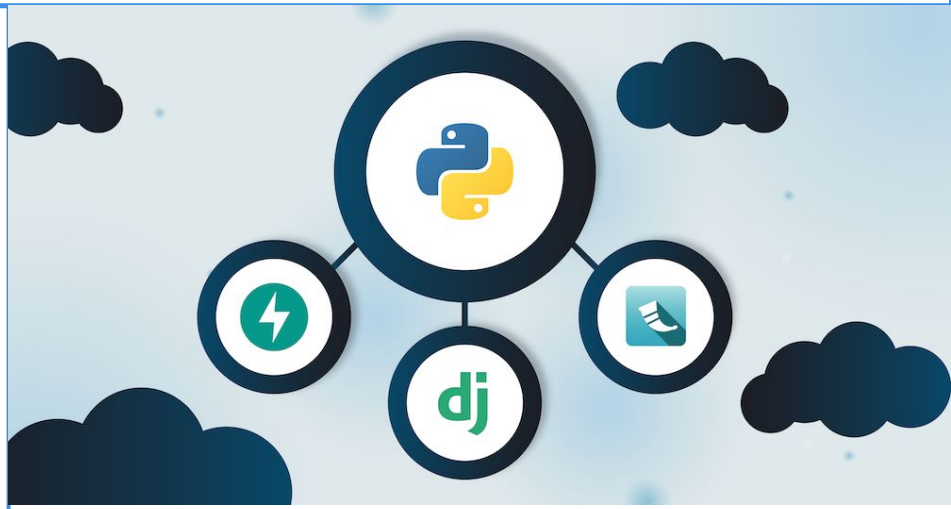
Frameworks populares:

Flask, Django, FastAPI.

Servidores:

uvicorn, uWSGI, ASGI.

APIs REST e integração com bancos de dados.



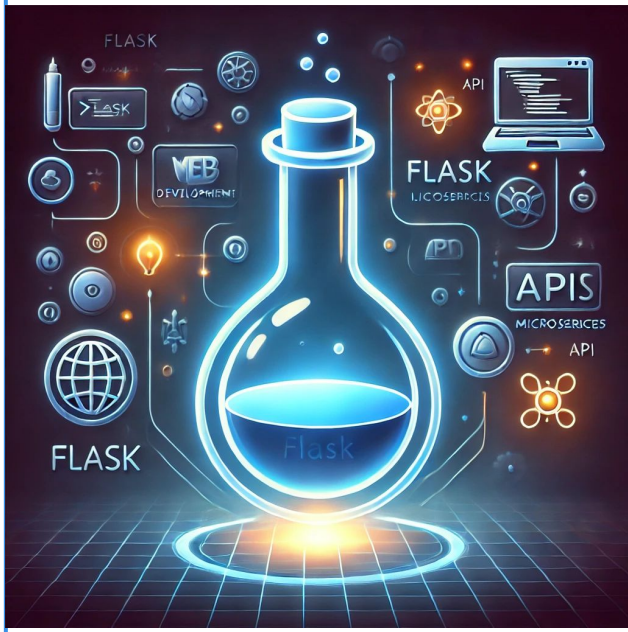
Python e o Framework Django

Django é um framework web Python completo que fornece um conjunto abrangente de ferramentas e recursos para construir aplicativos web complexos e escaláveis. Ele vem com tudo o que os desenvolvedores precisam para começar, incluindo autenticação, uma interface administrativa e um framework leve para tipos de conteúdo, bem como muitas outras ferramentas.

Python e o Framework Flask

Flask é um microframework Python leve e flexível que se concentra na simplicidade; é conhecido por seu design minimalista e sintaxe direta, o que o torna uma excelente escolha para projetos de pequeno e médio porte e uma ótima ferramenta para iniciantes, graças à pouca quantidade de código clichê que possui.

O framework permite que os desenvolvedores comecem a construir aplicativos da web rapidamente, sem uma curva de aprendizado íngreme, o que fornece muita liberdade em termos de personalização, permitindo que os desenvolvedores escolhem apenas os componentes de que precisam para seus requisitos específicos de projeto.



Python e o Framework FastAPI

FastAPI é um framework web de alto desempenho que está rapidamente ganhando popularidade para construir APIs. Ele é construído sobre Starlette, um framework web assíncrono, e usa dicas de tipo Python para validação automática de dados. Ao utilizar programação assíncrona, ele fornece desempenho excepcional e é uma ótima escolha para construir APIs e microsserviços.

