Programação de Computadores

- Processamento de textos (Strings)

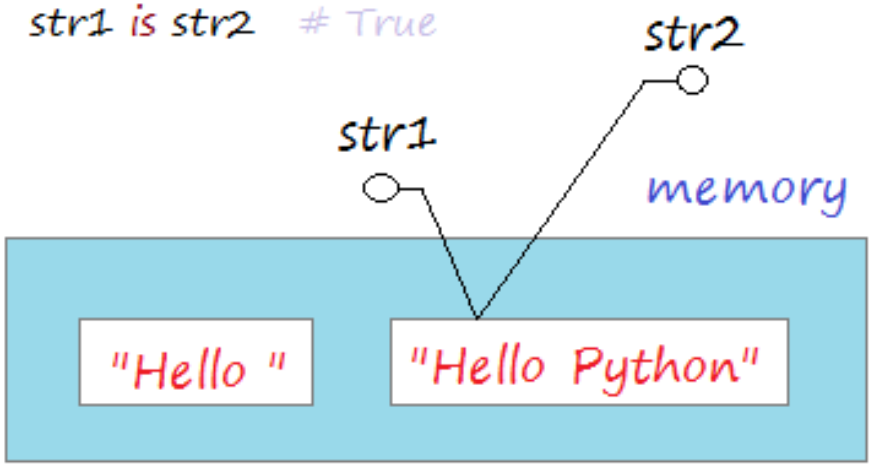
Professor Marco Antonio

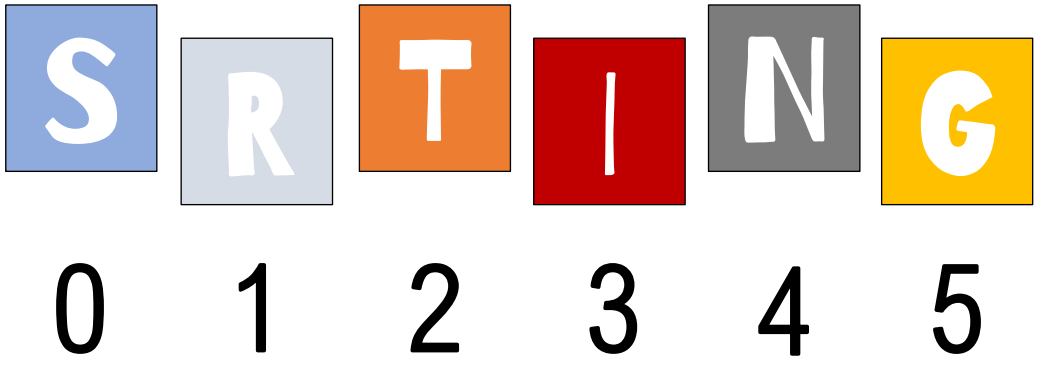
1º Semestre 2023

Conceitos abordados nesta aula

- A proposta desta aula é apresentar o processamento de textos com variáveis do tipo String.

```
str1 = "Hello Python"
str2 = "Hello Python"

str1 == str2  # True
str1 is str2  # True
```





O que são Strings?

Definição: Uma String é um tipo de dados que armazena uma cadeia de caracteres como nomes, textos e símbolos.

No **Python**, assim como em diversas outras linguagens de programação, as Strings podem ser expressas com aspas simples ('...'), duplas ("..."), ou até mesmo ("""...""") para múltiplas linhas.

Por exemplo:

exe.py

```
1  print('String é um elemento importante da programação')
2  print("Também podemos representá-las com aspas duplas")
3  print("""Podemos, até mesmo, representá-las com três aspas duplas, para textos grandes""")
```

Características das Strings em Python

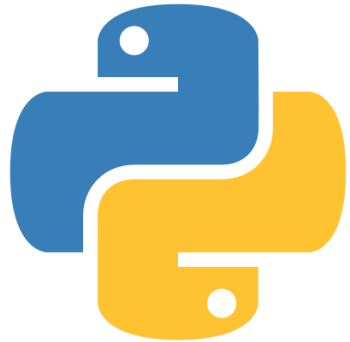
- As Strings em Python são **imutáveis**, ou seja, depois de criadas, não podem ser alteradas.
- As Strings em Python são **indexadas**, o que significa que cada caractere tem um índice, começando do 0 para o primeiro caractere.
- As Strings em Python podem ser **fatiadas (sliced)**, o que significa que é possível obter uma sub-sequência de caracteres de uma String maior.
- As Strings em Python suportam **operações de concatenação** (junção de Strings), **formatação** de Strings e muitos outros **métodos** úteis para **manipulação** de Strings.

