

# Strings

Joyce Teixeira



# Strings

*teste = 'Sorria! Hoje é terça!'*

# Strings

- Em Python, strings possuem dois conceitos importantes:
  - **Índices**: posições dos caracteres na string
  - **Imutabilidade**: string não pode ser modificada

# Strings: Índices

-21	-20	-1	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
S	o	r	r	i	a	!		H	o	j	e		é		t	e	r	ç	a	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

- Valor numérico que indica posição do caractere na string
- Contagem da **esquerda para a direita**
- Numeração **começa do 0** (zero)
- Próximas posições são 1, 2, 3, ...
- Última posição é N-1
- Índice pode ter numeração negativa
  - *Contagem inversa: da direita para a esquerda*

# Strings: Índices e Fatiamento

-21	-20	-1	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
S	o	r	r	i	a	!		H	o	j	e		é		t	e	r	ç	a	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

## O que os comandos abaixo retornam?

1. `teste[0]` S
2. `teste[-1]` !
3. `teste[0:5]` Sorri
4. `teste[0:6]` Sorria
5. `teste[:6]` Sorria
6. `teste[6:]` ! Hoje é terça!
7. `teste[3:6]` ria
8. `teste[3:-1]` ria! Hoje é terça
9. `teste[-3:-1]` ça
10. `teste[:]` Sorria! Hoje é segunda!
11. `teste[15:21:2]` tra
12. `teste[:15:2]` Sri!Hj
13. `teste[15::2]` tra
14. `teste[::3]` Sr!o tç
15. `teste[100]` `IndexError: string index out of range`

# Strings: Índices e Fatiamento

-21	-20	-1	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
S	o	r	r	i	a	!		H	o	j	e		é		t	e	r	ç	a	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

- Como mostrar a frase invertida?
- `teste[::-1]`
  - `!açret é ejoH !airroS`

# Strings: Análise - Tamanho

- Tamanho da string: `len(string)`

```
teste = 'Sorria! Hoje é terça!'
print(len(teste))
```

21

# Strings: Análise - Encontrar

- Encontrar ocorrências em uma string:
  - `string.find('o que quer encontrar')`

```
teste = 'Sorria! Hoje é terça!'
print(teste.find('ria'))
```

3

```
teste = 'Sorria! Hoje é terça!'
print(teste.find('ontem'))
```

-1



# Strings: Análise - Encontrar

- Encontrar ocorrências em uma string:
  - 'O que quer encontrar' in string

```
teste = 'Sorria! Hoje é terça!'
print('ria' in teste)
```

True

```
teste = 'Sorria! Hoje é terça!'
print('ontem' in teste)
```

False

# Strings: Imutabilidade

- Não é possível alterar o conteúdo de uma string uma vez definida
- Deve-se criar uma nova string a partir de substrings desejadas
- Exemplo:

```
teste = 'Sorria! Hoje é terça!'
teste[8] = 'haha'
```

```
TypeError: 'str' object does not support item assignment
```

```
teste = 'Sorria! Hoje é terça!'
novo_teste = teste[:8] + 'Olha' + teste[14:]
print(novo_teste)
```

```
Sorria! Olha terça!
```

# Strings: Transformação - Replace

```
teste = 'Sorria! Hoje é terça!'
teste.replace('Sorria', 'Força')
print(teste)
```

Sorria! Hoje é terça!

**IMUTABILIDADE**

```
teste = 'Sorria! Hoje é terça!'
novo_teste = teste.replace('Sorria', 'Força')
print(novo_teste)
```

Força! Hoje é terça!

# Strings: Transformação - Alguns Métodos

- `frase.upper()` → passa toda a frase para letras maiúsculas.
- `frase.lower()` → passa toda a frase para letras minúsculas.
- `frase.capitalize()` → deixa somente a primeira letra da string em maiúsculo.
- `frase.title()` → deixa a primeira letra de todas as palavras em maiúsculo.
- `frase.strip()` → elimina todos os espaços no início e no fim.

# Strings: Transformação - Alguns Métodos

```
frase = 'Sorria! Hoje é terça!'

print(frase.upper())
print(frase.lower())
print(frase.capitalize())
print(frase.title())
```

```
SORRIA! HOJE É TERÇA!
sorria! hoje é terça!
Sorria! hoje é terça!
Sorria! Hoje É Terça!
```

# Strings: Transformação - Alguns Métodos

```
frase = '      Sorria! Hoje é terça!      '  
  
print(f'--- {frase.strip()} ---')  
print(f'--- {frase.rstrip()} ---')  
print(f'--- {frase.lstrip()} ---')
```

```
--- Sorria! Hoje é terça! ---  
---      Sorria! Hoje é terça! ---  
--- Sorria! Hoje é terça!      ---
```

# Strings: Divisão - Split()

```
frase = 'Sorria! Hoje é terça!'
frase2 = frase.split()
print(frase2)
```

0 1 2 3

['Sorria!', 'Hoje', 'é', 'terça!']

