Lista - Python

Jacqueline Midlej

Definição

- Estrutura de dados usadas para agrupar valores
  - operações primitivas para inserir e remover elementos
  - outras podem ser definidas: ordenar, inserir ordenado, buscar...

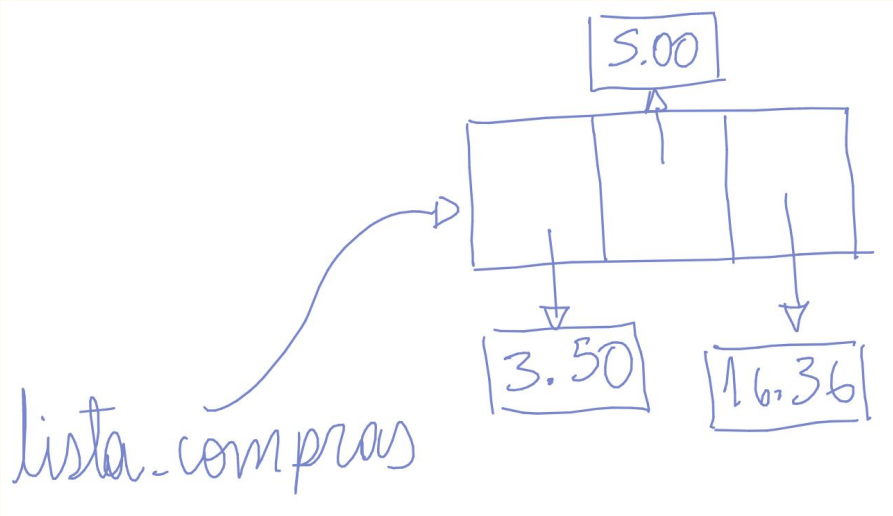
- Em Python, os dados não precisam ser do mesmo tipo

- Tem tamanho dinâmico, diferente do vetor, que possui tamanho fixo e simulamos a expansão e crescimento por meio de uma variável de controle (como o top e front e rear em pilhas e filas)

- Em C, chamamos esta lista dinâmica de lista encadeada (veremos nas aulas seguintes)

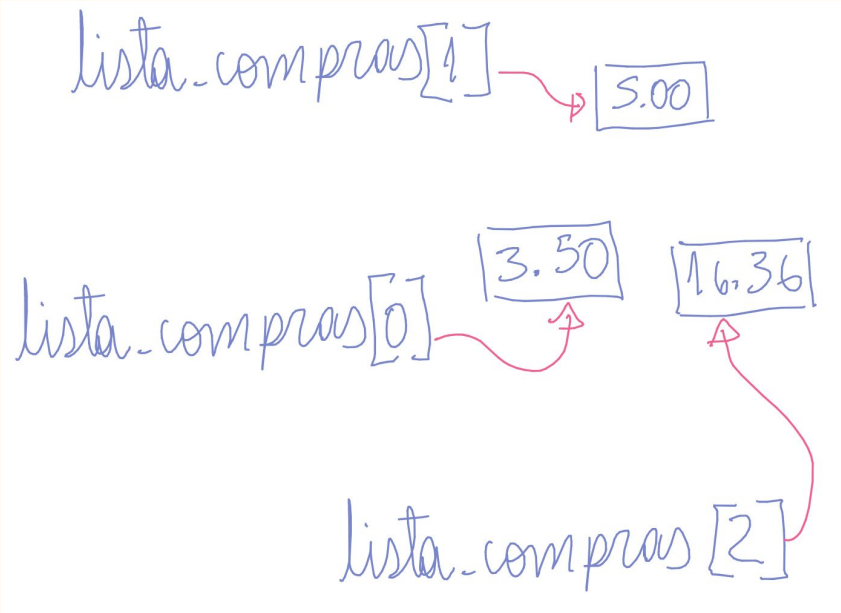
Definição

- uma lista é uma variável que contém um conjunto de referências para outras variáveis



Definição

- não precisa estar armazenada sequencialmente na memória, mas nós acessamos de forma sequencial



Inicialização

- Definimos uma lista por [] ou list()

```
lista_vazia = []
outra_lista_vazia = list()

primos = [2, 3, 5, 7, 11]
cinco_zeros = [0] * 5

escritores = ["Vinicius de Moraes",
              "Cecília Meireles",
              "Mary Shelley",
              "Cora Coralina",
              "Pedro dos Anjos"]

escritoras = escritores[1:4]
notas = [10.0, 7.5, 3.14]
```