Uma string é uma sequência

Como String é uma sequência de caracteres, podemos acessar um caractere de cada vez com o operador de colchete:

```
fruta = 'banana'
letra = fruta[1]
```

- A segunda instrução seleciona o caractere número 1 de fruta e o atribui à variável letra.
- A expressão entre colchetes chama-se índice. O índice aponta qual caractere da sequência você quer (daí o nome).



Uma string é uma sequência

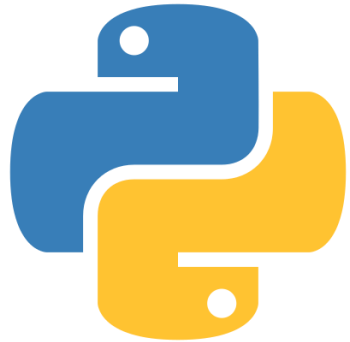
Como String é uma sequência de caracteres, podemos acessar um caractere de cada vez com o operador de colchete:

```
fruta = 'banana'
letra = fruta[1]
```

- Porém, veja que quando mandamos imprimir o conteúdo de letra, teremos uma surpresa:

```
fruta = 'banana'
letra = fruta[1]
letra
```

```
'a'
```



Uma string é uma sequência

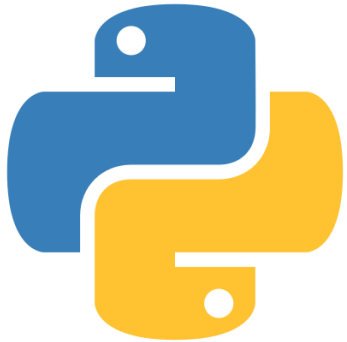
- Porém, veja que quando mandamos imprimir o conteúdo de letra, teremos uma surpresa:

```
fruta = 'banana'
letra = fruta[1]
letra
```

```
'a'
```

- O índice é uma referência do começo da String, e a referência da primeira letra é sempre zero.

