Senac

FACULDADE DE TECNOLOGIA E INOVAÇÃO

LISTAS

CLENIO EMIDIO

Lista simples

- Permite armazenar uma coleção de itens em uma única variável.
  - permite membros duplicados
  - possui índice (0,1,2,3...)

```
5   lista = ['Clenio', 'Andre', "Andre",'Ana Paula','Marilza']
6   print (lista)
7
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio Emidio\Desktop\python\exemplolistas.py"
['Clenio', 'Andre', 'Andre', 'Ana Paula', 'Marilza']
PS C:\Users\Clenio Emidio\Desktop\python>
```

  - Para imprimir uma posição bastaríamos colocar o número da posição:

```
5   lista = ['Clenio', 'Andre', "Andre",'Ana Paula','Marilza']
6   print (lista)
7   print(lista[3])
8
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio Emi
['Clenio', 'Andre', 'Andre', 'Ana Paula', 'Marilza']
Ana Paula
PS C:\Users\Clenio Emidio\Desktop\python>
```

Listas

- **LISTA VAZIA**
  - lista_vazia = [] ou lista_vazia = list()
- **Como criar uma lista dentro de lista**
  - lista_aninhada = [1, 2, ["olá", "mundo"], 3, 4, 5]
- **Adicionar um elemento a uma lista**
  - Para adicionar um elemento a uma lista, podemos utilizar o método append():
  - minha_lista = [1, 2, 3]
  - minha_lista.append(4)
- **Para acessar um elemento da lista**
  - print(minha_lista[2])
  - Há dois pontos a destacar sobre os índices de uma lista. O primeiro é que podemos passar índices negativos para pegar elementos partindo do final da lista. O segundo ponto é que índices além dos limites da lista causam um exceção do tipo IndexError.
  - print(minha_lista[-1]) # Último elemento
  - print(minha_lista[-2]) # Penúltimo elemento

- **Modificar um elemento em uma lista:**
  - minha_lista = ["olá", "mundo", “Faculdade", “Tecnologia"]

  - minha_lista[1] = "Novo Valor"
  - print(minha_lista)
    - # output: ['olá', 'Novo Valor’, Faculdade’, Tecnologia’]
- **Remover um elemento de uma lista**
  - del minha_lista[1]

- **Verificando se uma lista está vazia**

```
minha_lista = []
if len(minha_lista) == 0:
    print('Lista está vazia')
else:
    print('Lista não está vazia')
```

- **Buscar um elemento em uma lista**
  - minha_lista = [1, 2, 3, 4, 5]
  - print(1 in minha_lista)
    - # output: True
  - print(10 in minha_lista)
    - # output: False
  - print('1' in minha_lista)
    - # output: False
- **Concatenar 2 listas:**
  - lista1 = [1, 2, 3]
  - lista2 = [4, 5, 6]
  - minha_lista = lista1 + lista2
  - print(minha_lista)
    - # output: [1, 2, 3, 4, 5, 6]

Os principais métodos de lista em Python

- **Adicionar elementos com append() e extend()**
  - minha_lista = [1, 2, 3]
  - minha_lista.append(4)
  - print(minha_lista)
    - # output: [1, 2, 3, 4]
- **Podemos também usar o método de lista extend() para adicionar valores novos a uma lista**
  - minha_lista = [1, 2, 3]
  - minha_lista.extend([4, 5])
  - print(minha_lista)
    - # output: [1, 2, 3, 4, 5]

- **Inserir elementos em uma posição específica com insert()**
  - minha_lista = [“Olá", "tudo", "bem?"]
  - minha_lista.insert(1, "Python")
  - print(minha_lista)
    - # output: [‘Olá', 'Python', 'tudo', 'bem?']
- **Remover elementos com pop() e remove(**)
  - Além da palavra-chave del, podemos usar os métodos pop() e remove() para remover elementos de uma lista. O método pop() retira o último elemento da lista e o retorna, funcionando quase como um “append() ao contrário”:
  - minha_lista = [“Olá", "tudo", "bem?"]
  - ultimo_elemento = minha_lista.pop()
  - print(minha_lista)
    - # output: [‘Ola', 'tudo']
  - print(ultimo_elemento)
    - # output: bem?
- Já o método **remove() retira um item de dentro da lista**, o qual devemos passar como argumento
  - minha_lista = [“Olá", "tudo", "bem?"]
  - minha_lista.remove("tudo")
  - print(minha_lista)
  - # output: [‘Olá', 'bem?']
- **Limpar uma lista com clear()**
  - minha_lista = [“Olá", "tudo", "bem?"]
  - minha_lista.clear()

