

Profa. Dra. Raquel C. de Melo-Minardi
Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade Federal de Minas Gerais



MÓDULO 2 – PROGRAMAÇÃO

Sequências – Strings

TIPOS DE VARIÁVEIS OU ESTRUTURAS DE DADOS EM PYTHON

- ▶ As **sequências** podem ser de dois tipos:
 - ▶ **sequências imutáveis**: são objetos ordenados e finitos
 - ▶ **strings**: cadeias de caracteres
 - ▶ **tuples**: dois ou mais elementos de qualquer tipo dentro de parênteses e separados por vírgula
 - ▶ **sequências mutáveis**
 - ▶ **lists**: conhecidas em outras linguagens como vetores ou arranjos
 - ▶ **sets**: coleções não ordenadas e não indexadas escritas entre chaves

STRINGS

- ▶ Uma variável do tipo ***string*** armazena uma cadeia de caracteres
 - ▶ Em Python, uma string é uma variável composta
 - ▶ Veja um exemplo de uma *string*

```
sequencia = 'ACTTGGCAGTGACAAAGTGCATGGGGGACT'
```



MÉTODOS PARA MANIPULAÇÃO DE STRINGS

- ▶ As **strings**, como todos os outros tipos de dados, são **objetos** em Python
- ▶ Isso implica na existência de diversos **métodos** para se manipular strings embutidos no próprio objeto

SUBSTITUIÇÕES DE PARTES EM STRINGS

- ▶ **Substituição** de partes (trechos) em *strings* é realizada usando o método `replace()` conforme a sintaxe abaixo, recebendo como argumentos
 - ▶ A subsequência a ser encontrada e substituída: primeiro argumento
 - ▶ A subsequência que substituirá a subsequência buscada: segundo argumento

```
sequencia = sequencia.replace('AAA', 'CCC')
```

CONTAGEM DE OCORRÊNCIAS DE PARTES EM STRINGS

- ▶ Através do método `count()`, podemos realizar a **contagem** de ocorrências de um trecho ou subsequência em uma *string*:

```
sequencia.count( 'AAA' )
```

ENCONTRAR A POSIÇÃO DE UM PARTES EM STRINGS

- ▶ O método `find()` recebe como argumento uma sequência e retorna **em que posição se encontra essa subsquência**

```
sequencia.find('AAA')  
sequencia[sequencia.find('AAA')]
```

- ▶ Se ela não for encontrada, retorna -1

SEPARAÇÃO / QUEBRA DE STRINGS

- ▶ O método `split()` **separa / quebra** uma *string* por um certo trecho e retorna uma lista:

```
sequencia.split('A')
```

- ▶ Se o trecho for passado em branco, retorna uma lista com um elemento que é a sequência

```
sequencia.split()
```


CONCATENAÇÃO DE STRINGS

- ▶ O método `join()` é usado para **unir strings**:

```
'AAA'.join(sequencia)
```

MUDANÇA DE CASE EM STRINGS

- ▶ Há diversas funções que nos permitem mudar os caracteres para **maiúsculo** / **minúsculo**

```
sequencia.upper()  
sequencia.lower()  
sequencia.lower().capitalize() # Apenas a primeira letra em maiúsculo  
sequencia.title() # Cada primeira letra de palavra em maiúsculo  
sequencia.swapcase() # Inversão de maiúsculo para minúsculo e vice-versa
```

- ▶ Existem ainda as funções `isupper()`, `islower()`, `istitle()`, `isalnum()`, `isalpha()`, `isdigit()`, `isspace()` que retornam valores *booleanos* de acordo com os testes realizados nas *strings*

TRABALHANDO COM ESPAÇOS EM STRINGS

- ▶ É possível através de métodos **justificar** uma *string* em um determinado espaço:

```
'AAA'.ljust(15)  
'AAA'.rjust(15)  
'AAA'.center(15)
```

- ▶ Bem como **remover os espaços** no início e fim da *string*

```
sequencia.strip()  
sequencia.lstrip()  
sequencia.rstrip()
```

OBTENDO PARTE DE STRINGS

- ▶ Para obter subsequências de *strings*, não se usa nenhum método em particular
 - ▶ Basta trabalhar com os índices

```
sequencia[6:9]
```

```
sequencia[:9]
```

```
sequencia[6:]
```

COMPRIMENTO DE UMA STRING

- ▶ O método `len()` é um método nativo de Python e não um método de sequência
- ▶ Ele é usado para retornar o **comprimento** de uma *string*

```
len(sequencia)
```