

LISTAS



LISTAS

LISTAS

O QUE SÃO LISTAS ?

- Listas são **um tipo de variável que permite o armazenamento de vários valores**, acessados por um índice.
- Uma lista pode conter zero ou mais elementos de um mesmo tipo ou de tipos diversos, podendo conter outras listas.
- O tamanho é igual à quantidade de elementos que nela contem.



Lista é análogo a um trem de carga (lista) no qual cada vagão (índice) pode armazenar um valor, uma quantidade de informação. Enumerando os vagões (índice) de 0 a n valores;

A locomotiva recebe um nome (L) e cada vagão um numero [1, 2, 3, 4, ... N]

Pode-se acessar os vagões da seguinte forma:
L[0], L[1], até o ultimo vagão L[n]

LISTAS

CARACTERISTICAS:

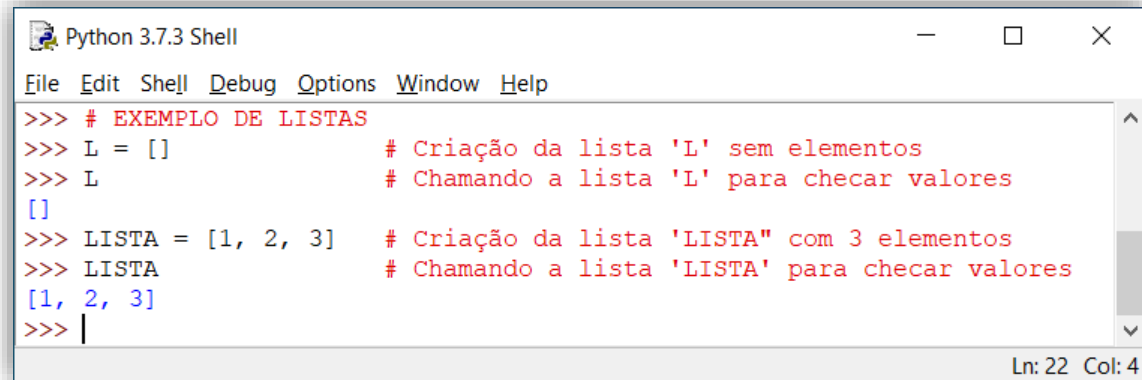
NOME_LISTA =

[0]	[1]	[2]	[3]	[4]	[5]	...	[n]
Valor 1	Valor 2	Valor 3	Valor 4	Valor 5	Valor 6	...	Valor n+1

SINTAXES:

```
<NOME_LISTA> = [<VALOR1>, <VALOR2>, <VALOR3>, ... <VALORn>]
```

- Uma lista é definida por uma sequencia de valores separados por virgulas e envolvidos por colchetes.
- Listas são flexíveis, podem crescer ou diminuir com o tempo.



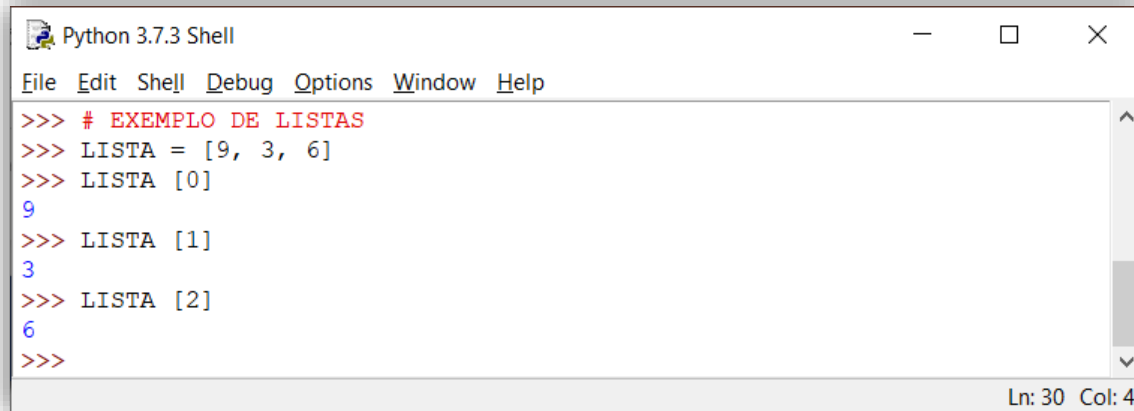
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # EXEMPLO DE LISTAS
>>> L = []                                # Criação da lista 'L' sem elementos
>>> L                                     # Chamando a lista 'L' para checar valores
[]
>>> LISTA = [1, 2, 3]                    # Criação da lista 'LISTA' com 3 elementos
>>> LISTA                                # Chamando a lista 'LISTA' para checar valores
[1, 2, 3]
>>> |
```

Ln: 22 Col: 4

LISTAS

ACESSANDO O CONTEUDO DE UMA LISTA:

- Uma lista “LISTA” com 3 elementos, tem tamanho 3.
- Para acessar os elementos, através do índice, necessita tomar cuidado com a ordem numérica dos índice. Como o primeiro índice é 0, 1 e 2 sucessivamente. Para acessar o ultimo elemento o ultimo será o tamanho da lista menos um, isso é, **LISTA [2]**.

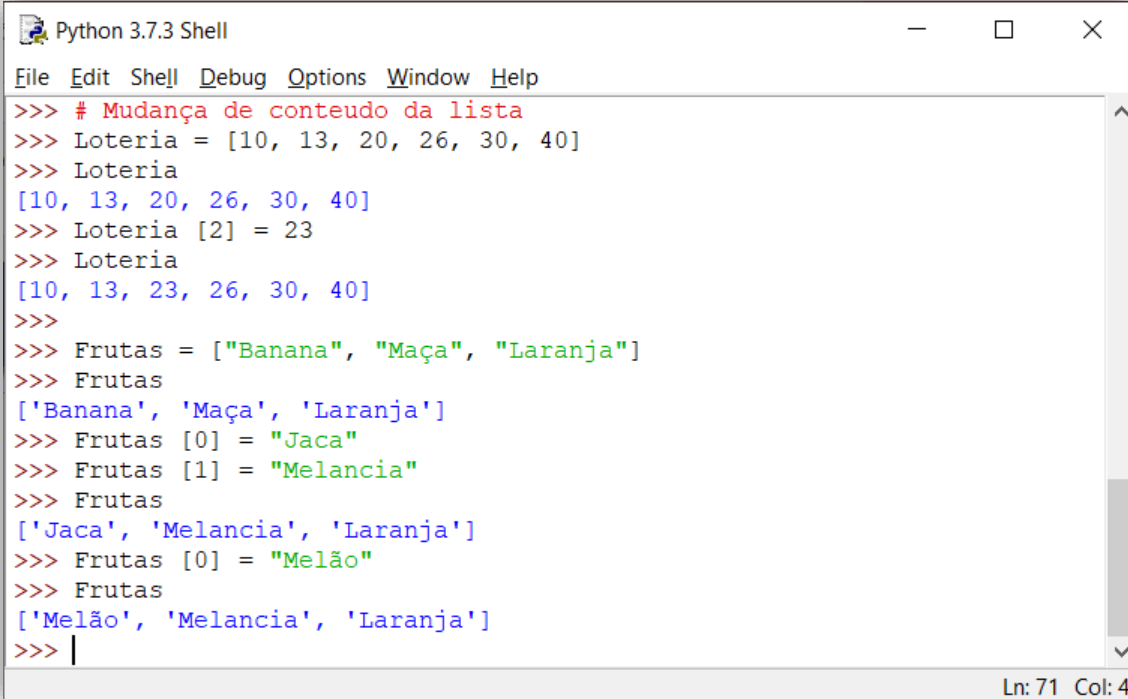


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # EXEMPLO DE LISTAS
>>> LISTA = [9, 3, 6]
>>> LISTA [0]
9
>>> LISTA [1]
3
>>> LISTA [2]
6
>>>
Ln: 30 Col: 4
```

LISTAS

ALTERANDO O CONTEUDO DE UMA LISTA:

- Para alterar o conteúdo de uma lista, deve utilizar o nome da lista e o índice desejado, mudando o conteúdo do elemento:

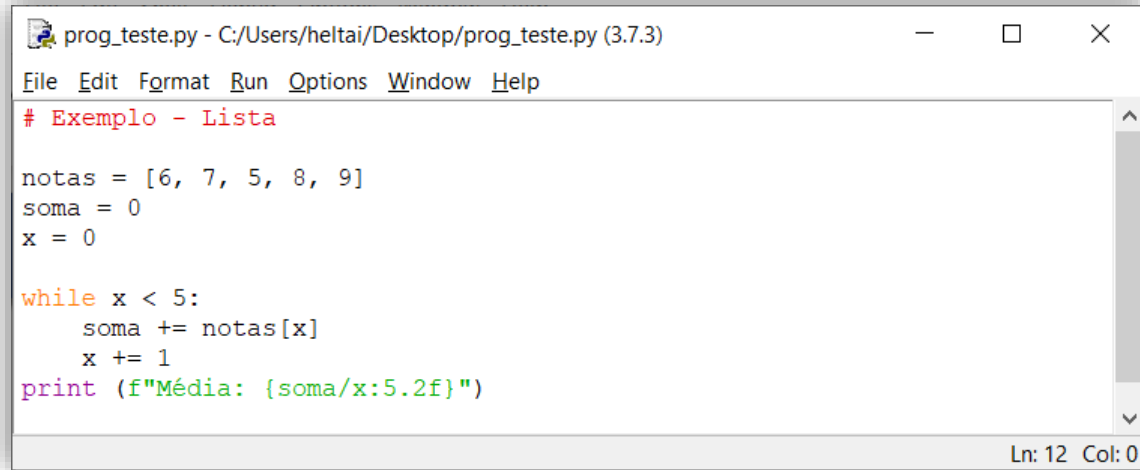


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Mudança de conteúdo da lista
>>> Loteria = [10, 13, 20, 26, 30, 40]
>>> Loteria
[10, 13, 20, 26, 30, 40]
>>> Loteria [2] = 23
>>> Loteria
[10, 13, 23, 26, 30, 40]
>>>
>>> Frutas = ["Banana", "Maça", "Laranja"]
>>> Frutas
['Banana', 'Maça', 'Laranja']
>>> Frutas [0] = "Jaca"
>>> Frutas [1] = "Melancia"
>>> Frutas
['Jaca', 'Melancia', 'Laranja']
>>> Frutas [0] = "Melão"
>>> Frutas
['Melão', 'Melancia', 'Laranja']
>>> |
```

Ln: 71 Col: 4

LISTAS

EXEMPLO: Programa calcule a média aritmética de 5 notas de um aluno



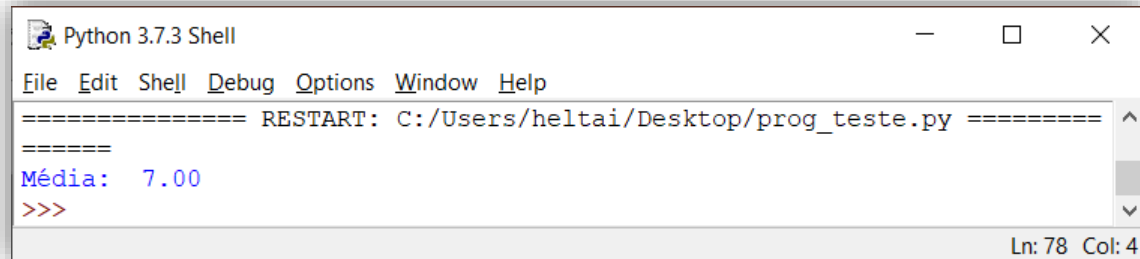
The screenshot shows a Python IDE window titled "prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code is as follows:

```
# Exemplo - Lista

notas = [6, 7, 5, 8, 9]
soma = 0
x = 0

while x < 5:
    soma += notas[x]
    x += 1
print (f"Média: {soma/x:5.2f}")
```

The status bar at the bottom right indicates "Ln: 12 Col: 0".



The screenshot shows a "Python 3.7.3 Shell" window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The output of the script is displayed:

```
===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====
=====
Média:  7.00
>>>
```

The status bar at the bottom right indicates "Ln: 78 Col: 4".



TRABALHANDO COM INDICES

INDICES – ACESSO DESLOCADO

ACESSO AO CONTEUDO DA LISTA – DESLOCANDO INDICE:

- É natural que o usuário conte na sequência: 1, 2, 3, ... Para que isso seja “resolvido” na seleção do conteúdo de uma lista, a solução é adicionar o valor 1. Observe que isso não é uma solução e sim um “mascaramento”

EXEMPLO: Entre com 5 valores e selecione um dos números para ser impresso de forma continua. Para sair há necessidade de digitar 0 (zero).

