Computação I - Python
Aula 1 - Teórica: Manipulação de Strings, Tuplas e Listas

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ

Manipulação de Strings

- Para obter ajuda a respeito de um tipo de dado, digite help(tipo).
- Por exemplo: help(str) para obter ajuda sobre strings, help(int) para ajuda sobre inteiros, etc.
- 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("Esperanca")
    ESPERANCA

>>> str.lower("Pe de Laranja Lima")
    pe de laranja lima
```

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**.
- **inicio** e **fim** são opcionais.
- **Exemplo**

```
>>> frase="macaco come banana"
>>> 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**.
- **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**.
- **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
```

Formatação de Strings

str.format(formatString, p0, p1, ...,), retorna uma string formatada segundo a formatString, contendo os dados indicados em p0, p1, ...

Formatação de Strings

str.format(formatString, p0, p1, ...,), retorna uma string formatada segundo a formatString, contendo os dados indicados em p0, p1, ...

```
1 >>> str.format('A soma de {0} e {1} eh {2}', 2, 3, 2+3)
2 # entre {} estao os indices dos valores que devem ser mostrados.
3 'A soma de 2 e 3 eh 5'
```

Formatação de Strings

str.format(formatString, p0, p1, ...,), retorna uma string formatada segundo a formatString, contendo os dados indicados em p0, p1, ...

```
1 >>> str.format('A soma de {0} e {1} eh {2}', 2, 3, 2+3)
2 # entre {} estao os indices dos valores que devem ser mostrados.
3 'A soma de 2 e 3 eh 5'
```

- A string de formato (formatString) é uma string contendo um ou mais códigos (que indicam campos a serem substituídos) inseridos em texto;
- Os campos a serem substituídos são códigos que aparecem entre { }. Tudo o que estiver entre essas chaves será substituído por um valor;
- Qualquer outra coisa que apareça na string de formato será copiada para a string de retorno;
- p0,p1,... são parâmetros posicionais. Eles indicarão que valor será inserido na string de formato, no lugar indicado pelos { }.

Formatação de Strings

str.format(formatString, p0, p1, ...,), retorna uma string formatada segundo a formatString, contendo os dados indicados em p0, p1, ...

- Os parâmetros p1,p2,... são inseridos na string de formato nas respectivas posições entre chaves

```
>>> str.format('A soma de {0} e {1} eh {2}', 2, 3, 2+3)
# entre {} estao os indices dos valores que devem ser mostrados.
'A soma de 2 e 3 eh 5'

>>> str.format('A soma de {1} e {2} eh {3}', 2, 3, 2+3)
# lembrando que o primeiro indice eh zero!
  Traceback (most recent call last):
  File "<ipython-input-44-287c69f8968c>", line 1, in <module>
    str.format('A soma de {1} e {2} eh {3}', 2, 3, 2+3)
  IndexError: tuple index out of range
```

Formatação de Strings

str.format(formatString, p0, p1, ...,), retorna uma string formatada segundo a formatString, contendo os dados indicados em p0, p1, ...

```
>>> str.format('{2} eh a soma de {0} e {1}', 2, 3, 2+3)
# entre {} estao os indices dos valores que devem ser
    mostrados.
'5 eh a soma de 2 e 3'
```

- Podemos omitir os índices dos dados dentro das chaves caso apareçam na mesma ordem nos argumentos.

```
>>> str.format('A soma de {} e {} eh {}', 2, 3, 2+3)
'A soma de 2 e 3 eh 5'
```

Formatação de Strings

str.format(formatString, p0, p1, ...,), retorna uma string formatada segundo a formatString, contendo os dados indicados em p0, p1, ...

- São usados códigos específicos para definir o formato dos dados a serem inseridos na string.

