

STRING

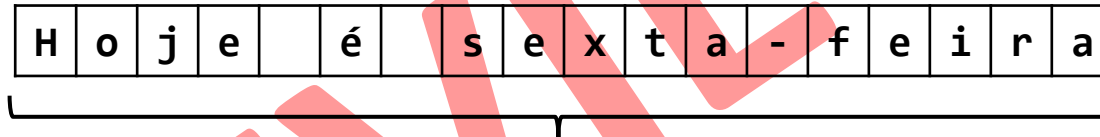


STRING – REVISÃO

TIPOS DE VARIÁVEIS – STRING

VARIAVEIS DO TIPO STRING

- São variáveis que armazenam cadeias de caracteres como nomes e textos em geral.
- Chama-se **CADEIA DE CARACTERES** uma **SEQUENCIA DE SÍMBOLOS** como letras, números, sinais de pontuação, espaço, etc.
- Exemplo: A frase “Hoje é sexta-feira” temos a seguinte string:



String

- É utilizado aspas (") para delimitar o inicio e o fim da sequencia de caracteres. Exemplo:

Print ("mensagem")

- O espaço é entendido como uma string vazia, porem é contabilizada.

TIPOS DE VARIÁVEIS – STRING

ÍNDICE DE UMA STRING

- Cada caractere de uma string é chamada de **ÍNDICE**.
- O tamanho de uma string depende da quantidade de índices no qual a compõem. No exemplo “**Hoje é sexta-feira**” a string é composta por 18 índices.
- A contagem do índice começa da esquerda para direita e começa pelo numeral 0;
- Exemplo: A frase “Hoje é sexta-feira” temos a seguinte string e índices:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	← índices
H	o	j	e		é		s	e	x	t	a	-	f	e	i	r	a	← Conteúdo

String com 18 índices

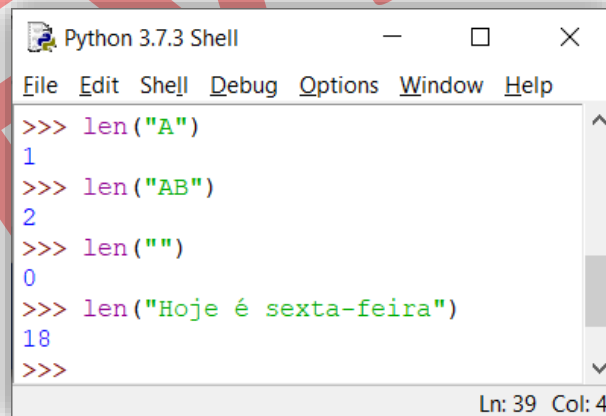
TIPOS DE VARIÁVEIS – STRING

TRABALHANDO COM STRING

- Cada string tem um tamanho associado e o seu conteúdo pode ser acessado caractere a caractere.

FUNÇÃO LEN

- Para consultar o tamanho de uma string utiliza-se a função **len**. Essa associação retorna o numero de caracteres na string.
- A função len retorna um valor do tipo inteiro, representando a quantidade de caracteres contidos na string.



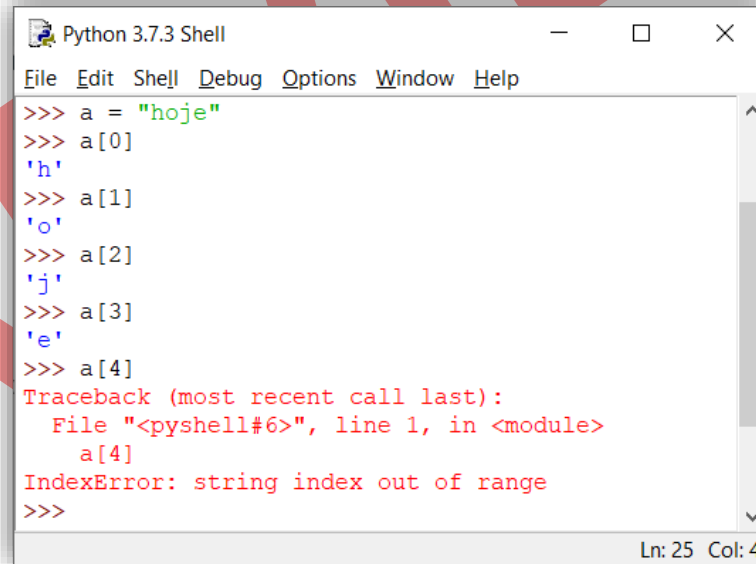
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> len("A")
1
>>> len("AB")
2
>>> len("")
0
>>> len("Hoje é sexta-feira")
18
>>>
```

Ln: 39 Col: 4

TIPOS DE VARIÁVEIS – STRING

ACESSANDO O INDICIE DE UMA STRING

- Para acessar os caracteres (índice) de uma string, deve-se informar o índice ou posição do caractere entre colchetes.
- Como o primeiro caractere de uma string é índice 0, pode-se acessar valores de 0 até o tamanho da string menos 1;
- Exemplo: A string “hoje” tem tamanho 4 e pode ser acessada os índices de 0 a 3. O acesso de um índice maior que a quantidade de caracteres da string, o interpretador emitirá uma mensagem de erro:



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> a = "hoje"
>>> a[0]
'h'
>>> a[1]
'o'
>>> a[2]
'j'
>>> a[3]
'e'
>>> a[4]
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    a[4]
IndexError: string index out of range
>>>
```

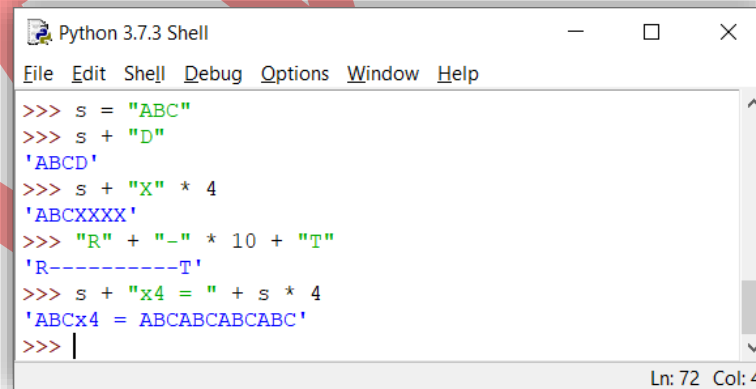
Ln: 25 Col: 4

TIPOS DE VARIÁVEIS – STRING

OPERAÇÕES COM STRING

CONCATENAÇÃO

- Concatenação é um termo usado em computação para designar a operação de unir o conteúdo de duas strings.
- Para concatenar duas string, utiliza-se o operador de adição (+).
- A concatenação pode ocorrer apenas com strings.
- Para concatenar repetindo uma string por varias vezes (caso especial) é utilizado o operador de multiplicação (*)



