Orientador

Prof. Golbery Aguiar

Coorientadores

Prof. Danyllo Wagner

Prof.ª Ianna Sodre

Pesquisadores

Gabriel William

Jhonnata Virginio

João Gabriel

Luiz Eduardo

LAPLIN

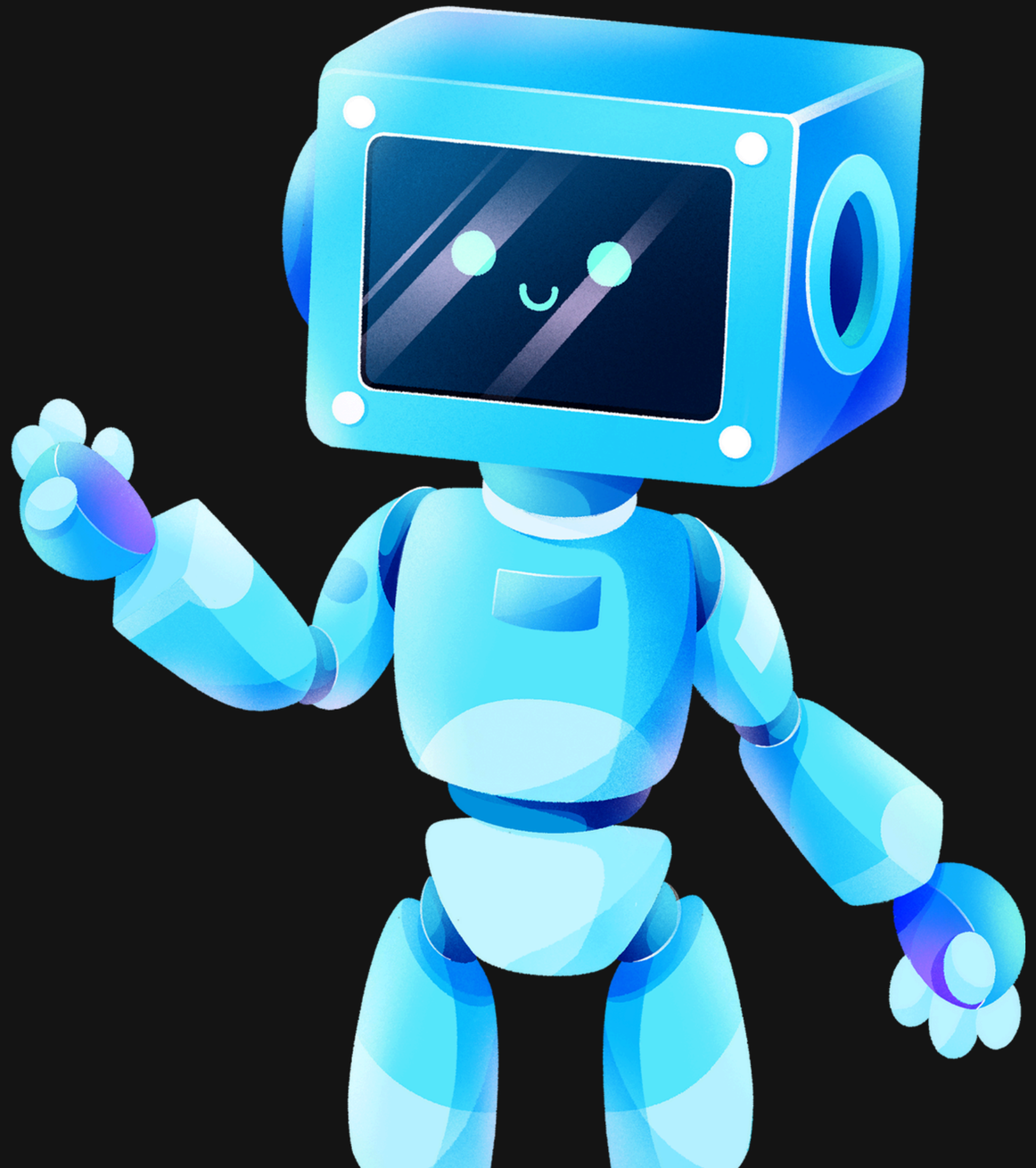
LABORATÓRIO DE ANÁLISE E PROCESSAMENTO DE LINGUAGEM NATURAL

INSTITUTO FEDERAL

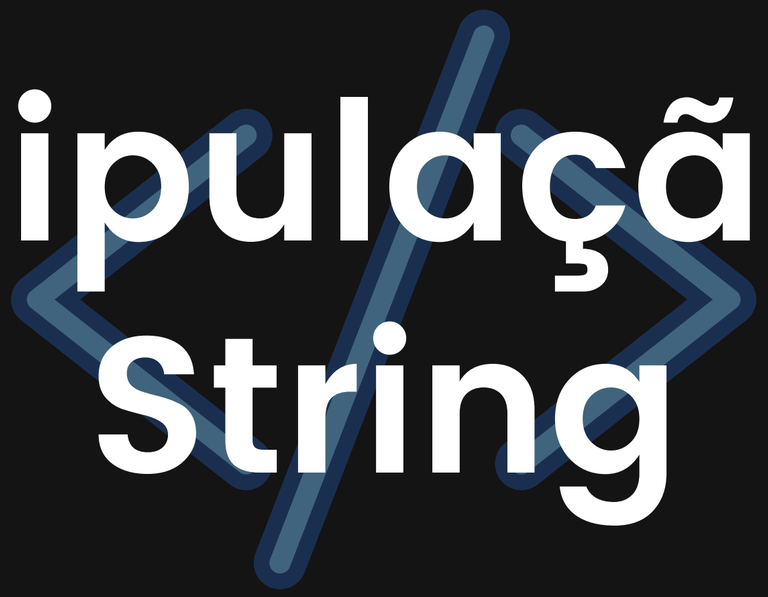
Paraíba

Campus Campina Grande

O que vamos aprender hoje?



Manipulação de String



Acesso e Index

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

Strings tem um funcionamento muito similar à lista, conteúdo já antes estudado, e podem ser indexadas da mesma maneira.

```
nome = 'Jhonnata'
print(nome[0]) # J
```

Atenção: Acessar um index fora do alcance da string gera um erro, assim como na lista.

Acesso e Index

- **Assim como nas listas, é possível utilizar indexação com indexes negativos, começando do fim até o início.**

```
nome = "Jhonnata"
print(nome[-2]) # t
```

Fatiamento

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

```
nome = "Jhonnata"
print(nome[0:3]) # Jho
```

Imutabilidade

- **Uma string no Python é uma sequência de caracteres imutável e por causa disto, não é possível alterar o valor de uma determinada posição de uma string.**

```
nome = 'Eduardo'
nome[6] = 'a'
```

Estre trecho de código **gera um erro**, pois é impossível alterar um caractere de uma string diretamente.

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.**

```
nome = "Jhonnata"
sobrenome = "Virgínio"
nome_completo = nome + " " + sobrenome
print(nome_completo)
```

