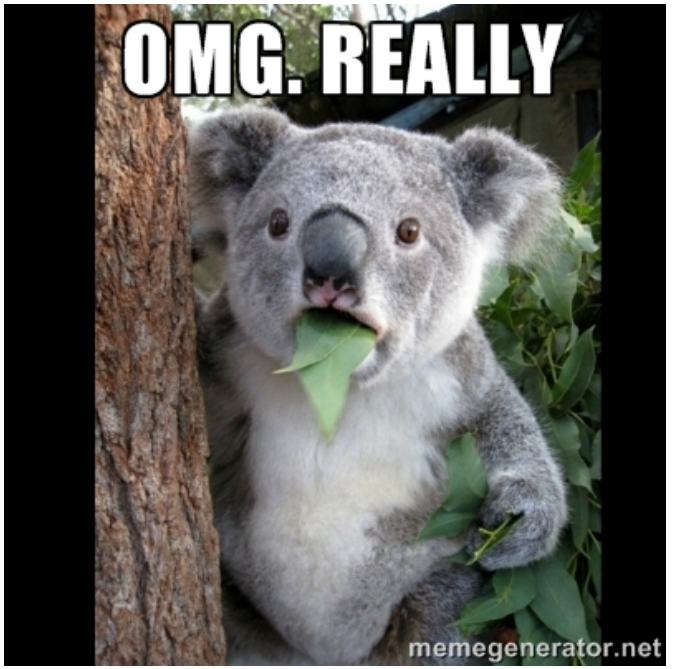
COMPUTAÇÃO 1 – PYTHON
AULA 5 TEÓRICA

SILVIA BENZA

SILVIA@BENZA.COM.PY

MANIPULAÇÃO DE STRINGS

MAIS STRINGS?



COMO ASSIM, TEM AINDA MAIS FUNÇÕES?

COMO ACHAR ELAS????

LEMBRA!!!!

Para obter ajuda a respeito de um tipo de dado, digite help(tipo).

- Por exemplo: help(str) para obter ajuda sobre strings, assim como help(int) para ajuda sobre inteiros.

Existem várias funções disponíveis para executar diferentes tarefas com strings. A sintaxe para estas funções é:

- str._nomeFunção_ (umaString, _parâmetros_)

Exemplo

>>>str.upper('abcde') 'ABCDE'

MANIPULAÇÃO DE STRINGS

lower(): retorna a string com todos os caracteres maiúsculos convertidos para minúsculos.

upper(): retorna a string com todos os caracteres minúsculos convertidos para maiúsculos.

Exemplo

```
>>> str.upper("TeamCap")
TEAMCAP
>>> str.lower("MENTIRA, nem vi ainda (2017 e continuo sem ver!)")
mentira, nem vi ainda (2017 e continuo sem ver!)
```

MANIPULAÇÃO DE STRINGS

str.count(umaString, elemento, inicio, fim): retorna quantas vezes o elemento aparece na string, procurando-se a partir da posição **inicio** e indo até a posição **fim - 1**.

inicio e fim são opcionais.

Exemplo

```
>>> frase="Teria sido melhor assistir o filme de Pele"
>>> str.count(frase, "a", 2, 10)
```

MANIPULAÇÃO DE STRINGS

str.count(umaString, elemento, inicio, fim): retorna quantas vezes o elemento aparece na string, procurando-se a partir da posição **inicio** e indo até a posição **fim - 1**.

inicio e fim são opcionais.

Exemplo

```
>>> frase="Teria sido melhor assistir o filme de Pele"
>>> str.count(frase, "a", 2, 10)
1
```

MANIPULAÇÃO DE STRINGS

str.index(umaString,elemento, inicio, fim): retorna o índice da primeira ocorrência de elemento na string, a partir da posição **inicio**, até a posição **fim - 1**.

inicio e fim são opcionais.

Exemplo

```
>>> str.index("mariana", "a”)

>>> str.index("mariana", "a", 2)

>>> str.index("mariana", "a",5, 7)

>>> str.index(’Mariana’, ’ana’)

>>> str.index(’Mariana’, ’x’)
```

MANIPULAÇÃO DE STRINGS

str.index(umaString,elemento, inicio, fim): retorna o índice da primeira ocorrência de elemento na string, a partir da posição **inicio**, até a posição **fim - 1**.

inicio e fim são opcionais.