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> s = "ABC"
>>> s + "D"
'ABCD'
>>> s + "X" * 4
'ABCXXXX'
>>> "R" + "-" * 10 + "T"
'R-----T'
>>> s + "x4 = " + s * 4
'ABCx4 = ABCABCABCABC'
>>> |
```

Ln: 72 Col: 4

TIPOS DE VARIÁVEIS – STRING

COMPOSIÇÃO

- Composição de string é utilizada para apresentar mensagens com conteúdo de variável ou variáveis.
- Exemplo: “**João tem x anos**” onde x é o valor da idade de João.
- Existem 3 formas diferentes de executar essa finalidade.

OPÇÃO 1 : MARCADOS DE POSIÇÃO

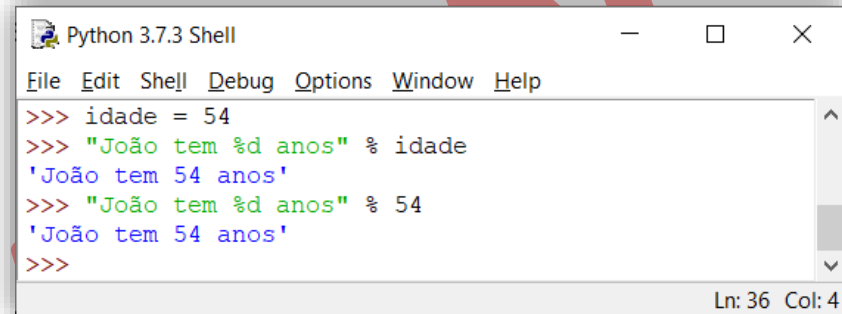
- Escreve-se a string, substituindo a variável por um marcador de posição e referenciando a variável na qual será alocado na marcação
- A marcação pode ser do tipo numero inteiro, string ou numero decimal, conforme representado na tabela abaixo:

MARCADOR	TIPO
%d	Numero Inteiro
%s	Strings
%f	Numero Decimal

TIPOS DE VARIÁVEIS – STRING

NUMERO INTEIRO - %d

- É alocado na string o %d onde será substituído pela variável ou valor indicado do tipo inteiro.
- Exemplo: “João tem 54 anos”



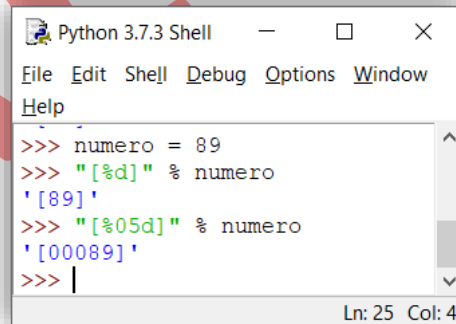
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> idade = 54
>>> "João tem %d anos" % idade
'João tem 54 anos'
>>> "João tem %d anos" % 54
'João tem 54 anos'
>>>
```

Ln: 36 Col: 4

FORMATAÇÃO DO NUMERO INTEIRO:

➤ FIXAR POSIÇÕES NUMÉRICAS, COMPLETANDO COM ZERO A ESQUERDA.

- **%0nd** Onde: **n** é o número de posições numéricas reservadas para a variável
- Exemplo:



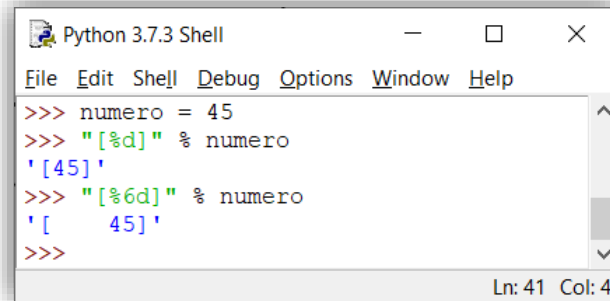
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> numero = 89
>>> "[%d]" % numero
'[89]'
>>> "[%05d]" % numero
'[00089]'
>>> |
```

Ln: 25 Col: 4

TIPOS DE VARIÁVEIS – STRING

➤ FIXAR POSIÇÕES NUMÉRICAS, SEM COMPLETAR ZERO A ESQUERDA.

- **%nd** Onde: **n** é o numero de posições numéricas reservadas para a variável

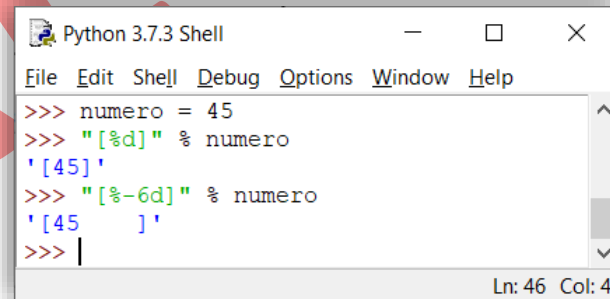


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> numero = 45
>>> "[%d]" % numero
'[45]'
>>> "[%6d]" % numero
'[    45]'
>>>
```

Ln: 41 Col: 4

➤ FIXAR POSIÇÕES NUMÉRICAS, ALINHADO A PARTIR DA ESQUEURDA.

- **%-nd** Onde: **n** é o numero de posições numéricas reservadas para a variável

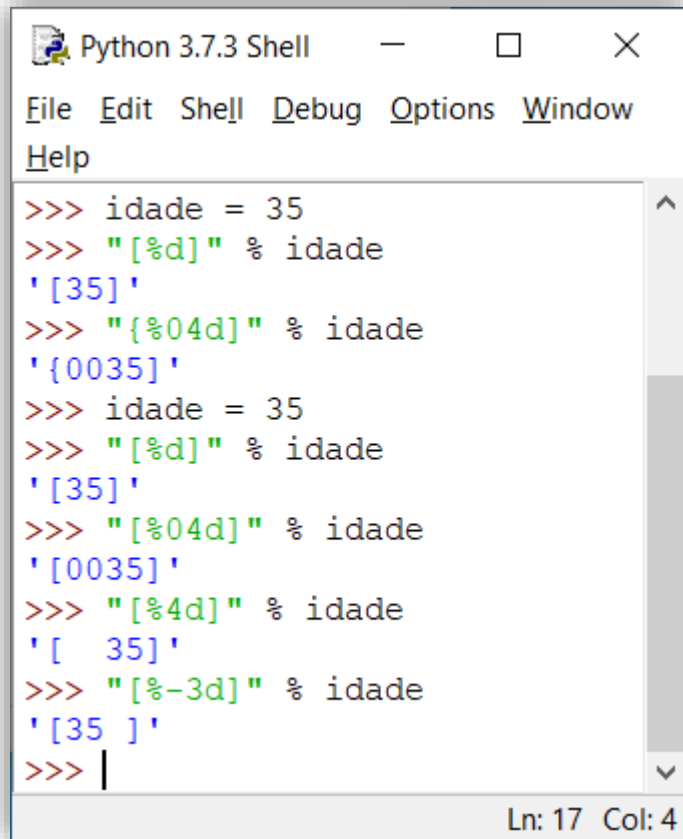


```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> numero = 45
>>> "[%d]" % numero
'[45]'
>>> ["%-6d]" % numero
'[45      ]'
>>> |
```

