Aula 09 - Coleções Parte 1: Lista

Na aula 05, vimos quee uma variável é um valor que varia e não sabemos qual é. Também vimos que ao declarar essa variável, o computador reserva um pedaço da memória RAM para armazenar esse valor. Agora, vamos descobrir que podemos armazenar mais de um valor em uma única variável. As coleções são muito parecidas com o que nas outras linguagens chamamos de *arrays*. Há 3 tipos de coleções em Python: **lista**, **tupla** e **dicionário**. É justamente nesse ponto onde o Python se encontra tão mais forte e poderoso em relação à otras linguagens de programação.

Obs: para os que já conhecem os *arrays* das outras linguagens, há uma importante diferença para as listas: diferentemente dos *arrays*, uma lista pode conter dados de diferentes tipos.

Lista

É o equivalente ao *vetor* nas outras linguagens de programação. Consiste em um conjunto de dados de um mesmo tipo armazenados em uma única variável, que por sua vez reserva vários espaços na memória RAM para isso. Sua estrutura é identificada por `nome_lista = [valor1, valor2, valor3, ...]`, quando a lista já recebe os valores diretamente na declaração, e `nome_lista =[]` para inicializar uma lista vazia. Segue o exemplo abaixo:

In [ ]:
```
# lista
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']

# imprime lista na tela
print(nomes_lista)
```

```
['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']
```

É possível exibir apenas nomes específicos da lista, desde que se saiba em qual posição ele se encontra. É por isso que aprendemos na aula passada sobre a forma de contagem numérica dos computadores, pois esse detalhe é importante aqui. Lembre-se: o computador começa a contar do 0. Isso significa que, se eu quiser que ele exiba o primeiro nome da lista, na verdade ele tem que trazer o nome na posição 0, enquanto que o nome na posição 1, na verdade, é o segundo nome da lista.

In [ ]:
```
# lista
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']

print(nomes_lista[0]) # exibindo o primeiro nome da lista
print(nomes_lista[2]) # exibindo o terceiro nome da lista
print(nomes_lista[5]) # exibindo o sexto e último nome da lista
```

```
Fulano
Beltrano
Mariana
```

Também posso exibir todos os nomes da lista de uma forma mais organizada. Basta apenas percorrer a lista utilizando o laço `for` aprendido na aula passada:

In [ ]:
```python
# lista
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']

# percorrendo a lista
for nome in nomes_lista:
    print(nome)
```

```
Fulano
Cicrano
Beltrano
Joana
Maria
Mariana
```

Muitos programadores vão se surpreender com a forma como o `for` está sendo usado acima, pois em outras linguagens, há um outro laço de repetição chamado `foreach`, que é utilizado exatamente dessa forma como no código acima. Mas no Python, não temos `foreach`. Ao invés disso, podemos usar o `for` da mesma forma.

No próximo exemplo, iremos reexibir toda a lista, mas desta vez com um contador do lado:

In [ ]:
```python
# lista
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']

# percorrendo a lista com um contador
for i in range(len(nomes_lista)):
    print(f'{i + 1}º nome da lista: {nomes_lista[i]}')
```

```
1º nome da lista: Fulano
2º nome da lista: Cicrano
3º nome da lista: Beltrano
4º nome da lista: Joana
5º nome da lista: Maria
6º nome da lista: Mariana
```

Diferentes tipos de dados em uma única lista

Como dito anteriormente, é possível armazenar dados de diferentes tipos em uma única lista. No exemplo abaixo, estamos armazenando na mesma lista: *strings*, *ints* e *floats*.

In [ ]:
```python
# lista com diferentes tipos de dados
lista_dados = ['Alex', 39, 1.72]

# exibindo os dados da lista na tela e seus respectivos tipos
for dado in lista_dados:
    print(dado, type(dado))
```

```
Alex <class 'str'>
39 <class 'int'>
1.72 <class 'float'>
```

Percorrendo variável simples como se fosse lista

É possível pegar o valor de uma variável e destrinchá-lo como uma lista. Veja o código-fonte a seguir:

In [ ]:
```python
# variável
nome = 'SENAI'

# percorrendo cada caractere do valor da variável
for i in nome:
    print(i)
```

```
S
E
N
A
I
```

Ordenando dados da lista

É possível ordenar os dados da lista, tanto em ordem crescente (no caso dos números) como em ordem alfabética (no caso de textos) através da função `sort()`, como veremos no código-fonte abaixo:

In [ ]:
```python
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']

# ordena os dados da lista
nomes_lista.sort()

for nome in nomes_lista:
    print(nome)
```

```
Beltrano
Cicrano
Fulano
Joana
Maria
Mariana
```

Pesquisando dados em uma lista

Poss usar algoritmos de pesquisa para encontrar um determinado valor em uma lista. O recurso é útil para verificar se um determinado valor existe ou não em uma lista.