Caractere	b	a	n	a	n	a
índice	0	1	2	3	4	5

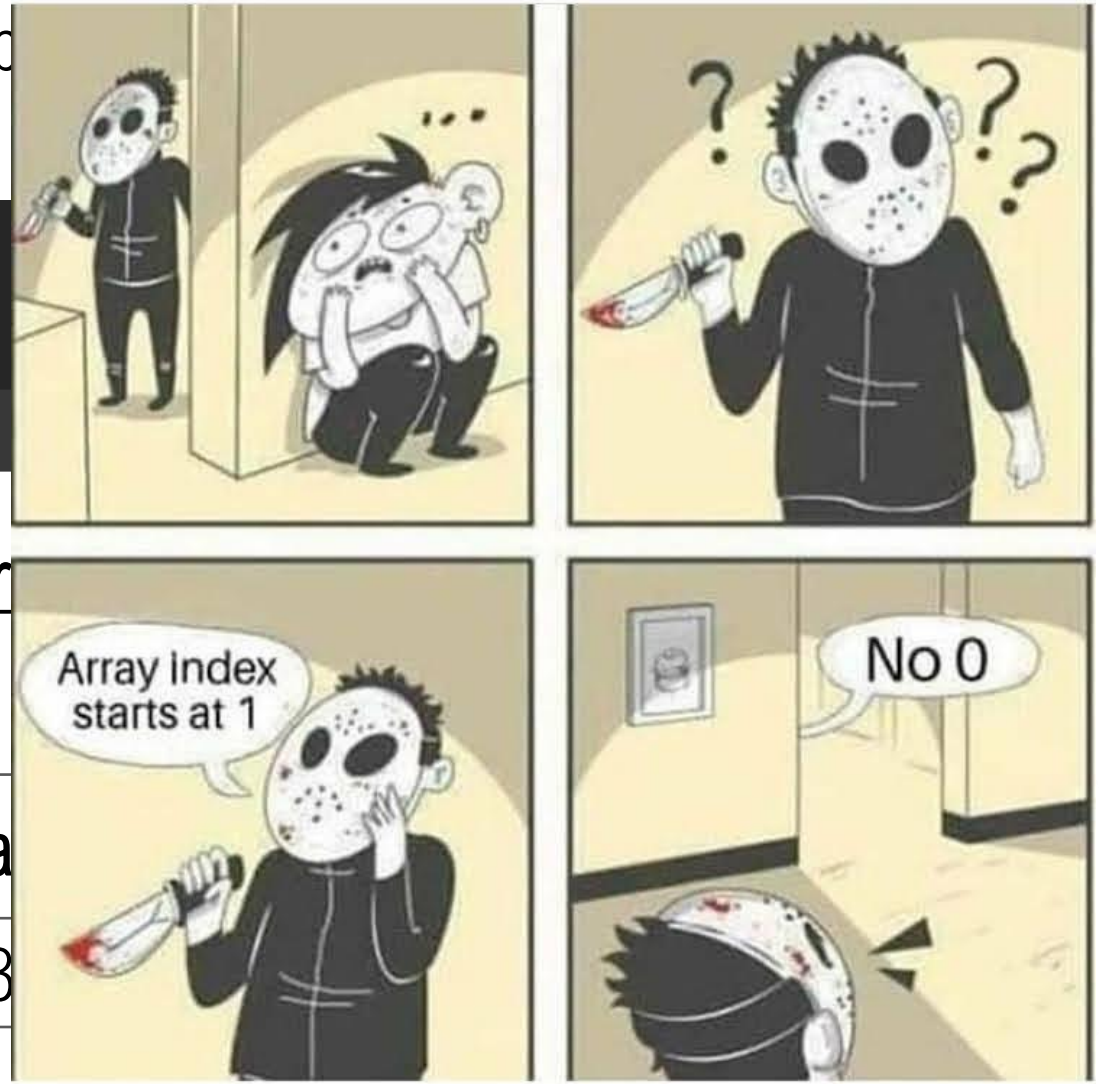


Uma string é uma sequência

- Porém, veja que quando mandamos imp
  uma surpresa:

```
fruta = 'banana'
letra = fruta[1]
letra
```

```
'a'
```



- O índice é uma referência do começo da Str
  sempre zero.

Caractere	b	a	n	a
índice	0	1	2	3

Uma string é uma sequência

Toda String possui um tamanho e, da mesma forma como podemos acessar caractere a caractere, também podemos obter o tamanho utilizando-se a função len().

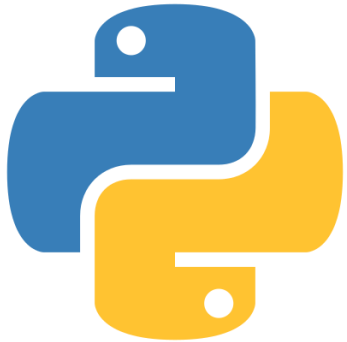
```
fruta = 'banana'
len(fruta)
```

```
6
```

- A função len() retorna o número de caracteres da string.

Caractere	b	a	n	a	n	a
índice	0	1	2	3	4	5

n = 6



Uma string é uma sequência

Exemplo 01: Faça um programa que exiba a quantidade de dígitos de um determinado número inteiro digitado pelo usuário.

main.py

```
1  num = input("Digite um número inteiro: ")
2  qt = len(num)
3  print(qt)
```

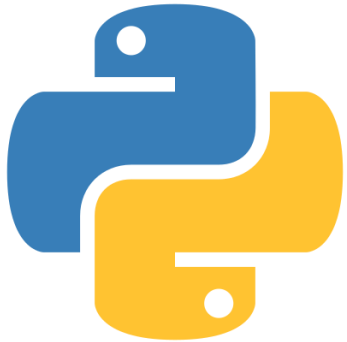
ou

int converte o valor digitado em um número inteiro

main.py

```
1  num = int(input("Digite um número inteiro: "))
2  qt = len(str(num))
3  print(qt)
```

str converte o número em string

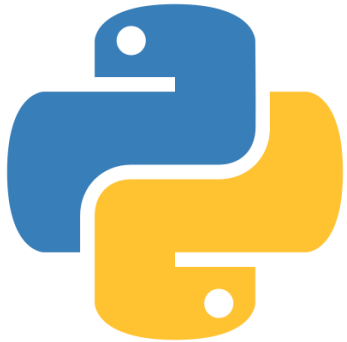


Strings como listas

Exemplo 02: Crie um programa que leia uma String e imprima a inversa dela.

main.py

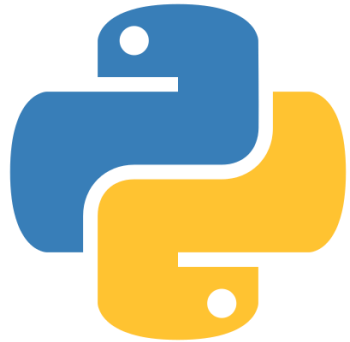
```
1  string = input("Digite um texto: ")
2  inversa = " "
3  for x in string:
4      inversa = x + inversa
5  print(inversa)
```



Uma string é uma sequência

Exemplo 03: Faça um programa em Python que solicite um número decimal e uma base, faça a conversão e exiba o número digitado na base escolhida.