Ln: 46 Col: 4

TIPOS DE VARIÁVEIS – STRING

- Exemplo: “João tem **idade** anos”



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> idade = 35
>>> "[%d]" % idade
'[35]'
>>> "{%04d}" % idade
'[0035]'
>>> idade = 35
>>> "[%d]" % idade
'[35]'
>>> "[%04d]" % idade
'[0035]'
>>> "[%4d]" % idade
'[ 35]'
>>> "[% -3d]" % idade
'[35 ]'
>>> |
```

Ln: 17 Col: 4

RESUMO:

%d – Padrão, sem formatar.

%0nd – Limitando 'n' casas e preenchendo 0 a esquerda.

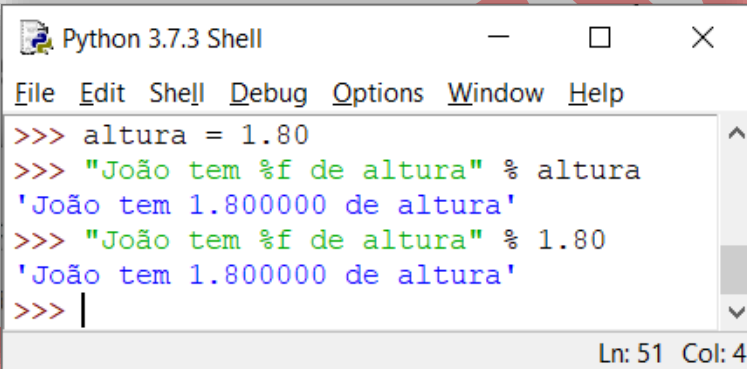
%nd – Limitando 'n' casas e sem preencher 0 a esquerda.

%-nd – Limitando 'n' casas e alinhando a esquerda.

TIPOS DE VARIÁVEIS – STRING

NUMERO DECIMAL - %f

- É alocado na string o %f onde será substituído pela variável ou valor indicado do tipo decimal.
- Exemplo: “João tem 1.80 de altura”



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> altura = 1.80
>>> "João tem %f de altura" % altura
'João tem 1.800000 de altura'
>>> "João tem %f de altura" % 1.80
'João tem 1.800000 de altura'
>>> |
```

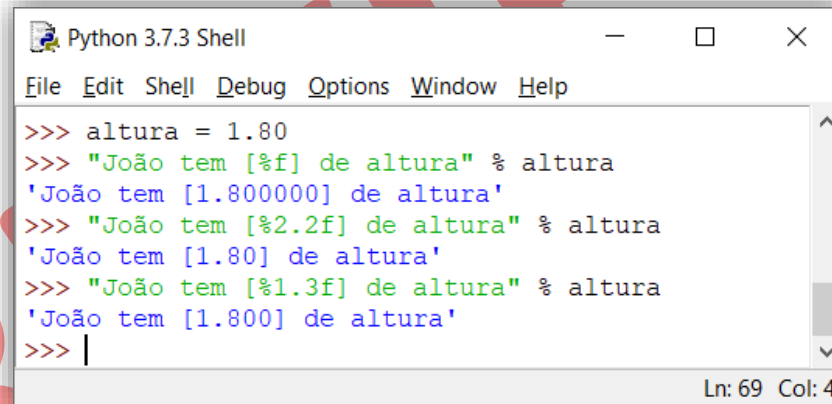
Ln: 51 Col: 4

- A formatação dos números decimais utilizam dois valores entre o símbolo % e a letra f

TIPOS DE VARIÁVEIS – STRING

FORMATAÇÃO DO NUMERO DECIMAL:

- `% X . Y f`
- Onde **X** é o numero de posição da **parte inteira** e **Y** é o numero de posição da **parte decimal**.
- Exemplo: “João tem 1.80 de altura”



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> altura = 1.80
>>> "João tem [%f] de altura" % altura
'João tem [1.800000] de altura'
>>> "João tem [%2.2f] de altura" % altura
'João tem [1.80] de altura'
>>> "João tem [%1.3f] de altura" % altura
'João tem [1.800] de altura'
>>> |
```

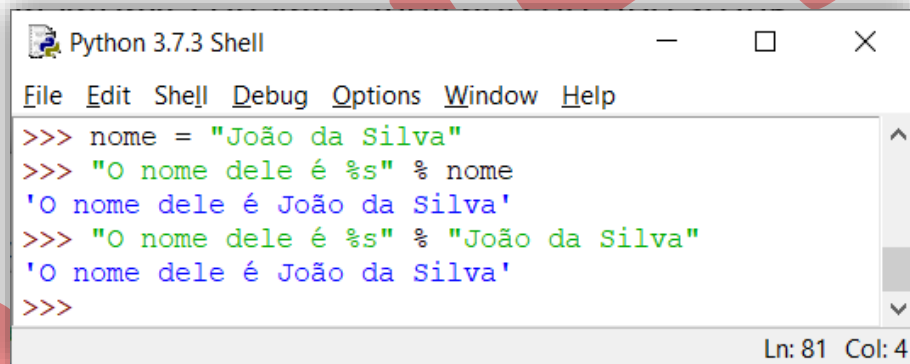
Ln: 69 Col: 4

- O valor da parte inteira é reservado, porem o interpretador não muda em decorrência dessa variação.

TIPOS DE VARIÁVEIS – STRING

STRING - %s

- É alocado na string o %s onde será substituído pela variável ou valor indicado do tipo string.
- Exemplo: **“O nome dele é João da Silva”**



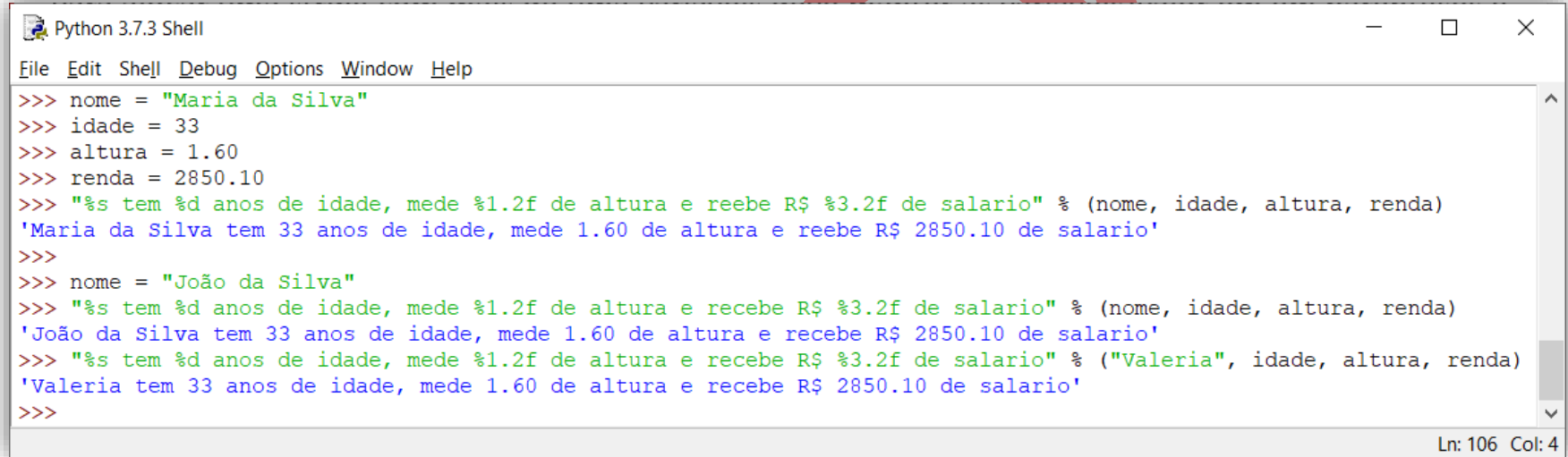
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "João da Silva"
>>> "O nome dele é %s" % nome
'O nome dele é João da Silva'
>>> "O nome dele é %s" % "João da Silva"
'O nome dele é João da Silva'
>>>
```