Concatenação de Strings

- **Strings também suportam o operador `+=` para concatenação, tornando incrementando uma à outra.**

```
nome = "João"
nome += " "
nome += "Gabriel"
print(nome) # João Gabriel
```

Comparação de Strings

- **No Python podemos comparar strings com o operador ==. Com o operador == verificamos se o conteúdo de duas strings é igual. O operador != tem o comportamento inverso**

```
nome_1 = 'Eduardo'
nome_2 = 'Eduardo'
if nome_1 == nome_2:
    print('iguais')
else:
    print('diferentes')
```

```
nome_1 = 'Eduardo'
nome_2 = 'Eduardo'
if nome_1 != nome_2:
    print('diferentes')
else:
    print('iguais')
```

Métodos

len(): dá como resposta o a quantidade de caracteres da string;

```
texto = 'bob1234'
print(len(texto))


# 7
```

Métodos

.capitalize(): Coloca a 1ª letra Maiúscula;

```
texto = 'jhonnata'
print(texto.capitalize())

# 'Jhonnata'
```

Métodos

.casefold(): Transforma todas as letras em minúsculas (existe também o .lower())

```
texto = 'JhoNnaTa'
print(texto.casefold())

# 'jhonnata'
```

Métodos

.count(): Conta a quantidade de vezes que uma string aparece em outra string;

```python
texto = 'jhonnata@gmail.com.br'
print(texto.count('.'))

# 2
```

```python
texto = 'jhonnata@gmail.com.br'
print(texto.count('jhonnata'))

# 1
```

Métodos

.find(): Procura um texto dentro de outro texto e dá como resposta a posição (index) do texto encontrado.

```
texto = 'jhonnata@gmail.com.br'
print(texto.find('@'))

# 8
```

```
texto = 'jhonnata@gmail.com.br'
print(texto.find('com'))

# 15
```