Sem soar muito previsível, uma das vantagens significativas do FastAPI é sua velocidade. Ao fazer uso de programação assíncrona, o FastAPI pode lidar com um grande número de solicitações simultâneas de forma eficiente. Isso o torna perfeito para aplicativos de alto tráfego que exigem processamento de dados em tempo real, e uma vantagem é que ele fornece documentação automática usando Swagger UI, economizando tempo e esforço na documentação de APIs.

Introdução ao Python e Estrutura do Código

O que é Python? Linguagem interpretada, tipagem dinâmica e alto nível.
Como rodar Python no terminal (`python` ou `python3`) e em arquivos (`python script.py`).

Características fundamentais:

- Sensível a indentação (uso obrigatório de espaços ou tabulação).
- Scripts podem ser executados diretamente no terminal.

Iniciando com Python

podemos usar o Colab do google ou com IDE de nossa preferência como VSCode, o python é uma linguagem de script logo não é necessário compilação explicita como java ou c# e outras, você escreve e roda o script.

Algumas regras fundamentais.

1. **Indentação de código.**
   espaço inicial quando houver bloco
   if xxxx : “dois pontos abre um bloco”
2. **chamar o compilador antes do nome do arquivo** a ser executado

```
python nome.py
```

Isolando o ambiente

Configurando um ambiente virtual

```
python -m venv venv
```

Ativando:

- Windows: `venv\Scripts\activate`
- Linux/macOS: `source venv/bin/activate`

instalar dependências

```
pip install nome_dependencia
```

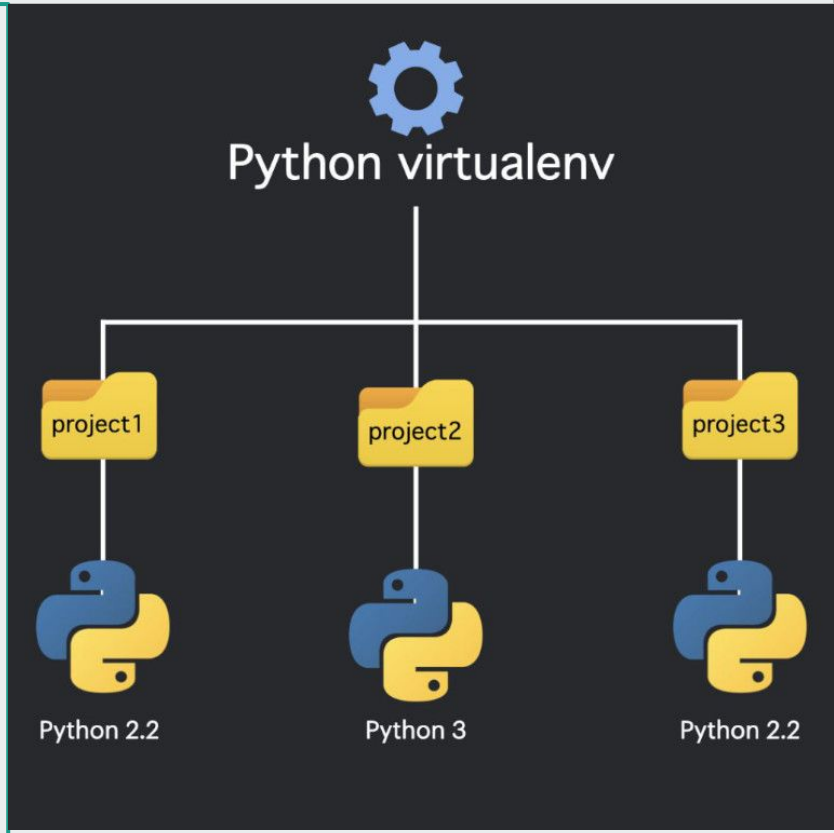
Congelar dependências

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

Instalando pacotes a partir do `requirements.txt`:

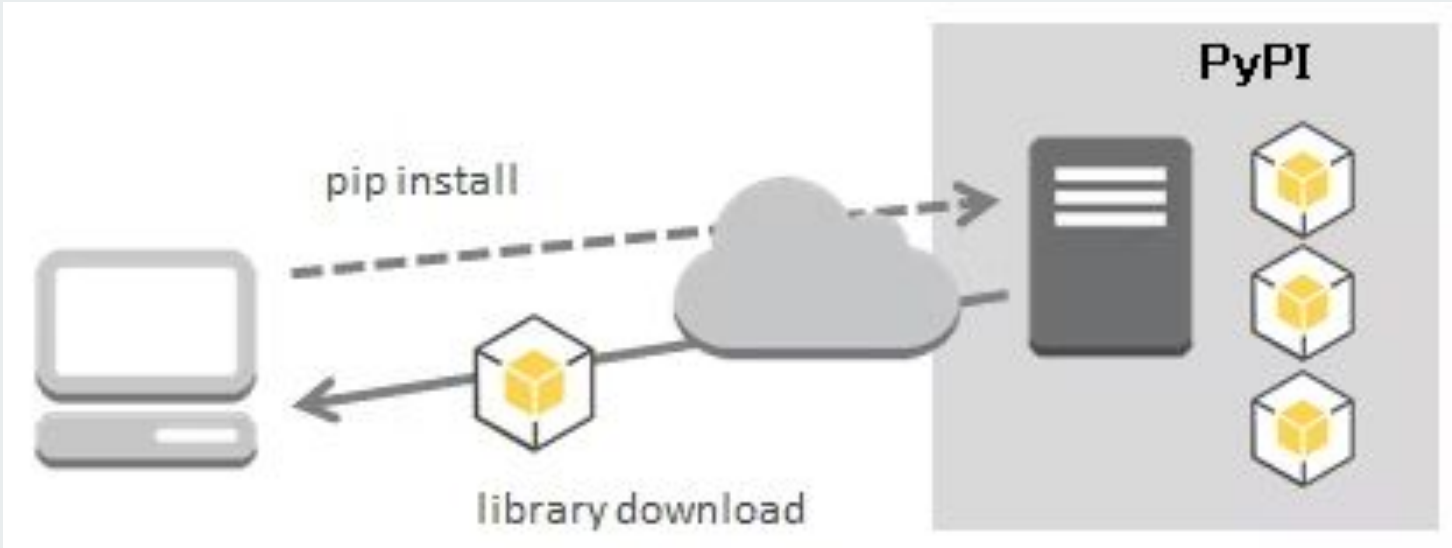
```
pip install -r requirements.txt
```

Um ambiente virtual é uma ferramenta que ajuda a manter as dependências de diferentes projetos Python separadas, garantindo que bibliotecas específicas não conflitem umas com as outras. Isso é especialmente útil quando você trabalha em múltiplos projetos com diferentes requisitos de bibliotecas.



Pip é o instalador de pacotes para Python. Ele permite instalar e gerenciar bibliotecas e dependências que não são distribuídas com o Python.

O Python Package Index (PyPI) é um repositório online que abriga milhares de pacotes Python prontos para uso. **(https://pypi.org/)**



Criar um Arquivo de Requisitos:

pip freeze > requirements.txt

Instalar do Arquivo de Requisitos:

pip install -r requirements.txt

Instalar uma Biblioteca:

pip install nome_da_biblioteca

Listar Bibliotecas Instaladas:

pip list

Fundamentos Python - variáveis

Identificadores
Um identificador é o nome dado a uma variável, função, classe ou módulo
Identificadores pode ter um ou mais caracteres e devem obedecer às seguintes regras:

- Deve **começar** com uma letra (a-z ou A-Z) ou sublinhado (_)
- Pode **conter** letras minúsculas (a-z), maiúsculas (A-Z), números (0-9) ou sublinhado (_)

Não pode ser utilizado qualquer uma das palavras-chaves reservados da linguagem:

```
and continue finally is raise as
def for lambda return assert
del from
None True async elif global non local
try await else if not while break except
import or with class False in pass yield
```

Nem caracteres especiais como @, #, $, % e nem espaço

Variáveis e Tipos de Dados: Tipagem em Linguagens de Programação

Tipagem Fraca vs. Tipagem Forte

- **Linguagens de Tipagem Fraca**: Permitem operações entre diferentes tipos de dados sem exigir conversão explícita. Isso pode gerar resultados inesperados.
- **Linguagens de Tipagem Forte**: Exigem que os tipos sejam compatíveis ou convertidos explicitamente antes de operações.

