

CURSO DE PYTHON



PARA INICIANTE



AULA 05: LOOPS - ESTRUTURAS DE REPETIÇÃO

Neste módulo, vamos explorar dois conceitos fundamentais da programação em Python: listas e funções.

- Listas são estruturas de dados que permitem armazenar múltiplos valores em uma única variável, oferecendo uma forma eficiente de organizar e manipular coleções de dados.
- Funções são blocos de código reutilizáveis que realizam uma tarefa específica. Elas tornam o código mais organizado e fácil de manter.

Compreender listas e funções é essencial para escrever programas mais poderosos e flexíveis. Vamos mergulhar nesses conceitos e ver como utilizá-los em seus projetos!

```
class Produto(object):  
    def __init__(self, nome, valor, quantidade):  
        self.__nome = nome  
        self.__valor = valor  
        self.__quantidade = quantidade
```

```
    def __repr__(self):  
        return self.__nome
```

```
    def get_nome(self):  
        return self.__nome
```

```
    def get_valor(self):  
        return self.__valor
```

```
    def get_quantidade(self):  
        return self.__quantidade
```

```
    def impressao(self):
```

```
        print(f"nome: {self.__nome} valor: {self.__valor} quantidade: {self.__quantidade}")
```

INTRODUÇÃO À LISTAS

As listas são um dos tipos de dados mais usados em Python. Elas permitem armazenar múltiplos itens em uma única variável. Diferente de outros tipos de coleções, as listas são:

- Mutáveis: você pode alterar os itens depois de criados.
- Ordenadas: a ordem dos elementos é preservada.
- Heterogêneas: podem conter diferentes tipos de dados (inteiros, strings, etc.).

CRIANDO LISTAS E ACESSANDO ELEMENTOS

Para criar uma lista, basta usar colchetes [] e separar os itens por vírgulas:

```
frutas = ["Maçã", "Banana", "Laranja"]
```

Você pode acessar elementos usando índices, começando em 0:

```
print(frutas[0]) # Saída: "maçã"
```

MÉTODOS COMUNS PARA LISTAS

1. `append(item)`: Adiciona um item ao final da lista.

```
frutas.append("uva") # Adiciona "uva" à lista
```

2. `remove(item)`: Remove o primeiro item da lista com o valor especificado

```
frutas.remove("uva") # Remove "banana" da lista
```

3. `sort()`: Ordena a lista em ordem crescente.


```
frutas.sort() # Ordena a lista em ordem alfabética
```

FUNÇÕES EM PYTHON

Funções são blocos de código reutilizáveis que realizam uma tarefa específica. Elas permitem que você organize seu código, tornando-o mais legível e fácil de manter. Ao usar funções, você evita a repetição de código e pode dividir seu programa em partes menores e mais gerenciáveis.

DEFININDO E CHAMANDO FUNÇÕES

Para definir uma função, utilizamos a palavra-chave `def`, seguida do nome da função e parênteses:



```
def saudacao():  
    print("Olá, seja bem-vindo!")
```


Para chamar a função, basta usar seu nome seguido de parênteses:



```
saudacao()  
# Saída: "Olá, seja bem-vindo!"
```

ARGUMENTOS E VALORES DE RETORNO

As funções podem receber argumentos, que são valores passados para a função para serem utilizados dentro dela. Além disso, uma função pode retornar um valor usando a palavra-chave `return`:



```
def soma(a, b):  
    return a + b  
  
resultado = soma(5, 3) # resultado agora é 8
```


Neste exemplo, a função `soma` aceita dois argumentos (`a` e `b`) e retorna a soma deles. O valor retornado pode ser armazenado em uma variável para uso posterior.

FUNÇÕES COM LISTAS

As funções podem interagir de maneira poderosa com listas, permitindo realizar operações complexas de forma simples e eficiente. Isso torna seu código mais flexível e modular.

PASSANDO LISTAS COMO ARGUMENTOS

Você pode passar listas como argumentos para funções, permitindo que a função opere sobre os elementos da lista. Por exemplo:




```
def imprimir_frutas(frutas):  
    for fruta in frutas:  
        print(fruta)  
  
minhas_frutas = ["maçã", "banana", "laranja"]  
imprimir_frutas(minhas_frutas) # Imprime cada fruta da lista
```

Nesse exemplo, a função `imprimir_frutas` recebe uma lista de frutas e imprime cada uma delas.

RETORNANDO LISTAS DE FUNÇÕES

Assim como você pode passar listas para funções, também pode retornar listas delas. Isso é útil para processar dados e devolver uma nova lista resultante. Veja um exemplo:



```
def filtrar_frutas(frutas, letra):  
    return [fruta for fruta in frutas if letra in fruta]  
  
todas_frutas = ["maçã", "banana", "laranja", "uva"]  
frutas_com_a = filtrar_frutas(todas_frutas, "a")  
# Retorna ["maçã", "banana", "laranja"]
```

Neste exemplo, a função `filtrar_frutas` retorna uma nova lista contendo apenas as frutas que têm a letra especificada.

PARABÉNS!

Você concluiu a QUINTA aula do curso de introdução a linguagem PYTHON.

Agora um desafio:

Escreva uma função que receba uma lista de números e retorne a soma de todos os números da lista.

Parabéns por chegar até aqui! Lembrem-se de que a programação é uma jornada, e cada linha de código é um passo em direção à realização dos seus sonhos. Continuem explorando e criando, e que o aprendizado nunca pare! Até a próxima!

