Lógica de Programação

Desvendando os Segredos das Listas

As listas, como estruturas de dados versáteis, são peças fundamentais na construção de programas eficientes e organizados. Nesta aula, desvendaremos seus segredos, explorando desde a criação e manipulação até aplicações práticas em cenários reais.

Listas: O que são e como criá-las?

Imagine uma lista de compras: leite, ovos, pão... As listas em Python funcionam de forma similar, armazenando diversos elementos em um único lugar. Cada elemento possui sua posição, como os itens em uma prateleira.

Criando sua Lista

Lista Simples

```
minha_lista = ["maçã", "banana", "laranja"]
```

Desvendando os Segredos

- Os colchetes [ ] definem o início e o fim da lista.
- Os elementos, separados por vírgulas, podem ser de diversos tipos (textos, números, etc.).
- A ordem dos elementos é crucial: minha_lista[0] retorna "maçã", o primeiro da fila.

Acessando os Elementos da Lista: Desvendando Posições e Índices

Cada elemento da lista possui um índice, como um número de identificação. O primeiro índice é 0, e os demais aumentam em 1:

```
minha_lista = ["uva", "pera", "morango"]

print(minha_lista[0])  # Exibe "uva"
print(minha_lista[1])  # Exibe "pera"
print(minha_lista[2])  # Exibe "morango"
```

Acessando os Elementos da Lista: Desvendando Posições e Índices

Atenção: Índices negativos acessam elementos a partir do final da lista:

```
print(minha_lista[-1])  # Exibe "morango"
print(minha_lista[-2])  # Exibe "pera"
```

Modificando Listas: Inserindo, Removendo e Alterando Elementos

As listas são mutáveis, permitindo que você as modifique após a criação. Explore algumas operações:

Inserindo Elementos:

Modificando Listas: Inserindo, Removendo e Alterando Elementos

Inserindo Elementos:

append( ) adiciona um elemento no final:

```
minha_lista.append("abacaxi")
print(minha_lista)  # Exibe ["uva", "pera", "morango", "abacaxi"]
```

Modificando Listas: Inserindo, Removendo e Alterando Elementos

Inserindo Elementos:

insert(indice, elemento) insere um elemento em uma posição específica:

```
minha_lista.insert(1, "kiwi")
print(minha_lista)  # Exibe ["uva", "kiwi", "pera", "morango",
```

Modificando Listas: Inserindo, Removendo e Alterando Elementos

Removendo Elementos:

remove(elemento) remove a primeira ocorrência do elemento:

```
minha_lista.remove("morango")
print(minha_lista)  # Exibe ["uva", "kiwi", "pera", "abacaxi"]
```

Modificando Listas: Inserindo, Removendo e Alterando Elementos

Removendo Elementos:

pop(indice) remove o elemento na posição especificada e retorna-o:

```
fruta_removida = minha_lista.pop(2)
print(fruta_removida)  # Exibe "pera"
print(minha_lista)  # Exibe ["uva", "kiwi", "abacaxi"]
```

Modificando Listas: Inserindo, Removendo e Alterando Elementos

Alterando Elementos

Atribuição direta modifica o valor na posição indicada:

```
minha_lista[1] = "manga"
print(minha_lista)  # Exibe ["uva", "manga", "abacaxi"]
```

Explorando Funções Úteis para Listas: Facilitando sua Vida

Python oferece diversas funções para facilitar a manipulação de listas. Vamos conhecer algumas:

len(lista) retorna o tamanho da lista (número de elementos):

```
tamanho_lista = len(minha_lista)
print(tamanho_lista)  # Exibe 3
```

Explorando Funções Úteis para Listas: Facilitando sua Vida

sorted(lista) cria uma nova lista ordenada (crescente ou decrescente):

```
lista_ordenada = sorted(minha_lista)
print(lista_ordenada)  # Exibe ["abacaxi", "kiwi", "manga", "uva"]
```

Explorando Funções Úteis para Listas: Facilitando sua Vida

reversed(lista) inverte a ordem dos elementos:

```
lista_invertida = list(reversed(minha_lista))
print(lista_invertida)  # Exibe ["uva", "manga", "kiwi", "
```

Operações com Listas

As listas em Python permitem operações que vão além da simples manipulação de elementos. Exploremos algumas delas:

Concatenando Listas

O operador + junta duas listas em uma nova:

```
lista1 = ["abacate", "manga"]
lista2 = ["kiwi", "uva"]

lista_combinada = lista1 + lista2
print(lista_combinada)  # Exibe ["abacate", "manga", "kiwi", "uva"]
```

Repetindo Elementos

O operador * repete um elemento o número desejado de vezes

```
fruta_repetida = "banana" * 4
print(fruta_repetida)  # Exibe "bananabananabanana"
```