- **Encontrar a posição de um elemento com index()**
  - minha_lista = [“olá", "tudo", "bem?"]
  - indice = minha_lista.index(“Olá")
  - print(indice)
    - # output: 0
- **Contar ocorrências de um elemento com count()**
  - minha_lista = [0, 1, 1, 0, 1, 1, 1, 0, 0, 1]
  - numeros_zero = minha_lista.count(0)
  - print(numeros_zero)
    - # output: 4
- **Ordenar ou inverter uma lista com sort() ou reverse()**
  - Dvalores = [10, 3, 9, 11, 12, 5, -1]
  - valores.sort()
  - print(valores)
    - # output: [-1, 3, 5, 9, 10, 11, 12]
- Obs: Um ponto importante com o método sort() é que os elementos devem ser comparáveis entre si, senão apresenta erro.
  - valores = [10, 3, “Tecnologia"]
  - valores.sort()
    - # TypeError: '<' not supported between instances of 'str' and 'int'

- **Método pop(índice)**
- Remove da lista o elemento na posição índice e o retorna Se índice não for mencionado, é assumido o último
- Lista = [1,2,3,4,5,6]
- Lista.pop(1)
- Print (lista)
  - #[1,3,4,5,6]
- Lista.pop()
- Print (lista)
  - #[1,3,4,5]
- **Reverse()**
  - minha_lista = [“Olá", "tudo", "bem?"]
  - minha_lista.reverse()
  - print(minha_lista)
    - # output: ['bem?', 'tudo', ‘Olá']

Outras operações

- Elemento=[1,2,3,4,5]
- Sum(elemento)
- Len(elemento)
- max(elemento)
- min(elemento)

- **Como transformar uma lista em um string?**
  - minha_lista = [1, 2, 3]
  - minha_lista_texto = str(minha_lista)

  - print(minha_lista_texto)
    - # output: [1, 2, 3]
  - print(type(minha_lista_texto))
    - # output: <class 'str'>

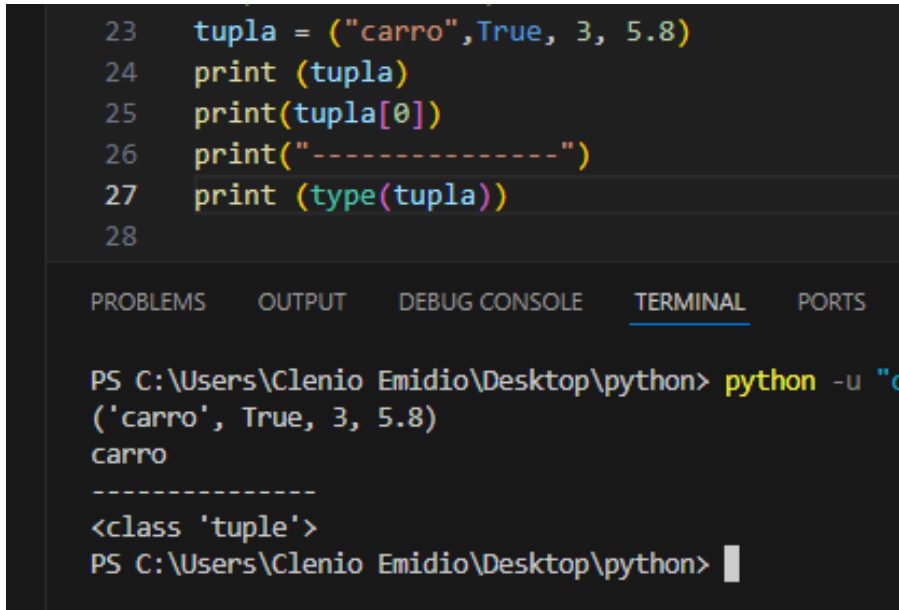
- **Se quisermos adicionar algum caractere específico ou formatação entre os valores da lista, podemos utilizar o método de string join()**
  - minha_lista = [1, 2, 3]
  - texto = "---".join(minha_lista)
  - print(texto)
    - # output: 1---2---3

Qual é a diferença entre uma lista e um vetor (array) em Python?

