Linguagem de Programação

08: LISTAS EM PYTHON

Nossos **objetivos** nesta aula são:

- Entender o que é uma lista na programação de computadores.
- Conhecer quais as operações comumente efetuadas nas listas ● Criar e usar listas em Python
- Entender a diferença entre listas e tuplas em Python
- Entender o que são e como utilizar outras sequências em Python.
- Aprender a percorrer por todos os elementos de uma sequência.

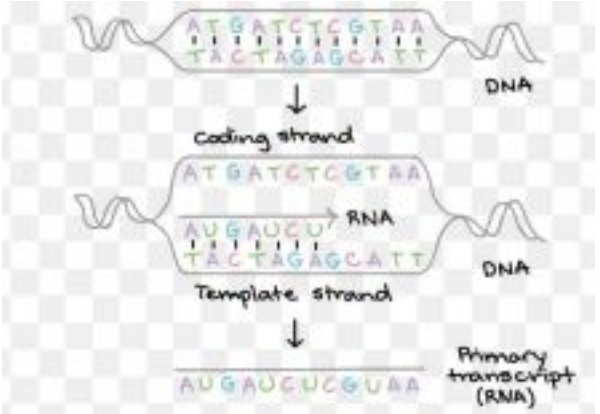
A referência para esta aula é o **Capítulo 4 (Listas)** do livro:

DIERBACH, C. Introduction to Computer Science Using Python: A Computational Problem Solving Focus. 1st Edition, New York: Wiley, 2012.

INTRODUÇÃO

A forma como dados são organizados tem um impacto significativo sobre a eficácia com que podem ser usados. Uma das maneiras mais óbvias e úteis de organizar os dados é como uma lista.

Nós usamos listas no nosso dia-a-dia, fazemos listas de compras, listas de tarefas, listas de convidados, etc.



Listas também ocorrem na natureza. Nosso DNA é essencialmente uma longa lista de moléculas na forma de uma dupla hélice, encontrada no núcleo de todas as células humanas e todos os organismos vivos. Sua finalidade também é armazenar informações - especificamente, as instruções usadas para construir e operar todas as células do corpo - que chamamos de genes.

Dados os 2,85 bilhões de nucleotídeos que compõem o genoma humano, determinar seu

sequenciamento (e, assim, entender nossa composição genética) é fundamentalmente um problema computacional.

Veremos o uso de listas na programação. O conceito de uma lista é semelhante à nossa noção cotidiana de uma lista. Nós lemos (acessamos) itens em nossa lista de tarefas, adicionamos itens, cruzamos (excluímos) itens e assim por diante.

DEFINIÇÃO DE LISTA

Uma lista é uma **estrutura de dados linear** que mantém a posição dos seus elementos em uma ordem linear. Ou seja, tem o primeiro elemento, o segundo elemento e assim por diante até o último elemento.

Abaixo temos um exemplo de uma lista de compras. Note que o primeiro elemento da lista é Cereal, o segundo é Suco, o terceiro é Banana, etc.

Posição	Item
0	Cereal
1	Suco
2	Banana
3	Maçã
4	Azeite

A ordem em que os elementos aparecem na lista pode ser a ordem em que o autor planeja pegar os itens no mercado, ou pode ser simplesmente a ordem em que ele anotou os itens à medida que foi se lembrando.

A numeração desta lista segue um padrão muito comum em diversas linguagens de programação, e que é adotado também pelo Python, começando com o primeiro elemento na posição zero. E a posição de um elemento na lista é também chamada de **índice**.

OPERAÇÕES BÁSICAS EM UMA LISTA

As operações básicas normalmente são realizadas em uma lista são: recuperar (ler), substituir, inserir, remover e acrescentar.

1) **Recuperar** um item da lista é a operação de obter o valor que está em uma determinada posição na lista. No exemplo anterior, na posição 2 da lista temos o item “Banana”.

2) **Substituir** um item da lista é a operação de atribuir um novo valor a uma determinada posição da lista, sobrescrevendo o valor anteriormente guardado naquela posição.

Por, exemplo, podemos trocar o elemento na posição 3 da lista por Mamão.

Posição	Antes	Depois
0	cereal	cereal
1	suco	suco
2	banana	banana
3	**maçã**	**mamão**
4	azeite	azeite

3) **Inserir** um item na lista é a operação de adicionar um novo item em uma posição específica, análogo ao comando do Excel de inserir uma nova linha ou coluna em uma tabela. Suponhamos que o autor da lista se lembrou de que ele deve pegar macarrão antes de ir para a seção de frutas. Neste caso, ele irá inserir na posição 2 da lista o novo item.

Posição	Antes	Depois
0	cereal	cereal
1	suco	suco
2	banana	**macarrão**
3	mamão	banana
4	azeite	mamão
5	azeite	

Note que os demais itens avançam uma posição, pois foi criado um novo espaço na lista, e agora o total de itens na lista aumentou de 5 para 6. Note também que nenhum item foi sobrescrito.

4) **Remover** um item da lista é a operação de excluí-lo. Se o autor da lista não precisar mais de Cereal, ele pode remover o item da posição número 0, e o tamanho da lista diminui de 6 para 5 nesse caso.