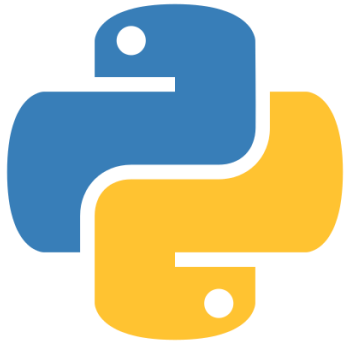
```
decimal = int(input('Digite um número decimal: '))
print('[2] binário\n[8] Octal\n[16] Hexadecimal')
base = int(input('Escolha uma base para conversão: '))

num_conv = ''
digitos = '0123456789ABCDEF'
while decimal > 0:
    num_conv = str(digitos[decimal % base]) + num_conv
    decimal = decimal // base

print(f'O número convertido para a base {base} é igual a {num_conv}')
```



Operações com strings

As variáveis do tipo String suportam várias operações, como veremos a seguir:

- **Concatenação:** consiste em juntar duas ou mais Strings em uma nova String maior.
- **Composição:** é muito utilizada em mensagens exibidas na tela e consiste em utilizar Strings com modelos onde podemos inserir outras Strings.
- **Fatiamento:** é uma técnica utilizada para obter somente uma parte da String, ou seja, uma fatia.



Operações com strings

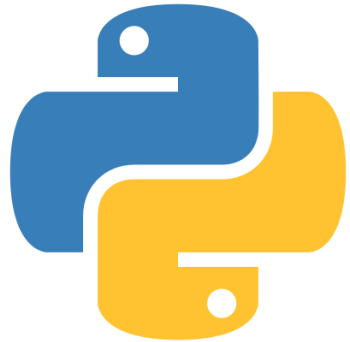
Concatenação: consiste em juntar duas ou mais Strings em uma nova String maior.

Para concatenar duas Strings, utilizaremos o operador de **adição(+)**. Veja o exemplo:

ABC + D = ABCD

```
s = "ABC"
print(s + "D")
```

```
ABCD
```



Operações com strings

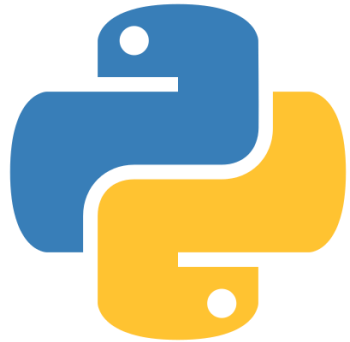
Composição: é muito utilizada em mensagens exibidas na tela e consiste em utilizar strings com modelos para inserir outras Strings.

Exemplo: Queremos imprimir "João tem x anos de idade", onde x é uma variável numérica inteira.

```
x = 18
print("João tem", x,"anos de idade.")
```

```
João tem 18 anos de idade.
```

Podemos fazer isso de forma simples, utilizando um composição de Strings.



Operações com strings

Composição: é muito utilizada em mensagens exibidas na tela e consiste em utilizar strings com modelos para inserir outras Strings.

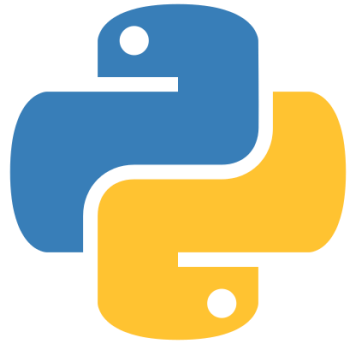
Exemplo: Queremos imprimir "João tem x anos de idade", onde x é uma variável numérica inteira.

Podemos fazer isso de forma simples, utilizando um composição de Strings.

Observe o símbolo %d

```
x = 18
print("João tem %d anos de idade." %x)
```

```
João tem 18 anos de idade.
```



Operações com strings

Composição: é muito utilizada em mensagens exibidas na tela e consiste em utilizar Strings com modelos para inserir outras Strings.

- O símbolo % é utilizado para indicar a composição da String anterior com o conteúdo da variável x.
- O %d é chamado de marcador de posição e indica onde será exibido o valor de x.
- O Python suporta diversos marcadores, entre eles:

Marcador	Tipo
%d	Numérico inteiro
%s	Strings
%f	Numéricos decimais

Operações com strings

Composição: é muito utilizada em mensagens exibidas na tela e consiste em utilizar strings com modelos para inserir outras Strings.

- Podemos formatar a exibição utilizando a composição.

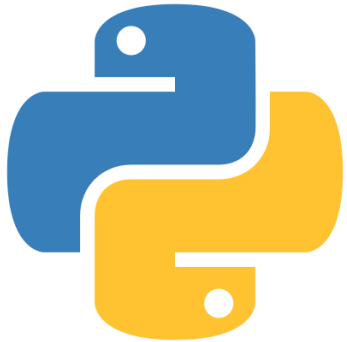
```
idade = 18
print("%03d" %idade)
```

```
018
```

Representamos o número com três casas, completando com zeros à esquerda

```
idade = 8
print("%03d" %idade)
```

```
008
```



Operações com strings

Composição: é muito utilizada em mensagens exibidas na tela e consiste em utilizar strings com modelos para inserir outras Strings.