- Uma lista em Python é diferente de um vetor (também chamado de arranjo ou array). As listas não possuem as operações que esperamos de um vetor matemático, como soma de vetor e escalar ou produto vetorial. Se quisermos criar uma sequência de números que representem um vetor matemático, teremos que usar outras estruturas de dados, como os arrays do NumPy.
- Python possui um objeto array (vetor) em sua biblioteca padrão, que é considerado um tipo de dado embutido na linguagem. Dito isso, a comunidade de ciência de dados usa quase que exclusivamente a biblioteca NumPy (nome derivado do inglês Numerical Python) para trabalhar com vetores, que precisa ser instalada com pip.
- https://hub.asimov.academy/blog/listas-em-python/#:~:text=Uma%20lista%20em%20Python%20%C3%A9,din%C3%A2mica%20ao%20longo%20do%20c%C3%B3digo.

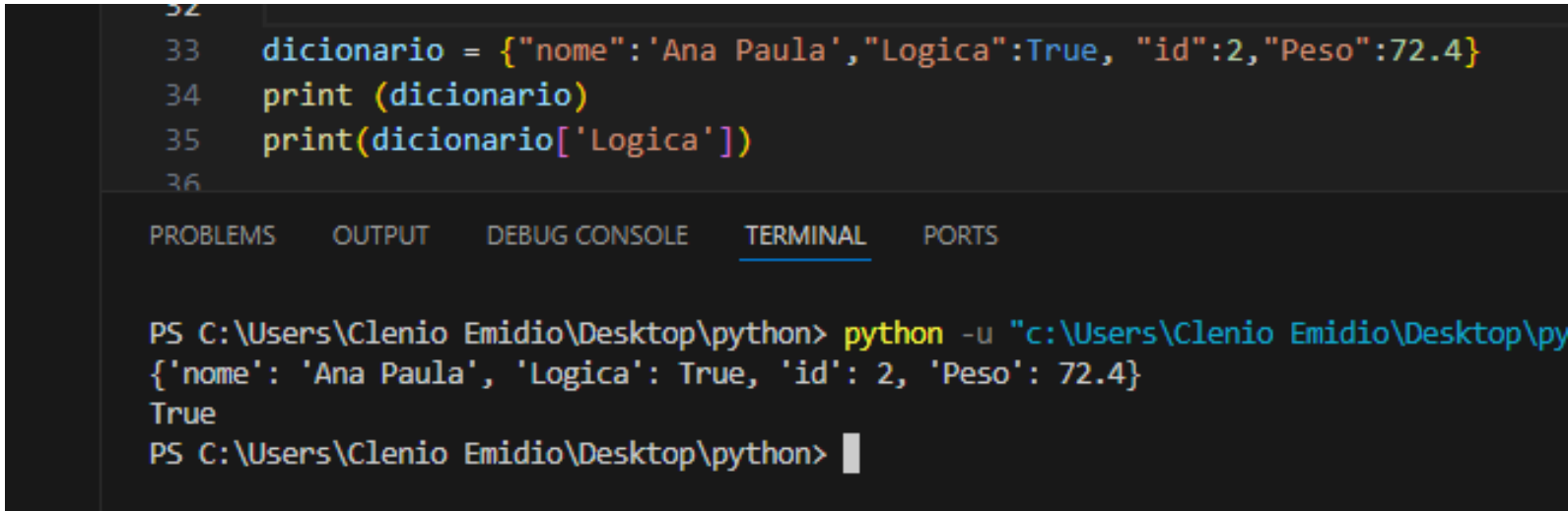
Listas

- **Tupla** - é uma sequência imutável de valores de qualquer tipo. Permite membros duplicados
- Possui índices (0,1,2,3...)
- Caso quiséssemos imprimir a posição “0” basta digitar print(tupla[0])conforme linha 25 na imagem ao lado
- Na linha 27 na imagem permite imprimir o tipo
- De lista <class ‘tuple’>