Posição	Antes	Depois
0	**cereal**	suco
1	suco	macarrão
2	macarrão	banana
3	banana	mamão
4	mamão	azeite

5 azeite

Note que todos os itens que estavam abaixo do Cereal na lista "subiram" uma

posição. 3

5) **Acrescentar** um item a lista é a operação de incluir um item ao final da lista. Ao adicionarmos um novo item na lista, temos:

Posição	Antes	Depois
0	suco	suco
1	macarrão	macarrão
2	banana	banana
3	mamão	mamão
4	azeite	azeite
5		**sal**

Note que novamente o tamanho da lista aumentou de 5 para 6 itens, mas dessa vez, não houve nenhuma alteração na posição dos itens que já estavam na lista, e que ao acrescentar um elemento na lista, não precisamos dizer em que posição o faremos, pois será sempre ao final da lista.

LISTAS EM PYTHON

Uma lista em Python é uma estrutura de dados linear mutável e heterogênea, isto é, que pode ser modificada e permite itens de tipos diferentes entre si. Ela é representada em Python por colchetes, sendo os itens separados por vírgula.

Exemplos de listas:

```
[1, 10, 3] # lista de inteiros
['cereal', 'suco', 'banana', 'maçã', 'azeite'] # lista de strings
[5, 'maçã', True, 10.3] # lista mista
[] # lista vazia
```

Vamos criar então a lista de compras do começo do capítulo e repetir as mesmas operações, só que agora em Python. Para criar a lista usamos a sintaxe acima e atribuímos o resultado a uma variável de nossa escolha.

```
>>> compras = ['cereal', 'suco', 'banana', 'maçã', 'azeite']
```

Ao acessarmos o conteúdo da variável na shell do Python, obtemos a lista completa:

```
>>> compras
['cereal', 'suco', 'banana', 'maçã', 'azeite']
```


1) **Recuperar**: Para recuperar um item da lista, usamos um par de colchetes após a variável e passamos o índice do elemento que queremos acessar. Lembrando que a contagem dos índices (posição) começa em zero.

```
>>> compras[2]
'banana'
```

2) **Substituir**: Para substituir um item da lista, fazemos a atribuição do novo valor à uma posição específica na lista.

```
>>> compras[3] = 'mamão'
>>> compras
['cereal', 'suco', 'banana', 'mamão', 'azeite']
```

3) **Inserir**: Para inserir um item na lista usamos o método **insert** da lista e passamos dois argumentos: o índice e o valor que queremos inserir. A utilização de métodos de um objeto é feita com o uso do ponto após a variável, seguido pelo nome do método e seus parâmetros, com funcionamento análogo aos parâmetros de uma função.

```
>>> compras.insert(2, 'macarrão')
>>> compras
['cereal', 'suco', 'macarrão', 'banana', 'mamão', 'azeite']
```

4) **Remover**: Para remover um item da lista usamos o comando **del** seguido do nome da variável e o índice do item que será removido.

```
>>> del compras[0]
>>> compras
['suco', 'macarrão', 'banana', 'mamão', 'azeite']
```

5) **Acrescentar**: Para acrescentar um item à lista usamos o método **append** da lista e passamos como argumento o item que será incluído à lista.

```
>>> compras.append('sal')
```

```
>>> compras
['suco', 'macarrão', 'banana', 'mamão', 'azeite', 'sal']
```

OUTROS MÉTODOS DE LISTA

A lista completa dos métodos existentes em uma lista pode ser encontrada na documentação oficial do Python sobre sequências. Veremos alguns deles no decorrer deste capítulo.

pop

O método **pop** é utilizado para remover um item da lista a partir do seu índice, assim como o comando **del**, mas diferentemente do comando, o método pop retorna o item removido, então podemos atribuí-lo a uma variável se quisermos.

Como estamos utilizando um método, o índice deve ser passado como argumento:

```
>>> numeros = [1, 5, 6, 82, 25, 11]
>>> x = numeros.pop(0)
>>> numeros
[5, 6, 82, 25, 11]
>>> x
1
```

O método **pop** pode também ser usado sem nenhum parâmetro, e nesse caso o item removido será sempre o último da lista.

```
>>> y = numeros.pop()
>>> numeros
[5, 6, 82, 25]
>>> y
11
```

remove

O método **remove** é também remove um item da lista, mas ao contrário dos outros do que vimos até agora, isso é feito pelo valor do item e não pelo índice. O método irá fazer uma busca na lista e remover o primeiro item com o valor correspondente que encontrar.

```
>>> numeros = [1, 5, 3, 27, 5, 11]
>>> numeros.remove(5)
>>> numeros
[1, 3, 27, 5, 11]
```

Note que a primeira ocorrência do número 5 foi removida, e que este método, assim como o comando **del**, não retorna o item removido, apenas o exclui da lista.

reverse

O método **reverse** inverte os elementos de uma lista, colocando os de trás pra frente na lista.

```
>>> numeros = [1, 5, 3, 27]
>>> numeros.reverse() # O método reverse não tem parâmetros >>> numeros
[27, 3, 5, 1]
```