Ln: 81 Col: 4

TIPOS DE VARIÁVEIS – STRING

COMPOSIÇÃO DE STRING COM MARCADOR DE POSIÇÃO:

- Python suporta diversas operações com marcadores.
- Quando se tem mais de um marcador na string, os valores devem ser escritos na ordem na qual é chamada na string e entre parênteses.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "Maria da Silva"
>>> idade = 33
>>> altura = 1.60
>>> renda = 2850.10
>>> "%s tem %d anos de idade, mede %1.2f de altura e reebe R$ %3.2f de salario" % (nome, idade, altura, renda)
'Maria da Silva tem 33 anos de idade, mede 1.60 de altura e reebe R$ 2850.10 de salario'
>>>
>>> nome = "João da Silva"
>>> "%s tem %d anos de idade, mede %1.2f de altura e recebe R$ %3.2f de salario" % (nome, idade, altura, renda)
'João da Silva tem 33 anos de idade, mede 1.60 de altura e recebe R$ 2850.10 de salario'
>>> "%s tem %d anos de idade, mede %1.2f de altura e recebe R$ %3.2f de salario" % ("Valeria", idade, altura, renda)
'Valeria tem 33 anos de idade, mede 1.60 de altura e recebe R$ 2850.10 de salario'
>>>
```

Ln: 106 Col: 4

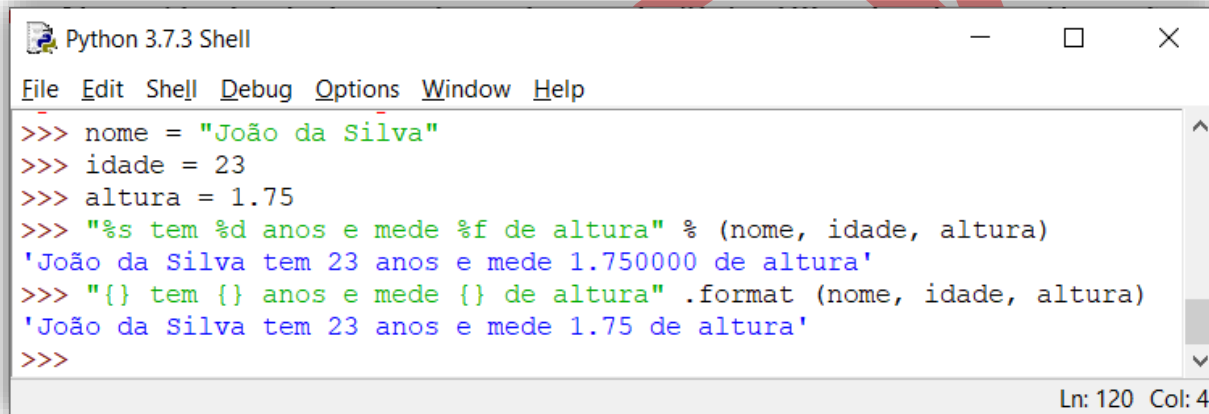
- O método de marcador de posição é análogo ao utilizado em outras linguagens, porem tem caído em desuso com métodos mais avançados. Porem seu domínio ajuda na interpretação de programas escritos no método (antigos).

TIPOS DE VARIÁVEIS – STRING

OPÇÃO 2 : METODO FORMAT

- No método de format, no lugar do % é utilizado chaves {} e dos parênteses o **.format**.

EXEMPLO:



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "João da Silva"
>>> idade = 23
>>> altura = 1.75
>>> "%s tem %d anos e mede %f de altura" % (nome, idade, altura)
'João da Silva tem 23 anos e mede 1.750000 de altura'
>>> "{} tem {} anos e mede {} de altura".format(nome, idade, altura)
'João da Silva tem 23 anos e mede 1.75 de altura'
>>>
```

