Relatório 2 - Prática: Linguagem de Programação Python (I)

Beatriz Almeida Felício

Descrição da atividade

Esse relatório aborda uma série de conceitos e práticas essenciais para a programação em Python, abordando desde noções básicas, como variáveis e operadores, até recursos mais avançados, incluindo funções, estruturas de dados e programação orientada a objetos (POO). Esses conceitos foram apresentados com exemplos práticos nas referências passadas para o estudo e a seguir mostro a compreensão que tive acerca do funcionamento da linguagem Python.

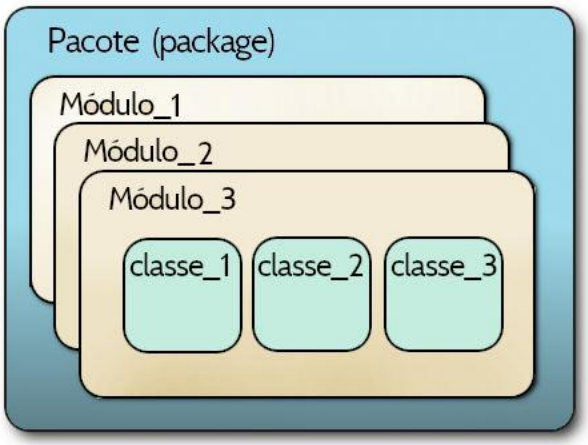
Conteúdos Abordados:

1. Conceitos Iniciais e Execução de Scripts

Python é uma linguagem interpretada e o uso do *shebang* (#!) no início de um script indica ao sistema qual interpretador deve processar o arquivo. Por exemplo, #!python3 especifica o interpretador do Python 3. Esse comando é importante para garantir a execução correta dos scripts, especialmente em sistemas Unix.

2. Módulos e Pacotes

Em Python, módulos são arquivos (arquivo.py) que contêm definições e implementações de classes, funções, variáveis, etc., que podem ser acessados por outros arquivos usando o comando import. Pacotes são coleções de módulos organizados em diretórios, permitindo uma estruturação eficiente do código em projetos de maior escala.



3. Variáveis Built-in e Definição de Variáveis

A linguagem fornece diversas funções embutidas (*built-ins*), como print(), len() e type(), que são utilitárias para manipular variáveis e estruturas de dados. Em Python, as variáveis são declaradas e inicializadas sem a necessidade de especificar seu tipo, que é determinado automaticamente e, embora a linguagem não tenha suporte para constantes, é uma convenção nomear variáveis constantes em maiúsculas.

4. Tipos Básicos

Python suporta tipos básicos como:

- int: Números inteiros.
- float: Números decimais.
- str: Cadeias de texto.
- bool: Valores booleanos (True ou False).

5. Estruturas de Dados

- **Listas**: Utilizam colchetes [], suportam valores duplicados e podem ser modificadas. Funções úteis incluem len() para o tamanho e insert() para inserção de elementos.
- **Tuplas**: Imutáveis, usam parênteses (), e aceitam valores repetidos. Mesmo com um único elemento, a presença de uma vírgula é necessária para definir uma tupla.
- **Conjuntos**: Criados com chaves {}, não permitem elementos duplicados e não são indexados. São usados para armazenar elementos únicos.
- **Dicionários**: Estruturas de mapeamento com pares chave: valor, acessados por meio de suas chaves.

6. Operadores

Python suporta operadores de várias categorias:

- **Unários**: Operam em um único operando, como -.
- **Binários**: Operam em dois operandos, como +, -, *, /.
- **Ternário**: Em Python, um operador ternário é expresso com if e else em uma única linha (valor1 if condicao else valor2).

7. Estruturas de Controle

Blocos de código em Python são definidos pela tabulação (indentação). Estruturas comuns incluem:

- **If/else**: Estruturas condicionais.
- **For**: Estrutura de repetição com quantidade determinada.
- **While**: Estrutura de repetição indeterminada.

8. Funções em Python

As funções em Python são blocos de código reutilizáveis que permitem a organização de instruções de maneira modular e estruturada. Elas recebem entradas, processam dados e retornam saídas. Têm-se flexibilidade na definição de funções, permitindo a passagem de parâmetros de diversas formas, incluindo o uso de *args e **kwargs para manipulação de argumentos variáveis.

