Aula 8 – Estrutura de Dados

Estrutura de Dados

Utilizadas para manter e organizar informação (dados)

Tipos de Estrutura de Dados

- Sequências
    - String
    - Listas
    - Tuplas

- Dicionários

Sequências

Série de valores contíguos que frequentemente estão relacionados.

Exemplos

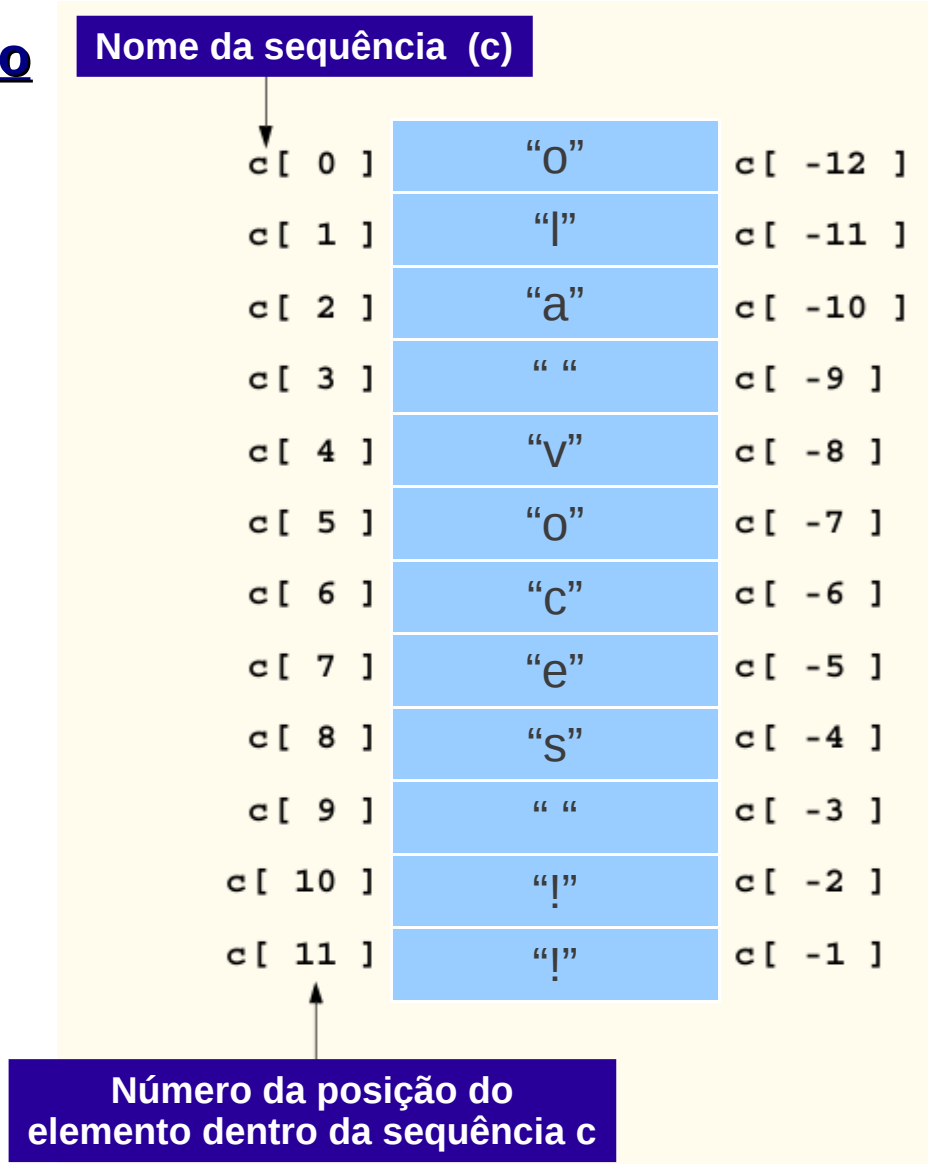
- Strings
- Lista
- Função range

```
>>> range(5)
    [0,1,2,3,4]
```

```
>>> range(0,10,2)
    [0,2,4,6,8]
```