```
>>> str.format('A soma de {0:3d} e {1:3d} eh {2:5d}', 200, 37, 200+37)
#formato {indice: codigo de formatacao }
# d -> inteiro no formato decimal
# {Nd} N eh um numero que indica quantos digitos
'A soma de 200 e  37 eh   237'

>>> str.format('A soma de {0:3.2f} e {1:3.3f} eh {2:3.4f}', 2, 37.005, 2+37.005)
#formato {indice: codigo de formatacao }
# f -> float,
# {numero1.numero2f} indica digitos da parte inteira e da parte fracionaria
'A soma de 2.00 e 37.005 eh 39.0050'
```

Formatação de Strings

str.format(formatString, p0, p1, ...,), retorna uma string formatada segundo a formatString, contendo os dados indicados em p0, p1, ...

- Todos os dados float abaixo foram formatados com o código {6.2f}

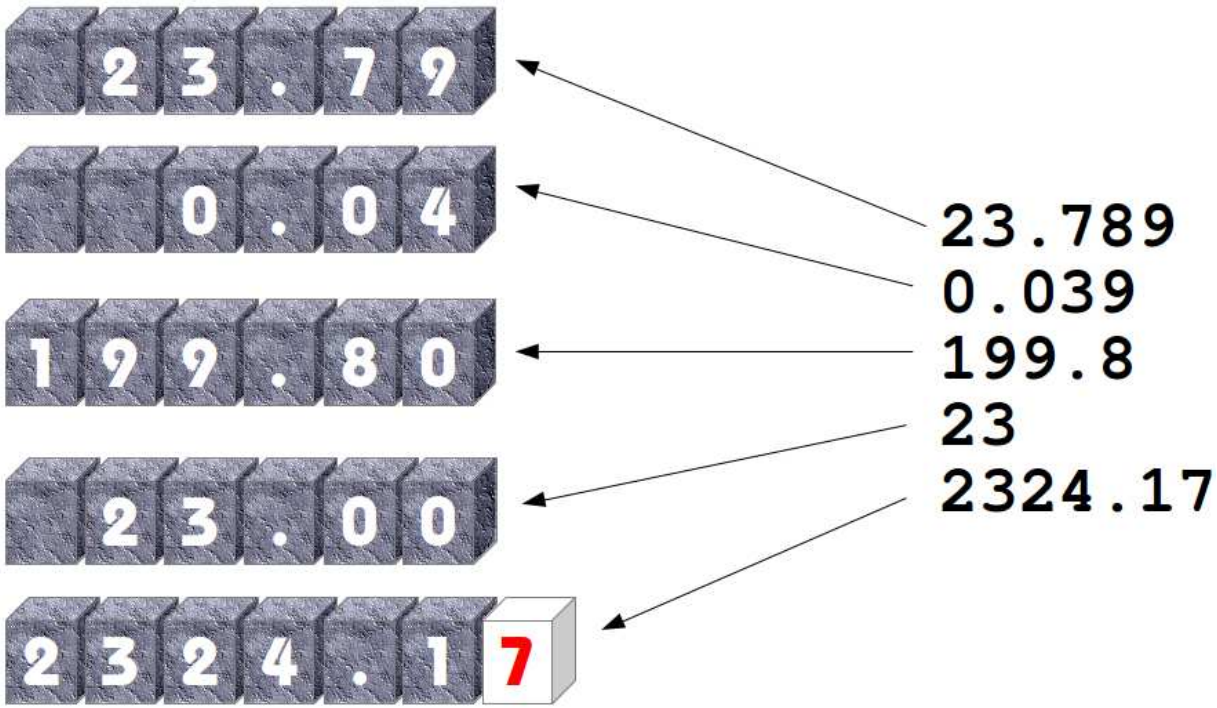


Figure: Exemplos de códigos de formatação para floats.
Fonte: www.python-course.eu

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,)
```

- 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

- **Tupla Vazia**: tupla sem elementos.
- **Tupla unitária**: contém um único elemento, que deve ser sucedido por uma vírgula.
- Os parênteses são opcionais se não provocarem ambiguidade.
- Um valor entre parênteses sem vírgula no final é meramente uma expressão.

Qual o tipo de dado da variável A em cada um dos casos abaixo:

```
1  >>> A = ()
2
3  >>> A = (10)
4
5  >>> A = 10,
6
7  >>> A = (10,)
8
9  >>> A = 3*(10+3)
10
11 >>> A = 3*(10+3,)
```

Tuplas

- **Tupla Vazia**: tupla sem elementos.
- **Tupla unitária**: contém um único elemento, que deve ser sucedido por uma vírgula.
- Os parênteses são opcionais se não provocarem ambiguidade.
- Um valor entre parênteses sem vírgula no final é meramente uma expressão.