Forma 1: operador in

Podemos fazer um `if` econjunto com o operador `in` para encontrar um valor em uma lista:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']
nome = input('Informe o nome que deseja encontrar: ')

# busca o nome desejado
if nome in nomes_lista:
    print(nome)
else:
    print(f'{nome} não encontrado.')
```

```
Joana
```

Forma 2: index

Outro método bastante utilizado para buscar elementos em uma lista é o método "index". Esse método retorna o índice do primeiro elemento encontrado na lista que corresponde ao valor buscado:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']
nome = input('Informe o nome que deseja encontrar: ')

# procura pelo índice através do valor
indice = nomes_lista.index(nome)

# retorna resultado
try:
    print(f'{nome} encontrado no índice {indice}.')
except:
    print(f'{nome} não encontrado.')
```

```
Cicrano encontrado no índice 1.
```

Forma 3: count

O Python também oferece a função "count" para contar a quantidade de ocorrências de um determinado elemento em uma lista:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Fulano']
nome = input('Informe o nome que deseja encontrar: ')

# conta a quantidade de vezes que o valor foi encontrado
quantidade = nomes_lista.count(nome)

# retorna resultado
try:
    print(f'{nome} foi encontrado na lista {quantidade} vezes.')
except:
    print(f'{nome} não foi encontrado.')
```

```
Fulano foi encontrado na lista 2 vezes.
```

Alterando dados de uma lista

Uma vez que você pode consultar e acessar os dados de uma lista, pode facilmente alterar seu valor. No exemplo abaixo, vamos alterar o primeiro nome da lista. Para isso, precisaremos apenas saber e acessar exatamente o seu índice:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Fulano']
nomes_lista[0] = 'Alex'

for nome in nomes_lista:
    print(nome)
```

```
Alex
Cicrano
Beltrano
Joana
Maria
Fulano
```

Mas se não souber, pode fazer uma pesquisa pelo nome e pedir para retornar seu índice, como no exemplo abaixo:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Fulano']
nome_a_alterar = input('Informe o nome que deseja alterar: ')
nomes_lista[nomes_lista.index(nome_a_alterar)] = input('Informe o novo nome: ')

for nome in nomes_lista:
    print(nome)
```

```
Fulano
Alex
Beltrano
Joana
Maria
Fulano
```

Inserindo dados em uma lista

É possível inserir manualmente os dados de uma lista, caso o usuário queira criar sua própria lista. Uma das facilidades do Python é que com uma lista simples o usuário pode adicionar quantos itens da lista desejar, sem limites no algoritmo. Vamos criar, por exemplo, um programa para ajudar a dona de casa a montar uma lista de compras:

In [ ]:
```
# Lista vazia
lista_itens = []

# Loop
while True:
    # informa o nome do item
    item = input('Insira um item, ou deixe em branco para terminar a lista: ')

    # verifica se o valor foi inserido ou não
    if item != '':
        # adiciona o item na lista
        lista_itens.append(item)
        continue
    else:
```

```
        # encerra a lista
        break

# exibe a lista
for item in lista_itens:
    print(item)
```

```
Papel higiênico
Macarrão
Feijão
Chocolate
```

Adicionando um item em uma posição da lista

Podemos adicionar um item da lista em uma posição específica, seja no início, no meio ou no fim, não importa. No exemplo abaixo iremos solicitar ao usuário qual valor e qual a posição ele deseja inserir o novo valor:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']

novo_nome = input('Informe o novo nome a ser inserido na lista: ')
posicao = int(input('Informe a posição que deseja inserir (de 1 a 6):'))

# corrige a posição para a contagem do computador
posicao -= 1

# tenta inserir o nome na posição desejada
try:
    nomes_lista.insert(posicao, novo_nome)
    print(f'{novo_nome} inserido na posição {posicao + 1} com sucesso.')
except:
    print('Não foi possível inserir o nome.')

# exibe os nomes na lista
for nome in nomes_lista:
    print(nome)
```

```
Alex inserido na posição 2 com sucesso.
Fulano
Alex
Cicrano
Beltrano
Joana
Maria
Mariana
```

Deletando dados de uma lista

Fareamos agora o inverso: deletar um item que já está em uma lista. Para isso, precisaremos informar a sua posição na lista. Mais uma vez, para evitar erros por parte do usuário, iremos aplicar uma correção subtraindo o valor que o usuário informar por `-1` para que ele delete exatamente o item na posição que o usuário desejar:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']
```

```
posicao = int(input('Informe a posição do item que deseja excluir (de 1 a 6): ')
posicao -= 1

# tenta deletar o item na posição informada
try:
    del(nomes_lista[posicao])
except:
    print('Não foi possível deletar.')