Sequências - Strings

Exemplo



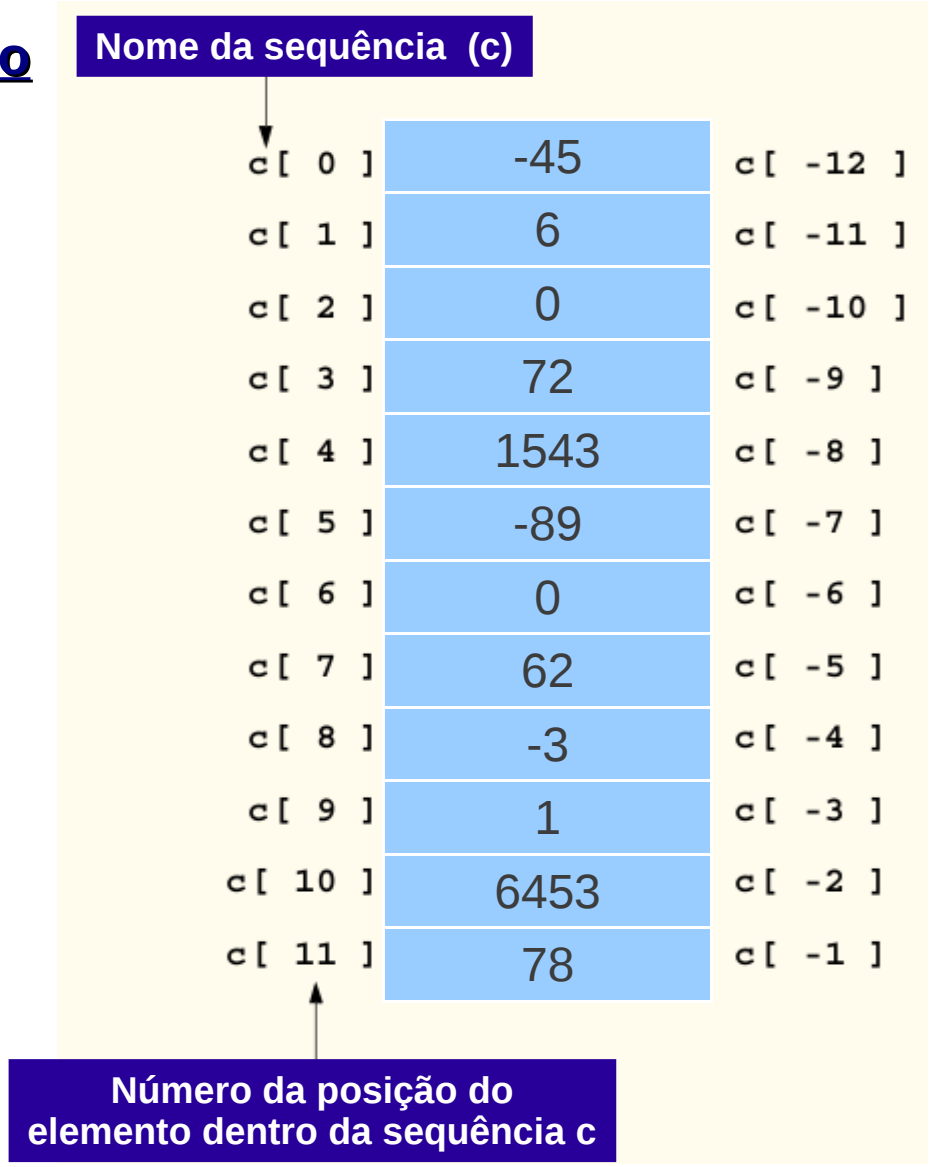

```
>>> c = "ola voces !!"

>>> c[4]
    "v"

>>> c[7] == c[-4]
    False
```

Sequências - Listas

Exemplo




```
>>> c = [-45, 6, 0, 72, 1543,
-89, 0,62, -3, 1, 6453, 78]

>>> c[3]
    72

>>> c[9]==c[-3]
    True
```

Listas

São formadas por sequência de valores, *não necessariamente do mesmo tipo*.

Lista Vazia: representada por [ ]

Como criar uma lista vazia: lista = [ ]

Atenção: *Uma lista vazia não contém nenhum elemento*

```
>>> lista = [ ]
>>> lista[0]
      Traceback (most recent call last):
        File "<pyshell#18>", line 1, in <module>
          lista[0]
      IndexError: list index out of range
```

Listas

Exercício: Faça um programa que construa a seguinte lista e a imprima:

0	-45
1	“ola”
2	[0]
3	“J”
4	[1,2]
5	[ ]
6	0
7	“62”
8	-3.0
9	“1”
10	6453L
11	[[23],[“alo”]]

Listas

Exercício: Faça um programa que construa a seguinte lista e a imprima:

0	-45
1	“ola”
2	[0]
3	“J”
4	[1,2]
5	[ ]
6	0
7	“62”
8	-3.0
9	“1”
10	6453L
11	[[23],[“alo”]]

```
lista = [-45, 'ola', [0], 'J', [1, 2], [ ], 0, '62', -3.0, '1',
6453L, [[23], ['alo']]]
for i in range(12):
        print lista[i]
```

```
lista = [ ]
for i in range(12):
    a=input("entre o próximo elemento: ")
    lista = lista + [a]
for i in range(12):
        print lista[i]
```

Listas

Exercício: Faça um programa que cor

0	-45
1	“ola”
2	[0]
3	“J”
4	[1,2]
5	[ ]
6	0
7	“62”
8	-3.0
9	“1”
10	6453L
11	[[23],[“alo”]]

Variável que permite armazenar mais de um valor ao mesmo tempo

```
lista = [-45, 'ola', [0], 'J', [1, 2], [ ], 0, '62', -3.0, '1',
6453L, [[23], ['alo']]]
for i in range(12):
        print lista[i]
```

```
lista = [ ]
for i in range(12):
    a=input("entre o próximo elemento: ")
    lista = lista + [a]
for i in range(12):
        print lista[i]
```

Inicializando a variável como uma lista vazia

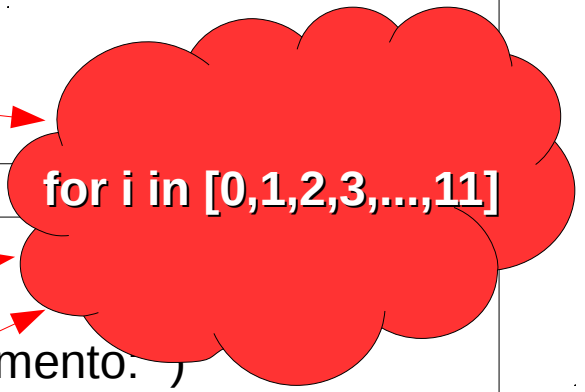
Listas

Exercício: Faça um programa que construa a seguinte lista e a imprima:

0	-45
1	“ola”
2	[0]
3	“J”
4	[1,2]
5	[ ]
6	0
7	“62”
8	-3.0
9	“1”
10	6453L
11	[[23],[“alo”]]

```
lista = [-45, 'ola', [0], 'J', [1, 2], [ ], 0, '62', -3.0, '1',
6453L, [[23], ['alo']]]
for i in range(12):
        print lista[i]
```

```
lista = [ ]
for i in range(12):
    a=input("entre o próximo elemento: ")
    lista = lista + [a]
for i in range(12):
        print lista[i]
```



Listas

Exercício: Faça um programa que construa a seguinte lista e a imprima:

0	-45
1	“ola”
2	[0]
3	“J”
4	[1,2]
5	[ ]
6	0
7	“62”
8	-3.0
9	“1”
10	6453L
11	[[23],[“alo”]]

```
lista = [-45, 'ola', [0], 'J', [1, 2], [ ], 0, '62', -3.0, '1',
6453L, [[23], ['alo']]]
for i in range(12):
        print lista[i]
```

```
lista = [ ]
for i in range(12):
    a=input("entre o próximo elemento: ")
    lista = lista + [a]
for i in range(12):
        print lista[i]
```

A operação + representa a *concatenação* de listas

Listas

Exercício: Faça um programa que construa a seguinte lista e a imprima:

0	-45
1	“ola”
2	[0]
3	“J”
4	[1,2]
5	[ ]
6	0
7	“62”
8	-3.0
9	“1”
10	6453L
11	[[23],[“alo”]]

```
lista = [-45, 'ola', [0], 'J', [1, 2], [ ], 0, '62', -3.0, '1',
6453L, [[23], ['alo']]]
for i in range(12):
        print lista[i]
```

[...] + [a]
=
[..., a]
A lista funciona como uma *FILA*

```
lista = [ ]
for i in range(12):
    a=input("entre c...no e...: ")
    lista = lista + [a]
for i in range(12):
        print lista[i]
```

```
>>> lista
    [-45, 'ola', [0], 'J', [1, 2], [ ], 0, '62', -3.0, '1', 6453L, [[23], ['alo']]]
```

Listas

Exercício: Faça um programa que construa a seguinte lista e a imprima:

0	-45
1	“ola”
2	[0]
3	“J”
4	[1,2]
5	[ ]
6	0
7	“62”
8	-3.0
9	“1”
10	6453L
11	[[23],[“alo”]]

```
lista = [-45, 'ola', [0], 'J', [1, 2], [ ], 0, '62', -3.0, '1',
6453L, [[23], ['alo']]]
for i in range(12):
        print lista[i]
```

[a] + [...]
=
[a, ...]
A lista funciona como uma *PILHA*

```
lista = [ ]
for i in range(12):
    a=input("entre c[illegible] ")
    lista = [a] + lista
for i in range(12):
        print lista[i]
```

```
>>> lista
    [[[23], ['alo']], 6453L, '1', -3.0, '62', 0, [ ], [1, 2], 'J',[0], 'ola', -45]
```

Listas

```
>>> [1,2] + [3]
```

```
>>> [1,2] + [[3]]
```

```
>>> [[1,2]] + [[3]]
```

```
>>> [1,2] * 3
```

```
>>> [1,2] * [3]
```

```
>>> [1,2] – [3]
```

Listas

```
>>> [1,2] + [3]
    [1, 2, 3]
>>> [1,2] + [[3]]
    [1, 2, [3]]
>>> [[1,2]] + [[3]]
    [[1, 2], [3]]
>>> [1,2] * 3
    [1, 2, 1, 2, 1, 2]
>>> [1,2] * [3]
    Traceback (most recent call last):
      File "<pyshell#35>", line 1, in <module>
        [1,2]*[3]
    TypeError: can't multiply sequence by non-int of type 'list'
```

Equivalente a [1,2]+[1,2]+[1,2]

```
>>> [1,2] – [3]

Traceback (most recent call last):
  File "<pyshell#37>", line 1, in <module>
    [1,2]-[2]
TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

Listas

Exercício: Faça um programa que soma 1 aos elementos pares da lista:

0	3
1	4
2	2
3	5
4	1
5	87
6	12
7	34
8	67
9	89
10	90
11	4

Listas

Exercício: Faça um programa que soma 1 aos elementos pares da lista:

0	3
1	4
2	2
3	5
4	1
5	87
6	12
7	34
8	67
9	89
10	90
11	4

```
lista = [3,4,2,5,1,87,12,34,67,89,90,4]
for i in range(12):
        if lista[i] % 2 == 0 :
                lista[i] = lista[i]+1
print lista
```

```
>>> [3, 5, 3, 5, 1, 87, 13, 35, 67, 89, 91, 5]
```

Listas

Exercício: Faça um programa que soma 1 aos elementos nas posições ímpares da lista:

0	3
1	4
2	2
3	5
4	1
5	87
6	12
7	34
8	67
9	89
10	90
11	4

Listas

Exercício: Faça um programa que soma 1 aos elementos nas posições ímpares da lista:

0	3
1	4
2	2
3	5
4	1
5	87
6	12
7	34
8	67
9	89
10	90
11	4

```
lista = [3,4,2,5,1,87,12,34,67,89,90,4]
for i in range(12):
        if (i+1) % 2 == 1 :
                lista[i] = lista[i]+1
print lista
```

```
>>> [4, 4, 3, 5, 2, 87, 13, 34, 68, 89, 91, 4]
```

Listas

Exercício: Faça um programa que construa uma lista de inteiros de tamanho indeterminado. A construção deve terminar quando for digitado a palavra "fim". Considere que a lista vazia pode ser um dos resultados deste programa.

Listas

Exercício: Faça um programa que construa uma lista de inteiros de tamanho indeterminado. A construção deve terminar quando for digitado a palavra “fim”. Considere que a lista vazia pode ser um dos resultados deste programa.

```
lista = [ ]
elemento = raw_input("Entre com o primeiro elemento da lista (digite
                    'fim' para terminar): ")
while elemento != "fim":
    lista = lista + [int(elemento)]
    elemento = raw_input("Entre com o próximo elemento da lista
                        (digite 'fim' para terminar): ")
print lista
```

Listas

Exercício: O que faz o seguinte programa?

```
v1 = [ ]
for i in range( 10 ):
    nv = int( raw_input( "Entre com valor %d: " % ( i + 1 ) ) )
    v1 += [ nv ]

for i in range( len( v1 ) ):
    print "%7d %10d %s" % ( i, v1[ i ], "*" * v1[ i ] )
```

Listas

Exercício: O que faz o seguinte programa?

```
v1 = [ ]
for i in range( 10 ):
    nv = int( raw_input( "Entre com valor %d: " % ( i + 1 ) ) )
    v1 += [ nv ]

for i in range( len( v1 ) ):
    print "%7d %10d %s" % ( i, v1[ i ], "*" * v1[ i ] )
```

Retorna o tamanho de uma lista

Listas

Exercício: O que faz o seguinte programa?

