

Curso Preparatório

Semana 9:

Coleções: Listas

Admissão 2025



Coleções



- As estruturas compostas, também conhecidas como **coleções** permitem armazenar múltiplos itens dentro de uma única unidade, que funciona como um container. Entre as coleções, temos:
- **Tuplas:**
 - Coleção de valores indexada estrutura de dados semelhante a vetores. Ela tem a característica de ser imutável, ou seja, após uma tupla ser criada, ela não pode ser alterada.
- **Sets:**
 - Coleções não ordenadas, que não permite elementos duplicados, ou seja, cada elemento é único. Um set em si pode ser modificado, contudo os elementos contidos dentro dele precisam ser de tipos imutáveis.
- **Listas:**
 - Lista é uma coleção de valores indexada, em que cada valor é identificado por um índice. O primeiro item na lista está no índice 0, o segundo no índice 1 e assim por diante, pode ser manipulado.
- **Dicionários:**
 - Os dicionários representam coleções de dados que contém na sua estrutura um conjunto de pares chave/valor, nos quais cada chave individual tem um valor associado. Esse objeto representa a ideia de um mapa, que entendemos como uma coleção associativa desordenada. A associação nos dicionários é feita por meio de uma chave que faz referência a um valor.



Listas

- Para criar uma lista com elementos deve-se usar **colchetes[]** e adicionar os itens entre eles separados por vírgula:

```
1 programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
2 print(type(programadores)) # type 'list'
3 print(len(programadores)) # 5
4 print(programadores[4]) # Luana
```

- As listas no Python são **mutáveis**, podendo ser alteradas depois de terem sido criadas. Em outras palavras, podemos adicionar, remover e até mesmo alterar os itens de uma lista. EX:

```
1 programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
2 print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
3
4 programadores[1] = 'Carolina'
5 print(programadores) # ['Victor', 'Carolina', 'Samuel', 'Caio', 'Luana']
```



Listas

- Além de alterar elementos em listas, também é possível adicionar itens nelas, pois já vêm com uma coleção de métodos predefinidos que podem ser usados para manipular os objetos que ela contém. No caso de adicionar elementos, podemos usar o método **append()**, que adiciona elementos no final de uma lista.

```
1 programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
2 print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
3
4 programadores.append('Renato')
5 print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana', 'Renato']
```

- Há outra forma de adicionarmos itens na lista, que é através do método **insert()**. Ele usa dois parâmetros: o primeiro para indicar a posição da lista em que o elemento será inserido e o segundo para informar o item a ser adicionado na lista:

```
1 programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
2 programadores.insert(1, 'Rafael')
3
4 print(programadores) # ['Victor', 'Rafael', 'Juliana', 'Samuel', 'Caio', 'Luana']
```



Listas

- Também é possível remover itens, para isso temos dois métodos: **remove()** para a remoção pelo valor informado no parâmetro, e **pop()** para remoção pelo índice do elemento na lista.

```
1 programadores = ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
2 print(programadores) # ['Victor', 'Juliana', 'Samuel', 'Caio', 'Luana']
3
4 programadores.remove('Victor')
5 print(programadores) # ['Juliana', 'Samuel', 'Caio', 'Luana']
```

```
4 programadores.pop(0)
5 print(programadores) # ['Juliana', 'Samuel', 'Caio', 'Luana']
```

- Assim como nas Tuplas, as Listas podem receber tipos numéricos também. Inclusive misturar INTEIROS, REAIS e TEXTOS em uma mesma Lista. Mas tenha cuidado, você quem deve tratar tudo, não é a linguagem.



Outras opções das Listas:

- Nas tuplas, tínhamos o **sorted()**, nas listas tem o **sort()** também, que já muda a ordem, mas não se usa o **sort()** dentro do **print()**, so o **sorted()**.
- **reverse()** -> Inverte a ordem da lista.

```
1 valores = [1,5,7,3,9,2,15]
2 print(valores)
3 valores.sort()
4 print(valores)
5 valores.reverse()
6 print(valores)
```



```
[1, 5, 7, 3, 9, 2, 15]
[1, 2, 3, 5, 7, 9, 15]
[15, 9, 7, 5, 3, 2, 1]
```



Outras opções das Listas:

- As listas podem ser preenchidas, do zero, para isso usamos o **append()**. Inicialmente é preciso criar (declarar) a lista, usa apenas os colchetes [].

