Aplicações em Python

Senai Londrina: Pedro Capelari

December 3, 2024

Contents

1 História e Evolução do Python

Python foi criado por Guido van Rossum e lançado em 1991. A linguagem foi desenvolvida com o objetivo de ser simples e de fácil leitura, facilitando o aprendizado e a utilização por parte de programadores iniciantes e experientes. Ao longo dos anos, Python evoluiu significativamente, com a migração das versões 2.x para 3.x sendo um marco importante. A versão 3.x trouxe melhorias e novas funcionalidades, mas também quebrou a compatibilidade com código escrito para a versão 2.x.

2 Características Principais

Python é conhecido por diversas características que o tornam uma escolha popular entre programadores:

- **Simplicidade e Legibilidade**: A sintaxe do Python é clara e concisa, utilizando indentação para definir blocos de código, o que torna a leitura e escrita de código mais fácil e intuitiva.
- **Linguagem Interpretada**: Python é interpretado, o que significa que o código é executado linha a linha, sem a necessidade de compilação prévia. Isso facilita a depuração e o desenvolvimento iterativo.
- **Suporte a Múltiplos Paradigmas**: Python suporta programação procedural, orientada a objetos e funcional, oferecendo flexibilidade para os desenvolvedores escolherem o paradigma que melhor se adapta às suas necessidades.

3 Áreas de Aplicação

Python é amplamente utilizado em diversas áreas, incluindo:

- **Desenvolvimento Web**: Frameworks como Django e Flask permitem a criação de aplicações web robustas e escaláveis.
- **Ciência de Dados e Machine Learning**: Bibliotecas como Pandas, NumPy e Scikit-learn facilitam a análise de dados e a implementação de algoritmos de machine learning.
- **Automação e Scripts**: Python é amplamente utilizado para automatizar tarefas repetitivas e scripts de manutenção, utilizando bibliotecas como `os`, `sys` e `subprocess`.
- **Desenvolvimento de Jogos**: Bibliotecas como Pygame permitem a criação de jogos 2D.

4 Atividade: Discussão em Grupo

Estimule os alunos a compartilharem suas experiências com Python e a discutirem casos de uso que conhecem ou já experimentaram, destacando como a linguagem ajudou a resolver problemas ou a facilitar o desenvolvimento de projetos.

5 Introdução ao Flask

Flask é um microframework para desenvolvimento web. Comparado ao Django, que é um framework completo, o Flask é mais leve e flexível, ideal para projetos menores ou que necessitam de maior personalização.

5.1 Vantagens do Flask

- **Simplicidade**: Fácil de aprender e usar, com uma curva de aprendizado baixa.
- **Flexibilidade**: Permite adicionar apenas os componentes necessários, evitando uma estrutura monolítica.
- **Extensibilidade**: Suporte a diversas extensões que ampliam suas funcionalidades.

5.2 Estrutura de um Projeto Flask

Para começar com Flask, é recomendado criar um ambiente virtual para isolar as dependências do projeto:

```
$ python3 -m venv venv
$ source venv/bin/activate  # No Windows, use ‘venv\Scripts\activate‘
$ pip install Flask
```

Estrutura básica de diretórios:

```
project/

 app.py
 templates/
    index.html
 static/
    style.css
 venv/
```

6 Criação de uma Aplicação Web Simples

Para criar uma aplicação básica com Flask:

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "Hello, World!"

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

6.1 Adição de Funcionalidades com Templates

Flask utiliza o motor de templates Jinja2 para gerar HTML dinâmico. Exemplo de uso de templates:

```
# app.py
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")

if __name__ == "__main__":
    app.run(debug=True)

# templates/index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
</head>
<body>
    <h1>Welcome to Flask!</h1>
</body>
</html>
```

7 Atividade: Implementação de Funcionalidades Extra

Os alunos devem adicionar novas rotas e funcionalidades, como um formulário de contato, utilizando templates para renderizar páginas dinâmicas.

8 Manipulação de Arquivos com os e shutil

Python oferece bibliotecas poderosas para manipulação de arquivos e diretórios.