Exemplo

```
>>> str.index("mariana", "a”)
1
>>> str.index("mariana", "a", 2)
4
>>> str.index("mariana", "a",5, 7)
6
>>> str.index(’Mariana’, ’ana’)
4
>>> str.index(’Mariana’, ’x’)
Traceback (most recent call last): File "<pyshell#1>", line 1, in <module> str.index(’Mariana’, ’x’)
ValueError: substring not found
```

TUPLAS

TUPLAS

Uma tupla é uma sequência heterogênea (permite que seus elementos sejam de tipos diferentes):

```
>>> a = (1,2,3,4)
>>> b = (1.0, 2, '3', 4+0j)
>>> c = 1,2,3,4
>>> d = (1,)
```

TUPLAS

Valores em uma tupla podem ser distribuídos em variáveis como uma atribuição múltipla:

```
>>> x = 1, 2, 3
>>> x


>>> a, b, c = x
>>> a


>>> b


>>> c
```

TUPLAS

Valores em uma tupla podem ser distribuídos em variáveis como uma atribuição múltipla:

```
>>> x = 1, 2, 3
>>> x
(1, 2, 3)
>>> a, b, c = x
>>> a
1
>>> b
2
>>> c
3
```

TUPLAS

Uma tupla vazia se escreve ()

Os parênteses são opcionais se não provocarem ambigüidade

Uma tupla contendo apenas um elemento deve ser escrita com uma vírgula ao final

- Um valor entre parênteses sem vírgula no final é meramente uma expressão:

```
>>> (10)
>>> 10,
>>> (10,)
>>> 3*(10+3)
>>> 3*(10+3,)
```

TUPLAS

Uma tupla vazia se escreve ()

Os parênteses são opcionais se não provocarem ambigüidade

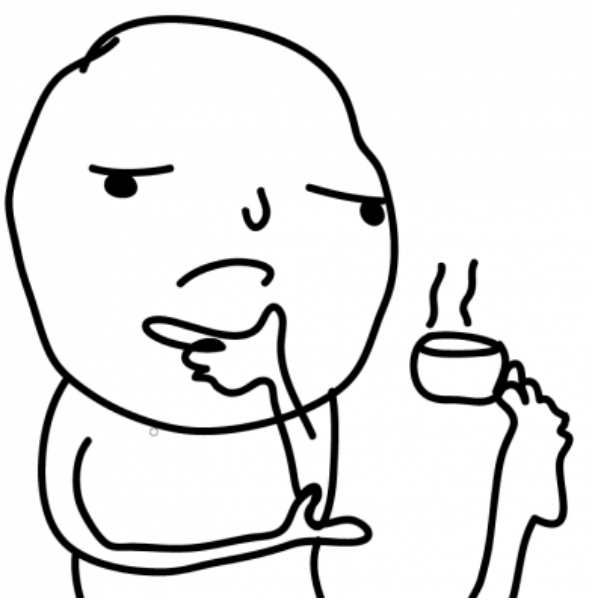
Uma tupla contendo apenas um elemento deve ser escrita com uma vírgula ao final

- Um valor entre parênteses sem vírgula no final é meramente uma expressão:

```
>>> (10)
10
>>> 10,
(10,)
>>> (10,)
(10,)
>>> 3*(10+3)
39
>>> 3*(10+3,)
(13, 13, 13)
```

UMA TUPLA É UMA SEQUÊNCIA HETEROGÊNEA

MAS UMA STRING TAMBÉM É UMA SEQUÊNCIA



OPERAÇÕES COM TUPLAS

Tuplas são muito similares às strings em relação às operações.

Tamanho de uma tupla: é dado pela função len.

```
>>> x = (1,2,3)
>>> len(x)
3
```

Indexação: começando do 0 à esquerda, ou de -1 à direita.

```
>>> x[0]
1
```

Fatiamento: idêntico às strings.

```
>>> x[0:2]
(1,2)
```

OPERAÇÕES COM TUPLAS

Concatenação e Replicação

```
>>> x*2
(1,2,3,1,2,3)
>>> x + (5,4)
(1,2,3,5,4)
```

Imutabilidade : uma vez criada, uma tupla não pode ser alterada !

```
>>> x[0] = 9
Traceback (most recent call last):
File "<pyshell#2>", line 1, in <module> x[0]=9
TypeError: 'tuple' object does not support item assignment
```

LEMBRAM DO JOÃOZINHO?

Joãozinho quer comprar o maior número de bombons possível com o dinheiro que tem. Faça funções para:

a. calcular o número de bombons e o troco, dados o dinheiro e o preço de um bombom.

LEMBRAM DO JOÃOZINHO?

Joãozinho quer comprar o maior número de bombons possível com o dinheiro que tem. Faça funções para:

a. calcular o número de bombons e o troco, dados o dinheiro e o preço de um bombom.

```
def bombom(dinheiro,preco):
    return (dinheiro)//preco, dinheiro%preco
```

LEMBRAM DO JOÃOZINHO?