```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)
File Edit Format Run Options Window Help
# Exemplo - Trabalhando com Lista

numeros = [0,0,0,0,0]
x = 0

while x < 5:
    numeros[x] = float(input(f"Numero {x+1}: "))
    x += 1

while True:
    escolhido = int(input("Que posição você quer imprimir (0 para sair): "))
    if escolhido == 0:
        break
    print (f"Você escolheu o numero; {numeros[escolhido - 1]}")

Ln: 17 Col: 0
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====
Numero 1: 3.5
Numero 2: 2.0
Numero 3: 1
Numero 4: 8.443
Numero 5: 10.22
Que posição você quer imprimir (0 para sair): 1
Você escolheu o numero; 3.5
Que posição você quer imprimir (0 para sair): 4
Você escolheu o numero; 8.443
Que posição você quer imprimir (0 para sair): 5
Você escolheu o numero; 10.22
Que posição você quer imprimir (0 para sair): 0
>>> |

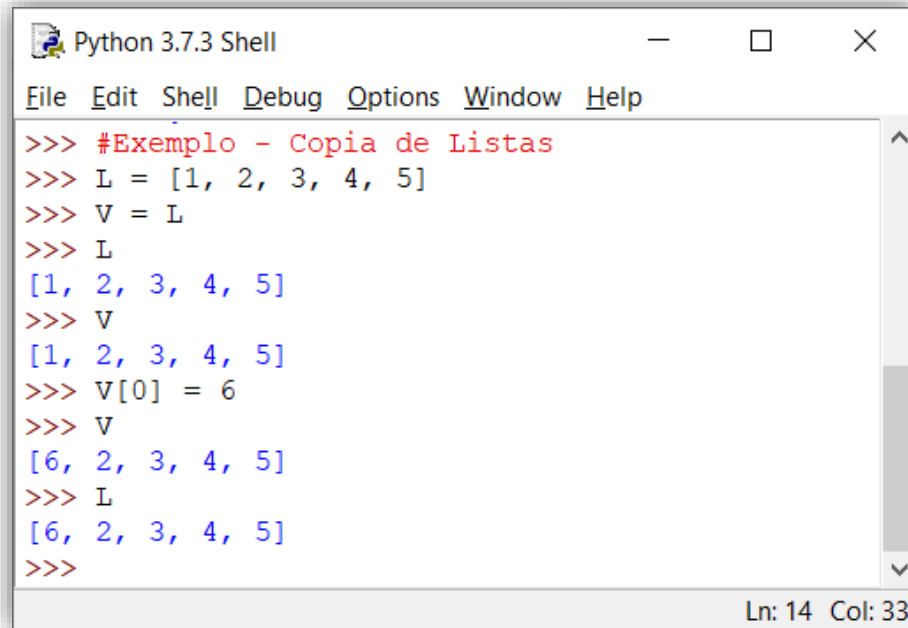
Ln: 17 Col: 4
```

INDICES – COPIA DE LISTA

COPIA DE LISTAS:

- Quando se realiza a copia de uma lista, a modificação de uma lista altera a outra lista copiada. Isso é causada pelo fato de uma lista ser um objeto e, quando atribui um objeto a outro, está se copiando a referencia da lista e não seus dados em si.

EXEMPLO: Programa que copia e modifica duas listas distintas.

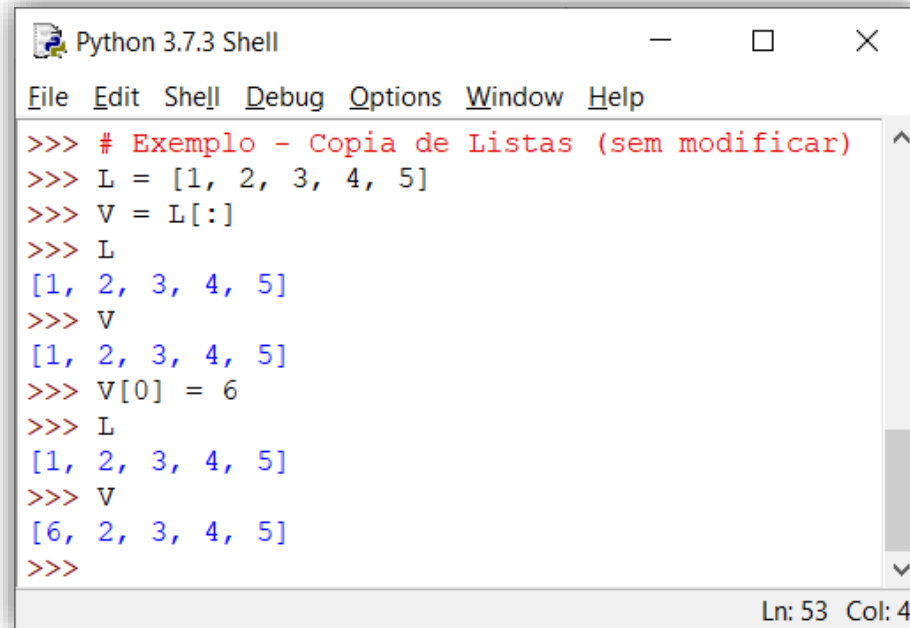
A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text 'Python 3.7.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a Python script. The script starts with a comment '#Exemplo - Copia de Listas' in red. It then defines a list 'L' with values [1, 2, 3, 4, 5]. Next, it assigns 'V' to 'L'. The script then prints 'L' and 'V', both showing as [1, 2, 3, 4, 5]. Then, it modifies 'V[0]' to 6. Finally, it prints 'V' and 'L' again, both showing as [6, 2, 3, 4, 5]. The status bar at the bottom right of the window shows 'Ln: 14 Col: 33'.

```
>>> #Exemplo - Copia de Listas
>>> L = [1, 2, 3, 4, 5]
>>> V = L
>>> L
[1, 2, 3, 4, 5]
>>> V
[1, 2, 3, 4, 5]
>>> V[0] = 6
>>> V
[6, 2, 3, 4, 5]
>>> L
[6, 2, 3, 4, 5]
>>>
```

INDICES – COPIA DE LISTA

- Para criar uma copia independente de uma lista, utiliza-se a sintaxe `[:]`

EXEMPLO: Programa que copia e sem modifica as duas listas.

A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text 'Python 3.7.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a Python script. The script starts with a comment: '>>> # Exemplo - Copia de Listas (sem modificar)'. It then defines a list L: '>>> L = [1, 2, 3, 4, 5]'. Next, it creates a copy of L: '>>> V = L[:]'. Then, it prints L: '>>> L' followed by the output '[1, 2, 3, 4, 5]'. It prints V: '>>> V' followed by the output '[1, 2, 3, 4, 5]'. Then, it modifies V: '>>> V[0] = 6'. It prints L again: '>>> L' followed by the output '[1, 2, 3, 4, 5]'. It prints V again: '>>> V' followed by the output '[6, 2, 3, 4, 5]'. The prompt '>>>' is shown at the end. At the bottom right of the window, the status bar shows 'Ln: 53 Col: 4'.

```
>>> # Exemplo - Copia de Listas (sem modificar)
>>> L = [1, 2, 3, 4, 5]
>>> V = L[:]
>>> L
[1, 2, 3, 4, 5]
>>> V
[1, 2, 3, 4, 5]
>>> V[0] = 6
>>> L
[1, 2, 3, 4, 5]
>>> V
[6, 2, 3, 4, 5]
>>>
```

- Ao escrever `L[:]` está se copiando uma nova copia de `L`. Assim `L` e `V` se referem a áreas diferentes na memoria, permitindo alterá-las de forma independente

INDICES – FATIAMENTO DE LISTA

FATIAMENTO DE LISTAS:

- Da mesma forma que foi feito com string, com lista é possível fazer o fatiamento.

CÓDIGO	OBJETIVO
Lista [i]	Seleciona o elemento de índice i da lista Lista
Lista [i:j]	Seleciona os elementos da lista Lista cujos índices estão compreendidos entre i e j-1
Lista [:j]	Seleciona os elementos da lista Lista do início até o elemento j-1 da lista
Lista [i:]	Seleciona os elementos da lista Lista do índice i até o final da lista
Lista [-i:]	Seleciona os i últimos elementos da lista Lista

EXEMPLO: Programa que fatia de diversas formas uma lista (igual string).

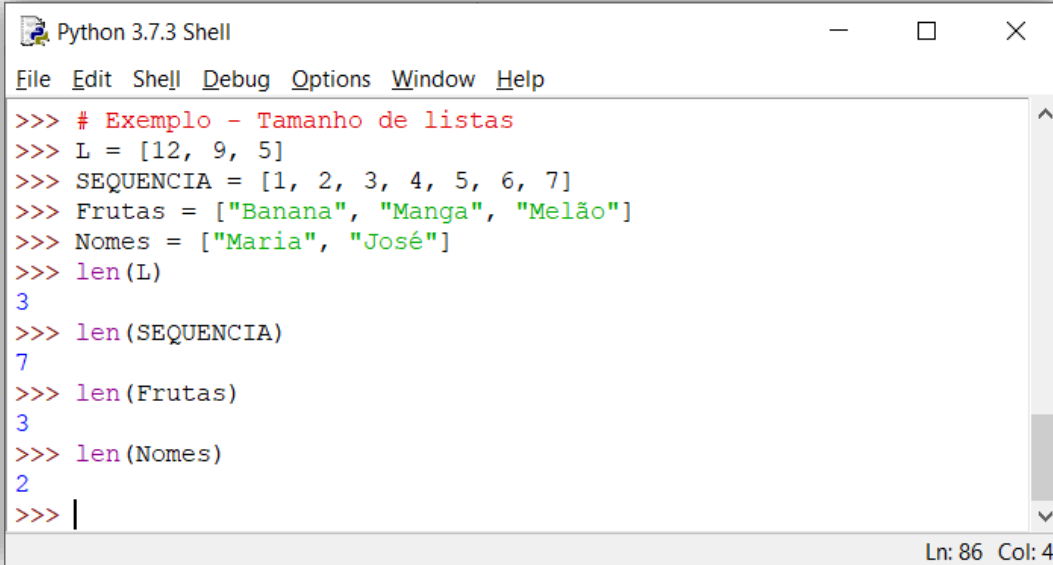
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Exemplo - Fatiamento de lista
>>> L = [1, 2, 3, 4, 5]
>>> L [0:5]
[1, 2, 3, 4, 5]
>>> L [:5]
[1, 2, 3, 4, 5]
>>> L[:-1]
[1, 2, 3, 4]
>>> L [1:3]
[2, 3]
>>> L [1:4]
[2, 3, 4]
>>> L [3:]
[4, 5]
>>> L [:3]
[1, 2, 3]
>>> L [-1]
5
>>> L [-2]
4
>>> |
```

Ln: 73 Col: 4

INDICES – TAMANHO DE LISTA

TAMANHO DE LISTAS:

- Usa-se a função **len** com listas. O valor retornado é igual ao numero de elementos da lista.