# exibe a nova lista
for nome in nomes_lista:
    print(nome)
```

```
Fulano
Beltrano
Joana
Maria
Mariana
```

Deletando valor específico da lista

Nesse tipo de algoritmo, você pode deletar um item pelo valor dele, caso saiba que esse valor esteja na lista, mas não sabe em qual posição, esse algoritmo pode ser útil. Mas esse algoritmo tem um problema: se por acaso o valor a ser deletado tiver repetido na lista, só irá deletar a primeira ocorrência em que ela ocorre:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']
nome_a_excluir = input('Informe o nome que deseja retirar da lista: ')

# tenta excluir o nome
try:
    nomes_lista.remove(nome_a_excluir)
except:
    print('Nome não pode ser excluído.')

# exibe a nova lista
for nome in nomes_lista:
    print(nome)
```

```
Cicrano
Beltrano
Joana
Maria
Mariana
```

Serapando item da lista em uma variável

Se por acaso precisar isolar especificamente um item da lista, também é perfeitamente possível através da função `pop()`.

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']
indice = int(input('Informe o número do índice do elemento que deseja separar da
indice -= 1

# tenta separar o nome da lista
```

```
try:
    nome_separado = nomes_lista.pop(indice)
except:
    print('Não foi possível separar.')

print(nome_separado)
```

```
Fulano
```

Se preferir, pode fazer uma procura pelo valor e retornar um índice para separar o valor:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']
nome_a_separar = input('Informe o nome que deseja separar da lista: ')

try:
    nome_separado = nomes_lista.pop(nomes_lista.index(nome_a_separar))
except:
    print('Não foi possível separar.')

print(nome_separado)
```

```
Maria
```

Join

Dependendo do tipo de operação que precisar fazer, pode haver a necessidade de juntar os valores de uma lista em uma única variável. Caso haja essa necessidade, você pode fazer uso da função `join()`, conforme demonstrado abaixo:

In [ ]:
```
nomes_lista = ['Fulano', 'Cicrano', 'Beltrano', 'Joana', 'Maria', 'Mariana']

# insere um separador
separador = ','

# junta os nomes em uma única variável
nomes_string = separador.join(nomes_lista)

print(nomes_string)
```

```
Fulano,Cicrano,Beltrano,Joana,Maria,Mariana
```

Split

A função `split()` faz exatamente o contrário do `join()`, ou seja, pega os valores de uma variável e separa em elementos diferentes de uma lista:

In [ ]:
```
meses_string = 'Janeiro Fevereiro Março Abril Maio Junho Julho Agosto Setembro O

# insere o separador para separar os valores e os insere em uma lista
meses_lista = meses_string.split(' ')

# exibe a lista
for mes in meses_lista:
    print(mes)
```

```
Janeiro
Fevereiro
Março
Abril
Maio
Junho
Julho
Agosto
Setembro
Outubro
Novembro
Dezembro
```

Lista numérica

É possível também montar uma lista com números, não só com strings. Veja o exemplo abaixo:

In [ ]:
```
# lista numérica
numeros = [1, 2, 3, 4, 5]
print(numeros)
```

```
[1, 2, 3, 4, 5]
```

A vantagem de montar uma lista numérica são algumas operações a mais que podem ser feitas.

Soma dos números de uma lista

Um exemplo de uma operação que pode ser feita com números de uma lista é uma soma. Veja o exemplo abaixo:

In [ ]:
```
numeros = [1, 2, 3, 4, 5]

# soma os números da lista
print(sum(numeros))
```

```
15
```

Ordem crescente e decrescente

Também é possível fazer uma ordenação dos números, tanto em ordem crescente quanto em ordem decresccente. Veja no código abaixo:

In [ ]:
```
numeros = [1,3,2,6,4,5]

# imprime em ordem crescente
numeros.sort()
print(numeros)

# imprime em ordem decrescente
numeros.sort(reverse=True)
print(numeros)
```

```
[1, 2, 3, 4, 5, 6]
[6, 5, 4, 3, 2, 1]
```

Exercícios

---

1. Crie um programa que possa incluir, pesquisar, alterar e excluir nomes em uma lista, que pode ter quantos nomes o usuário desejar.

2. Crie um programa que o usuário possa digitar quantos números digitar, e ao terminar, imprima os números em ordem crescente.

3. Crie um programa que o usuário possa digitar uma quantidade desejada de notas de um determinado aluno (nota mínima 0 e nota máxima 10), e o programa calcula a média desse aluno, e ao final informe se o aluno está aprovado ou reprovado (média mínima para aprovação = 7).