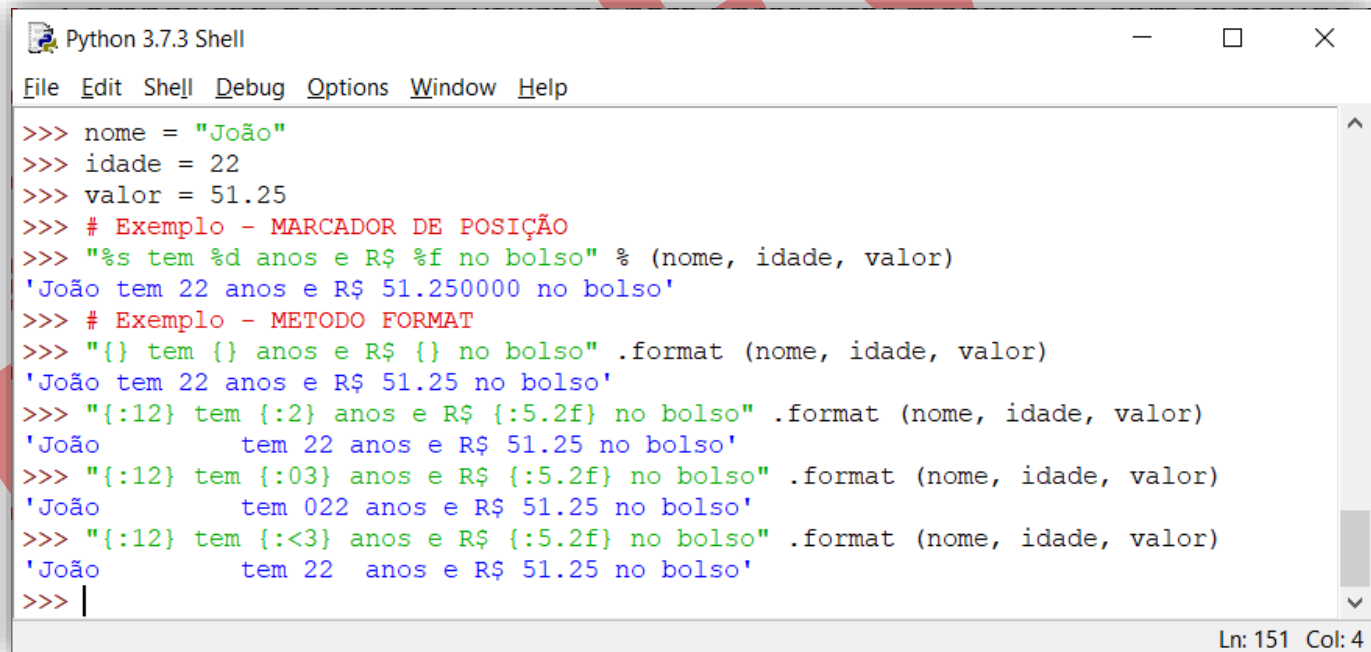
- No Método Format apresenta de forma mais “inteligente” os valores.
- Em resumo, substitui-se o % por **.format** e os %d, %s, %f por {}

TIPOS DE VARIÁVEIS – STRING

FORMATAÇÃO DE STRING NO METODO FORMAT:

- No método de format, para escrever o tamanho da mascaras é utilizado o `:` internamente nas chaves `{}`. Para numero inteiro, utiliza-se apenas o `:` com numero decimal exige o `f` completando e no caso de alinhamento a esquerda, ao invés de `-` utiliza o `<`

EXEMPLO:

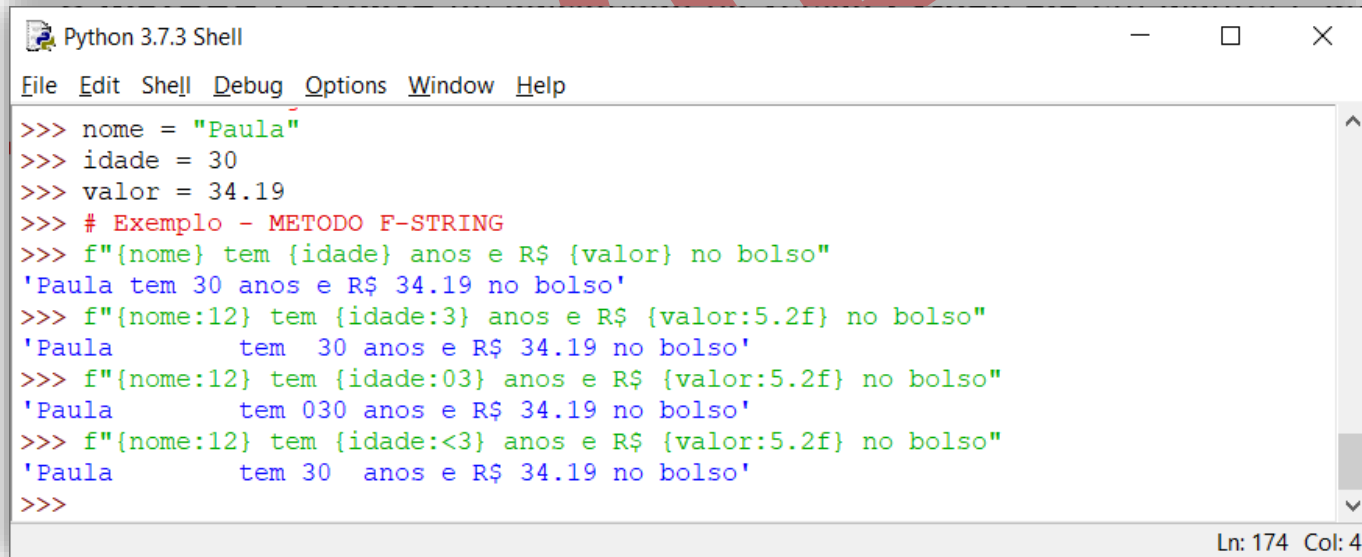
A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text 'Python 3.7.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a series of Python code snippets and their outputs. The code starts with variable assignments: 'nome = "João"', 'idade = 22', and 'valor = 51.25'. It then shows two examples of string formatting. The first example, labeled '# Exemplo - MARCADOR DE POSIÇÃO', uses the % operator to format a string: '%s tem %d anos e R\$ %f no bolso' % (nome, idade, valor), resulting in the output 'João tem 22 anos e R\$ 51.250000 no bolso'. The second example, labeled '# Exemplo - METODO FORMAT', uses the .format() method. It shows several variations: 1) '{} tem {} anos e R\$ {} no bolso' .format (nome, idade, valor) resulting in 'João tem 22 anos e R\$ 51.25 no bolso'. 2) '{:12} tem {:2} anos e R\$ {:.5.2f} no bolso' .format (nome, idade, valor) resulting in 'João tem 22 anos e R\$ 51.25 no bolso'. 3) '{:12} tem {:03} anos e R\$ {:.5.2f} no bolso' .format (nome, idade, valor) resulting in 'João tem 022 anos e R\$ 51.25 no bolso'. 4) '{:12} tem {:<3} anos e R\$ {:.5.2f} no bolso' .format (nome, idade, valor) resulting in 'João tem 22 anos e R\$ 51.25 no bolso'. The window status bar at the bottom right shows 'Ln: 151 Col: 4'.