- **JavaScript (tipagem fraca)**

```
console.log("5" + 3);

// Resultado: "53" (concatenação)
```

- **Python (tipagem forte)**:

```
print("5" + 3)

# Erro! Não pode somar string com inteiro

print(int("5") + 3) # Resultado: 8
```

Variáveis e Tipos de Dados: Tipagem Estática vs. Tipagem Dinâmica

- **Linguagens de Tipagem Estática**: O tipo da variável é definido no momento da declaração e não pode mudar durante a execução.
- **Linguagens de Tipagem Dinâmica**: O tipo da variável pode mudar durante a execução.

- **C (tipagem estática)**

int **idade** = 25;

idade = "vinte e cinco"; // Erro! Não pode mudar o tipo

- **Python (tipagem dinâmica)**

x = 10 # Inteiro

x = "dez" # Agora é string

print(x) # "dez"

Python - fortemente tipada e dinâmica

Dinamicamente tipada, isso significa que o compilador irá manipular uma variável de acordo com o valor que está sendo definido para a mesma.
Fortemente tipada, as conversões de tipos são feitas Explicitamente. O Python permite que variáveis mudem de tipo se não foi declarada de forma explícita.

Variáveis em Python (forte e Dinâmica)

```python
idade_int = 25

idade_string = "25"

print(type(idade_int) )
print(type(idade_string))

soma = idade_int + int(idade_string)

print(soma)
```

Fundamentos Python - variáveis

```
nome = "Alice"
idade = 25
altura = 1.65
ativo = True
```

Principais tipos de dados:

- `int`: números inteiros
- `float`: números decimais
- `str`: strings (texto)
- `bool`: valores booleanos (`True`, `False`)

Fundamentos Python - variáveis compostas

```
lista = ["apple", "banana",
"cherry"]

tupla = ("apple", "banana",
"cherry")

set = {"apple", "banana",
"cherry"}

dicionario = {
  "nome": "maria",
  "sexo": "feminino",
  "nascido": 1990
}
```

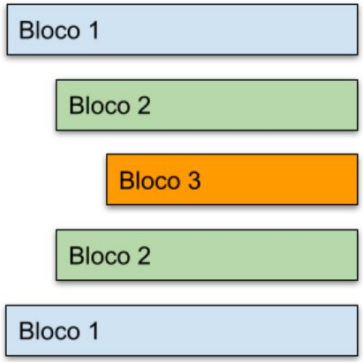
Principais tipos de dados:

- `list`, `tuple`, `set`, `dict`: coleções

Indentação

Em Python, os programas são estruturados através de indentação.

Em outras linguagens de programação, a indentação é opcional, mas em Python ela é um **requisito obrigatório**.




```
valor = 12
if valor > 0:
    print("Valor positivo")
    if valor % 2 == 0:
        print("Valor par")
else:
    print(“valor negativo”)
print("Fim do programa")
```

Comentários

Comentários são textos inseridos no programa, que não são executados e ajudam ao programador ou outros programadores a entender, manter e depurar o programa. Python utiliza dois tipos de comentários de: Linhas simples e Múltiplas Linhas

Comentários de Linha Simples

```
# Comentário em uma linha

b = 7
```

Comentários de Múltiplas Linhas

```
'''

Três aspas (simples ou duplas)

delimita um texto que são será

executado.

'''
```

Comando de saída

O comando de saída print permite ao programa apresentar texto no terminal.

O print irá imprimir tudo como string e qualquer coisa que não seja string será convertido.

Basicamente há duas formas de formart strings dentro do comando print

- str.format()
- f-strings

```
a = 4
b = 3
print("Valor A: {0}, valor de B: {1}".format(a, b))
```

```
a = 4
b = 3
print(f"Valor A: {a}, valor de B: {b}")
```

Comando de Entrada

Em Python é utilizado o comando **input()** para receber dados que o usuário digitar

Para converter um float ou uma string para um valor inteiro, utilizamos a função **int()**.

```
valorA = input("Digite o valor A: ")


valorA = int(input("Digite o valor A: "))
```

Comando de Entrada

Em Python é utilizado o comando **`input()`** para receber dados que o usuário digitar

Para converter um float ou uma string para um valor inteiro, utilizamos a função **`int()`**.