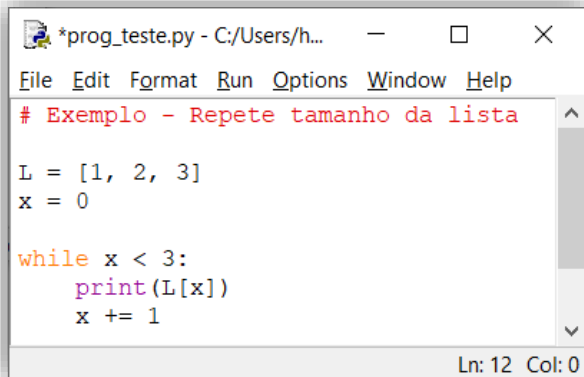


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Exemplo - Tamanho de listas
>>> L = [12, 9, 5]
>>> SEQUENCIA = [1, 2, 3, 4, 5, 6, 7]
>>> Frutas = ["Banana", "Manga", "Melão"]
>>> Nomes = ["Maria", "José"]
>>> len(L)
3
>>> len(SEQUENCIA)
7
>>> len(Frutas)
3
>>> len(Nomes)
2
>>> |
```

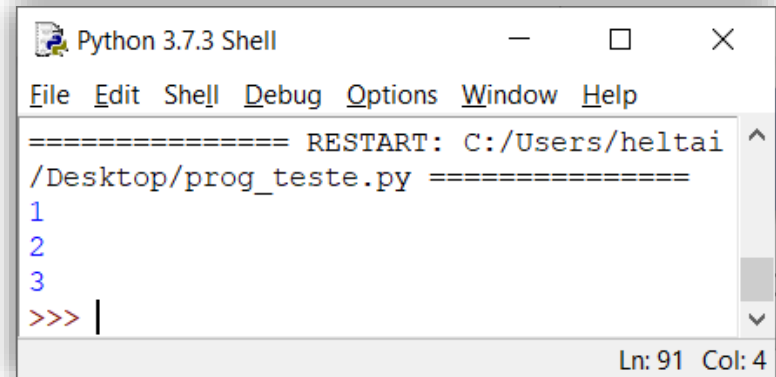
Ln: 86 Col: 4

INDICES – TAMANHO DE LISTA

EXEMPLO: Programa controla o numero de repetições pelo tamanho da lista (primeira opção).

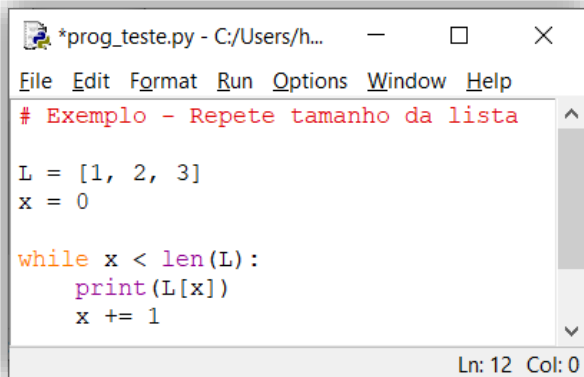


```
*prog_teste.py - C:/Users/h...  
File Edit Format Run Options Window Help  
# Exemplo - Repete tamanho da lista  
  
L = [1, 2, 3]  
x = 0  
  
while x < 3:  
    print(L[x])  
    x += 1  
  
Ln: 12 Col: 0
```

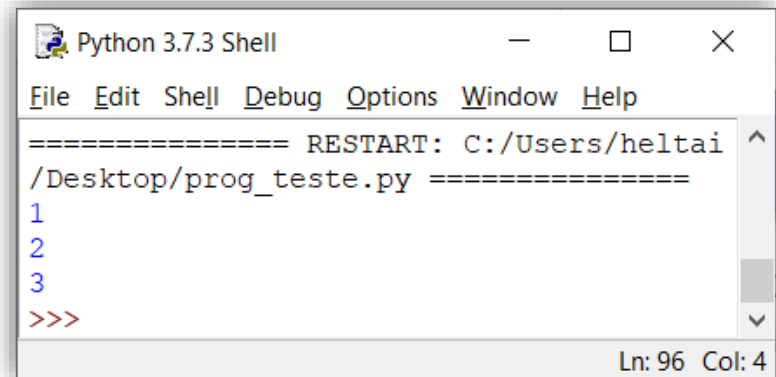


```
Python 3.7.3 Shell  
File Edit Shell Debug Options Window Help  
===== RESTART: C:/Users/hel...  
/Desktop/prog_teste.py =====  
1  
2  
3  
>>> |  
  
Ln: 91 Col: 4
```

EXEMPLO: Programa controla o numero de repetições pelo tamanho da lista (segunda opção). A grande vantagem da 2ª opção é o fato que se trocar o tamanho de L o programa continua funcionando normal.



```
*prog_teste.py - C:/Users/h...  
File Edit Format Run Options Window Help  
# Exemplo - Repete tamanho da lista  
  
L = [1, 2, 3]  
x = 0  
  
while x < len(L):  
    print(L[x])  
    x += 1  
  
Ln: 12 Col: 0
```



```
Python 3.7.3 Shell  
File Edit Shell Debug Options Window Help  
===== RESTART: C:/Users/hel...  
/Desktop/prog_teste.py =====  
1  
2  
3  
>>>  
  
Ln: 96 Col: 4
```


INDICES – ADIÇÃO DE ELEMENTOS

ADIÇÃO DE ELEMENTOS:

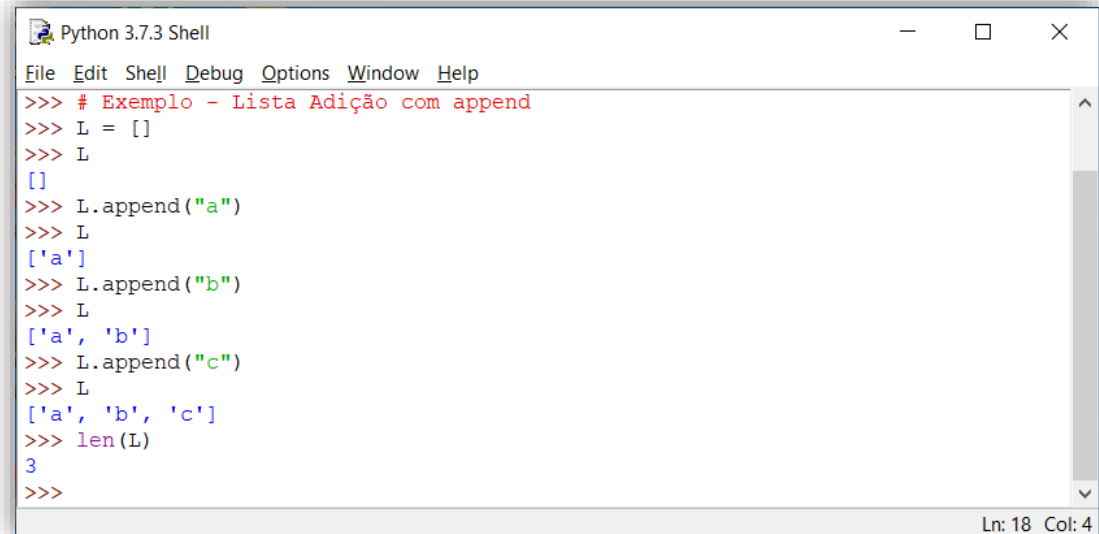
- Para adicionar um elemento na lista utiliza-se dois métodos **append** ou **insert**. Em Python, chama-se um método escrevendo o nome dele após o nome do objeto (lista é um objeto).

append

- Para invocar o método **append** deve seguir a seguinte sintaxe:

SINTAXE:

nome_lista.append(valor)



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Exemplo - Lista Adição com append
>>> L = []
>>> L
[]
>>> L.append("a")
>>> L
['a']
>>> L.append("b")
>>> L
['a', 'b']
>>> L.append("c")
>>> L
['a', 'b', 'c']
>>> len(L)
3
>>>
```

Ln: 18 Col: 4

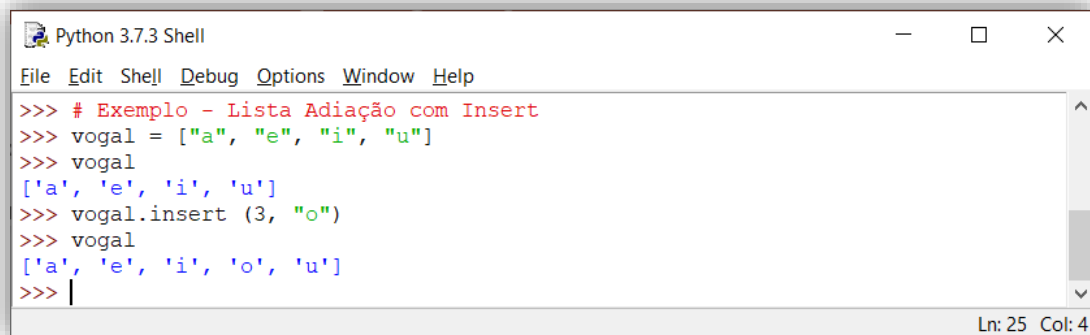
INDICES – ADIÇÃO DE ELEMENTOS

insert

- Para invocar o método **insert** deve seguir a seguinte sintaxe:

nome_lista.insert(local, valor)

- Onde local é a posição na lista que deseja inserir.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Exemplo - Lista Adição com Insert
>>> vogal = ["a", "e", "i", "u"]
>>> vogal
['a', 'e', 'i', 'u']
>>> vogal.insert(3, "o")
>>> vogal
['a', 'e', 'i', 'o', 'u']
>>> |
```

Ln: 25 Col: 4

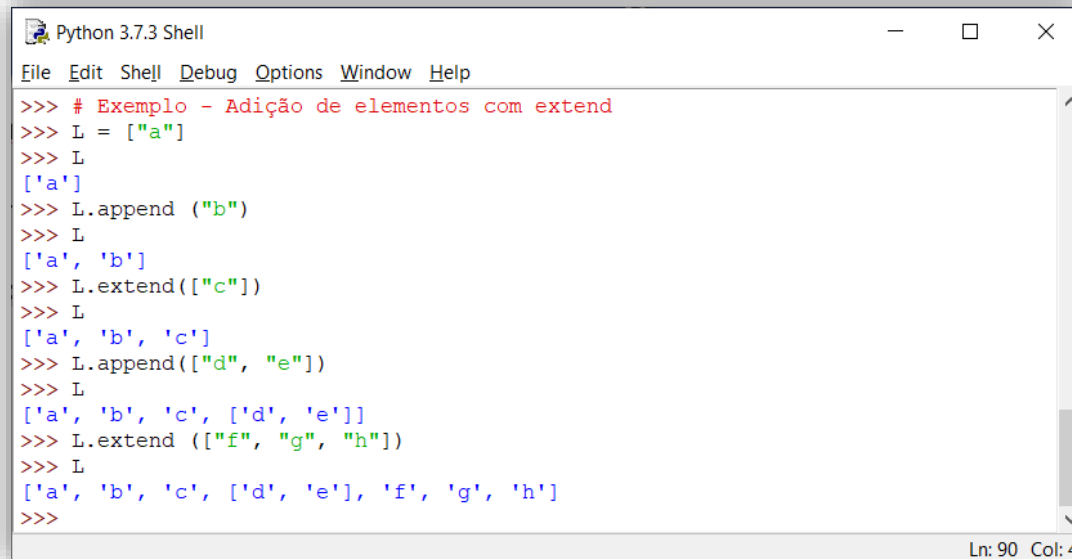
DIFERENÇA ENTRE APPEND E INSERT:

- append** – Insere um novo elemento no **final da lista**. Tem apenas um argumento (valor a inserir)
- insert** – Insere um novo elemento em **qualquer posição**. Tem dois argumentos (onde, valor)

INDICES – ADIÇÃO DE ELEMENTOS

extend

- O método **extend** adiciona uma lista em outra lista. Não aceita outro parâmetro que não for lista.
- Utilizando o método **append** com uma lista com parâmetros, em vez de adicionar os elementos no fim da lista, **append** adicionará a lista inteira, mas como apenas um novo elemento. Com o uso de **extend** a lista adicionada com parâmetros, será adicionado os elementos. (muito utilizado em data Science).

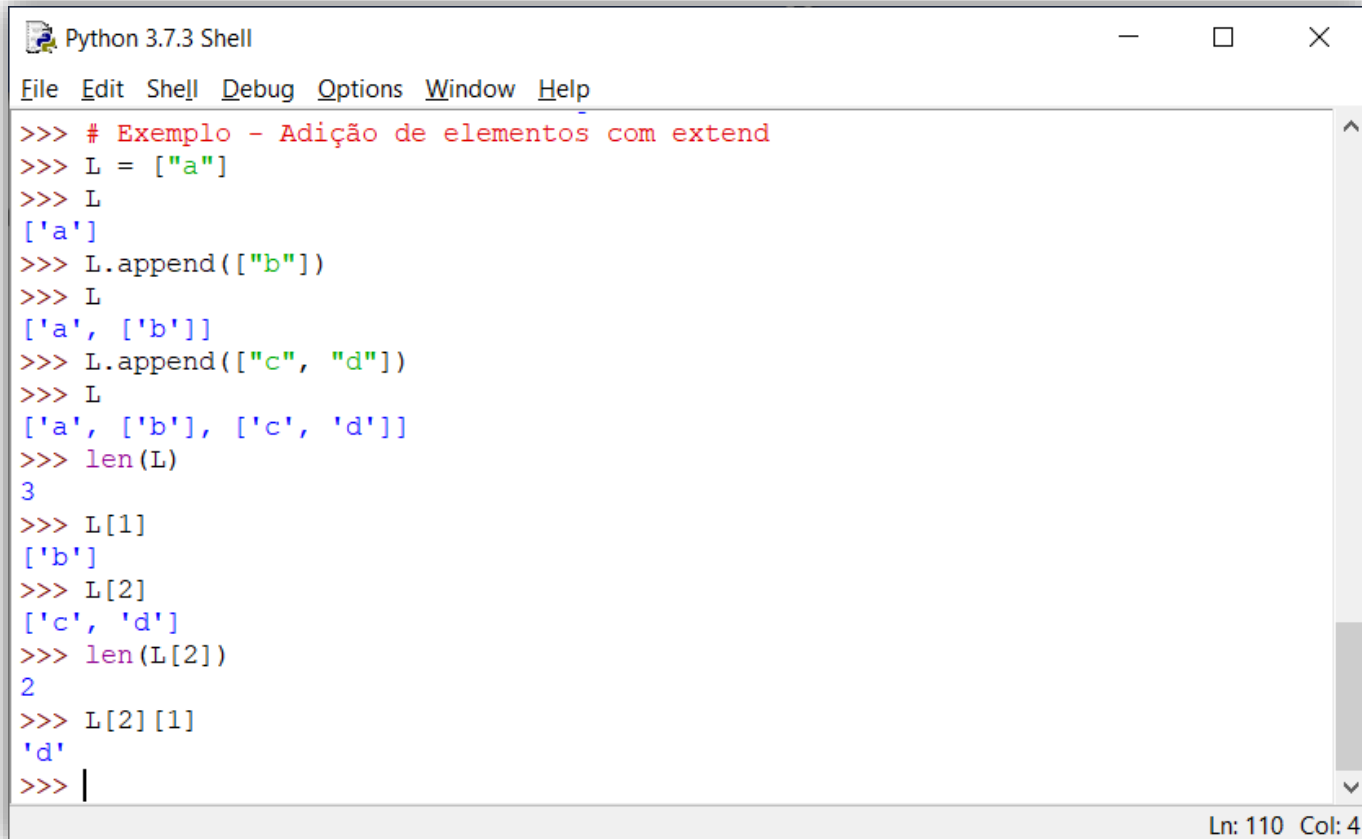
A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text 'Python 3.7.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a Python script with the following code:

```
>>> # Exemplo - Adição de elementos com extend
>>> L = ["a"]
>>> L
['a']
>>> L.append("b")
>>> L
['a', 'b']
>>> L.extend(["c"])
>>> L
['a', 'b', 'c']
>>> L.append(["d", "e"])
>>> L
['a', 'b', 'c', ['d', 'e']]
>>> L.extend(["f", "g", "h"])
>>> L
['a', 'b', 'c', ['d', 'e'], 'f', 'g', 'h']
>>>
```

The code demonstrates the use of the `append` and `extend` methods on a list `L`. It starts with `L = ["a"]`. Then `L.append("b")` is called, resulting in `['a', 'b']`. Next, `L.extend(["c"])` is called, resulting in `['a', 'b', 'c']`. Then `L.append(["d", "e"])` is called, resulting in `['a', 'b', 'c', ['d', 'e']]`. Finally, `L.extend(["f", "g", "h"])` is called, resulting in `['a', 'b', 'c', ['d', 'e'], 'f', 'g', 'h']`. The status bar at the bottom right of the window shows 'Ln: 90 Col: 4'.

INDICES – ADIÇÃO DE ELEMENTOS

- O conceito do método **extend** permite a utilização de estruturas de dados complexas, como matrizes, árvores e registros (conhecimentos futuros em estrutura de dados).

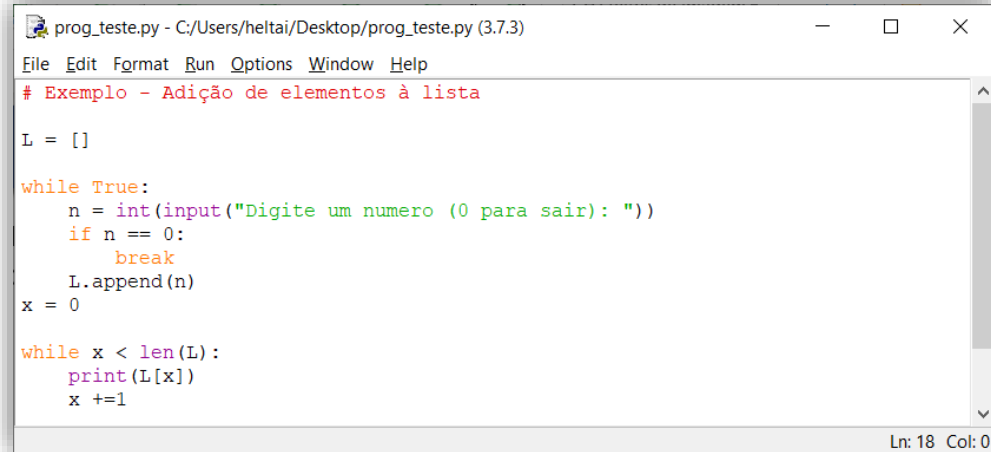


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Exemplo - Adição de elementos com extend
>>> L = ["a"]
>>> L
['a']
>>> L.append(["b"])
>>> L
['a', ['b']]
>>> L.append(["c", "d"])
>>> L
['a', ['b'], ['c', 'd']]
>>> len(L)
3
>>> L[1]
['b']
>>> L[2]
['c', 'd']
>>> len(L[2])
2
>>> L[2][1]
'd'
>>> |
```

Ln: 110 Col: 4

INDICES – ADIÇÃO DE ELEMENTOS

EXEMPLO: Programa que adiciona elemento à lista de forma continua e imprime os elementos após encerrado. Ao digitar 0 (zero) o programa é encerrado.



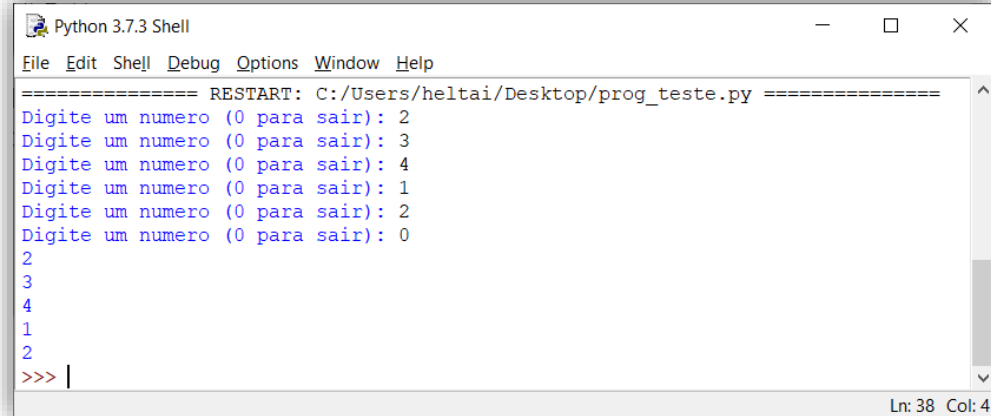
```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)
File Edit Format Run Options Window Help
# Exemplo - Adição de elementos à lista

L = []

while True:
    n = int(input("Digite um numero (0 para sair): "))
    if n == 0:
        break
    L.append(n)
x = 0

while x < len(L):
    print(L[x])
    x +=1

Ln: 18 Col: 0
```



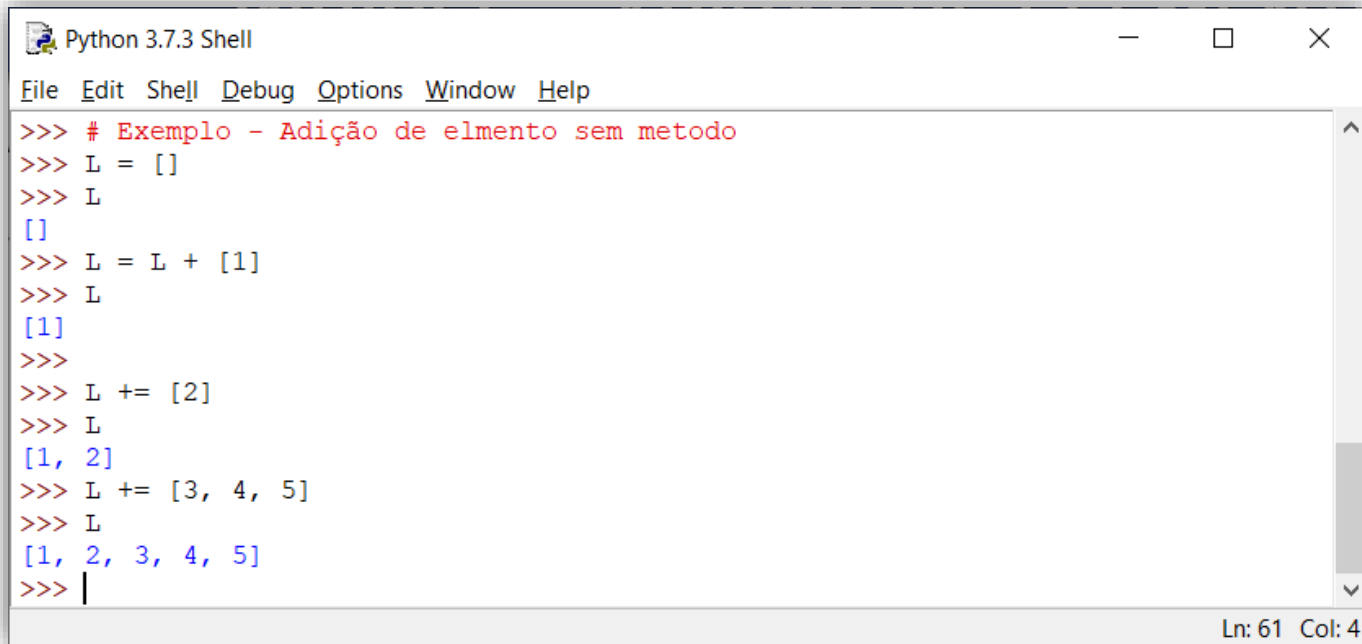
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====
Digite um numero (0 para sair): 2
Digite um numero (0 para sair): 3
Digite um numero (0 para sair): 4
Digite um numero (0 para sair): 1
Digite um numero (0 para sair): 2
Digite um numero (0 para sair): 0
2
3
4
1
2
>>> |

Ln: 38 Col: 4
```

INDICES – ADIÇÃO DE ELEMENTOS

CONCATENAÇÃO DE LISTAS

- Outra forma de adicionar elementos a uma lista é através da concatenação que é utilizando (+)

A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text 'Python 3.7.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a Python script demonstrating list concatenation. The script starts with a comment '# Exemplo - Adição de elemento sem metodo' in red. It then initializes an empty list 'L' and prints it. Next, it appends the value 1 to the list using 'L = L + [1]' and prints the result. Then, it appends the value 2 using 'L += [2]' and prints the result. Finally, it appends the list [3, 4, 5] using 'L += [3, 4, 5]' and prints the final list. The output shows the list growing from empty to [1], then [1, 2], and finally [1, 2, 3, 4, 5]. The status bar at the bottom right shows 'Ln: 61 Col: 4'.

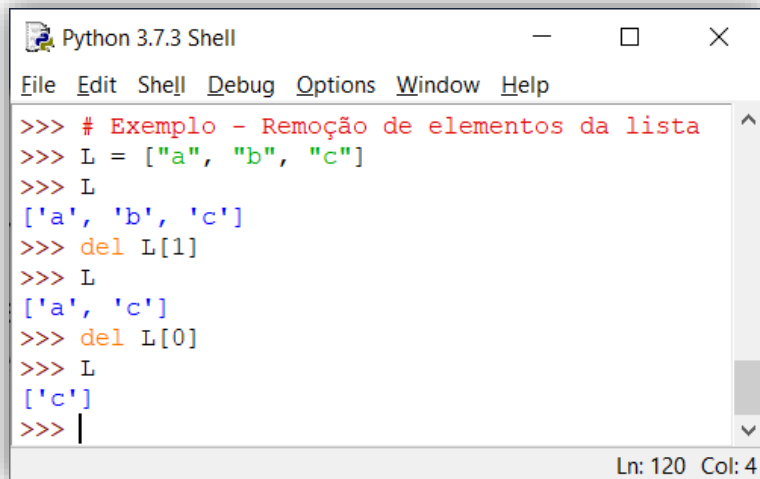
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Exemplo - Adição de elemento sem metodo
>>> L = []
>>> L
[]
>>> L = L + [1]
>>> L
[1]
>>>
>>> L += [2]
>>> L
[1, 2]
>>> L += [3, 4, 5]
>>> L
[1, 2, 3, 4, 5]
>>> |
Ln: 61 Col: 4
```