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "João"
>>> idade = 22
>>> valor = 51.25
>>> # Exemplo - MARCADOR DE POSIÇÃO
>>> "%s tem %d anos e R$ %f no bolso" % (nome, idade, valor)
'João tem 22 anos e R$ 51.250000 no bolso'
>>> # Exemplo - METODO FORMAT
>>> "{} tem {} anos e R$ {} no bolso" .format (nome, idade, valor)
'João tem 22 anos e R$ 51.25 no bolso'
>>> "{:12} tem {:2} anos e R$ {:.5.2f} no bolso" .format (nome, idade, valor)
'João      tem 22 anos e R$ 51.25 no bolso'
>>> "{:12} tem {:03} anos e R$ {:.5.2f} no bolso" .format (nome, idade, valor)
'João      tem 022 anos e R$ 51.25 no bolso'
>>> "{:12} tem {:<3} anos e R$ {:.5.2f} no bolso" .format (nome, idade, valor)
'João      tem 22  anos e R$ 51.25 no bolso'
>>> |
Ln: 151 Col: 4
```

TIPOS DE VARIÁVEIS – STRING

OPÇÃO 3 : METODO F-STRING

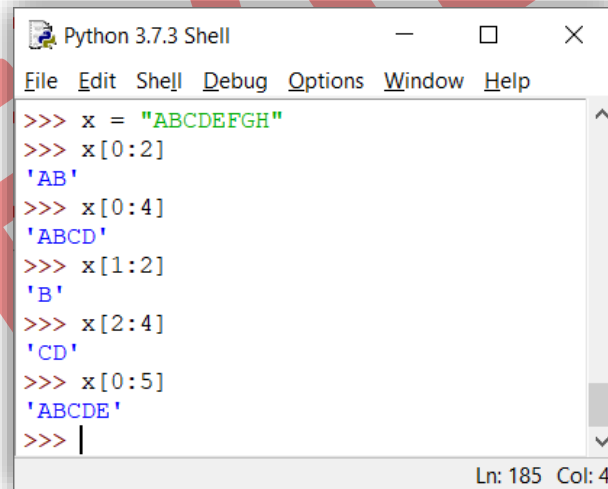
- O **MÉTODO F-STRING** foi adicionado na versão **Python 3.6** em diante. É uma forma mais moderna e compacta.
- Neste método escreve-se a letra **f** antes de abrir as aspas e escreve-se o nome da variável diretamente na string, entre **{}**.
- A **FORMATAÇÃO** utilizada no **METODO FORMAT** se mantém no **METODO F-STRING**

A screenshot of a Python 3.7.3 Shell window. The window has a title bar with the text 'Python 3.7.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a series of Python commands and their outputs, demonstrating the use of F-strings. The commands are: 1. 'nome = "Paula"' 2. 'idade = 30' 3. 'valor = 34.19' 4. '# Exemplo - METODO F-STRING' 5. 'f"{nome} tem {idade} anos e R\$ {valor} no bolso"' 6. 'f"{nome:12} tem {idade:3} anos e R\$ {valor:5.2f} no bolso"' 7. 'f"{nome:12} tem {idade:03} anos e R\$ {valor:5.2f} no bolso"' 8. 'f"{nome:12} tem {idade:<3} anos e R\$ {valor:5.2f} no bolso"' The outputs are: 1. 'Paula tem 30 anos e R\$ 34.19 no bolso' 2. 'Paula tem 30 anos e R\$ 34.19 no bolso' 3. 'Paula tem 030 anos e R\$ 34.19 no bolso' 4. 'Paula tem 30 anos e R\$ 34.19 no bolso' The status bar at the bottom right shows 'Ln: 174 Col: 4'.

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> nome = "Paula"
>>> idade = 30
>>> valor = 34.19
>>> # Exemplo - METODO F-STRING
>>> f"{nome} tem {idade} anos e R$ {valor} no bolso"
'Paula tem 30 anos e R$ 34.19 no bolso'
>>> f"{nome:12} tem {idade:3} anos e R$ {valor:5.2f} no bolso"
'Paula      tem 30 anos e R$ 34.19 no bolso'
>>> f"{nome:12} tem {idade:03} anos e R$ {valor:5.2f} no bolso"
'Paula      tem 030 anos e R$ 34.19 no bolso'
>>> f"{nome:12} tem {idade:<3} anos e R$ {valor:5.2f} no bolso"
'Paula      tem 30  anos e R$ 34.19 no bolso'
>>>
```

FATIAMENTO DE STRING

- O **FATIAMENTO DE STRING** é um poderoso recurso do Python, muito utilizado em **DATA SCIENCE** e outros recursos para resolução de problemas.
- O fatiamento funciona com a utilização de dois pontos no índice da string.
- O numero a esquerda dos dois pontos indica a posição de inicio da fatia e o à direita do fim.
- O final da fatia não é incluso na apresentação.



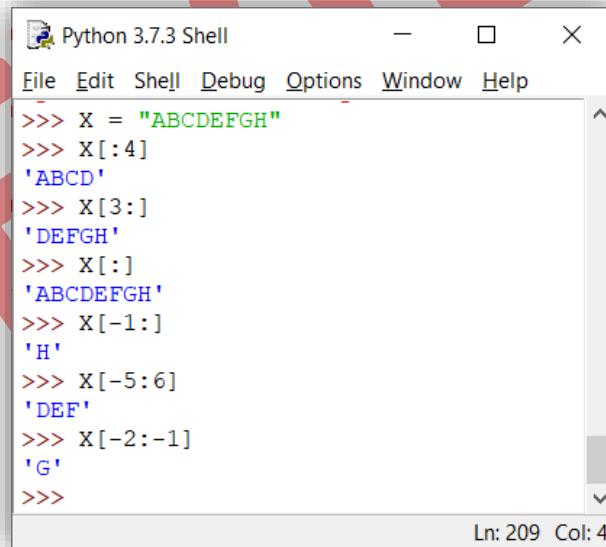
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> x = "ABCDEFGH"
>>> x[0:2]
'AB'
>>> x[0:4]
'ABCD'
>>> x[1:2]
'B'
>>> x[2:4]
'CD'
>>> x[0:5]
'ABCDE'
>>> |
```

Ln: 185 Col: 4

FATIAMENTO DE STRING

VARIAÇÕES DO FATIAMENTO DE STRING:

- **Omitir o numero a esquerda** representa do **inicio até o índice determinado**.
- **Omitir o numero a direita** representa no **índice determinado até o final**.
- **Omitir os números (esquerda e direita)** irá fazer uma copia de **todos os caracteres** da string.
- Utilizar valor negativo para indicar posições a partir da direita (-1 é o ultimo caractere e -2 o penúltimo, etc.).



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
>>> X = "ABCDEFGH"
>>> X[:4]
'ABCD'
>>> X[3:]
'DFGH'
>>> X[:]
'ABCDEFGH'
>>> X[-1:]
'H'
>>> X[-5:6]
'DEF'
>>> X[-2:-1]
'G'
>>>
```