Joãozinho quer comprar o maior número de bombons possível com o dinheiro que tem. Faça funções para:

a. calcular o número de bombons e o troco, dados o dinheiro e o preço de um bombom.

```
def bombom(dinheiro,preco):
    return (dinheiro)/preco, dinheiro%preco
```

b. calcular quanto Joãozinho terá que pedir para sua mãe para comprar um bombom a mais, dados o dinheiro que ele tem e o preço de um bombom. Utilize a função definida em a.

LEMBRAM DO JOÃOZINHO?

Joãozinho quer comprar o maior número de bombons possível com o dinheiro que tem. Faça funções para:

a. calcular o número de bombons e o troco, dados o dinheiro e o preço de um bombom.

```
def bombom(dinheiro,preco):
    return (dinheiro)//preco, dinheiro%preco
```

b. calcular quanto Joãozinho terá que pedir para sua mãe para comprar um bombom a mais, dados o dinheiro que ele tem e o preço de um bombom. Utilize a função definida em a.

```
def maisbombom(dinheiro,preco):
    return preco – bombom(dinheiro,preco)[1]
```

LISTAS

LISTAS

Tipo de dados mais versátil do Python.

Uma lista é representada como uma sequência de valores entre colchetes e separados por vírgula.

Os elementos de uma lista podem ser de tipos de dados diferentes.

LISTAS SÃO MUTÁVEIS

LISTAS

Tipo de dados mais versátil do Python.

Uma lista é representada como uma sequência de valores entre colchetes e separados por vírgula.

Os elementos de uma lista podem ser de tipos de dados diferentes.

LISTAS SÃO MUTÁVEIS

```
>>> lista1 = [’calculo’, ’fisica’, ’computacao’]
>>> lista2 = [’notas’, 5.4, ’aprovado’]
>>> lista2[1] = 6
>>> lista2
[’notas’, 6, ’aprovado’]
```

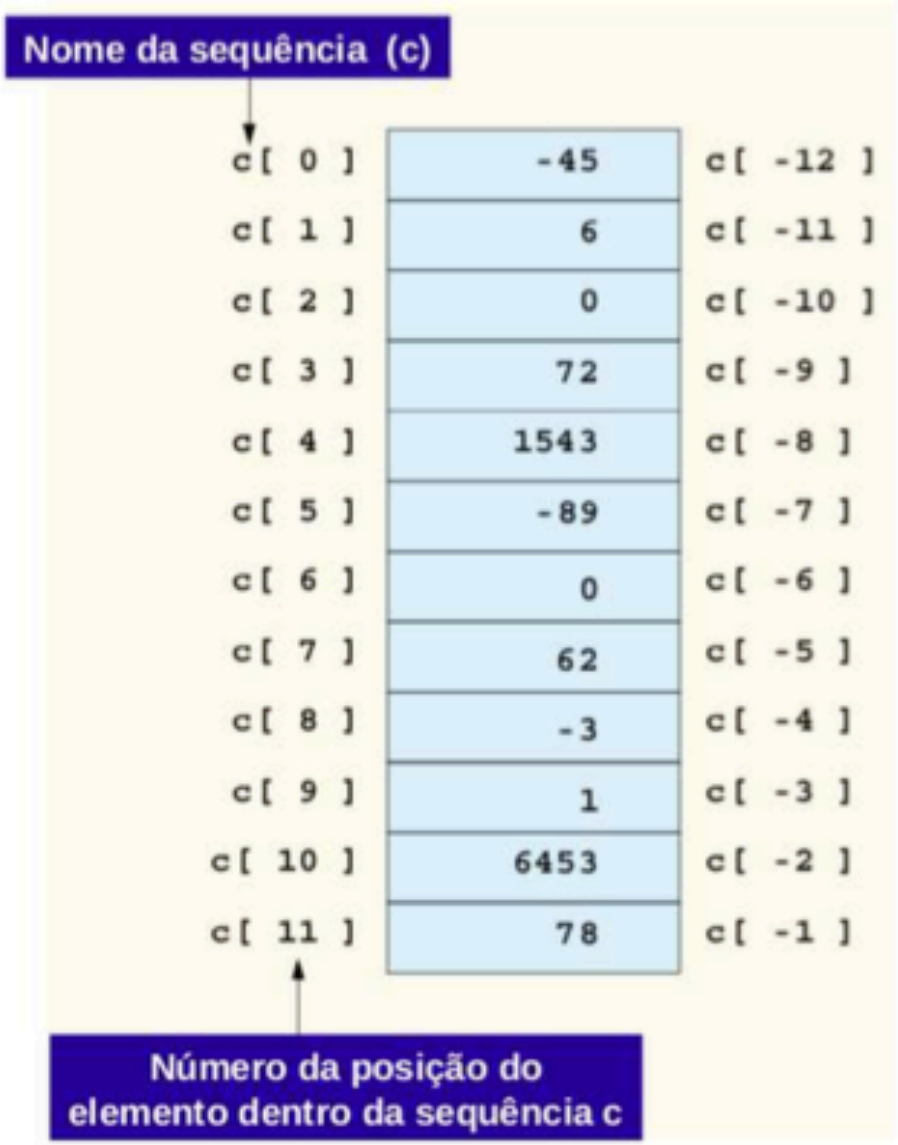
LISTAS

Atenção: Uma lista vazia não contém nenhum elemento

Exemplo

```
>>> lista3 = [ ]
>>> lista3[0]
Traceback (most recent call last):
File "<pyshell#18>", line 1, in <module> lista3[0]
IndexError: list index out of range
```