- Podemos formatar a exibição utilizando a composição.

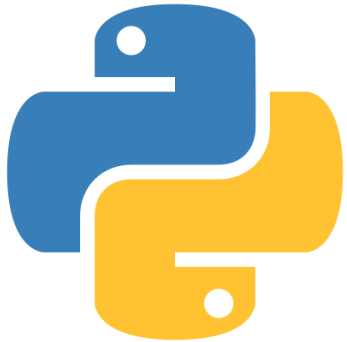
```
a = 18
print("%f" %a)
```

```
18.000000
```

Representamos o número com dois dígitos após a vírgula

```
a = 18
print("%.2f" %a)
```

```
18.00
```



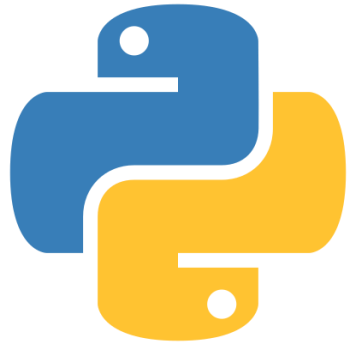
Operações com strings

Composição: é muito utilizada em mensagens exibidas na tela e consiste em utilizar strings com modelos para inserir outras Strings.

- Utilizando FStrings.

```
Nome = "Marco"
idade = 18
print(f"Meu nome é {nome} e eu tenho {idade} de idade")
```

```
Meu nome é Marco e eu tenho 18 anos de idade
```



- Usando format

```
Nome = "Marco"
idade = 18
print("Meu nome é {} e eu tenho {} de idade".format(nome, idade)
```

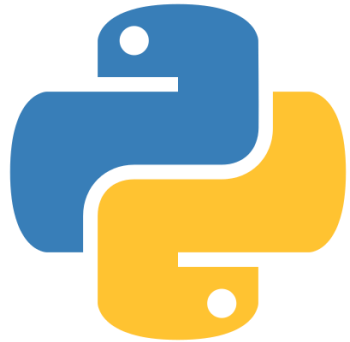
```
Meu nome é Marco e eu tenho 18 anos de idade
```

Operações com strings

Exemplo 04: Faça um programa que solicite ao usuário o valor de um produto e exiba o valor acrescido de 5%. A saída deverá exibir o valor no formato: **R$ xx.xx**

main.py

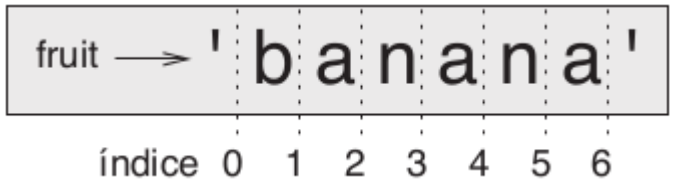
```
1  valor = float(input('Digite o valor de um produto: R$ '))
2  valor = valor * 1.05
3  print('O valor com o acréscimo será: R$ %.2f'%valor)
```



Operações com strings

Fatiamento: é uma técnica utilizada para obter somente uma parte da String, ou seja, uma fatia.

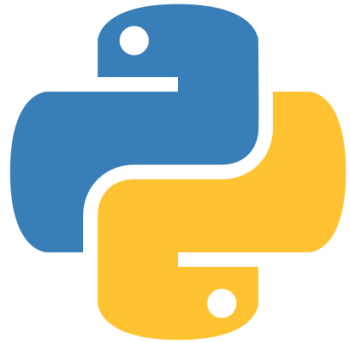
- A função slice (fatiar) devolve a string entre duas posições dadas.
- Pode-se fatiar (slice) strings usando [**inicio:fim**].




```
fruit = "banana"
print(fruit[0:2])
```

```
ba
```

Fatia da posição 0 até a posição 2



Operações com strings

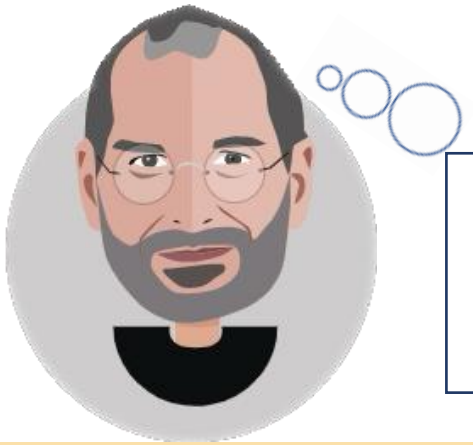
Fatiamento: é uma técnica utilizada para obter somente uma parte da String, ou seja, uma fatia.