Ln: 209 Col: 4

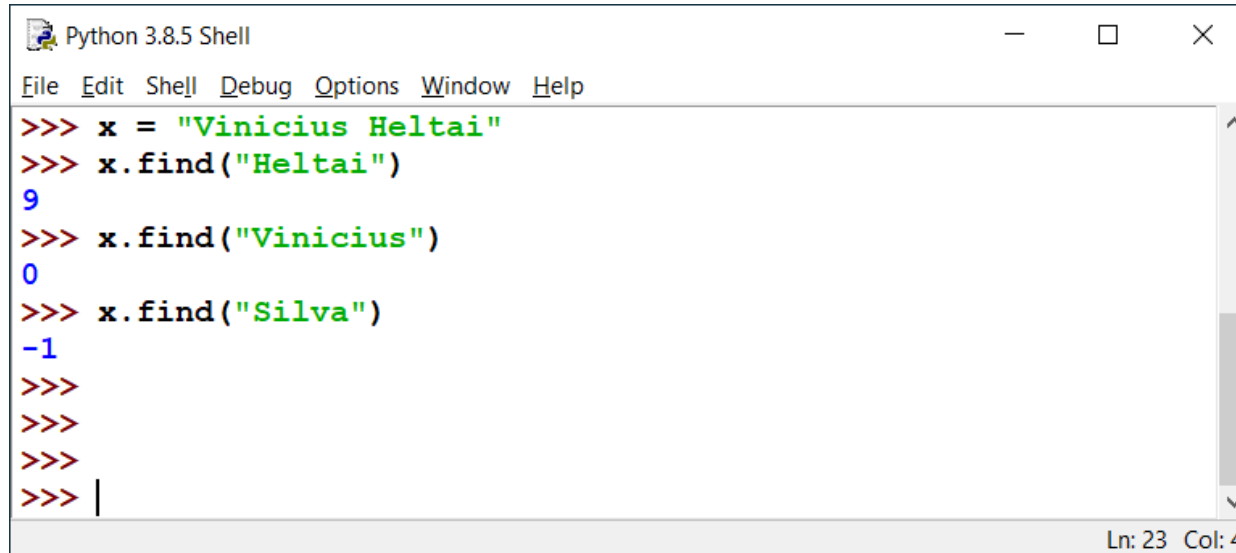


STRING – FUNÇÕES COMPLEMENTARES

FUNÇÕES COMPLEMENTARES

FUNÇÃO FIND()

- A Função find procura por uma ocorrência de determinado caractere em um String, e retorna o seu endereço dentro da String.
- Retornando -1 indica que não existe tal caractere na String.



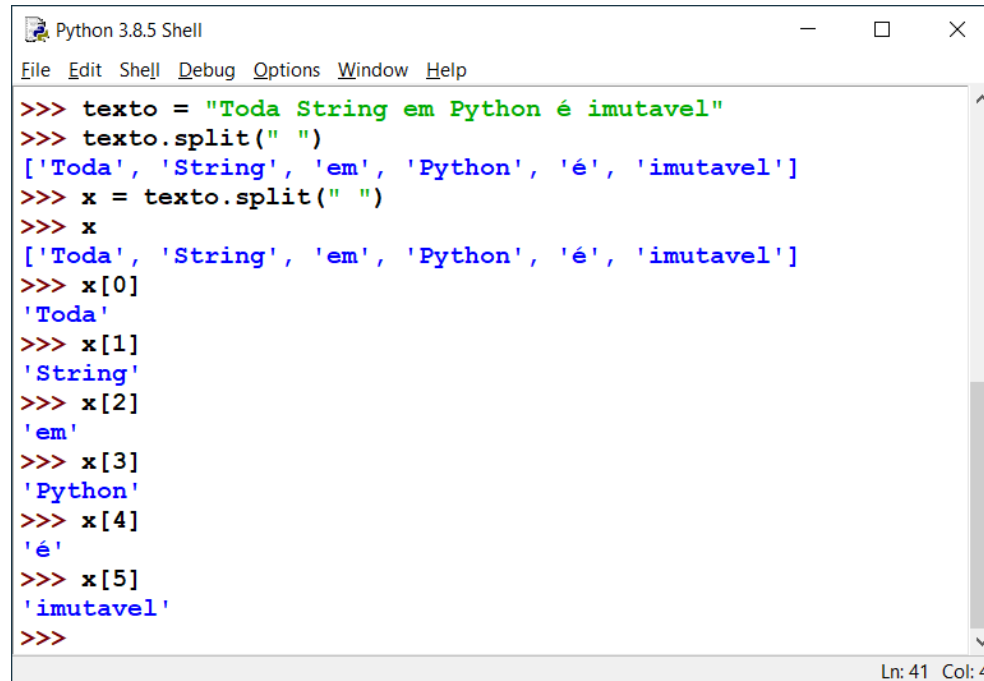
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> x = "Vinicius Heltai"
>>> x.find("Heltai")
9
>>> x.find("Vinicius")
0
>>> x.find("Silva")
-1
>>>
>>>
>>>
>>> |
```

Ln: 23 Col: 4

FUNÇÕES COMPLEMENTARES

FUNÇÃO SPLIT()

- A função split() divide um texto todas as vezes que a String passada como argumento for localizada.
- No código a seguir, definimos uma frase e sem seguida, invocamos a função split(), definindo como argumento, uma String que contém um único espaço em branco.



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help

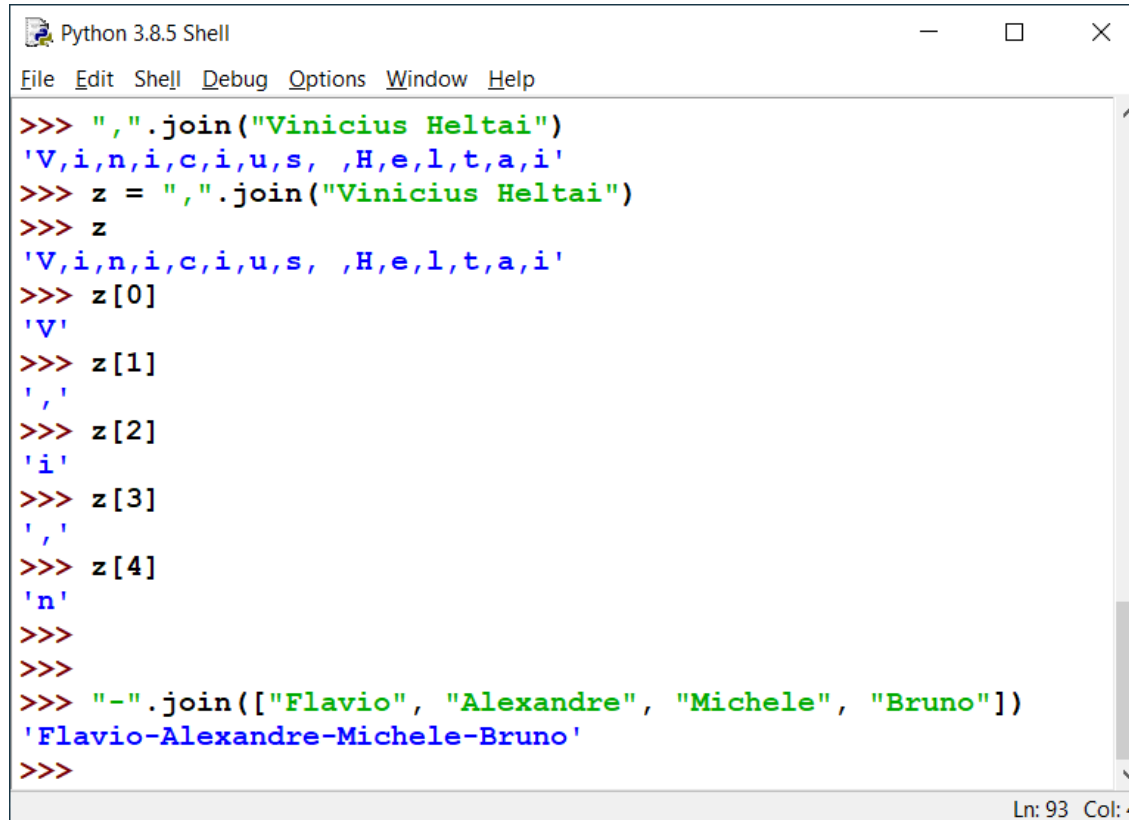
>>> texto = "Toda String em Python é imutavel"
>>> texto.split(" ")
['Toda', 'String', 'em', 'Python', 'é', 'imutavel']
>>> x = texto.split(" ")
>>> x
['Toda', 'String', 'em', 'Python', 'é', 'imutavel']
>>> x[0]
'Toda'
>>> x[1]
'String'
>>> x[2]
'em'
>>> x[3]
'Python'
>>> x[4]
'é'
>>> x[5]
'imutavel'
>>>
```