INDICES – REMOÇÃO DE ELEMENTOS

REMOÇÃO DE ELEMENTOS DA LISTA:

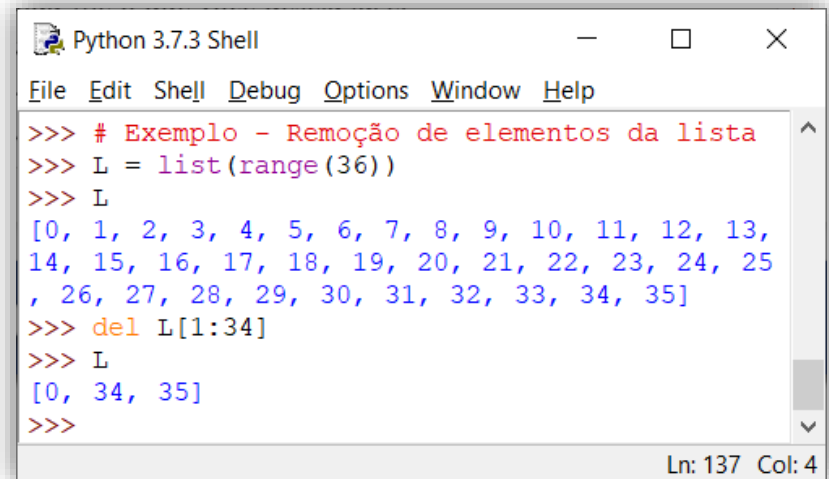
COMANDO DEL

- Para remover elementos de uma lista ou todos os elementos utilizando a instrução **del**.
- O elemento excluído não ocupa mais lugar na lista e a lista é reorganizado e passa a ser calculado sem o elemento deletado.
- A função **range** gera uma sequencia de elementos, mas um a cada chamada, sendo o que chama-se de generator. A função **list** é utilizada para transformar o resultado dessa função em uma lista.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Exemplo - Remoção de elementos da lista
>>> L = ["a", "b", "c"]
>>> L
['a', 'b', 'c']
>>> del L[1]
>>> L
['a', 'c']
>>> del L[0]
>>> L
['c']
>>> |
```

Ln: 120 Col: 4



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> # Exemplo - Remoção de elementos da lista
>>> L = list(range(36))
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]
>>> del L[1:34]
>>> L
[0, 34, 35]
>>>
```

Ln: 137 Col: 4

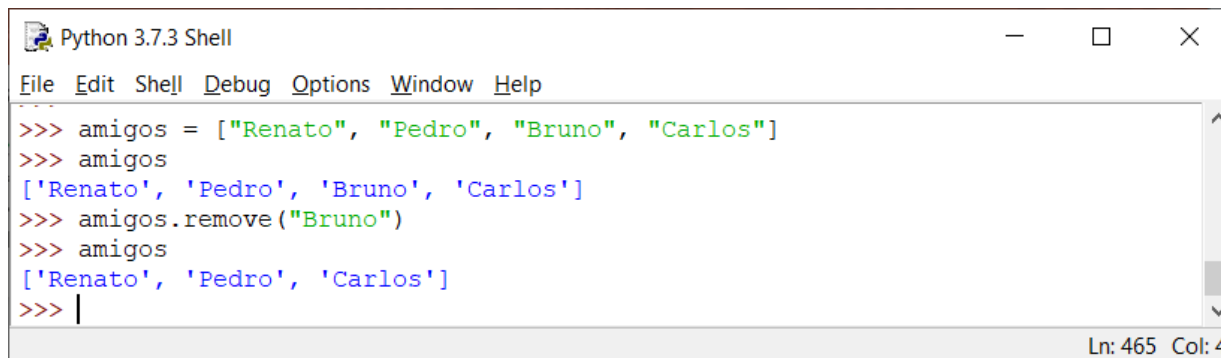
INDICES – REMOÇÃO DE ELEMENTOS

COMANDO REMOVE

- O método **remove** remove o primeiro elemento da lista correspondente ao argumento.

SINTAXE:

nome_lista.remove(conteúdo)



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> amigos = ["Renato", "Pedro", "Bruno", "Carlos"]
>>> amigos
['Renato', 'Pedro', 'Bruno', 'Carlos']
>>> amigos.remove("Bruno")
>>> amigos
['Renato', 'Pedro', 'Carlos']
>>> |
```

Ln: 465 Col: 4

DIFERENÇA ENTRE POP E REMOVE:

- **pop** – Remove da lista o primeiro elemento correspondente ao valor informado. Se houver outros valores iguais, eles permanecem na lista.
- **remove** – Remove da lista o elemento correspondente ao índice informado

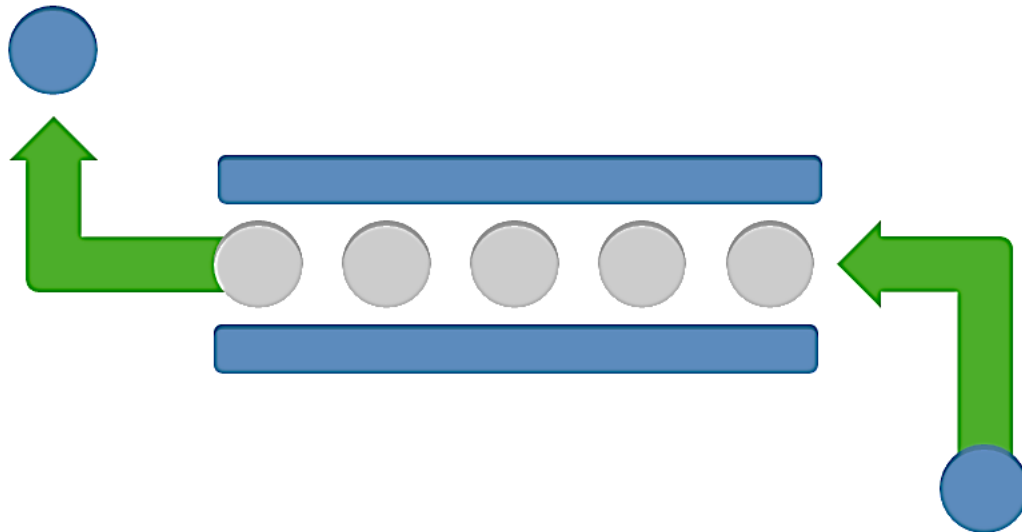


LISTA COMO FILA

LISTAS COMO FILA

USANDO LISTA COMO FILAS:

- Uma lista pode ser utilizada como fila se obedecer a certas regras de inclusão e eliminação de elementos.
- Em uma fila, a **inclusão é sempre realizada no fim**, e as **remoções são feitas no início**. “*Primeiro a chegar é o último a sair*” FIFO – *First In First Out*
- Para se utilizar esse sistema, adota-se o método **pop** que **retorna o valor do elemento e o exclui da lista**.



LISTA COMO FILA

EXEMPLO: Simulador de fila de banco com 10 clientes.

```
*prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)*
File Edit Format Run Options Window Help
# Exemplo - Simulador de fila de banco

ultimo = 10
fila = list(range(1, ultimo + 1))

while True:
    print(f"\nExistem {len(fila)} clientes na fila")
    print(f"Fila atual: {fila}")
    print("Digite F para adicionar um cliente ao fim da fila,")
    print("ou A para realizar o atendimento. S para sair")
    operação = input("Operação (F, A ou S):")
    if operação == "A":
        if len(fila) > 0:
            atendimento = fila.pop(0)
            print(f"Cliente {atendimento} atendido")
        else:
            print("Fila vazia! Ninguém para atender.")
    elif operação == "F":
        ultimo += 1
        fila.append(ultimo)
    elif operação == "S":
        break
    else:
        print("Operação inválida! Digite apenas F, A ou S")
```

Ln: 29 Col: 0

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====

Existem 10 clientes na fila
Fila atual: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Digite F para adicionar um cliente ao fim da fila, ou A para realizar o atendimento. S para sair
Operação (F, A ou S):F

Existem 11 clientes na fila
Fila atual: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
Digite F para adicionar um cliente ao fim da fila, ou A para realizar o atendimento. S para sair
Operação (F, A ou S):F

Existem 12 clientes na fila
Fila atual: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Digite F para adicionar um cliente ao fim da fila, ou A para realizar o atendimento. S para sair
Operação (F, A ou S):A
Cliente 1 atendido

Existem 11 clientes na fila
Fila atual: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Digite F para adicionar um cliente ao fim da fila, ou A para realizar o atendimento. S para sair
Operação (F, A ou S):A
Cliente 2 atendido

Existem 10 clientes na fila
Fila atual: [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Digite F para adicionar um cliente ao fim da fila, ou A para realizar o atendimento. S para sair
Operação (F, A ou S):S
>>>
```

Ln: 32 Col: 4

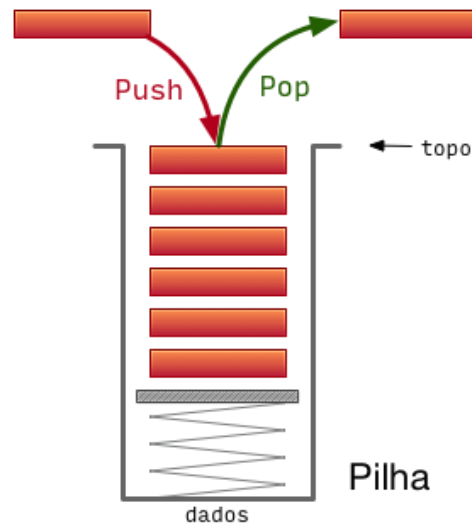


LISTA COMO PILHA

LISTA COMO PILHA

USANDO LISTA COMO PILHAS:

- Uma pilha tem uma politica de acesso: novos elementos são adicionados ao topo. A retirada de elementos também é feita pelo topo.
- Na pilha, “***o ultimo elemento a chegar é o primeiro a sair***”. LIFO – *Last In First Out*.
- Para que essa operação seja possível, o parâmetro de **pop** será **-1**



LISTA COMO PILHA

EXEMPLO: Simulador de pilha de pratos.

```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)
File Edit Format Run Options Window Help

# Exemplo - Simulador de Pilha de Prato

prato = 5
pilha = list(range(1, prato + 1))

while True:
    print(f"\nExistem {len(pilha)} pratos na pilha")
    print(f"Pilha Atual: {pilha}")
    print("Digite E para empilhar um novo prato,")
    print("ou D para desempilhar. S para sair.")

    operação = input("Operação (E, D, ou S): ")

    if operação == "D":
        if len(pilha) > 0:
            lavado = pilha.pop(-1)
            print(f"Prato {lavado} lavado")
        else:
            print("Pilha vazia! Nada para lavar.")

    elif operação == "E":
        prato += 1 #novo prato
        pilha.append(prato)

    elif operação == "S":
        break

    else:
        print("Operação invalida! Digite apenas E, D ou S")

Ln: 32 Col: 0
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help

===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====

Existem 5 pratos na pilha
Pilha Atual: [1, 2, 3, 4, 5]
Digite E para empilhar um novo prato,
ou D para desempilhar. S para sair.
Operação (E, D, ou S): E

Existem 6 pratos na pilha
Pilha Atual: [1, 2, 3, 4, 5, 6]
Digite E para empilhar um novo prato,
ou D para desempilhar. S para sair.
Operação (E, D, ou S): E

Existem 7 pratos na pilha
Pilha Atual: [1, 2, 3, 4, 5, 6, 7]
Digite E para empilhar um novo prato,
ou D para desempilhar. S para sair.
Operação (E, D, ou S): D
Prato 7 lavado

Existem 6 pratos na pilha
Pilha Atual: [1, 2, 3, 4, 5, 6]
Digite E para empilhar um novo prato,
ou D para desempilhar. S para sair.
Operação (E, D, ou S): D
Prato 6 lavado

Existem 5 pratos na pilha
Pilha Atual: [1, 2, 3, 4, 5]
Digite E para empilhar um novo prato,
ou D para desempilhar. S para sair.
Operação (E, D, ou S): S
>>>

Ln: 77 Col: 4
```