Acesso por índices

```
# 1 elemento do início ao fim
>>> escritores[0]
"Vinicius de Moraes"

# 1 elemento do fim ao início
>>> escritores[-1]
"Pedro dos Anjos"
>>> escritores[-3]
"Mary Shelley"
```

```
# partes da lista
>>> escritores[:]
["Vinicius de Moraes",
"Cecília Meireles",
"Mary Shelley",
"Cora Coralina",
"Pedro dos Anjos"]

# primeiros 2 elementos
>> escritores[:2]
["Vinicius de Moraes",
"Cecília Meireles"]

# a partir do índice 2
>> escritores[2:]
["Mary Shelley",
"Cora Coralina",
"Pedro dos Anjos"]
```

intervalo [a:b],
fechado no a
aberto no b

Isto é, o índice a é incluído,
o b não

Acesso por índices

```
# partes da lista com saltos
>>> escritores[::2]  #todos, de começo a fim, saltando de 2 em 2
["Vinicius de Moraes",
"Mary Shelley",
"Pedro dos Anjos"]
```

```
var[start:end:step]

default:
start=0
end=n #tamanho da lista
step=1
```

Operações

Inserir um elemento: append

Remover um elemento do final: pop

Remover um elemento do início: pop(0)

Remover um elemento da posição i: pop(i)

Concatenar listas: +

Exemplo

```
>>> escritores.append("Mário Quintana")
>>> escritores
["Vinicius de Moraes",
"Cecília Meireles",
"Mary Shelley",
"Cora Coralina",
"Pedro dos Anjos",
"Mário Quintana"]

>>> escritores.pop()
"Mário Quintana"
>>> escritores
["Vinicius de Moraes",
"Cecília Meireles",
"Mary Shelley",
"Cora Coralina",
"Pedro dos Anjos"]
```

```
>>> escritores.pop(2)
"Mary Shelley"
>>> escritores
["Vinicius de Moraes",
"Cecília Meireles",
"Cora Coralina",
"Pedro dos Anjos"]

>>> escritores.pop(2)
"Cora Coralina"
>>> escritores
["Vinicius de Moraes",
"Cecília Meireles",
"Pedro dos Anjos"]
```

Exemplo

```
>>> escritores + ["Mário Quintana"]
["Vinicius de Moraes",
 "Cecília Meireles",
 "Pedro dos Anjos",
 "Mário Quintana"]

>>> escritores
["Vinicius de Moraes",
 "Cecília Meireles",
 "Pedro dos Anjos"]

>>> escritores = escritores + ["Mário Quintana"]
>>> escritores
["Vinicius de Moraes",
 "Cecília Meireles",
 "Pedro dos Anjos",
 "Mário Quintana"]
```

Aplicações

- Podemos armazenar elementos sem saber previamente a quantidade
  - Exemplo: valores de itens de supermercado.
  - Condição de parada: digitar um número negativo

```
lista_compras = []

valor = float(input())
while valor >= 0:
    lista_compras.append(valor)

      valor = float(input())
```

Aplicações

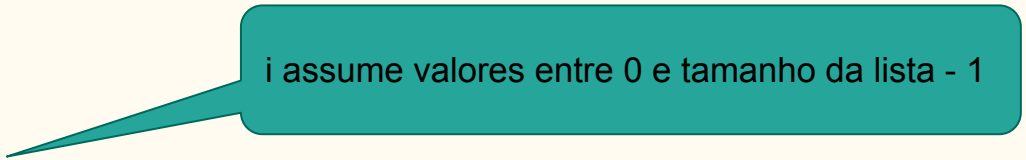
- Percorrendo listas

```
# somar todos os valores da lista

soma = 0.0

for valor in lista_compras:

    soma += valor

  print(soma)
```

Aplicações

- Percorrendo listas por índice

```
# somar todos os valores da lista
soma = 0.0
for i in range(len(lista_compras)):
    soma += lista_compras[i]
  print(soma)
```



Exemplos

```
# ler uma sequência de nomes
lista_nomes = []
nome = input()
while nome != "-":
    lista_nomes.append(nome)
    nome = input()

# guardar as iniciais
lista_iniciais = []
for nome in lista_nomes:
    inicial = nome[0]
    lista_iniciais.append(inicial)

print(lista_iniciais)
```