8.1 Usando a Biblioteca os

A biblioteca `os` permite interagir com o sistema operacional, incluindo operações de arquivos e diretórios:

```
import os

# Listar arquivos em um diret rio
arquivos = os.listdir('/caminho/do/diretorio')
print(arquivos)

# Criar um novo diret rio
os.mkdir('/caminho/do/novo_diretorio')

# Renomear um arquivo
os.rename('/caminho/arquivo_antigo.txt', '/caminho/arquivo_novo.txt')
```

8.2 Usando a Biblioteca shutil

A biblioteca `shutil` fornece operações de alto nível em arquivos e coleções de arquivos:

```
import shutil

# Copiar um arquivo
shutil.copy('/caminho/origem.txt', '/caminho/destino.txt')

# Mover um arquivo
shutil.move('/caminho/origem.txt', '/caminho/novo_diretorio/destino.txt')

# Deletar um arquivo ou diret rio
shutil.rmtree('/caminho/diretorio_para_deletar')
```

9 Criação de Scripts de Automação

Para organizar arquivos por extensão:

```
import os
import shutil

def organizar_pastas(diretorio):
    for filename in os.listdir(diretorio):
        if not os.path.isdir(os.path.join(diretorio, filename)):
            ext = filename.split('.')[-1]
            pasta = os.path.join(diretorio, ext)
            if not os.path.exists(pasta):
                os.makedirs(pasta)
            shutil.move(os.path.join(diretorio, filename), os.path.join(pasta, f

organizar_pastas('/caminho/do/diretorio')
```

10 Atividade: Desenvolvimento de Scripts

Os alunos devem criar scripts para organizar arquivos e melhorar os scripts para lidar com subpastas e arquivos duplicados.

11 Bibliotecas Essenciais

- **Pandas**: Biblioteca poderosa para manipulação e análise de dados, facilitando operações como filtragem, agregação e junção de dados.
- **NumPy**: Biblioteca fundamental para computação científica com Python, oferecendo suporte a arrays multidimensionais e operações matemáticas.
- **Matplotlib**: Biblioteca de plotagem para criar visualizações estáticas, animadas e interativas em Python.

12 Análise e Visualização de Dados

Para carregar e explorar um dataset:

```
import pandas as pd
import matplotlib.pyplot as plt

# Carregar um dataset CSV
df = pd.read_csv('caminho/do/arquivo.csv')

# Visualizar as primeiras linhas do dataframe
```

```
print(df.head())

# Agrupar dados por uma coluna e calcular a m dia
df_grouped = df.groupby('coluna').mean()

# Plotar os dados agrupados
df_grouped.plot(kind='bar')

# Exibir o gr fico
plt.show()
```

13 Atividade: Análise de Dados com Pandas

Os alunos devem carregar um dataset, realizar análises e criar gráficos. Eles serão desafiados a explorar diferentes aspectos dos dados e criar visualizações adicionais, utilizando bibliotecas como Seaborn para gráficos mais complexos.

14 Discussão sobre Aplicações Avançadas de Python

Nesta seção, discutiremos algumas aplicações avançadas de Python:

- **Machine Learning**: Introdução a Scikit-learn para criação e treinamento de modelos de aprendizado de máquina.
- **Desenvolvimento de APIs**: Uso do FastAPI para construção de APIs rápidas e eficientes.
- **Automação Web**: Utilização do Selenium para automação de interações web, como teste de interfaces e scraping de dados.

References

[1] SILVA, Leonardo Soares e; FORTES, Gabriel. *Aprenda a programar com python: descomplicando o desenvolvimento de software*. São Paulo, SP: Casa do Código, 2022. E-book. Disponível em: `https://plataforma.bvirtual.com.br`. Acesso em: 03 dez. 2024.

[2] GUILHON, André et al. (org.). *Jornada Python: uma jornada imersiva na aplicabilidade de uma das mais poderosas linguagens de programação do mundo*. Rio de Janeiro, RJ: Brasport, 2022. E-book. Disponível em: `https://plataforma.bvirtual.com.br`. Acesso em: 03 dez. 2024.