Listas

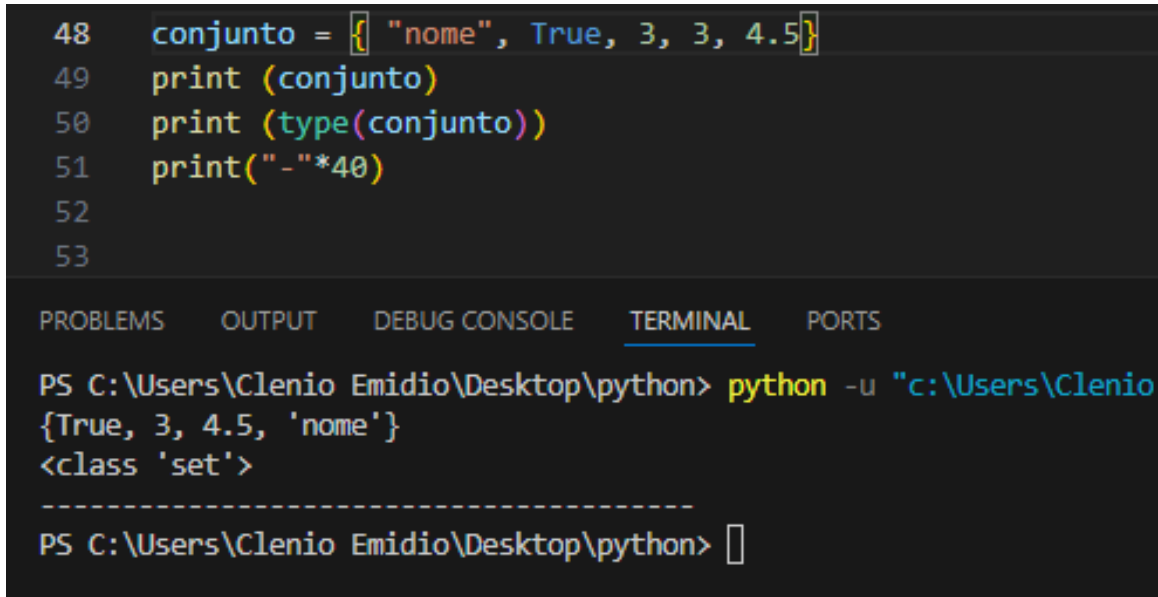
- **Dicionário**: - é uma coleção ordenada com elementos "**chave-valor**" que permite representar melhor o mundo real. Ambos são separados por “:”
- nenhum membro duplicado, possui índice!
- Porém para imprimir a posição de um índice, identificamos a chave. No exemplo abaixo (linha 35) queremos imprimir o valor da chave ‘Lógica’ e foi impresso na tela ‘True’.



Listas

- **Conjunto** - O mesmo que dicionário, porém "sem" a definição de chave e valor
- É uma coleção não ordenada e 'não indexada'. Nenhum membro duplicado. Ou seja, caso tenha algum valor que se repita, ele ignora, conforme mostra na imagem.
- Ele também altera a ordem de exibição

Dos valores.



