FIAP GRADUAÇÃO

FIAP

TDS

Computacional Thinking using Python
Strings – Definição e Manipulação

Prof. Dr. Daniel Trevisan Bravo

STRINGS – CONCEITOS E MANIPULAÇÃO

STRINGS

- O tipo String é usado principalmente para gravar informações de texto, como o nome de uma pessoa ou uma marca de roupas, por exemplo. Nele temos uma sequência de caracteres que formam uma string.

- O Python permite formar strings com um par de aspas simples ou duplas.

```
1  nome_1 = "Ricardo Alves"
2  nome_2 = 'Joana Melo'
3
4  print(type(nome_1)) # type 'str'
5  print(type(nome_2)) # type 'str'
```

STRINGS – CONCEITOS BÁSICOS

- As strings são sequências de caracteres, de forma que podemos acessar um caractere em uma dada posição utilizando um índice.

```
1 nome = 'Daniel'
2 print(nome[0]) # D
```

- A variável nome com o conteúdo Daniel foi acessada no índice 0, que é por onde começa a cadeia de caracteres. O resultado será o caractere D.

- ***OBS***: Caso tente acessar um índice inexistente, o seguinte erro será exibido: IndexError: string index out of range. Isso acontece porque o índice que se tentou acessar está fora do range da cadeia de caracteres da variável. No exemplo dado, a string "Daniel" só nos permitiria acessar até o índice 5. Se tentarmos acessar nome[6], esse erro seria gerado.

STRINGS – CONCEITOS BÁSICOS

- Há também a possibilidade de "fatiar" uma variável do tipo String, retornando um "pedaço" dela.

```
1 | nome = "Daniel"
2 | print(nome[0:3]) # Dan
```

- O resultado do código acima seria a impressão da substring Dan. Nela definimos o valor 0 como o início da string que será fatiada até antes da posição que será o limite, que nesse caso é o índice 3.
- Também podemos usar índices negativos para as posições dos caracteres nas strings. Nesse caso, a ordem será inversa, começando do último até o primeiro.

```
1 | nome = "Daniel"
2 | print(nome[-2]) # e
```

STRINGS – CONCEITOS INICIAIS

- Um recurso importante quando estamos trabalhando com esse tipo é a função *len()*, que retorna o comprimento de uma string.

```
1  nome_1 = 'Rodrigo'
2  nome_2 = 'Ana'
3
4  print(len(nome_1)) # 7
5  print(len(nome_2)) # 3
```

- **OBS**: no código acima, vemos que nas linhas 1 e 2 criamos duas variáveis que recebem textos. Nas linhas 4 e 5 exibimos com a função len() o tamanho de cada uma delas. A linha 4 vai retornar o resultado 7, enquanto a linha 5 vai retornar o resultado 3.

STRINGS - IMUTABILIDADE

- Uma string no Python é uma sequência de caracteres imutável.

- Para ver como isso ocorre na prática, vamos fazer uso da função id(), que retorna à identidade de um objeto.

```
1 nome = 'Eduardo'
2 print(id(nome))
3 nome = 'Felipe'
4 print(id(nome))
```

- **OBS**: o código acima cria a variável nome na linha 1 e depois imprime a identidade dela na linha 2. Já na linha 3 damos um novo valor à variável e novamente imprimimos a sua identidade na linha 4. Os resultados que serão impressos não serão iguais, mostrando que as identidades da variável nome são distintas, mostrando um objeto diferente do que foi criado. Então pode-se concluir que a string não foi alterada, mas que foi criada uma nova.

CONCATENAÇÃO DE STRINGS

- Há casos em que é necessário juntar informações textuais e para esses denominamos concatenação, que é a junção do conteúdo de strings.

```
1  nome = 'Daniel'
2  sobrenome = 'Silva'
3
4  nome_completo = nome + ' ' + sobrenome
5
6  print(nome_completo) # Daniel Silva
```

- **OBS**: nas linhas 1 e 2 do código acima são criadas as variáveis nome e sobrenome do tipo String. O processo de concatenar variáveis ocorre na linha 4, na qual a variável nome_completo recebe o conteúdo das variáveis declaradas. Para fazer a concatenação entre strings no Python é necessário usar o sinal de adição +.

COMPARAÇÃO DE STRINGS

- No Python podemos comparar strings de duas formas distintas: com o operador == ou is.
- Com o operador == verificamos se o conteúdo de duas strings é igual.

```
1  nome_1 = 'Eduardo'
2  nome_2 = 'Eduardo'
3
4  if nome_1 == nome_2:
5      print('iguais')
6  else:
7      print('diferentes')
```

- Já com o operador is, o que será comparado é a referência do endereço na memória.

```
1  nome_1 = 'Eduardo'
2  nome_2 = 'Eduardo'
3
4  if nome_1 is nome_2:
5      print('iguais')
6  else:
7      print('diferentes')
```

STRINGS – PRINCIPAIS MÉTODOS

- Método *find()*: Com o método find() podemos procurar uma substring dentro de uma string e retornar a posição onde ela foi encontrada. No caso de a ocorrência não ser encontrada, o resultado será -1.

```
1  mensagem = 'string no Python'
2  print(mensagem.find('Python')) # 10
```

- Método *replace()*: O método replace() é utilizado para substituir ocorrências de substrings dentro de uma string.

```
frase = 'Aprender a programar em qualquer linguagem é um ato de praticar.... treinar....'
nova_frase = frase.replace('qualquer linguagem','Python')
print(nova_frase)
```

```
Aprender a programar em Python é um ato de praticar.... treinar....
```

STRINGS – PRINCIPAIS MÉTODOS

- Método *split()*: Com o método split() desmembramos uma string em múltiplas strings através de um separador passado no parâmetro, retornando todas numa lista.

```
frase = 'Programar em Python é muito legal'
nova_frase = frase.split(' ')
print(nova_frase)
```

```
['Programar', 'em', 'Python', 'é', 'muito', 'legal']
```

- Note que a lista retornada pode ser acessada pelos seus índices

```
frase = 'Programar em Python é muito legal'
nova_frase = frase.split(' ')
print(nova_frase[2])
```

```
Python
```

STRINGS – PRINCIPAIS MÉTODOS

- Método *upper()*: Com o método upper() retornamos uma cópia da string com todas as letras minúsculas convertidas em maiúsculas.

```
frase = 'Programar em Python é muito legal'
nova_frase = frase.upper()
print(nova_frase)
```

```
PROGRAMAR EM PYTHON É MUITO LEGAL
```

- Método *lower()*: Com o método lower(), retornamos uma cópia da string com todas as letras maiúsculas convertidas em minúsculas.

```
frase = 'Programar em Python é muito legal'
nova_frase = frase.lower()
print(nova_frase)
```

```
programar em python é muito legal
```

EXERCÍCIOS

- Escreva um programa em Python que, considerando uma string digitada pelo usuário, converta-a em letras minúsculas e, em seguida, exiba os caracteres na vertical (um debaixo do outro).
- Dada uma string digitada pelo usuário, crie um programa em Python que faça a contagem de vogais existentes nessa string.
- Um palíndromo é um tipo de palavra ou frase que tem a propriedade de ser lida tanto da direita para a esquerda quanto da esquerda para a direita. Como exemplo, temos a palavra “asa”. Baseado nesse conceito, escreva um programa em Python que, dada uma palavra, verifique se ele é um palíndromo ou não. DICA: utiliza a notação de slice.
- Faça um programa em Python que solicite a data de nascimento (dd/mm/aaaa) do usuário e imprima a data com o nome do mês por extenso.