Verificando Pertencimento

O operador in verifica se um elemento existe na lista

```
fruta_presente = "manga" in lista_combinada
print(fruta_presente)  # Exibe True
```

Vamos a prática

- Crie uma lista contendo os nomes de 4 frutas diferentes

- Acesse e mostre o primeiro e o último elemento da lista.
- Mostre o tamanho da lista
- Adicione um novo item à lista usando o método append( )

Listas Dentro de Listas: Estruturas Aninhadas para Maior Organização

As listas podem conter outras listas, criando estruturas aninhadas. Imagine uma lista de alunos, onde cada aluno possui sua própria lista de notas.

```
turma = [
    ["João", [8, 7, 9]],
    ["Maria", [6, 8, 10]],
    ["Pedro", [5, 9, 7]],
]

print(turma[1][1])  # Exibe
```

Listas Dentro de Listas: Estruturas Aninhadas para Maior Organização

As listas podem conter outras listas, criando estruturas aninhadas. Imagine uma lista de alunos, onde cada aluno possui sua própria lista de notas.

```
turma = [
    ["João", [8, 7, 9]],
    ["Maria", [6, 8, 10]],
    ["Pedro", [5, 9, 7]],
]

print(turma[1][1])  # Exibe
```

Desvendando os Fundamentos dos Loops for em Listas

Imagine um ninja percorrendo um caminho, visitando cada vilarejo. O loop for funciona da mesma forma, permitindo que você visite cada elemento de uma lista, um por um.

Desvendando os Fundamentos dos Loops for em Listas

Estrutura básica

```
for item in lista:
    # Codigo a ser executado para cada item
```

Variável de Iteração

A variável item dentro do loop recebe cada elemento da lista sequencialmente, como um ninja visitando cada vilarejo.

Percorrendo uma Lista Simples

Utilizando o loop for, você pode executar um bloco de código para cada elemento da lista, como um ninja cumprimentando cada habitante.

```
frutas = ["banana", "maçã", "laranja", "uva"]

for fruta in frutas:
    print(f"Adoro comer {fruta}!")
```

Acessando Índices com range( )

O loop for pode ser combinado com a função range( ) para iterar sobre os índices da lista, como um ninja visitando casas numeradas.

```
frutas = ["banana", "maçã", "laranja", "uva"]

for i in range(len(frutas)):
    print(f"Fruta {i + 1}: {frutas[i]}")  # Exibe
```

Iterando sobre Dicionários

O loop for também funciona com dicionários, permitindo que você acesse chaves ou valores, como um ninja explorando um mapa.

```
pessoa = {"nome": "Maria", "idade": 31, "cidade": "Teresina"}

for chave in pessoa:
    print(f"{chave}: {pessoa[chave]}")  # Exibe chave e valor
```

Usando enumerate( ) para Acessar Índices e Elementos

A função enumerate( ) retorna uma tupla contendo o índice e o elemento em cada iteração, como um ninja recebendo um mapa e a localização de cada ponto de interesse.

```
frutas = ["banana", "maçã", "laranja", "uva"]

for indice, fruta in enumerate(frutas):
    print(f"Posição {indice}: {fruta}")
```

Modificando Elementos Durante a Iteração

O loop for permite modificar elementos da lista enquanto você itera, como um ninja ajustando sua rota conforme explora o caminho.

```
cidades = ["São Paulo", "Rio de Janeiro", "Salvador", "Brasília"]

for indice, cidade in enumerate(cidades):
    if cidade == "Brasília":
        cidades[indice] = "Brasília (Capital)"  # Modificando o nome

print(cidades)  # Exibe: ["São Paulo", "Rio de Janeiro", "Salvador",
```

Evitando Modificações com Cópia da Lista

Ao modificar elementos dentro de um loop for, é importante criar uma cópia da lista se você precisar da lista original intacta.

```
frutas = ["banana", "maçã", "laranja", "uva"]

frutas_modificadas = frutas[:]

for indice, fruta in enumerate(frutas_modificadas):
    frutas_modificadas[indice] = fruta.upper()

print(frutas)
print(frutas_modificadas)
```

Evitando Modificações com Cópia da Lista

Ao modificar elementos dentro de um loop for, é importante criar uma cópia da lista se você precisar da lista original intacta.

```
frutas = ["banana", "maçã", "laranja", "uva"]

frutas_modificadas = frutas[:]

for indice, fruta in enumerate(frutas_modificadas):
    frutas_modificadas[indice] = fruta.upper()

print(frutas)
print(frutas_modificadas)
```

Por hoje é só...

- Hoje vimos muita coisa, vamos descansar que amanhã tem mais!