USO DO COMANDO FOR EM LISTA

USO DO COMANDO FOR EM LISTA

USANDO O FOR EM LISTA

- O comando **for** em lista, a cada repetição é possível utilizar um elemento diferente da lista.
- A cada repetição, o próximo elemento da lista é utilizado, o que se repete até o fim da lista.

```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)
File Edit Format Run Options Window Help
# Exemplo - Uso do for em lista

L = [7, 9, 10, 12]

for e in L:
    print(e)

Ln: 9 Col: 0
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====
7
9
10
12
>>> |

Ln: 9 Col: 4
```

```
*prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)*
File Edit Format Run Options Window Help
# Exemplo - Uso do for em lista

L = [7, 9, 10, 12]
p = int(input("Digite um numero a pesquisar na lista:"))

for e in L:
    if e == p:
        print("Elemento encontrado")
    else:
        print("Elemento não encontrado")

Ln: 13 Col: 0
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====
Digite um numero a pesquisar na lista:1
Elemento não encontrado
Elemento não encontrado
Elemento não encontrado
Elemento não encontrado
>>> |

Ln: 22 Col: 4
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====
Digite um numero a pesquisar na lista:10
Elemento não encontrado
Elemento não encontrado
Elemento encontrado
Elemento não encontrado
>>>

Ln: 29 Col: 4
```



USO DO COMANDO RANGE

USO DO COMANDO RANGE

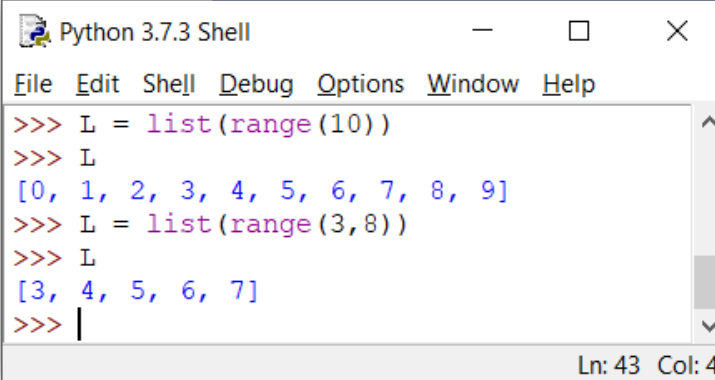
USANDO O COMANDO RANGE:

- O comando **range** é uma função para gerar listas simples.
- A função **range** não retorna uma lista propriamente dita, mas um gerador ou *generator*.
- O comando **range** gera valores a partir de 0, logo, ao especificar um parâmetro 10, o mesmo informa de 0 a 10 (ponto a parar a geração).
- Quando se limita o início e o fim do intervalo, como é um intervalo aberto, isto é, não incluso na faixa de valores o ultimo numero não é mostrado (assim como em fatiar string).

SINTAXE:

range (**fim**)

range (**inicio**, **fim**)

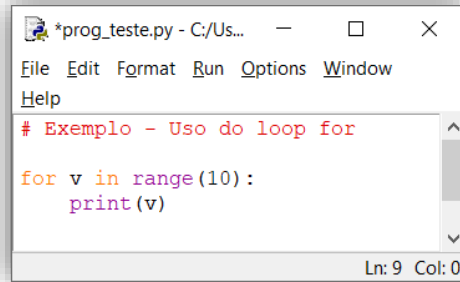


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> L = list(range(10))
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> L = list(range(3,8))
>>> L
[3, 4, 5, 6, 7]
>>> |
```

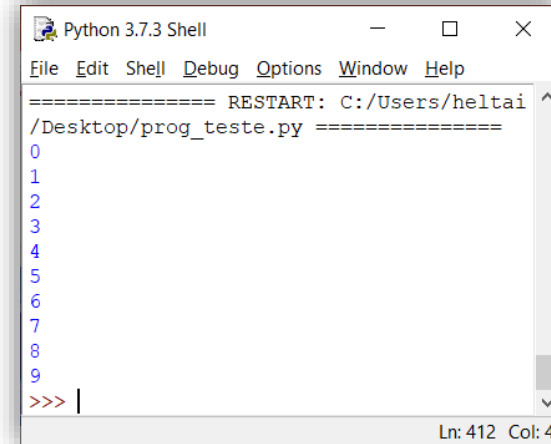
Ln: 43 Col: 4

USO DO COMANDO RANGE

EXEMPLO: Programa que imprime 0 a 9 na tela com o uso do parâmetro **fim** do **range**.

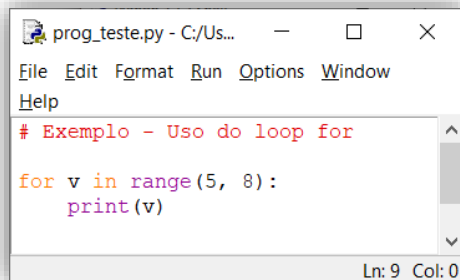


```
*prog_teste.py - C:/Us...  
File Edit Format Run Options Window  
Help  
# Exemplo - Uso do loop for  
  
for v in range(10):  
    print(v)  
  
Ln: 9 Col: 0
```

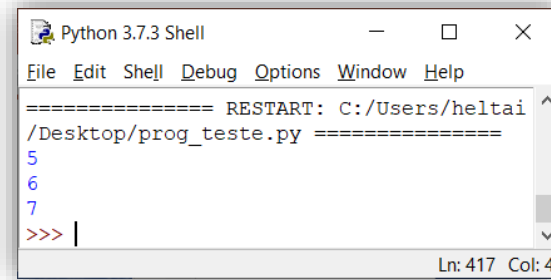


```
Python 3.7.3 Shell  
File Edit Shell Debug Options Window Help  
===== RESTART: C:/Users/heltai  
/Desktop/prog_teste.py =====  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
>>> |  
  
Ln: 412 Col: 4
```

EXEMPLO: Programa que imprime na tela de acordo com o **inicio** e **fim** do comando **range**.



```
prog_teste.py - C:/Us...  
File Edit Format Run Options Window  
Help  
# Exemplo - Uso do loop for  
  
for v in range(5, 8):  
    print(v)  
  
Ln: 9 Col: 0
```



```
Python 3.7.3 Shell  
File Edit Shell Debug Options Window Help  
===== RESTART: C:/Users/heltai  
/Desktop/prog_teste.py =====  
5  
6  
7  
>>> |  
  
Ln: 417 Col: 4
```

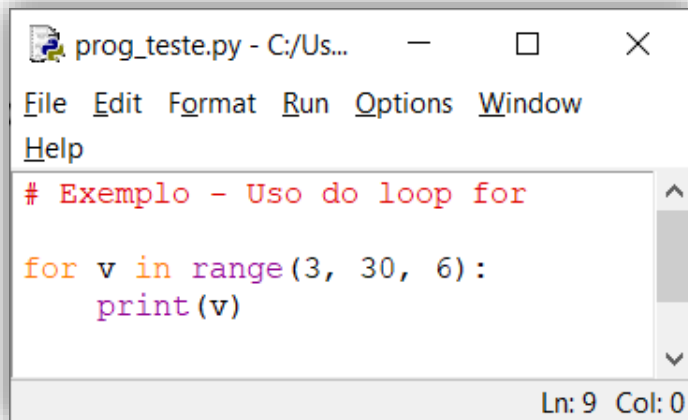
USO DO COMANDO RANGE

- A função **range** pode apresentar um terceiro parâmetro que representa o “salto” (**step**) entre os valores gerados.

SINTAXE:

range (INICIO, FIM, STEP)

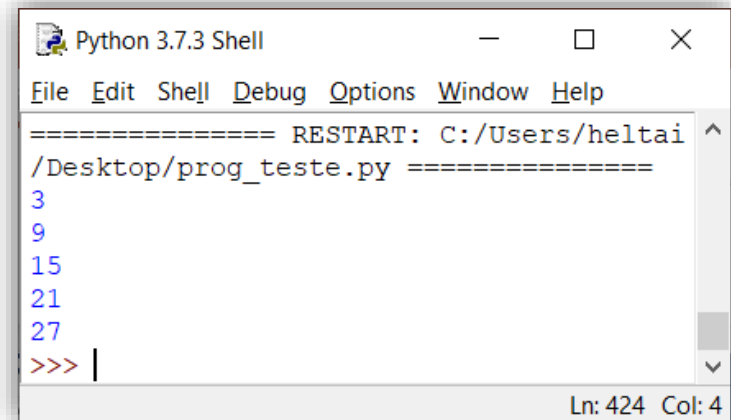
EXEMPLO: Programa que imprime na tela de acordo com o **inicio**, **fim** e **step** do comando **range**.



```
File Edit Format Run Options Window
Help
# Exemplo - Uso do loop for

for v in range(3, 30, 6):
    print(v)
```

Ln: 9 Col: 0

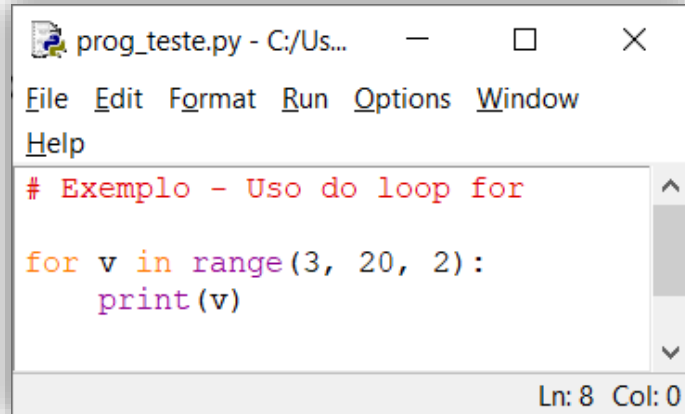


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai
/Desktop/prog_teste.py =====
3
9
15
21
27
>>> |
```

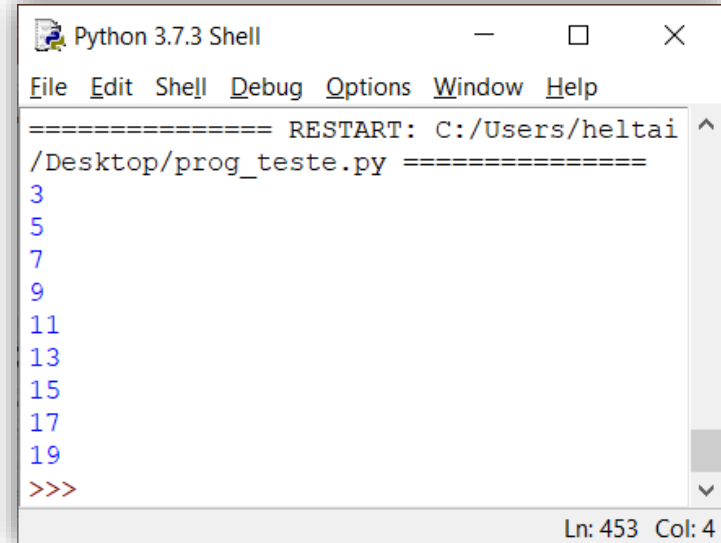
Ln: 424 Col: 4

USO DO COMANDO RANGE

EXEMPLO: Programa que imprime na tela de acordo com o **inicio**, **fim** e **step** do comando **range**.



```
File Edit Format Run Options Window Help
# Exemplo - Uso do loop for
for v in range(3, 20, 2):
    print(v)
Ln: 8 Col: 0
```