`float()` Semelhante a função `int()`, essa função converte para números decimais.

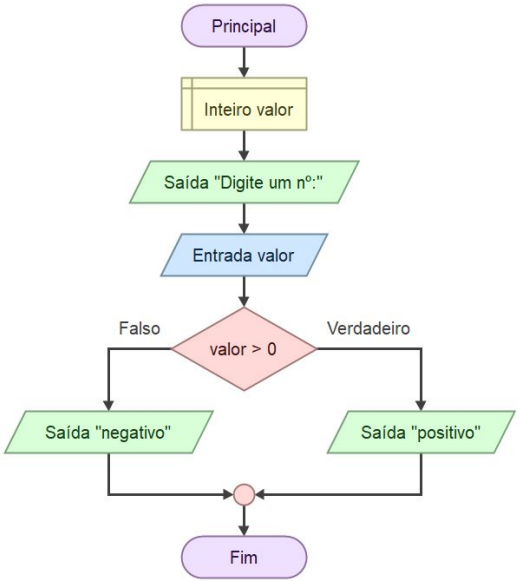
Para os casos em que é necessário a conversão de números para string, utilizamos a função **`str()`**

```
valorA = input("Digite o valor A: ")


valorA = int(input("Digite o valor A: "))
```

Comandos de decisão

A estrutura mais básica de decisão é formada apenas pelo comando `if`




```
valor = int(input('Digite um nº: '))
if valor > 0:
   print('positivo')
else:
   print('negativo')
```

Operadores Aritméticos

Operador	Significado	Tipos de operandos	Tipos de resultado
↑, ^, **	Exponencial	Inteiro ou real	Inteiro ou real
+	Soma	Inteiro ou real	Inteiro ou real
–	Resto	Inteiro ou real	Inteiro ou real
*	Multiplicação	Inteiro ou real	Inteiro ou real
/	Divisão	Real	Real

Operadores de Comparação

A estrutura mais básica de decisão é formada apenas pelo comando `if`

```
print(5<3) # menor que
print(5>3) # maior que
print(5==3) # igual a
print(5!=3) # diferente de
print(5>=3) # maior ou igual a
print(5<=3) # menor ou igual a
```

```
ython .\main.py
False
True
False
True
True
False
(venv) PS C:\Apache24
```

Laço de repetição FOR

A sintaxe do comando for é

for variável_da_interação in sequência:

comando(s)

O comando for do Python difere do comando em C em seu princípio de funcionamento.

O for iterage numa lista de valores dada na sequência e carrega cada um desses valores para a variável **variável_da_interação**, continuando esse processo enquanto houver valores na sequência.

```
for valor in [2, 3, 5, 8]:
    print(f"{valor}")
----------------------------
for valor in range(0, 10):
    print(f"{valor}")
----------------------------
for valor in range(0, 10, 2):
    print(f"{valor}")
```

Laço de repetição While

O `while` irá repetir a lista de `comando(s)` enquanto a `expressão` for verdadeira

O laço de repetição `while` possui a seguinte forma de construção:

```
while expressão:
    comando(s)
```

```
i = 0
while i < 10:
    print(f'Valor de i: {i}')
    i += 1
 ---------------------------
i = True
x = 0
while i:
    print(f'Valor de x: {x}')
    if x >=10:
        i = False
    x += 1
```

Declarações Continue e Break

As declarações `continue` e `break` podem alterar bastante o funcionamento do laço de repetição.

- `continue` **faz o fluxo pular ao encontrar a condição**
- `break` **faz o fluxo do loop parar**

```
for valor in range(-3, 4):
    if valor == 0:
        continue
    print(f"{valor}")

----------------------------

for valor in range(-3, 4):
    if valor == 0:
        break
    print(f"{valor}")
```

Referências

BEAZLEY D.; JONES, B.K. Python Cookbook: Receitas para dominar Python. 3 ed. São Paulo: Novatec, 2019.

CRUZ, FJ. Python: escreva seus primeiro programas. 1.ed. São Paulo: Casa do Código, 2021.

PEREIRA, E, DOUGLAS MICHAEL. Trilhas Python: Programação multiparadigma e desenvolvimento web com Flask. 1.ed. São Paulo: Casa do Código, 2020.