- Você pode usar a seguinte estrutura:

valores = []

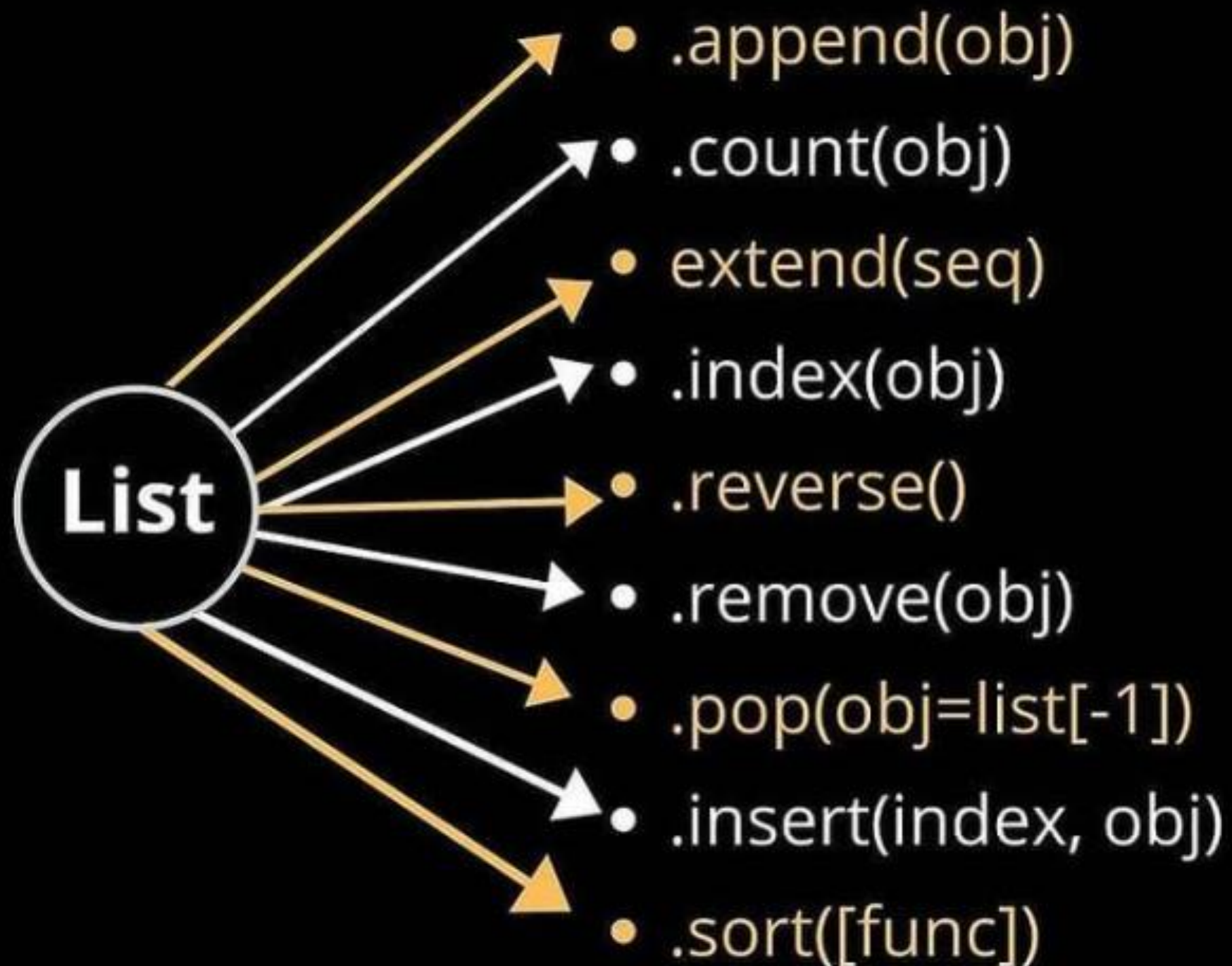
#Estrutura de repetição. EX:

while True:

valores.append(int(input("Digite um valor qualquer: ")))



PYTHON LIST METHODS



Aprenda mais sobre Python no Instagram:

@pycodebr






























(<https://www.instagram.com/pycodebr/>)

@python.hub

<https://www.instagram.com/python.hub/>



List Cheat Sheet

	<code>.append()</code>		
	<code>.clear()</code>		
	<code>.copy()</code>		
	<code>.count()</code>		2
	<code>.index()</code>		1
	<code>.insert(1, )</code>		
	<code>.pop(1)</code>		
	<code>.remove()</code>		
	<code>.reverse()</code>		