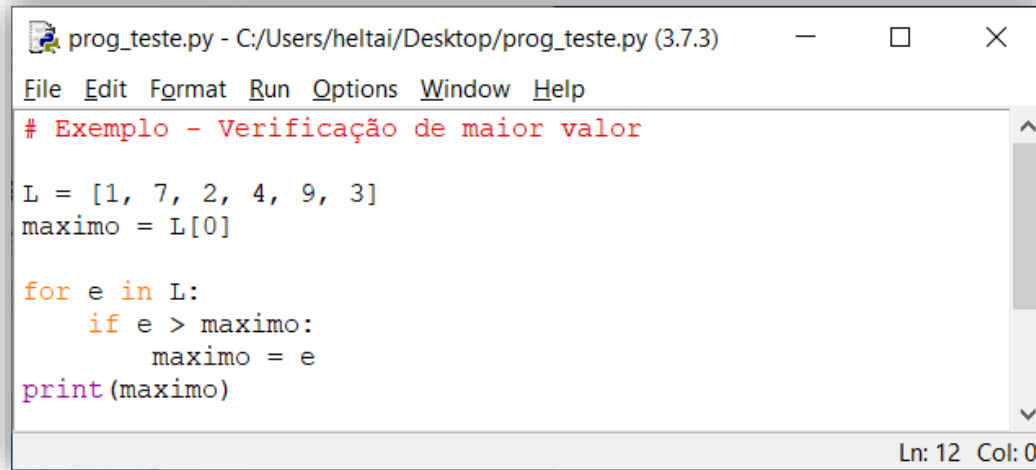
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai
/Desktop/prog_teste.py =====
3
5
7
9
11
13
15
17
19
>>>
Ln: 453 Col: 4
```



APLICAÇÕES COM LISTA

APLICAÇÃO COM LISTA

EXEMPLO: Procura o maior valor em uma lista

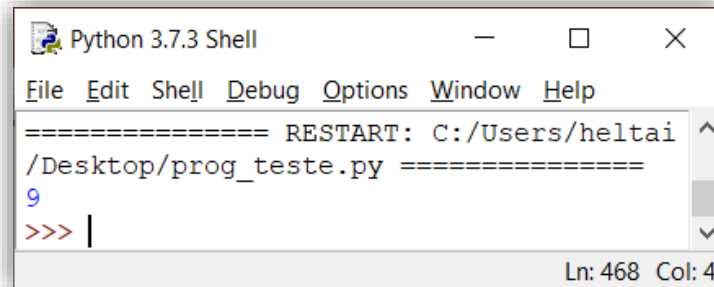


```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)
File Edit Format Run Options Window Help
# Exemplo - Verificação de maior valor

L = [1, 7, 2, 4, 9, 3]
maximo = L[0]

for e in L:
    if e > maximo:
        maximo = e
print(maximo)

Ln: 12 Col: 0
```

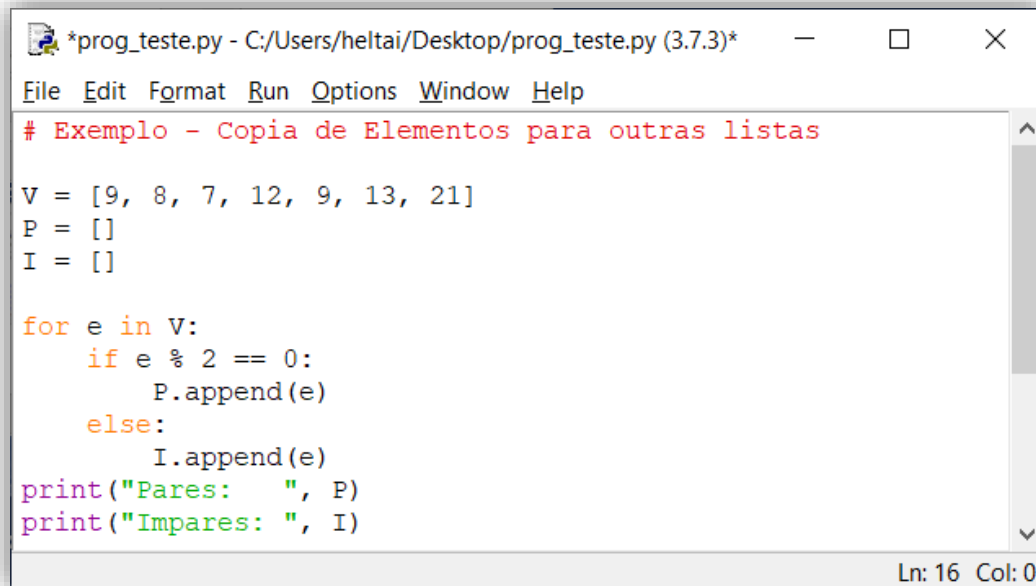


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
==== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====
9
>>> |

Ln: 468 Col: 4
```

APLICAÇÃO COM LISTA

EXEMPLO: Programa que percorre uma lista de forma a verificar o menor e o maior valor.

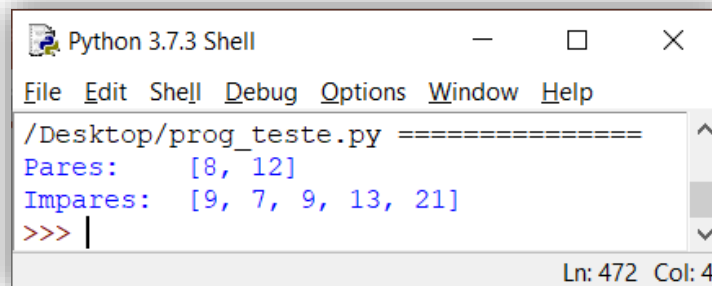


```
*prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)*
File Edit Format Run Options Window Help
# Exemplo - Copia de Elementos para outras listas

V = [9, 8, 7, 12, 9, 13, 21]
P = []
I = []

for e in V:
    if e % 2 == 0:
        P.append(e)
    else:
        I.append(e)
print("Pares: ", P)
print("Impares: ", I)
```

Ln: 16 Col: 0



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
/Desktop/prog_teste.py =====
Pares:      [8, 12]
Impares:    [9, 7, 9, 13, 21]
>>> |
```

Ln: 472 Col: 4

APLICAÇÃO COM LISTA

EXEMPLO: Programa que controla a utilização das salas de um cinema. Considerando que a lista `lugares_vagos = [10, 2, 1, 3, 0]` contenha o numero de lugares vagos nas salas 1, 2, 3, 4 e 5, respectivamente. O programa lê o numero da sala e a quantidade de lugares solicitados e informa se é possível vender o numero de lugares solicitados (se houve lugares livre). Caso seja possível vender os bilhetes, atualiza o numero de lugares livres. A saída ocorre quando se digita 0 no numero da sala.

```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)
File Edit Format Run Options Window Help
# Exemplo - Controle da atualização de salas de um cinema

lugares_vagos = [10, 2, 1, 3, 0]

while True:

    sala = int(input("Sala (0 sai): "))

    if sala == 0:
        print("Fim")
        break
    if sala > len(lugares_vagos) or sala < 1:
        print("Sala inválida")
    elif lugares_vagos[sala - 1] == 0:
        print("Desculpe, sala lotada!")

    else:
        lugares = int(input(f"Quantos lugares desejados ({lugares_vagos[sala - 1]} vagos):"))
        if lugares > lugares_vagos[sala - 1]:
            print("Esse numero de lugares não estão disponivel.")
        elif lugares < 0:
            print("Numero invalido")
        else:
            lugares_vagos[sala - 1] -= lugares
            print(f"{lugares} lugares vendidos")

    print("Utilização das salas")
    for x, l in enumerate(lugares_vagos):
        print(f"Sala {x + 1} - {l} lugar(es)_vazio(s)")
```

Ln: 34 Col: 4

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/
Desktop/prog_teste.py =====
Sala (0 sai): 2
Quantos lugares desejados (2 vagos):1
1 lugares vendidos
Utilização das salas
Sala 1 - 10 lugar(es)_vazio(s)
Sala 2 - 1 lugar(es)_vazio(s)
Sala 3 - 1 lugar(es)_vazio(s)
Sala 4 - 3 lugar(es)_vazio(s)
Sala 5 - 0 lugar(es)_vazio(s)
Sala (0 sai): 2
Quantos lugares desejados (1 vagos):1
1 lugares vendidos
Utilização das salas
Sala 1 - 10 lugar(es)_vazio(s)
Sala 2 - 0 lugar(es)_vazio(s)
Sala 3 - 1 lugar(es)_vazio(s)
Sala 4 - 3 lugar(es)_vazio(s)
Sala 5 - 0 lugar(es)_vazio(s)
Sala (0 sai): 2
Desculpe, sala lotada!
Utilização das salas
Sala 1 - 10 lugar(es)_vazio(s)
Sala 2 - 0 lugar(es)_vazio(s)
Sala 3 - 1 lugar(es)_vazio(s)
Sala 4 - 3 lugar(es)_vazio(s)
Sala 5 - 0 lugar(es)_vazio(s)
Sala (0 sai): 0
Fim
>>>
```

Ln: 33 Col: 4

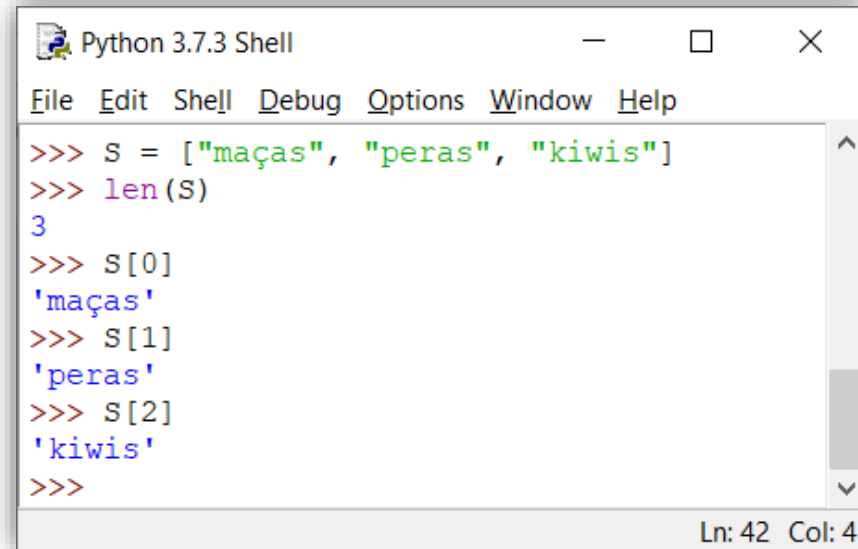


LISTA COM STRING

LISTA COM STRING

LISTA COM STRINGS

- Conforme já conceituado, string podem ser indexadas ou acessadas letra por letra. Lista em Python funcionam da mesma forma, permitindo o acesso a vários valores e se tornando uma das principais vantagens da linguagem.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> S = ["maças", "peras", "kiwis"]
>>> len(S)
3
>>> S[0]
'maças'
>>> S[1]
'peras'
>>> S[2]
'kiwis'
>>>
Ln: 42 Col: 4
```

LISTA COM STRING

EXEMPLO: Programa que lê e imprime uma lista de compras até que seja digitado fim.

```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3...
File Edit Format Run Options Window Help
# Exemplo - Lendo e imprimindo uma lista de compras

compras = []

while True:
    produto = input("Produto: ")
    if produto == "fim":
        break
    compras.append(produto)

for p in compras:
    print(p)

Ln: 16 Col: 4
```

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/D
esktop/prog_teste.py =====
Produto: Sabão
Produto: Arroz
Produto: Feijão
Produto: Batata
Produto: Leite
Produto: fim
Sabão
Arroz
Feijão
Batata
Leite
>>>

Ln: 55 Col: 4
```

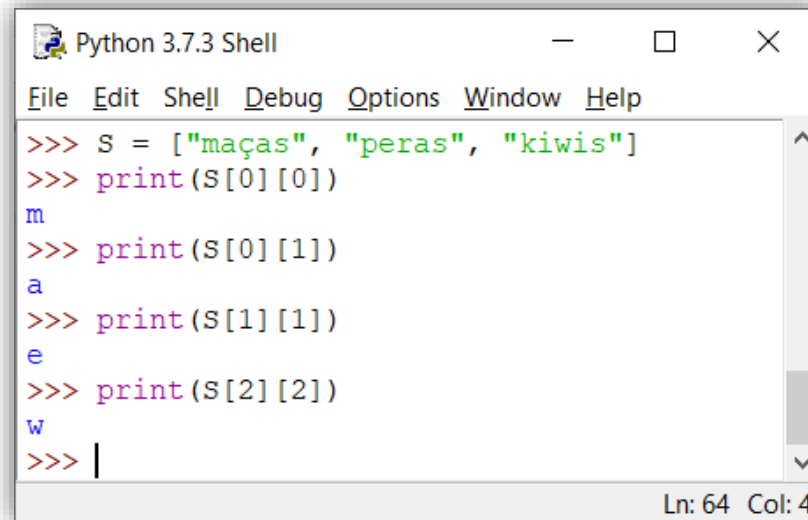


LISTAS DENTRO DE LISTAS

LISTAS DENTRO DE LISTAS

LISTAS DENTRO DE LISTAS

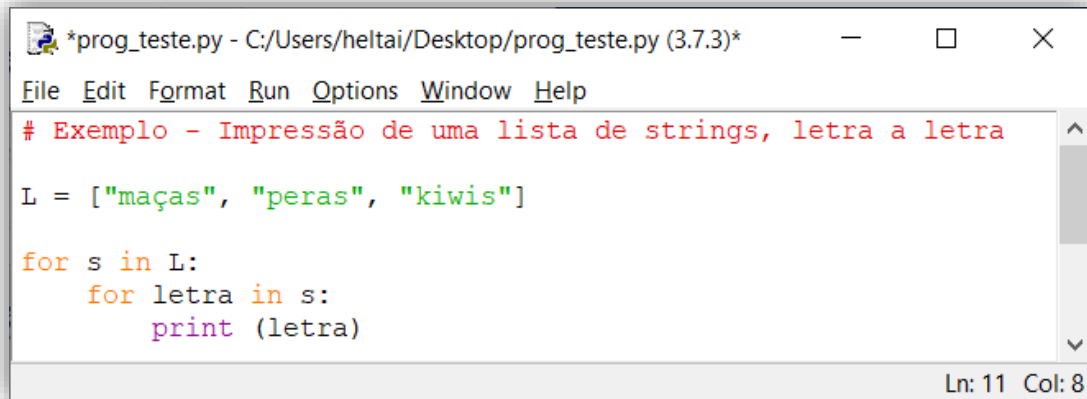
- É possível acessar as strings dentro da lista, letra por letra, usando um segundo índice:

A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text "Python 3.7.3 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a series of Python commands and their outputs. The commands are: `>>> S = ["maças", "peras", "kiwis"]`, `>>> print(S[0][0])`, `>>> print(S[0][1])`, `>>> print(S[1][1])`, and `>>> print(S[2][2])`. The outputs are: `m`, `a`, `e`, and `w` respectively. The prompt `>>>` is followed by a vertical bar `|` on the last line. At the bottom right of the window, the status bar shows "Ln: 64 Col: 4".

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> S = ["maças", "peras", "kiwis"]
>>> print(S[0][0])
m
>>> print(S[0][1])
a
>>> print(S[1][1])
e
>>> print(S[2][2])
w
>>> |
Ln: 64 Col: 4
```

LISTAS DENTRO DE LISTAS

EXEMPLO: Impressão de uma lista de string, letra a letra

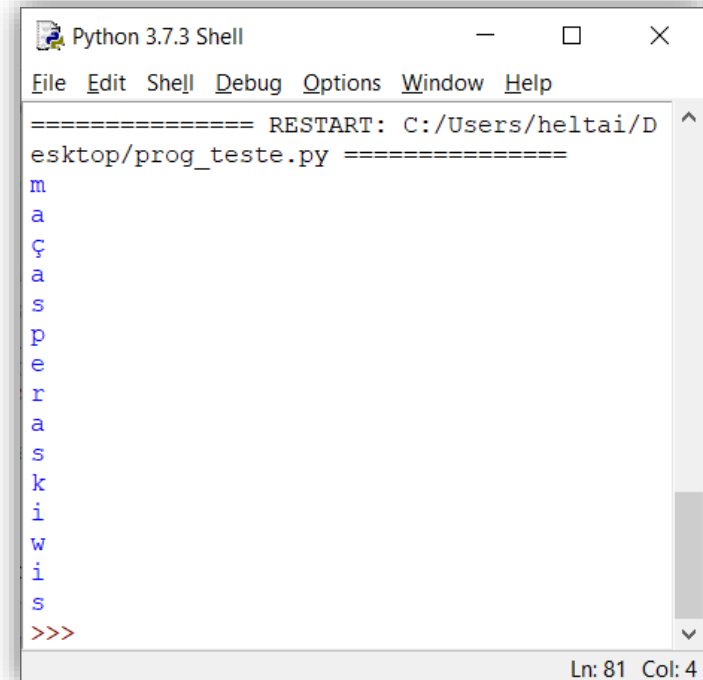


```
*prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)*
File Edit Format Run Options Window Help
# Exemplo - Impressão de uma lista de strings, letra a letra

L = ["maças", "peras", "kiwis"]

for s in L:
    for letra in s:
        print (letra)
```

Ln: 11 Col: 8



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/D
esktop/prog_teste.py =====
m
a
Ç
a
s
p
e
r
a
s
k
i
w
i
s
>>>
```

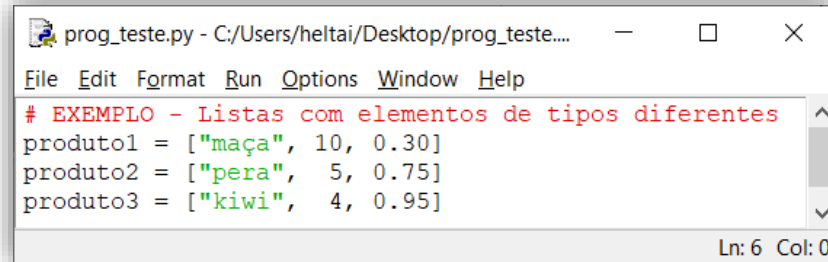
Ln: 81 Col: 4

LISTAS DENTRO DE LISTAS

STRING COM OUTROS TIPOS DE DADOS:

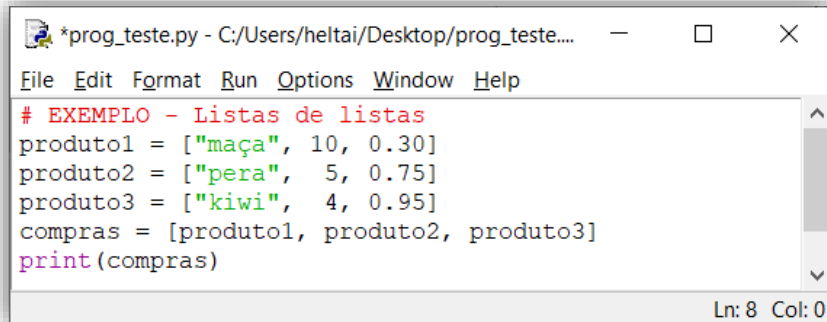
- Os elementos de uma lista não precisam ser do mesmo tipo.

EXEMPLO: O primeiro elemento seria o nome do produto; o segundo, a quantidade; e o terceiro, seu preço.

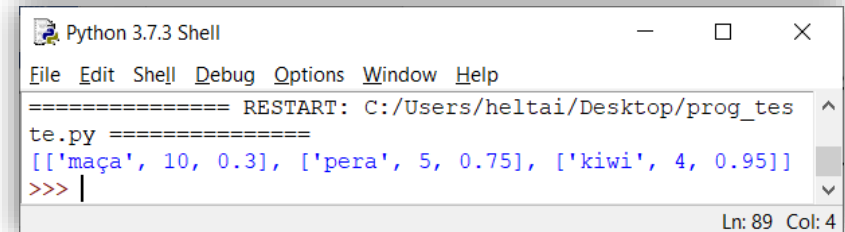


```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste....
File Edit Format Run Options Window Help
# EXEMPLO - Listas com elementos de tipos diferentes
produto1 = ["maça", 10, 0.30]
produto2 = ["pera", 5, 0.75]
produto3 = ["kiwi", 4, 0.95]
Ln: 6 Col: 0
```

EXEMPLO: Lista de lista com elementos diferentes



```
*prog_teste.py - C:/Users/heltai/Desktop/prog_teste....
File Edit Format Run Options Window Help
# EXEMPLO - Listas de listas
produto1 = ["maça", 10, 0.30]
produto2 = ["pera", 5, 0.75]
produto3 = ["kiwi", 4, 0.95]
compras = [produto1, produto2, produto3]
print(compras)
Ln: 8 Col: 0
```



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/Desktop/prog_teste.py =====
[['maça', 10, 0.3], ['pera', 5, 0.75], ['kiwi', 4, 0.95]]
>>>
Ln: 89 Col: 4
```

LISTAS DENTRO DE LISTAS

EXEMPLO: Impressão de Lista de lista com elementos diferentes

```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste....  
File Edit Format Run Options Window Help  
# EXEMPLO - Impressão das compras  
produto1 = ["maça", 10, 0.30]  
produto2 = ["pera", 5, 0.75]  
produto3 = ["kiwi", 4, 0.95]  
compras = [produto1, produto2, produto3]  
for e in compras:  
    print(f"Produtos: {e[0]}")  
    print(f"Quantidade: {e[1]}")  
    print(f"Preço: {e[2]:5.2f}")  
Ln: 12 Col: 4
```

```
Python 3.7.3 Shell  
File Edit Shell Debug Options Window Help  
===== RESTART: C:/Users/heltai/  
Desktop/prog_teste.py =====  
Produtos: maça  
Quantidade: 10  
Preço: 0.30  
Produtos: pera  
Quantidade: 5  
Preço: 0.75  
Produtos: kiwi  
Quantidade: 4  
Preço: 0.95  
>>> |  
Ln: 100 Col: 4
```



ORDENAÇÃO

ORDENAÇÃO

ORDENAÇÃO DOS ELEMENTOS DA LISTA

- Para ordenar uma lista, realiza-se uma operação idêntica à pesquisa, mas trocando a ordem dos elementos quando necessários.

METODO BOLHA:

- Compara dois elementos de cada vez. Se o valor do primeiro elemento for maior que o do segundo, ele trocarão de posição.
- Essa operação é então repetida para o próximo elemento até o fim da lista.
- O método bolha exige que se percorra toda a lista várias vezes e para isso se utiliza um marcador a fim de saber se chegou-se ao fim da lista trocada.

ORDENAÇÃO

METODO BOLHA:

```
prog_teste.py - C:/Users/heltai/Desktop/prog_teste.py (3.7.3)
File Edit Format Run Options Window Help
# Exemplo - Ordenação pelo meotdo de bolhas

L = [7, 4, 3, 12, 8]
fim = 5 #numero de elementos da lista

while fim > 1:
    trocou = False
    x = 0
    while x < (fim - 1):
        if L[x] > L[x + 1]:
            trocou = True
            temp = L[x]
            L[x] = L[x + 1]
            L[x + 1] = temp
        x += 1
    if not trocou:
        break
    fim -= 1
for e in L:
    print(e)
```

Ln: 20 Col: 12

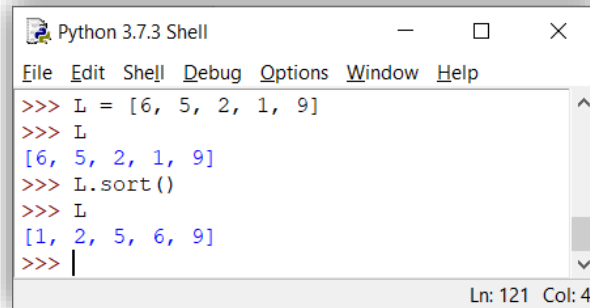
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/heltai/
Desktop/prog_teste.py =====
3
4
7
8
12
>>> |
```

Ln: 107 Col: 4

ORDENAÇÃO

METODO SORT:

- O método **sort** ordena rapidamente uma lista.

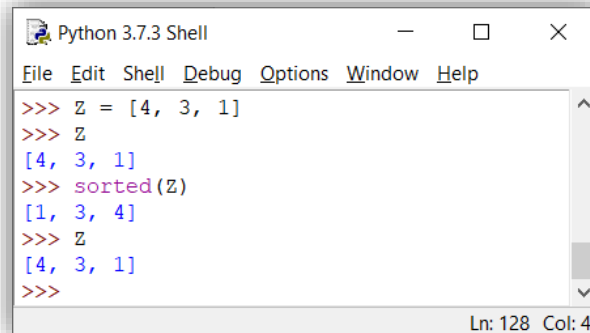


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> L = [6, 5, 2, 1, 9]
>>> L
[6, 5, 2, 1, 9]
>>> L.sort()
>>> L
[1, 2, 5, 6, 9]
>>> |
```

Ln: 121 Col: 4

METODO SORTED:

- O método **sorted**, ordena uma lista, sem alterar seus elementos.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> Z = [4, 3, 1]
>>> Z
[4, 3, 1]
>>> sorted(Z)
[1, 3, 4]
>>> Z
[4, 3, 1]
>>>
```

Ln: 128 Col: 4



PROF. VINICIUS HELTAI

vinicius.pacheco@docente.unip.br

www.heltai.com.br

(11) 98200-3932



/vheltai



@vheltai



/vheltai



@Vinicius Heltai



@vheltai