- Funções com Argumentos Variáveis (*args e **kwargs)

A linguagem permite que as funções aceitem uma quantidade variável de argumentos com o uso de *args e **kwargs. Esse recurso é útil quando a função precisa lidar com um número indeterminado de argumentos ou quando os argumentos podem ser opcionais.

- ***args**: É utilizado para receber uma lista de argumentos de comprimento variável sem nome (ou seja, sem palavra-chave associada). *args permite que a função aceite múltiplos valores, encapsulando-os como uma tupla. Os argumentos não nomeados (*args) sempre devem vir antes dos argumentos nomeados (**kwargs). Um exemplo prático é uma função de soma que aceita qualquer quantidade de números:

- ****kwargs**: Permite o uso de argumentos nomeados (com palavra-chave), que serão armazenados como um dicionário dentro da função. Cada argumento é representado por uma chave-valor no dicionário **kwargs, o que permite acessar os valores com base em seus nomes. Esse recurso é útil quando se deseja passar vários argumentos opcionais para uma função..

- Usando *args e **kwargs Juntos

Quando ambos os argumentos são usados em uma função, ***args** deve vir antes de ****kwargs** na definição da função. Dessa forma, a função pode aceitar uma lista de argumentos variáveis sem nome e, ao mesmo tempo, parâmetros nomeados.

- Funções Lambda

É possível a criação de funções anônimas de uma linha usando lambda. Essas funções são frequentemente usadas em funções como map(), filter() e reduce(), para operações rápidas sem a necessidade de declarar uma função completa.

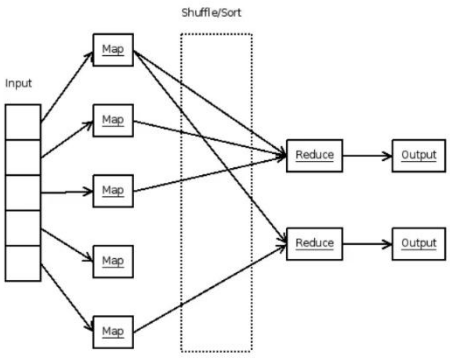
9. Programação Funcional

Na programação funcional em Python, funções como map(), filter() e reduce() permitem realizar operações em coleções de forma eficiente e concisa:

- **map()** aplica uma função a cada elemento de uma lista, criando uma nova lista com os resultados.

- **filter()** filtra elementos de uma lista com base em uma condição booleana, retornando apenas os que atendem ao critério especificado.

- **reduce()** reduz uma lista a um único valor ao aplicar uma função cumulativa aos elementos.



Essas funções tornam o código mais direto, especialmente em operações complexas, melhorando a legibilidade e desempenho ao manipular coleções de dados.

10. Programação Orientada a Objetos (POO)

Pode-se usar POO, onde @property transforma métodos em propriedades acessíveis como atributos. Para indicar atributos privados, utiliza-se o prefixo __ (__atributo), limitando seu acesso a partir de fora da classe.

Conclusões

Como conclusão, é possível perceber o quanto a programação em Python é versátil. Os conceitos apresentados neste relatório cobrem fundamentos essenciais para qualquer desenvolvedor, proporcionando uma base sólida para o desenvolvimento de scripts simples e aplicações complexas. Com o conhecimento sobre variáveis, estruturas de controle, funções e POO, é possível criar códigos organizados, eficientes e prontos para expansão, facilitando a construção de projetos robustos e escaláveis.

Referencias

CURSOS, C. **PYTHON 3 curso rápido 🐍 Parte #1 2020 - 100% prático!** Disponível em: <https://www.youtube.com/watch?v=oUrBHiT-lzo&ab_channel=Cod3rCursos>. Acesso em: 13 nov. 2024a.

CURSOS, C. **PYTHON 3 curso rápido 🐍 Parte #2 2020 - 100% prático!** Disponível em: <https://www.youtube.com/watch?v=iq7JLIH-sV0&ab_channel=Cod3rCursos>. Acesso em: 13 nov. 2024b.

VINÍCIUS, R. **Args e Kwargs no Python**. **Python Academy**, 25 Jun. 2022. Disponível em: <https://pythonacademy.com.br/blog/args-e-kwargs-do-python>. Acesso em: 13 nov. 2024