```
v1 = [ ]
for i in range( 10 ):
    nv = int( raw_input( "Entre com valor %d: " % ( i + 1 ) ) )
    v1 += [ nv ]

for i in range( len( v1 ) ):
    print "%7d %10d %s" % ( i, v1[ i ], "*" * v1[ i ] )
```

```
>>>
Entre com valor 1: 19
Entre com valor 2: 3
Entre com valor 3: 15
Entre com valor 4: 7
Entre com valor 5: 11
Entre com valor 6: 9
Entre com valor 7: 13
Entre com valor 8: 5
Entre com valor 9: 17
Entre com valor 10: 1
      0         19 *******************
      1          3 ***
      2         15 ***************
      3          7 *******
      4         11 ***********
      5          9 *********
      6         13 *************
      7          5 *****
      8         17 *****************
      9          1 *
```

Listas

Exercício: Faça um programa que leia 5 notas, armazene-as em uma lista, imprima cada uma delas e calcule a média de tais notas, com precisão de uma casa decimal. Use *while* nas estruturas de repetição.

Listas

Exercício: Faça um programa que leia 5 notas, armazene-as em uma lista, imprima cada uma delas e calcule a média de tais notas, com precisão de uma casa decimal. Use *while* nas estruturas de repetição.

```
notas = [0,0,0,0,0]
soma = 0
i = 0
while i < 5:
    notas[i]=float(input("Nota %d: " % (i+1)))
    soma = soma + notas[i]
    i = i + 1
i=0
while i < 5:
    print "Nota %d : %.1f" % (i+1,notas[i])
    i = i + 1
print "Média: %.1f" % (soma/i)
```

notas=[0]*5

Reescreva o programa com for

Listas

Exercício: Faça um programa que leia 5 notas, armazene-as em uma lista, e permita que o usuário possa escolher qual a nota ele quer imprimir. O programa deve terminar quando o usuário digitar 0. Use ***while*** nas estruturas de repetição. Não pode usar ***break***.

Listas

Exercício: Faça um programa que leia 5 notas, armazene-as em uma lista, e permita que o usuário possa escolher qual a nota ele quer imprimir. O programa deve terminar quando o usuário digitar 0. Use ***while*** nas estruturas de repetição. Não pode usar ***break***.

```
notas = [0]*5
i = 0
while i < 5:
    notas[i]=float(input("Nota %d: " % (i+1)))
    i = i + 1
escolha=int(input(“Qual nota você quer imprimir (0 para sair): “))
while escolha != 0:
    print "Nota %d : %.1f" % (escolha,notas[escolha-1])
    escolha=int(input(“Qual nota você quer imprimir (0 para sair): “))
```

Reescreva o programa com for

Listas

Manipulação de Listas

Além dos operadores + (concatenação) e * (usado para múltiplas concatenações) podemos manipular listas usando:

- **append** : outra forma de concatenação. Neste caso, a lista é tratada como uma **fila**.
- **extend** : permite adicionar os elementos de uma lista a outa

- **del** : remover elemento de uma lista

```
>>> Lista = [ ]

>>> Lista.append(“a”)

>>> Lista
['a']

>>> Lista.append(2)
```

```
>>> Lista
['a',2]
>>> Lista.append([3,"f"])

>>> Lista
['a',2,[3,'f']]
```

```
>>> Lista.extend([“q”])

>>> Lista
['a',2,[3,'f'], 'q']
>>> Lista.extend(2)
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    l.extend(2)
TypeError: 'int' object is not iterable
```

Listas

Manipulação de Listas

Além dos operadores + (concatenação) e * (usado para múltiplas concatenações) podemos manipular listas usando:

- **append** : outra forma de concatenação. Neste caso, a lista é tratada como uma **fila**.
- **extend** : permite adicionar os elementos de uma lista a outa

- **del** : remover elemento de uma lista

```
>>> Lista
    ['a',2,[3,'f'], 'q']
>>> del Lista[1]
>>> Lista
    ['a',[3,'f'], 'q']
>>> del Lista[2]
>>> Lista
    ['a',[3,'f']]
```

```
>>> del Lista[1] [1]
    ['a',[3]]
```

Como o segundo elemento de Lista é uma lista, posso retirar desta seu segundo elemento.

Aula 8 – Estrutura de Dados