COMPUTAÇÃO 1 – AULA 3

Strings

Prof. Cesar Raitz

1. Introdução

- Toda comunicação no Terminal (PowerShell, IDLE Shell, bash, *etc.*) é feita por texto (com `print` e `input`)
- Em programação, **strings** são usadas para guardar texto
- Então o que você precisa aprender?
  - **indexar** e **fatiar** strings para extrair caracteres ou trechos
  - **manipular** strings para extrair as informações interessantes
  - **formatar** strings para imprimir um texto mais bonito

2. Definição

- *Strings são sequências de caracteres* (no sistema Unicode)
- Em Python, escrevemos uma string entre aspas (duplas ou simples):

```
1. s1 = "O sapo na lagoa "
2. s2 = 'não lava o pé'
3. print(type(s1))
```

- Você pode pensar em strings como caixinhas na memória do computador, o nome da variável seria o endereço da string:

s1

O		s	a	p	o		n	a		l	a	g	o	a	

s2

n	ã	o		l	a	v	a		o		p	é

- O *tipo de variável* para strings é **`str`** (também é a função que converte números em strings)

```
3. print(type(s1))
```

```
<class 'str'>
```

3. Indexação

3.1. Com índices positivos

- Podemos localizar os caracteres individuais pelo seu índice, começando do zero, essa operação se chama **indexação**.

exemplo

I	s	t	o		é		u	m		e	x	e	m	p	l	o
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

```
1. exemplo = "Isto é um exemplo"
2. a = exemplo[0]
3. b = exemplo[16]
5. print(a)
6. print(b)
```

```
I
o
```

- Você já pegou a **sintaxe** para indexação?

```
<string>[<índice do caractere>]
```

- Para indexar, não pode esquecer dos colchetes! [ ]
- O resultado de uma indexação é uma string o caractere escolhido:

```
1. exemplo = "Isto é um exemplo"
2.                          # Equivalente:
3. print(exemplo[4])        # print(" ")
4. print(exemplo[5])        # print("é")
5. print(exemplo[20])       # Erro!
```

```
é
Traceback (most recent call last):
  File "teste.py", line 4, in <module>
    print(exemplo[20])
IndexError: string index out of range
```

- O programa é interrompido na linha 4, por causa de um erro de indexação **IndexError**, uma tentativa de ler um caractere que não existe neste índice!

3.2. Com índices negativos

- Também podemos usar índices negativos:

exemplo

I	s	t	o		é		u	m		e	x	e	m	p	l	o
-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
1. exemplo = "Isto é um exemplo"
2.                          # Equivalente:
3. print(exemplo[-1])       # print("o")
4. print(exemplo[-12])      # print("é")
5. print(exemplo[-17])      # print("I")
```

Exercício 1. Mostrando caracteres.

Escreva um código para imprimir os caracteres r da *string*

"O rato roeu a roupa" com índices positivos e negativos.

3.3. Comprimento de uma string

- O **comprimento/tamanho** de uma string é o número de caracteres que ela possui
- A função **`len()`** retorna o número de caracteres numa string:

```
1. exemplo = "Isto é um exemplo"
2. tamanho = len(exemplo)
3. print(tamanho)
4. print(len("Xico"))
5. print(len("X"))
```

```
17
4
1
```

Exercício 2. Comprimento de uma frase qualquer.

Escreva um programa que leia uma frase digitada pelo usuário e imprima quantos caracteres ela tem. Imprima também o índice do caractere que está na metade da string (o índice deve ser um inteiro).

4. Fatiamento

4.1. Regras para fatiar

- Conseguimos selecionar um caractere de uma string, mas podemos selecionar um trecho?
- Claro! Se chama **fatiamento**, se liga na sintaxe:

```
<string>[i1 : i2 : p]
```

  - `i1` é o **primeiro índice** da fatia
  - `i2` é o **último índice** (excluído)
  - `p` é o **passo**

```
1. ditado = "Quem não arrisca"
2. print(ditado[5:8])      # não
3. print(ditado[-11:-8])   # não
4. print(ditado[9:16])     # arrisca
5. print(ditado[-5:16])    # arrisca
```

`ditado`

Q	u	e	m		n	ã	o		a	r	r	i	s	c	a
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- Se `p = 2`, então um a cada 2 caracteres é selecionado:

```
1. msg = "Mensagem secreta"
2. xxx = msg[0:16:2] + msg[1:16:2]
3. print(xxx)
```

```
Mnae ertesgmscea
```

msg

M	e	n	s	a	g	e	m		s	e	c	r	e	t	a
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- E olha que interessante, **um passo negativo** coloca os caracteres de trás pra frente:

```
1. msg = "Mensagem secreta"
2. print(msg[15:-17:-1])
```

```
aterces megasneM
```

4.2. Fatiando like a pro 😎

- Em fatiamento, podemos omitir:
  - `i1` equivale a `i1 = 0`
  - `i2` equivale a `i2 = len(string)`
  - `p` equivale a `p = 1`
- E para passos negativos, é o contrário
- Dizemos que eles possuem um **valor padrão**
- Então veja o exemplo (teste no seu PC):