Aprenda mais sobre Python no Instagram:

[@pycodebr](#)

(<https://www.instagram.com/pycodebr/>)

[@python.hub](#)

<https://www.instagram.com/python.hub/>



Lista - Práticas

- Crie um programa para criar uma lista a partir de dados digitados pelo usuário (números inteiros). Depois da lista ser formada, seu programa deve:
 - A) Dizer quantos números foram digitados;
 - B) Ordenar a lista em ordem crescente e decrescente;
 - C) Dizer se o usuário digitou o numero 5 ou não.

```
1 valores = []
2 while True:
3     valores.append(int(input("Digite um valor qualquer: ")))
4     opcao = str(input("Deseja continuar?\n[s = Sim | n = Não]")).lower()
5     if (opcao=='n'):
6         break
7
8     print("Você digitou",len(valores),"numeros na lista")
9     print("A lista ficou assim:")
10    print(valores)
11    valores.sort()
12    print("A lista ordenada ficou assim:")
13    print(valores)
14    valores.reverse()
15    print("A lista ordenada de forma decrescente ficou assim:")
16    print(valores)
17    if 5 in valores:
18        print("Você digitou o numero 5")
19    else:
20        print("Você não digitou o numero 5")
```



- Crie um programa em Python para o jogo PEDRA, buscando as opções, você deve fazer com que seja

```
1  from random import randint
2  computador = randint(0,2)
3  escolha = ("Pedra", "Papel", "Tesoura")
4  while True:
5      print("opções:")
6      print("[1]-Pedra")
7      print("[2]-Papel")
8      print("[3]-Tesoura")
9      while True:
10         jogador = int(input("Qual sua jogada?: "))
11         jogador = jogador-1
12         if (jogador < 3):
13             break
14         else:
15             print("Opção invalida!")
16     print("O computador escolheu", escolha[computador].upper())
17     print("Você escolheu", escolha[jogador].upper())
```

```
18     if computador == 0: #pedra
19         if jogador==0:
20             print("Deu EMPATE")
21         elif jogador==1:
22             print("PARABENS! Você venceu")
23         elif jogador==2:
24             print("IHH! Você perdeu")
25     elif computador == 1: #computador = papel
26         if jogador==0:
27             print("IHH! Você perdeu")
28         elif jogador==1:
29             print("Deu EMPATE")
30         elif jogador==2:
31             print("PARABENS! Você venceu")
32     elif computador == 2: #computador = tesoura
33         if jogador==0:
34             print("PARABENS! Você venceu")
35         elif jogador==1:
36             print("IHH! Você perdeu")
37         elif jogador==2:
38             print("Deu EMPATE")
39     continuar = input("Deseja continuar? [s/n]: ")
40     if continuar in ('n','não','nao'):
41         print("Até a próxima")
42         break
```



Lista - Práticas

- Crie um programa que leia o nome e duas notas de um(a) aluno(a). O programa deve calcular a média dessas duas notas e guardar tudo em uma lista composta (tipo uma matriz). No final, o seu programa deve conseguir imprimir o boletim de cada aluno individualmente pelos métodos de manipulação das listas.

```
1 boletim = []
2 alunos = int(input("Quanto alunos? "))
3 cont = 1
4 while cont <= alunos:
5     nome = input(f"Digite o nome do aluno {cont}: ")
6     nota1 = float(input(f"Primeira nota de {nome}: "))
7     nota2 = float(input(f"Segunda nota de {nome}: "))
8     media = (nota1 + nota2) / 2
9     boletim.append([nome,[nota1, nota2],media])
10    cont = cont+1
11
12 print("N: | Alunos | Média")
13 for i, a in enumerate(boletim):
14     print(f"{i+1} |{a[0]:^9}|{a[2]} ")
```



Lista - Práticas

- Criar um programa para gerar jogos da MegaSena, onde o usuário vai informar apenas quantos jogos quer fazer e irão ser criados jogos com 6 jogos e ordenados.

```
1  from random import randint
2
3  n = int(input("Quantos jogos deseja fazer? "))
4  megasena = []
5  for i in range(n):
6      jogo = []
7      for i in range(6):
8          jogo.append(randint(1,60))
9          jogo.sort()
10     megasena.append(jogo)
11     cont = 1
12     for i in megasena:
13         print("Jogo",cont,"=",i)
14         cont += 1
```