0 1 2 3 4 5 6 0 1 2 3 0 0 1 2 3 4 5

```
print(frase2[1][3])
```

e

# Strings: Junção - join()

```
frase = 'Sorria! Hoje é terça!'
frase2 = frase.split()
juncao1 = '-'.join(frase2)
juncao2 = ' '.join(frase2)
print(juncao1)
print(juncao2)
```

```
Sorria!-Hoje-é-terça!
Sorria! Hoje é terça!
```



# Strings: Junção - Concatenação

Diagram illustrating a list of strings in memory:

- Index 0: 'Sorria!' (characters: 0 1 2 3 4 5 6)
- Index 1: 'Hoje' (characters: 0 1 2 3)
- Index 2: 'é' (character: 0)
- Index 3: 'terça!' (characters: 0 1 2 3 4 5)

```
frase = 'Sorria! Hoje é terça!'
frase2 = frase.split()
frase3 = frase2[1] + frase2[2] + frase2[3]
print(frase3)
```

Hoje é terça!

# Strings: Formatação

- Textos são dados por strings
- Strings são imutáveis
- Porém, é possível pré-processar a string antes de imprimi-la na tela
- Isso é feito dentro do print formatado

# Strings: Formatação

```
nome = 'Carlos'
idade = 30
peso = 85.0
altura = 1.90
imc = peso/altura**2

print(f'{nome} tem {idade} anos, pesa {peso} quilos e tem {altura} de altura.')
print(f'{nome} tem o IMC de {imc}')
print(f'{nome} tem o IMC de {imc:.2f}')
print(f'{nome} tem o IMC de {imc:.3f}')
```

```
Carlos tem 30 anos, pesa 85.0 quilos e tem 1.9 de altura.
Carlos tem o IMC de 23.545706371191137
Carlos tem o IMC de 23.55
Carlos tem o IMC de 23.546
```

# Strings: Formatação

Existem duas informações importantes para uso do print formatado

- A variável está **entre chaves**
- Existe a variável e a formatação que a variável terá, separados por ':' (dois pontos)
  - {variavel:formato\_variavel}

# Formatação de Strings - Tipo

- O **[tipo]** é usado para informar como o dado deve ser apresentado na saída;
- Existem várias notações para inteiros e ponto flutuante;

# Formatação de Strings - Tipo - Inteiros

Tipo	Descrição
'b'	Base 2 (binário)
'c'	Caractere. Converte o inteiro correspondente na tabela ASCII antes de imprimir
'd'	Base 10 (decimal)
'o'	Base 8 (octal)
'x'	Base 16 (hexadecimal), usando letras minúsculas
'X'	Base 16 (hexadecimal), usando letras maiúsculas
'n'	Número. O mesmo que 'd', mas usa o idioma local do sistema operacional para notação do valor

# Formatação de Strings - Tipo - Inteiros

```
print(f'int: {15:d}; bin: {15:b}; oct: {15:o}; hex: {15:x}')
```

Saída:

```
int: 15; bin: 1111; oct: 17; hex: f
```

# Formatação de Strings - Tipo - Ponto Flutuante

Tipo	Descrição
'e'	Apresenta o resultado em notação científica usando a letra <b>e</b> para indicar expoente. Precisão 6 por padrão
'E'	Apresenta o resultado em notação científica usando a letra <b>E</b> para indicar expoente. Precisão 6 por padrão
'f'	Mostra o número como um número real em ponto fixo. Precisão 6 por padrão
'g'	Para precisão $\geq 1$ , arredonda o número para $p$ dígitos significativos e formata o número resultado para ponto fixo ou notação científica, conforme a magnitude
'n'	Número. O mesmo que <b>'g'</b> , mas usa o idioma local do sistema operacional para notação do valor
'%'	Multiplica o número por 100 e o mostra como tipo <b>'f'</b> , seguido do caractere <b>'%'</b>
Sem letra	Similar ao <b>'g'</b> , mas com precisão variável (de acordo com o dado) e pelo menos 1 casa decimal após a vírgula



# Formatação de Strings - Tipo - Ponto Flutuante

```
num = 12345.123456789
print(f'Valor = {num:e} ')
print(f'Valor = {num:E} ')
print(f'Valor = {num:g} ')
print(f'Valor = {num:n} ')
print(f'Valor = {num:f} ')
print(f'Valor = {num:.2f} ')
print(f'Valor = {num:.3f} ')
```

```
Valor = 1.234512e+04
Valor = 1.234512E+04
Valor = 12345.1
Valor = 12345.1
Valor = 12345.123457
Valor = 12345.12
Valor = 12345.123
```

# Formatação de Strings - Tipo - Ponto Flutuante

```
resposta = 2/4  
print(f'Respostas corretas: {resposta:%}')
```

```
print(f'Respostas corretas: {resposta:.2%}')
```

```
Respostas corretas: 50.000000%  
Respostas corretas: 50.00%
```

# Exercícios

1. Faça um programa que receba uma frase digitada pelo usuário. O programa deve imprimir letra por letra na tela.
2. Faça um programa que receba uma frase digitada pelo usuário. O programa deve imprimir a primeira e última palavra, concatenadas em uma única frase, com espaço entre elas.
3. Faça um programa que receba uma frase e uma palavra digitada pelo usuário. O programa deverá analisar a frase, procurando pela palavra digitada. Se existir, substituirá a palavra por 'domingo', caso contrário, manterá a mesma frase. O programa deverá ter como saída:
  - a. Houve modificação na frase! A nova frase é: <frase modificada>
  - b. Não houve modificação na frase! A frase permanece: <frase original>
4. Faça um programa que receba uma palavra digitada pelo usuário. O programa deve informar se a palavra é palíndroma ou não.