For e While – Estruturas condicionais
Laço de Repetição ou loop

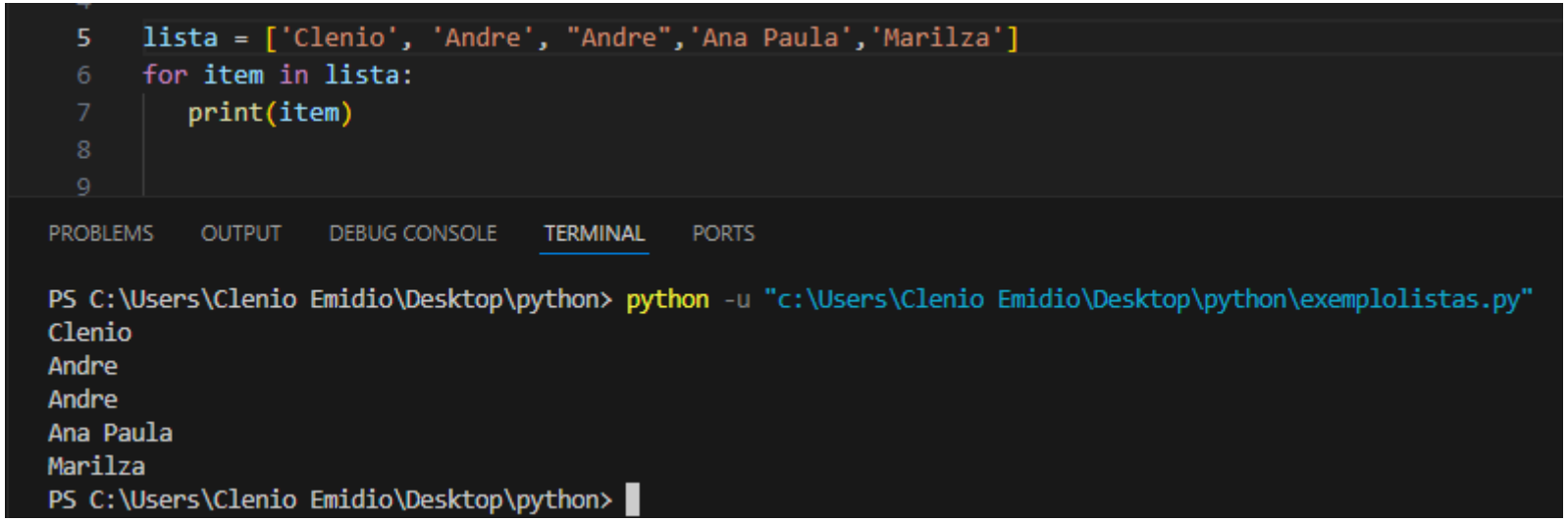
- Em Python, os loops são codificados por meio dos comandos for e while.
- O '**For**' permite percorrer os itens de uma coleção e, para cada um deles, executar um bloco de código ou um conjunto de instruções em cada item.., é utilizado para percorrer ou iterar sobre uma sequência de dados (seja esse uma lista, uma tupla, uma string)
- Já o **while**, executa um conjunto de instruções várias vezes enquanto uma condição é atendida.

```
1 | nomes = ['Pedro', 'João', 'Leticia']
2 | for n in nomes:
3 |     print(n)
```

```
1 | contador = 0
2 | while contador < 5:
3 |     print(contador)
4 |     contador = contador + 1
```

Listas com laço de Repetição ‘For’

- Permite percorrer a lista e imprime cada valor abaixo um do outro.
  - A indentação é primordial, observe que o print está um pouco à direita
  - Tudo o que estiver indentado faz parte do laço de repetição.



Laço de repetição For com outros argumentos

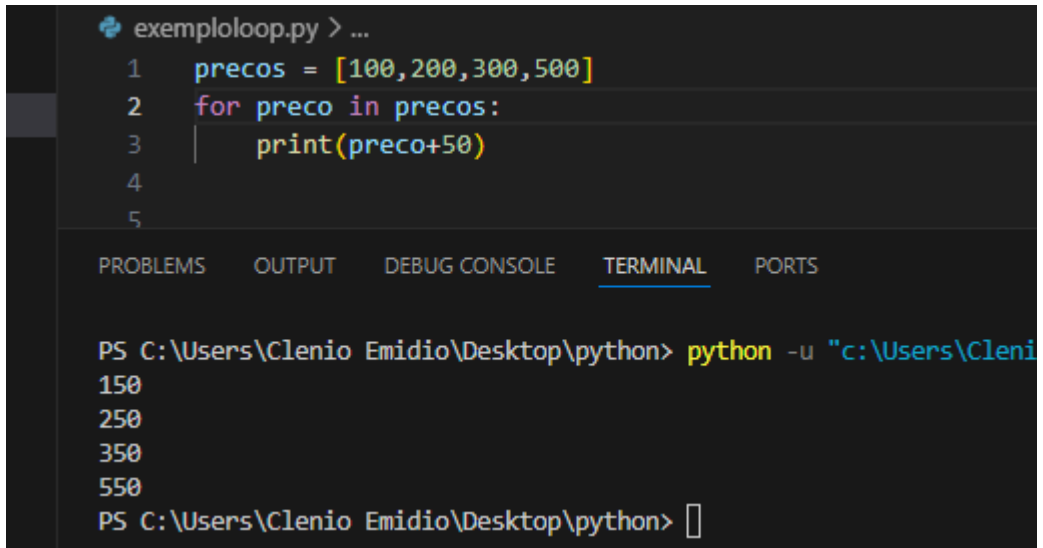
```
5   lista = ['Clenio', 'Andre', "Andre",'Ana Paula','Marilza']
6   for item in lista:
7       print(item)
8   print (lista[1])
9   print("*"*40)
10  print(type(lista))
```

