Introdução ao Python

Aula 2 - String e Intervalos

1h30

Prof. Cláudio A. Fleury
Abr-21

Conteúdo

- Strings
  - Definição: classe p/ seq.s de caracteres
  - Delimitadores e Tipo de Dado `str`
  - Índices positivos e negativos
  - Ações
    - Criação
    - Impressão / Instrução _future_
    - Comprimento
    - Fatiamento
    - Concatenação
    - Métodos: upper/lower, split, strip, capitalize, replace,...
    - Impressão formatada (como **printf()** da ling. C)
      - Moda Antiga (2.x e 3.x)
      - Moda Nova (3.x)
  - Características: imutabilidade
- Intervalos
  - Definição
  - Função nativa: `range()`
    - Argumentos do tipo inteiro
    - Operador de pertencimento: **in**

Strings

- Definição
  - Estruturas de dados usadas com informações do tipo textual

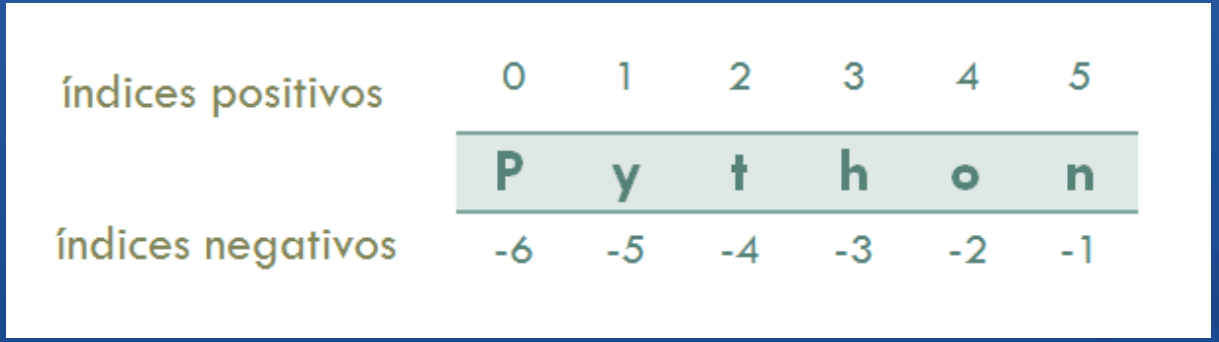
- Delimitadores
  - Apóstrofes ou aspas simples: `'texto'`
  - Aspas duplas: `"texto"`
  - Ambas formas:

```
'valor da var. "x" no script'
"valor da var. 'x' no script"
```

Strings

- Índice
  - Indica a posição de armazenamento de cada caractere na memória
  - Positivos e Negativos
    - Ex.:

```
ling = "Python"
ling[0]  ⟶ 'P'
ling[-1] ⟶ 'n'
```



Strings

- Ações
  - Criação: `'Bem-vindos ao Curso de do IFG 2019-1'`
  - Impressão
    - *Prompt* do Interpretador:

```
"Aprendizado de Máquina"
'Aprendizado de Máquina'
```

    - Impressão (3.x):

```
print("Aprendizado de Máquina")
Aprendizado de Máquina
```

  - Comprimento:

```
print(len("Entendeu?"))
9
```

Strings

- Ações
  - Indexação
    - Pode-se usar índices (valor(es) inteiro(s) entre colchetes) para acessar partes da sequência de caracteres armazenados numa string

```
curso = "Eng. de Controle e Automação"
print(curso[2])
g
```

  - Fatiamento (*slicing*) `string[início:final[:passo]]`
    - Acessando partes (fatias) da sequência de caracteres, a partir dos índices de inicio e final da parte desejada:

```
print(curso[8:9])
print(curso[8:16])
print(curso[8:])
print(curso[:4])
```

```
C
Controle
Controle e Automação
Eng.
```

Strings

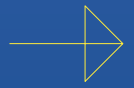
- Ações
  - Concatenação
    - Ajuntamento, justaposição de strings
      - Com o método 'join( )'

```
dias = ['Terça', 'Quinta', 'Sábado']          # lista de dias
concat = ' '.join(dias)
print(concat)
```

```
Terça Quinta Sábado
```

      - Com o operador '+'

```
print("Cláudio " + "Fleury")
```



```
Cláudio Fleury
```

  - Métodos
    - Funções associadas a um objeto que executam ações no próprio objeto
    - Uso: `nome_objeto.método(parâmetros)`
    - Exemplo:

```
s = 'Bom dia Bia!'
print(s.upper())
```

```
BOM DIA BIA!
```

Strings

- Ações
  - Impressão Formatada
    - Moda Antiga (printf() da ling. C) - formatação posicional com **operador %**
      - Indicada quando a ordem dos argumentos não precisa ser alterada

```
nome = "Matheus"
print("Nome: %s" % nome)
```

```
Nome: Matheus
```

    - Moda Nova - usa o método **format( )** e índice(s) posicional(is) explícito(s)
      - Permite a organização da ordem de exibição sem alterar os argumentos

```
print('{1}, {0}.'.format('Cláudio', 'Fleury'))
usuario = 'Maria'
print('{0}\n{1}\n{0}'.format('-'*30, usuario.center(30)))
```

```
Fleury, Cláudio.
------------------------------
             Maria
------------------------------
```

Strings

- Ações
  - Troca de caracteres
    - Faz a troca de um caractere por outro
    - Ex.:

```
print(s)
so = s.replace('a','o')
print(so)
```

```
Bom dia Bia!
Bom dio Bio!
```

Intervalos

- Função **`range([início,] final [,passo])`**
  - Gera um intervalo de valores iteráveis (contagem)
  - Função da biblioteca padrão do Python
  - O intervalo é uma sequência imutável de números inteiros
  - Números inteiros gerados dependem dos argumentos usados na função
    - Ex.:

```
for i in range(6):
    print(i, end=' - ')
```

```
0 - 1 - 2 - 3 - 4 - 5 -
```

Intervalos

- Exemplos
  - Exibição dos múltiplos de 3, de 3 até 50.
    - Ex.:

```
for i in range(3,51,3):
    print(i, end=' - ')
```

```
3 - 6 - 9 - 12 - 15 - 18 - 21 - 24 - 27 - 30 - 33 - 36 - 39 - 42 - 45 - 48 -
```