```
1. msg = "Mensagem secreta"
2. print(msg[0:8], "equivale a", msg[:8])
3. print(msg[9:16], "equivale a", msg[9:])
4. print(msg[15:-17:-1], "equivale a", msg[::-1])
```

5. Manipulando strings

- Já vimos como juntar (concatenar) strings
- Veremos mais algumas funções que podem ser usadas com strings
- *Você deve conseguir seus lembrar os nomes para a prova!*

5.1. Passando para maiúscula ou minúscula

```
1. a = str.lower("NÃO GRITE COMIGO")
2. b = str.uppper("fale mais alto")
3. print(a)
4. print(b)
```

```
não grite comigo
FALE MAIS ALTO
```

5.2. Procurando uma string em outra

```
1. trava = "O rato roeu a roupa do Rei de Roma"
2. indice_r = str.find(trava, "r")
3. indice_x = str.find(trava, "x")
4. print("O primeiro r está em", indice_r)
5. print("O primeiro x está em", indice_x)
```

```
O primeiro r está em 2
O primeiro x está em -1
```

- Também tem a função **`str.rfind()`** que procura da direita para a esquerda:

```
1. trava = "O rato roeu a roupa do Rei de Roma"
2. indice_R = str.rfind(trava, "R")
3. indice_r = str.rfind(trava, "x")
4. print("O último R está em", indice_R)
5. print("O último r está em", indice_r)
```

```
O último R está em 30
O último r está em 14
```

5.3. Contando ocorrências de uma string

```
1. s = "Quem tem borogodó tem, quem não tem, não tem"
2. print( str.count(s, "quem") )
3. print( str.count(s, "borogodó") )
4. print( str.count(s, "não") )
5. print( str.count(s, "tem") )
```

```
0
1
2
4
```

5.4. Substituindo strings

```
1. s = "Quem parte e reparte, fica com a maior parte"
2. print( str.replace(s, "parte", "prioriza") )
3. print( str.replace(s, "parte", "prioriza", 2) )
```

```
Quem prioriza e reprioriza, fica com a maior prioriza
Quem prioriza e reprioriza, fica com a maior parte
```

6. Formatando strings

- Você passará boa parte do tempo de Comp.1 imprimindo coisas na tela
- Então é melhor aprender formas mais eficientes de fazer isso!
- Já vimos duas formas:

```
1. nome = "Joãozinho"
2. dinheiro = 10
3. preco = 3.2
4. num_bombons = dinheiro // preco
5. troco = dinheiro % preco
6. print("Joãozinho comprou", num_bombons, "bombons e
   devolveu R$", troco)
7. print("Joãozinho comprou " + str(num_bombons) + "
   bombons e devolveu R$ " + str(troco))
```

- Você já percebeu que cada vírgula dentro de print adiciona um espaço no texto impresso?
- Agora vejamos como fazer com **f-strings** ou *Template Strings*:
  - coloque **f** antes da primeira aspa
  - coloque suas variáveis entre chaves { }

```
8. print(f"Joãozinho comprou {num_bombons} bombons e devolveu R$ {troco}")
```

- O resultado é que o Python substitui as variáveis com chaves por seus valores

```
Joãozinho comprou 3.0 bombons e devolveu R$ 0.3999999999999947
Joãozinho comprou 3.0 bombons e devolveu R$ 0.3999999999999947
```

- Meio ruim né? 😕 A quantidade de bombons é um número inteiro! Vamos converter para **`int`** dentro da *f-string*, substitua `{num_bombons}` por `{int(num_bombons)}`
- Queremos apenas dois dígitos depois da vírgula no troco, então depois do nome da variável, e antes de `}`, colocamos `:` seguidos de `.2f`, que significa "imprima este número **float** com 2 casas depois da vírgula"

```
8. print(f"Joãozinho comprou {int(num_bombons)} bombons e devolveu R$ {troco:.2f}")
```

- Agora sim:

```
Joãozinho comprou 3 bombons e devolveu R$ 0.40
```

7. Mais exercícios

Exercício 3. Shipando um casal.

Faça um programa que pergunte o nome de duas pessoas e junte a primeira metade do nome de uma com a segunda metade do nome da outra. Imprima o *nome shippado* com uma mensagem, por exemplo:

```
Nome da primeira pessoa? Raphinha <ENTER>
Nome da segunda pessoa? Lidiane <ENTER>
Oi casal Raphiane!
```

Exercício 4. Prevendo a saída no Terminal.

Para o código a seguir, o que será mostrado no Terminal?

```
1. print( "Beetlejuice, "*3 )
2. print( "Wick" in "John Wick" )
3. texto = "Um tigre, dois tigres"
4. print( texto.replace("tigre", "trigo") )
5. print( texto.count("tigre") )
6. print( texto.find("dois") )
7. print( "Um dia frio"[-4:] )
```

Exercício 5. Contando frases.

Escreva um programa para contar frases numa string. Utilize a string abaixo para testar:

```
redacao = """Numa floresta mágica, o sábio coelho
perguntou aos pássaros: "Quem sabe o segredo da
árvore dos desejos?" Os pássaros olharam uns para os
outros, intrigados... Até que o velho coruja, com um
olhar misterioso, exclamou: "Ah, os desejos se
realizam quando o coração é puro e a amizade
verdadeira floresce!" E assim, todos os animais
aprenderam a importância dos sentimentos sinceros e
da magia que reside no respeito mútuo.
"""
```

Três aspas foram usadas para delimitar uma string de várias linhas.