- A função slice (fatiar) devolve a string entre duas posições dadas.
- Pode-se fatiar (slice) strings usando [**inicio:fim**].

```
fruit = "banana"
print(fruit[0:2])
```

```
ba
```

Fatia da posição 0 até a posição 2



Observe que o fatiamento não inclui a última posição, ou seja, é um intervalo aberto entre 0 e 2.

Operações com strings

Fatiamento: é uma técnica utilizada para obter somente uma parte da String, ou seja, uma fatia.

- Se você omitir o primeiro índice (antes dos dois pontos), a fatia começa no início da string. Se omitir o segundo índice, a fatia vai ao fim da string:

```
fruit = "banana"
print(fruit[:3])
```

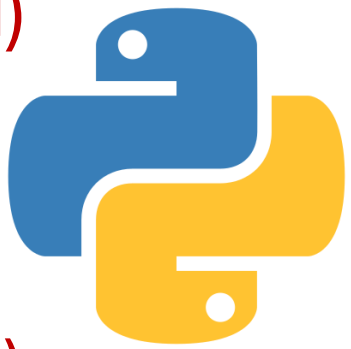
```
ban
```

Fatia do início até a posição 3 (exclui a posição final)

```
fruit = "banana"
print(fruit[3:])
```

```
ana
```

Fatia da posição 3 até o final (inclui a posição inicial)



Operações com strings

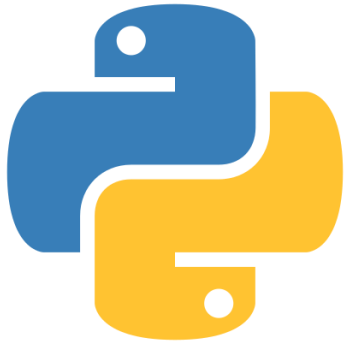
Fatiamento: é uma técnica utilizada para obter somente uma parte da String, ou seja, uma fatia.

- Podemos utilizar números negativos para posições a partir da direita

```
a = "ABCDEFGHI"
print(a[0:-4])
```

```
ABCDE
```

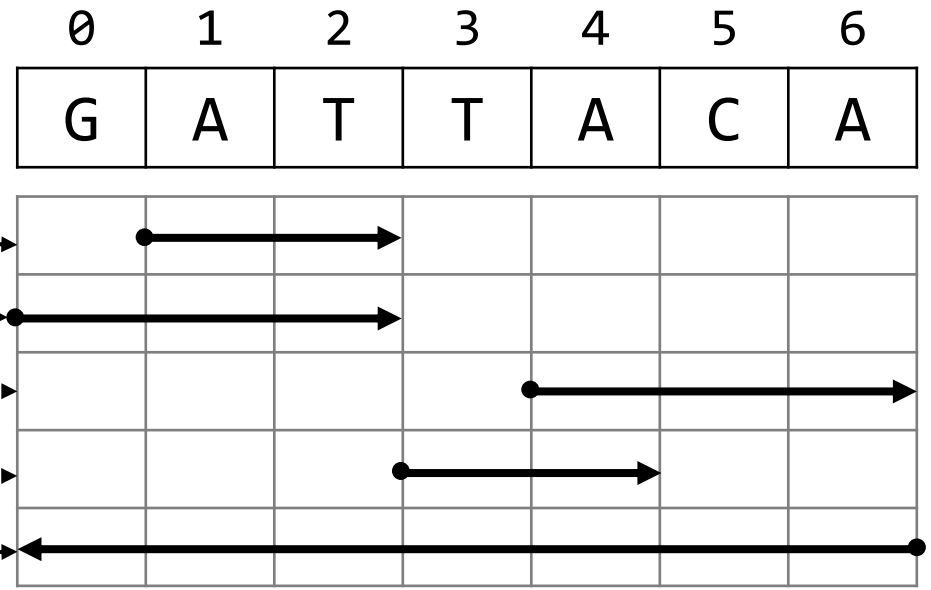
Fatia do final (direita) para o início da string (esquerda), excluindo as quatro últimas posições (-4)



Operações com strings

Fatiamento: é uma técnica utilizada para obter somente uma parte da String, ou seja, uma fatia.

```
movie = "GATTACA"
print(movie[1:3])
```
1
```
'AT'
```
```
print(movie[:3])
```
2
```
'GAT'
```
```
print(movie[4:])
```
3
```
'ACA'
```
```
print(movie[3:5])
```
4
```
'TA'
```
```
print(movie[::-1])
```
5
```
'ACATTAG'
```