Qual o tipo de dado da variável A em cada um dos casos abaixo:

```
>>> A = ()
    () # tupla vazia
>>> A = (10)
    10 # inteiro
>>> A = 10,
    (10,) # tupla unitaria
>>> A = (10,)
    (10,) # tupla unitaria
>>> A = 3*(10+3)
    39 # inteiro
>>> A = 3*(10+3,)
    (13, 13, 13) # tupla
```

Tuplas

- Tuplas são muito similares às strings em relação às operações.
- O 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)   # NOVA TUPLA
```

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
```

Tuplas

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.

Tuplas

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.

```
1 def bombom(dinheiro,preco):
2
3   """Os parametros de entrada sao do tipo (float,float).
4   O valor de retorno e do tipo tupla (float,float)"""
5
6   return dinheiro // preco , dinheiro % preco
```

Tuplas

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.

```
1 def bombom(dinheiro,preco):
2
3   """Os parametros de entrada sao do tipo (float,float).
4   O valor de retorno e do tipo tupla (float,float)"""
5
6   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.

Tuplas

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.

```
1 def bombom(dinheiro,preco):
2
3   """Os parametros de entrada sao do tipo (float,float).
4   O valor de retorno e do tipo tupla (float,float)"""
5
6   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.

```
1 def maisbombom(dinheiro,preco):
2
3   """Os parametros de entrada sao do tipo (float,float).
4   O valor de retorno e do tipo float"""
5
6   return preco - bombom(dinheiro,preco)[1]
```

Testes: bombom(10,3) e maisbombom(10,3)

Tuplas

a. Escreva uma função que recebe uma tupla e retorna **True** se o primeiro elemento for igual ao último elemento da tupla.

Tuplas

a. Escreva uma função que recebe uma tupla e retorna **True** se o primeiro elemento for igual ao último elemento da tupla.

```
1 def igual_if(tup):
2
3   """Funcao que retorna True se o inicio de uma tupla e igual
      ao seu final.
4   O parametros de entrada e do tipo tupla.
5   O valor de retorno e do tipo booleano."""
6
7   return tup[0] == tup[-1]
```

Tuplas

a. Escreva uma função que recebe uma tupla e retorna **True** se o primeiro elemento for igual ao último elemento da tupla.

```
1 def igual_if(tup):
2
3   """Funcao que retorna True se o inicio de uma tupla e igual
      ao seu final.
4   O parametros de entrada e do tipo tupla.
5   O valor de retorno e do tipo booleano."""
6
7   return tup[0] == tup[-1]
```

b. Escreva uma função *inverte* que recebe uma tupla de três elementos e retorna uma nova tupla com os elementos na ordem reversa.

Tuplas

b. Escreva uma função *inverte* que recebe uma tupla de três elementos e retorna uma nova tupla com os elementos na ordem reversa.

```
1 def inverte(tup):
2
3   """Funcao que inverte elementos de uma tupla de tamanho 3.
4   O parametros de entrada e uma tupla de tamanho 3.
5   O valor de retorno e uma tupla de tamanho 3."""
6
7   return tup[2], tup[1], tup[0]
```

```
1 def inverte(tup):
2
3   """Funcao que inverte elementos de uma tupla de tamanho 3.
4   O parametros de entrada e uma tupla de tamanho 3.
5   O valor de retorno e uma tupla de tamanho 3."""
6
7   return tup[::-1]
```

Tuplas

Exercícios

c. Escreva a função *intercala* que recebe duas tuplas de três elementos cada e retorna uma tupla de seis elementos intercalando as duas tuplas.

d. Escreva a função *opera* que recebe uma tupla com uma string e dois números; se a string for 'SOMA', retorna a soma dos dois números, se for 'MULT', retorna a multiplicação, se for 'DIV', retorna a divisão, se for 'SUB', retorna a subtração, se não for nenhuma das anteriores retorna *None*.

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 !!!