Métodos

.format(): Formata uma string de acordo com os valores passados;

```
faturamento = 1000
texto = 'O faturamento da loja foi de {} reais'.format(faturamento)
print(texto)

#  'O faturamento da loja foi de 1000 reais'
```

OBS: O format retira cada um dos {} e substitui para os valores que forem passados como argumento, então a quantidade de {} e argumentos devem ser iguais, senão gera erro

Métodos

.isalnum(): Verifica se um texto é todo feito com caracteres alfanuméricos (letras e números). Letras com acento ou ç são considerados letras para essa função;

```
texto = 'João123'
print(texto.isalnum())

# True
```

Obs: se o texto fosse 'Jo~ao' ou então 'Joao#' o resultado seria False

Métodos

.isalpha(): Verifica se um texto é todo feito de letras;

```python
texto = 'João'
print(texto.isalpha())

# True
```

Obs: nesse caso se o texto fosse 'Joao123' o resultado seria False, porque 123 são números.

Métodos

.isnumeric(): Verifica se um texto é todo feito por números;

```
texto = '123'
print(texto.isnumeric())

# True
```

Observação

Transformação em número: strings que são compostas apenas de números, podem ser transformadas. utilizando **int** ou **float**

```
texto = "1234567"
numero = int(texto)
print(numero + 1)


#1234568
```

```
texto = "1234567"
numero = float(texto)
print(numero + 1)


#1234568.0
```

Métodos

.replace(): Substitui um texto por um outro texto em uma string;

```
texto = '1000.00'
print(texto.replace('.', ','))

# '1000,00'
```

Obs: o replace precisa de 2 argumentos para funcionar. O 1º é o texto que você quer trocar. O 2º é o texto que você quer colocar no lugar daquele texto que você está tirando.

Métodos

.split(): Separa uma string de acordo com um delimitador em uma lista com os textos separados;

```
texto = 'jhonnata@gmail.com'
print(texto.split('@'))

#  ['jhonnata', 'gmail.com']
```

```
texto = 'jhonnata@gmail.com'
print(texto.split('a'))

#  ['jhonn', 't', '@gm', 'il.com']
```

Métodos

.startswith(): Verifica se a string começa com determinado texto;

```
texto = 'BOB123453'
print(texto.startswith('BOB'))

# True
```

Métodos

.strip(): Retira caracteres indesejados dos textos. Por padrão, retira espaços “extras” no início e no final;

```
texto = ' BOB123453 '
print(texto.strip())

# 'BOB123453'
```

Métodos

.title(): Coloca a 1ª letra de cada palavra em maiúscula;

```
texto = 'jhonnata virginio'
print(texto.title())

#  'Jhonnata Virginio'
```

Métodos

.upper(): Coloca o texto todo em letra maiúscula.

```
texto = 'bob12343'
print(texto.upper())

# 'BOB12343'
```


Agora, vamos praticar!

Exercício 1 - Senhas

seu programa deve receber uma senha do usuário e validar se esta segue os seguinte critérios:

- Ter 10 ou mais caracteres
- Ter pelo menos uma letra maiúscula
- Ter pelo menos uma letra minúscula
- Ter pelo menos um dígito
- Não conter símbolos
- Não iniciar ou terminar nas seguinte sequências:
- 123
- 321
- 000
- abc

SAÍDA:
para sucesso imprima: "sucesso"
para fracasso imprima: "fracasso"

Ignore os espaços em branco no início e no final da senha.
Caso a senha seja válida, imprima uma mensagem de sucesso, caso contrário imprima uma mensagem de fracasso

Exercício 2 - Soma

seu programa deve receber uma sequência de números inteiros separadas por espaços “ “ e mostrar na tela a soma destes valores.

Entrada:
1 2 3 4 5 6 -1 2

SAÍDA:
22

Exercício 3 – Criptografia

seu programa deve receber um texto e criptografá-lo seguindo a cifra de césar.

Cifra de césar consiste em transformar um caractere em outro alfabeticamente, por exemplo para a cifra 1 :

A -> B
B -> C
C -> D

Receba um número inteiro a cifra e um texto em seguida mostre na tela o texto tendo sido feito cifra de césar.

Entrada:

2

aaaaaaaaaaaaaaaaaaaaa

SAÍDA:

ccccccccccccccccccccc