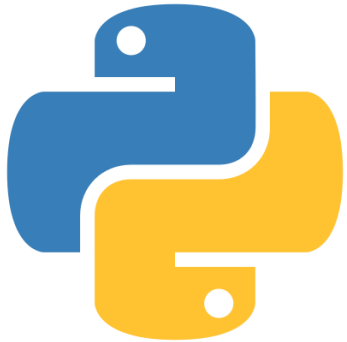


Operações com strings

Exemplo 05: Faça um programa em Python que leia um número inteiro digitado pelo usuário e exiba seu inverso.

main.py

```
1  num = input("Digite um número inteiro: ")
2  num = num[::-1]
3  print(num)
```



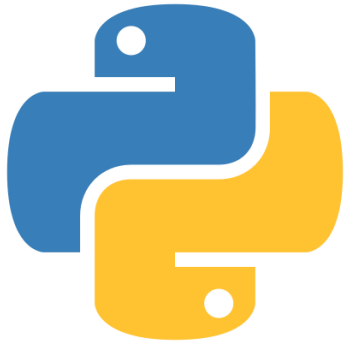
Strings são imutáveis

É tentador usar o operador [] no lado esquerdo de uma atribuição, com a intenção de alterar um caractere em uma String. Por exemplo:

```
saudacao = "Olá mundo"
saudacao[0] = "J"
```

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

As strings são imutáveis, o que significa que você não pode alterar uma string existente.

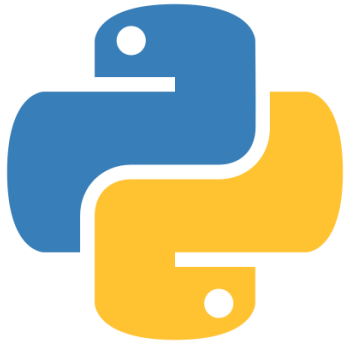


Strings: Operações, Funções e Métodos

- O método **strip** retorna uma String sem os espaços em branco e mudança de linhas no início e no final de uma String.

```
b = "\n Fizeram a atividade? "
b.strip()
```

```
'Fizeram a atividade?'
```



Strings: Operações, Funções e Métodos

- O operador in verifica se uma substring é parte de uma outra String.

```
a = "Fizeram a atividade?"
"atividade" in a
```

```
True
```

```
a = "Fizeram a atividade?"
“abril" in a
```

```
False
```

Strings: Operações, Funções e Métodos

- O método find retorna onde a substring começa na String.

```
a = "Fizeram a atividade?"
a.find("atividade")
```

```
10
```

```
a = "Fizeram a atividade?"
a.find("abril")
```

```
-1
```

retorna -1 quando a substring não ocorre na string.

Exemplos

Exemplo 6 - Faça um programa em Python que solicite um número e a correspondente base, faça a conversão e exiba o número digitado na base decimal.

```
num = input('Digite um número: ').upper()
base = int(input('Digite a base [2] [8] [16]'))

n = len(num)-1
decimal = 0
digitos = '0123456789ABCDEF'

for d in num:
    decimal += digitos.find(d)*base**n
    n = n-1

print(f'O número na base {base} é igual ao decimal {decimal}')
```

Strings: Operações, Funções e Métodos

- O método **lower** converte todos os caracteres da String para caixa baixa
- O método **upper** converte todos os caracteres da String para caixa alta.

```
a = "Atividade"
a1 = a.upper()
print(a1)
```

```
ATIVIDADE
```

```
a = "Atividade"
a2 = a.lower()
print(a2)
```

```
atividade
```

Strings: Operações, Funções e Métodos

- O método **replace** serve para trocar todas as ocorrências de uma substring por outra em uma String.

```
a = "Fizeram a atividade?"
a.replace("atividade", "avaliação")
```

```
'Fizeram a avaliação?'
```

```
a = "Fizeram a atividade?"
a.replace("atividade", " ")
```

```
'Fizeram a ?'
```

Mas as Strings não são imutáveis???

Strings: Operações, Funções e Métodos

Mas as Strings não são imutáveis???

Sim, experimente imprimir a variável a após o replace:

```
a = "Fizeram a atividade?"
a.replace("atividade", "avaliação")
print(a)
```

```
'Fizeram a atividade?'
```

O valor de a não foi alterado por replace!!!

Strings: Operações, Funções e Métodos

- O método **split(sep)** separa uma String usando sep como separador. Retorna uma lista das substrings.

```
num = "1; 2 ; 3"
num.split(";")
```

```
['1', ' 2 ', ' 3']
```

```
a = "Fizeram a atividade?"
a.split()
```

```
['Fizeram', 'a', 'atividade?']
```

OBS: Podem haver substrings vazias no retorno de split().