Cópia de listas

- Cópia se dá pela cópia da referência para a lista
- Teremos 2 variáveis apontando para o mesmo lugar da memória

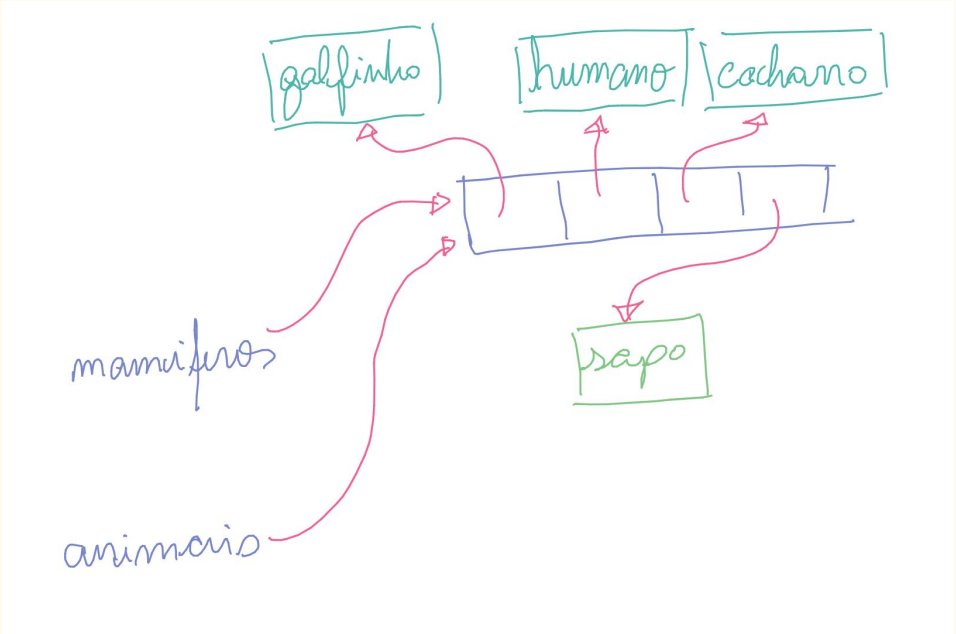
```
mamiferos = ["golfinho", "humano", "cachorro"]
animais = mamiferos
animais.append("sapo")
print(mamiferos)
```

```
['golfinho', 'humano', 'cachorro', 'sapo']
```

- Para copiar cada elemento: copy

```
mamiferos = ["golfinho", "humano", "cachorro"]
animais = mamiferos.copy()
animais.append("sapo")
print(mamiferos)
```

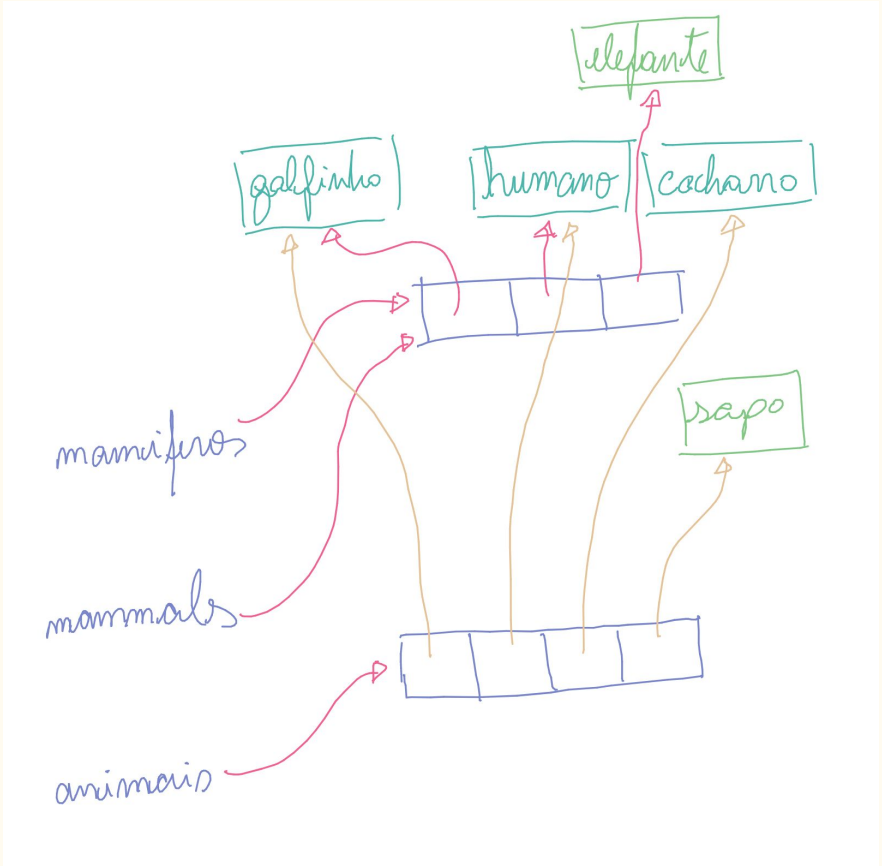
```
['golfinho', 'humano', 'cachorro']
```