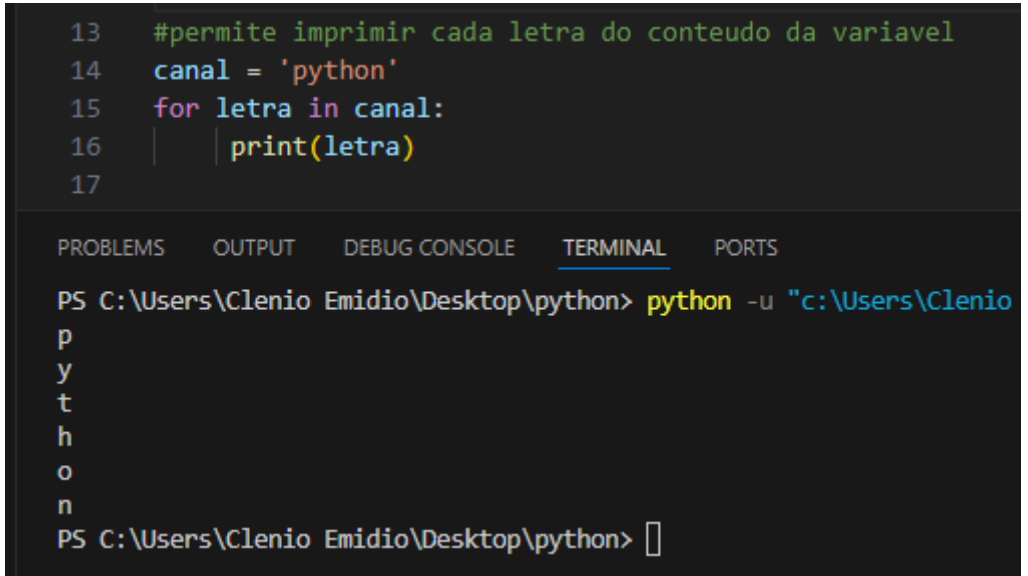
```
ROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

S C:\Users\Clenio Emidio\Desktop\python> python -u "c:\Users\Clenio Emidio\Desktop\python\exemplolistas.py"
lenio
ndre
ndre
na Paula
arilza
ndre
****************************************
class 'list'>
S C:\Users\Clenio Emidio\Desktop\python>
```

For com listas

- No exemplo abaixo será somado o valor 50 a cada preço.
  - Pois o print está dentro do laço de repetição.
  - No exemplo a direita podemos imprimir caracteres, embora não seja uma lista





Exercícios

Enviar até dia 05/09

1. *Escreva um programa que receba uma lista de de 10 inteiros via teclado, em seguida o programa deve solicitar um número e informar se o número também está na lista ou não.*
2. *Escreva um programa que leia e mostre uma lista de 10 elementos inteiros. Em seguida, conte quantos valores pares existem na lista, por fim, exiba a quantidade na tela.*
3. *Faça um programa que inicialize uma lista de compras com 5 itens diferentes e exiba todos utilizando um laço de repetição.*
4. *Faça um programa que inicialize uma lista vazia e solicite ao usuário 3 nomes de cidades, um por vez, cada vez que o usuário digitar um nome, o programa deve incluir este nome na lista de cidades.*
5. *Faça um programa que inicialize uma lista com vários números diferentes, depois disso, solicite ao usuário um número, verifique se o número está ou não na lista e exiba uma mensagem notificando o usuário do resultado.*
6. *Faça um programa que inicialize uma lista vazia e a preencha com 5 nomes diferentes digitados pelo usuário, depois disso solicite um número de 0 até 4 e remova o elemento desta posição.*
7. *Ler uma lista de 10 números reais e mostre-os na ordem inversa.*
8. *Ler uma lista de 5 números inteiros e mostre cada número juntamente com a sua posição na lista.*