Exemplo

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

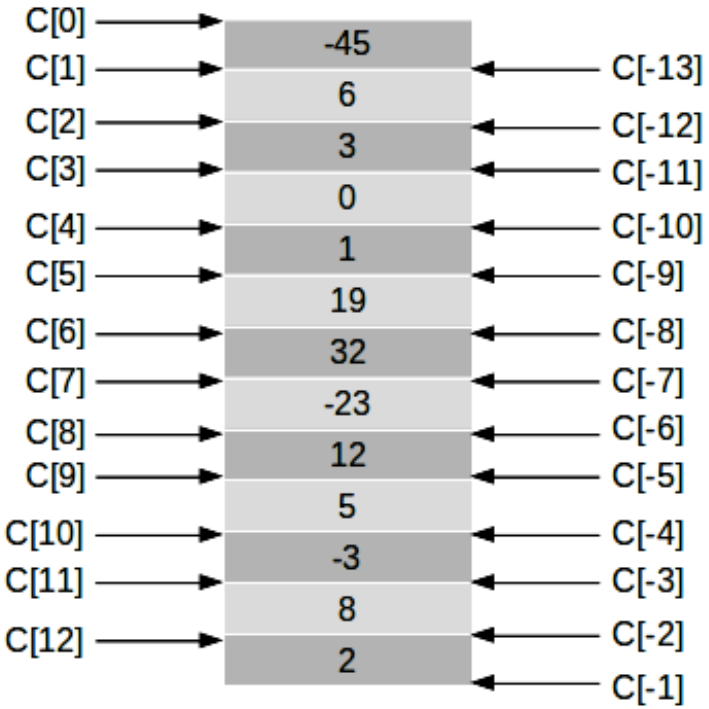
Listas

- 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
```

Listas




```
>>> c=[-45, 6, 3, 0, 1, 19, 32, -23, 12, 5, -3, 8, 2]
>>> c[3]
    0
>>> c[9]==c[-4]
    True
>>> len(c)
    13
```

Listas

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

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

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

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

Listas

```
>>> [1,2] + [3] # Concatenando Listas
    [1, 2, 3]

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

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

>>> [1,2] * 3 # Equivale a [1,2]+[1,2]+[1,2]
    [1, 2, 1, 2, 1, 2]
```

Listas

```
1 >>> [1,2] * [3]
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: cant multiply sequence by non-int of type '
    list'

>>> [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'
```

Como retirar um elemento de uma lista?
Aguarde

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.

```
1 def concatenaListas(Lista1,Lista2):
2
3   """Funcao que dadas duas listas, retorna a concatenacao das
      listas.
4   O parametros de entrada sao list,list.
5   O valor de retorno e list."""
6
7   return Lista1+Lista2
```

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

- 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

```
>>> list(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

```
>>> list(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

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

Listas

- **ATENÇÃO:** A função range(...) começa com zero
- São equivalentes:

range(10)

range(0,10)

range(0,10,1)

- **Exemplos**

```
1 >>> list(range(3))
2
3 >>> list(range(2,5,2))
4
5 >>> list(range(5,2,-2))
```

Listas

- **ATENÇÃO:** A função range(...) começa com zero
- São equivalentes:

range(10)

range(0,10)

range(0,10,1)

- **Exemplos**

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

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.

```
1 def lista(n):
2
3   """Funcao que dado um numero inteiro, retorna uma lista com todos
        os numeros pares entre 1 e o numero dado, inclusive.
4   O parametros de entrada e um int.
5   O valor de retorno e uma lista."""
6
7   return range(2,n+1,2)
```

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

Listas - Exercícios

1. Faça uma função que dada uma lista com 5 notas, retorne a média das notas.

2. Faça uma função que, dados dois inteiros x e y, retorna uma lista com todos os valores entre x e y (inclusive), funcionando tanto para $x <= y$ como para $x > y$.
   Exemplos
   x = 2, y = 6, resultado = [2, 3, 4, 5, 6]
   x = 10, y = 7, resultado = [10, 9, 8, 7]

3. Faça uma função que dadas duas listas de 3 numeros, representando dois vetores no espaço $\Re^3$, retorna uma lista que represente a soma destes dois vetores.
   Exemplo
   Lista1 = [1,4,6]
   Lista2 = [2,4,3]
   Lista resultante = [3,8,9]

Autores

- João C. P. da Silva ▸ Lattes
- Carla Delgado ▸ Lattes
- Ana Luisa Duboc ▸ Lattes

Colaboradores

- Anamaria Martins Moreira ▸ Lattes
- Fabio Mascarenhas ▸ Lattes
- Leonardo de Oliveira Carvalho ▸ Lattes
- Charles Figueiredo de Barros ▸ Lattes
- Fabrício Firmino de Faria ▸ Lattes

Computação I - Python
Aula 1 - Teórica: Manipulação de Strings, Tuplas e Listas

João C. P. da Silva

Carla A. D. M. Delgado

Ana Luisa Duboc

Dept. Ciência da Computação - UFRJ