Cópia de listas

- Cópia de cada elemento

```
mamiferos = ["golfinho", "humano", "cachorro"]

animais = []
for m in mamiferos:
    animais.append(m)

animais.append("sapo")

print(mamiferos)
print(animais)
```



- Vantagem:
  - Podemos remover e adicionar elementos nas listas em qualquer ordem
  - Se um elemento é alterado, a mudança reflete em ambas as listas

Cópia de lista – modificando um elemento - utilizando elementos mais complexos

```
mamiferos = [
    {'name': "golfinho", 'peso': 10},
    {'name': "humano", 'peso': 20},
    {'name': "cachorro", 'peso': 30}]

animais = []
for m in mamiferos:
    animais.append(m)

animais.append({'name': 'sapo', 'peso': 10})

mamiferos[2]['peso']=40

print(mamiferos)
print(animais)
```

Exemplo - pilha e fila

Exercício **4 da Lista 1 - pilha e fila**

Usando pilha ou fila, determine se uma string de caracteres tem a seguinte forma:
xCx onde x é uma string composta por As e Bs.
Um exemplo de xCx é ABBBCABBB.

```
x='AAABBBCAAABBB'
chegou_c=False
mal_formatada=False
queue=[]

for item in x:
    if item=='C':
        chegou_c=True
    else:
        if not chegou_c:
            queue.append(item)
        elif queue[0]==item:
            queue.pop(0)
        else:
            mal_formatada=True
            break

if mal_formatada:
    print("MAL FORMATADA")
else:
    print("BEM FORMATADA")
```

Erro se n de letras antes de após C for desbalanceado

Exemplo - pilha e fila

Criando uma função para verificar lista vazia
Criando uma função para retornar first

```
def vazia(q: list):
     if len(q)==0:
          return True
     else:
          return False

def first(q: list) -> chr:
    if not vazia(q):
        return q[0]
    else:
        return None
```

```
x='AABBCAAABBB'
chegou_c=False
mal_formatada=False
queue=[]

for item in x:
    if item=='C':
        chegou_c=True
    else:
        if not chegou_c:
            queue.append(item)
        elif first(queue)==item:
            queue.pop(0)
        else:
            mal_formatada=True
            break

if mal_formatada:
    print("MAL FORMATADA")
else:
    print("BEM FORMATADA")
```

Não existe mais erro.
Posição 0 é apenas acessada se lista não é vazia

Exercícios

- Dada uma lista de números, crie duas outras listas para separar os números pares dos ímpares
  - exemplo:
  - lista_inicial=[10, 20, 33, 44, 55, 56, 66, 89]
  - lista_par=[10, 20, 44, 56, 66]
  - lista_impar=[33, 55, 89]
- Q5 da lista 1:

  Usando pilha ou fila, determine se uma string de caracteres tem a seguinte forma:

  xCy onde: y é a string inversa de x (i.e, leitura de trás para frente de x); e x e y são strings compostas por As e Bs.

  Um exemplo de xCy é ABBBCBBBA.

  Em cada ponto você só poderá ler o próximo caracter da string, para decidir se está ou não no padrão esperado.

- Faça o exercício dos trens usando pilha (lista) em python
- Verifique se os parênteses de uma expressão estão balanceados. Já vimos esse problema em aula. Agora resolva com listas em python.

Referências

- https://docs.python.org/pt-br/3/tutorial/introduction.html#lists
- https://www.ic.unicamp.br/~lehilton/cursos/1s2020/mc102qr/unidades/05-listas.html