Strings: Operações, Funções e Métodos

- O método **count()** retorna quantas vezes o elemento aparece na string.

```
frase = "Macaco come banana"
frase.count("a")
```

```
5
```

Alguns métodos importantes

Método	Parâmetros	Descrição
strip	nenhum	Retorna uma string removendo caracteres em branco do início e do fim. Ex: a.strip()
find	substring	Retorna o índice onde a substring começa na string. Ex: a.find("texto")
split	nenhum	Separa uma string usando sep como separador e retorna uma lista das substrings. Ex: a.strip()
replace	substring1, substring2	Substitui todas as ocorrências de uma substring por outra. Ex: a.replace("prova","teste")
join	substring	Retorna uma string com a concatenação dos elementos da sequência/lista. Ex: "".join(a)
count	substring	Retorna o número de ocorrências de uma substring. Ex: a.count("as")
upper	nenhum	Retorna uma string toda em maiúsculas. Ex: a.upper()
lower	nenhum	Retorna uma string toda em minúsculas. Ex: a.lower()

Exemplos

Exemplo 7- Faça um programa que conta o número de palavras em um texto.

main.py

```
1  texto = input("Digite um texto: ")
2  pontuacao = [".", ",", ":", ";", "!", "?"]
3
4  # remove os sinais de pontuação
5  for p in pontuacao:
6      texto = texto.replace(p," ")
7
8  # split devolve lista com palavras como itens
9  numero_palavras = len(texto.split())
10 print("Número de palavras:", numero_palavras)
```

Formatação de strings

Exemplo 8 - Faça um programa que calcule o lucro de uma empresa, a partir da entrada do faturamento e do custo de produção:

lucro = faturamento – custo

O programa deverá exibir o lucro no formato: R$ x.xxx,xx

```
faturamento = float(input('Digite o faturamento: R$ '))
custo = float(input('Digite o custo: R$ '))
lucro = faturamento - custo
lucro = f'R$ {lucro:_.2f}'
lucro = lucro.replace('.', ',').replace('_', '.')
print(f'O lucro foi de {lucro}')
```

Alguma dúvida????

Atividade de bonificação

- Esta atividade é opcional e deverá ser postada no link disponibilizado na Aula 09 - Strings.
- Os códigos Python deverão ser feitos na ferramenta de sua preferência e deve ser salvo um arquivo para cada exercício, com a extensão .py.
- Por favor não envie link do Repl.it e tampouco compacte a pasta.
- Esta atividade tem valor de 0 a 0,5 e será somada à nota da avaliação parcial, desde que não ultrapasse o valor total de 3,0 pontos

Exercícios

1- Elabore um programa em Python que solicite o e-mail do usuário e imprima na tela somente o domínio. **Exemplo:**

- Entre com seu e-mail: teste@uol.com.br
- O domínio do seu e-mail é: http://uol.com.br

2- Faça um programa em Python que solicite um número inteiro de três algarismos e imprima a soma desse número com seu inverso. **Exemplo:**

- Digite um número inteiro com três algarismos: 123
- O inverso do número é: 321
- A soma é: 123 + 321 = 444

Exercícios

3- Elabore um programa em Python que leia uma String, converta para caixa alta e imprima quantas vezes cada caractere aparece nessa String. **Exemplo:**

Entrada:

- Entre com uma String: TTAAC

Saída:

- O caractere T aparece 2 vezes
- O caractere A aparece 2 vezes
- O caractere C aparece 1 vez

Exercícios

4- Um palíndromo é uma palavra ou frase que tenha a propriedade de poder ser lida tanto da direita para a esquerda como da esquerda para a direita.

Exemplo: as Strings "aaaaa", "1221", "bbaabb" são palíndromos, entretanto a String "chef" não é um palíndromo.

Crie um programa em Python que solicite uma palavra e exiba na tela se é um palíndromo ou não.

Exercícios

5- Fizz buzz é um jogo de palavras desenvolvido para ensinar divisão para crianças em idade escolar. Faça um programa em Python que implemente o FizzBuzz para os inteiros de 1 a 50. Pra tanto, observe as seguintes regras:

- Se o número é divisível por 3 e 5, exiba na tela a palavra "**FizzBuzz**".
- Se o número é divisível por 3, exiba na tela a palavra "**Fizz**".
- Se o número é divisível por 5, exiba na tela a palavra "**Buzz**".
- Se o número não é divisível nem por 3 nem por 5, apenas exiba na tela a o número.

Créditos

Esta aula foi elaborada com base no material produzido e cedido gentilmente pelos **Professores Alcides, Lédon, Amilton e Cristiane**.

marco.sanches@cruzeirodosul.edu.br