O ÍNDICE É DO MESMO JEITO???




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

>>> c[3]
72

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

>>> len(c)
12
```

AS OPERAÇÕES TAMBÉM??

Listas são variedades de seqüências assim como strings e portanto têm APIs semelhantes

- Podem ser indexadas e fatiadas
- Podem ser concatenadas (+) e repetidas

STRINGS VS LISTA

Há diferenças importantes entre listas e strings

- Seqüência genérica (similar a tuplas) X de seqüência de caracteres
- Elementos de listas podem ser alterados individualmente mas os de strings, não

Listas constituem o tipo de agregação de dados mais versátil e comum da linguagem Python

Podem ser usadas para implementar estruturas de dados mais complexas como matrizes e árvores, por exemplo

LISTAS

```
>>> [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]
```

(Concatenando Listas)

(Equivale a [1,2]+[1,2]+[1,2])

LISTAS

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

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

LISTAS

```
>>> [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’
```

Se é mutável, porque não posso retirar um elemento?
(Não se apressure)

```
>>> [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

Faça uma função que receba duas listas como entrada e retorne a concatenação destas listas.

LISTAS

Faça uma função que receba duas listas como entrada e retorne a concatenação destas listas.

```
# Função que dadas duas listas,
# retorna a concatenação das listas
# list,list → list
def concatenaListas(Lista1,Lista2):
    return Lista1+Lista2
```

>>> concatenaListas([1,2,3],[4,5,6])

[1,2,3,4,5,6]

LISTAS

Faça uma função que dado um número inteiro como entrada, retorne uma lista com todos os números pares entre 1 e o número dado, inclusive

LISTAS

Faça uma função que dado um número inteiro como entrada, retorne uma lista com todos os números pares entre 1 e o número dado, inclusive

Como faz isso?????

RANGE

FUNÇÃO RANGE

A função range(...) pode ter 1, 2 ou 3 argumentos:
range(numero): retorna uma lista contendo uma sequência de valores de 0 a numero-1

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

range(inf,sup): retorna uma lista contendo uma sequência de valores de inf a sup-1

```
>>> range(3, 8)
[3, 4, 5, 6, 7]
```

range(inf, sup, inc): retorna uma lista contendo uma sequência de valores de inf a sup-1 com incremento de inc

```
>>> range(3, 8, 2)
[3, 5, 7]
```

FUNÇÃO RANGE

ATENÇÃO:

A função range(...) começa com zero

São equivalentes:

range(10)

range(0,10)

range(0,10,1)

Exemplos:

```
>>> range(3)

>>> range(2,5,2)

>>> range(5,2,-2)
```

FUNÇÃO RANGE

ATENÇÃO:

A função range(...) começa com zero

São equivalentes:

range(10)

range(0,10)

range(0,10,1)

Exemplos:

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

LISTAS

Faça uma função que dado um número inteiro como entrada, retorne uma lista com todos os números pares entre 1 e o número dado, inclusive

Como faz isso?????
USANDO RANGE!!!

LISTAS

```
# Função que dado um número inteiro,
# retorna uma lista com todos os números
# pares entre 1 e o número dado, inclusive
# int → list
def lista(n):
    if n%2==0:
        return range(2,n+1,2)
    else:
        return range(2,n,2)
```

>>>lista(5)

[2,4]

>>>lista(6)

[2,4,6]

RESUMO

Para obter ajuda a respeito de um tipo de dado, digite help(tipo).

A sintaxe para utilizar as funções é:

str._nomeFunção_ (umaString, _parâmetros_)

Listas e Tuplas são sequências genéricas

- Listas são mutáveis, enquanto que tuplas não!
- Operações similares de str, como concatenação, repetição, indexação, fatiamento.

A função range(...) começa com zero. São equivalentes:
range(10) <==> range(0,10) <==> range(0,10,1)

COMPUTAÇÃO 1 – PYTHON
AULA 5 TEÓRICA

SLIDES BASEADOS NOS TRABALHOS:
AULAS TEÓRICAS DO DCC UFRJ
AULA DO CLAUDIO ESPERANÇA DO PESC