Ln: 41 Col: 4

FUNÇÕES COMPLEMENTARES

FUNÇÃO JOIN()

- Junta cada item da string com um delimitador especificado.



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help

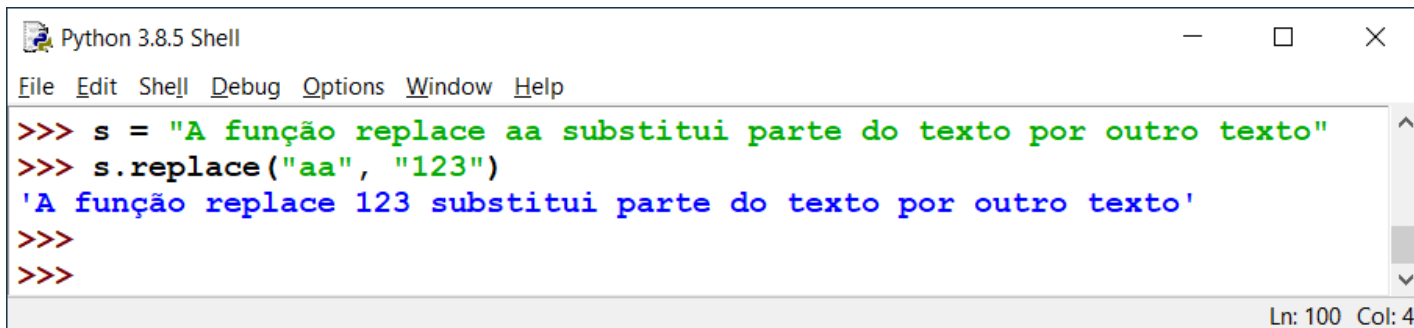
>>> ",".join("Vinicius Heltai")
'V,i,n,i,c,i,u,s, ,H,e,l,t,a,i'
>>> z = ",".join("Vinicius Heltai")
>>> z
'V,i,n,i,c,i,u,s, ,H,e,l,t,a,i'
>>> z[0]
'V'
>>> z[1]
','
>>> z[2]
'i'
>>> z[3]
','
>>> z[4]
'n'
>>>
>>>
>>> "-".join(["Flavio", "Alexandre", "Michele", "Bruno"])
'Flavio-Alexandre-Michele-Bruno'
>>>
```

Ln: 93 Col: 4

FUNÇÕES COMPLEMENTARES

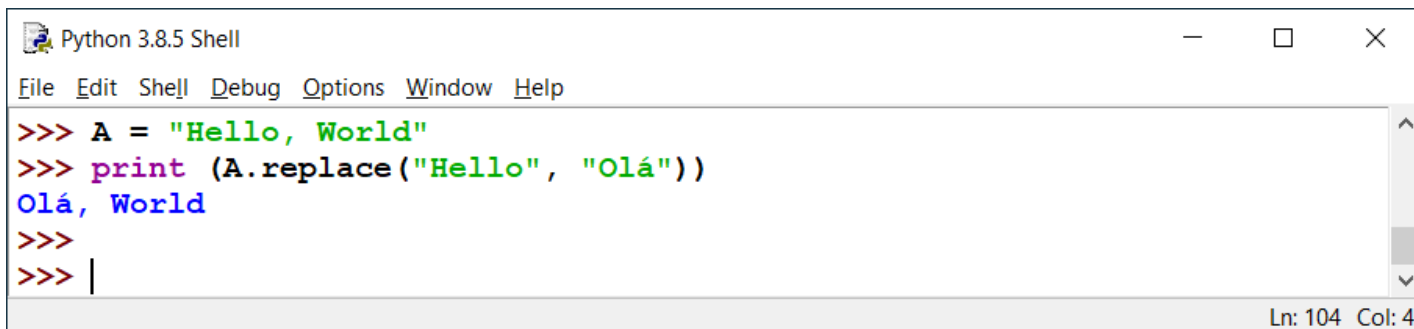
FUNÇÃO REPLACE()

- A função `replace()` substitui uma parte do texto por uma outra String.
- A palavra `replace()`, do Inglês, significa substituir e é isso que a função `replace()` da classe String do Python faz.



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> s = "A função replace aa substitui parte do texto por outro texto"
>>> s.replace("aa", "123")
'A função replace 123 substitui parte do texto por outro texto'
>>>
>>>
```

Ln: 100 Col: 4



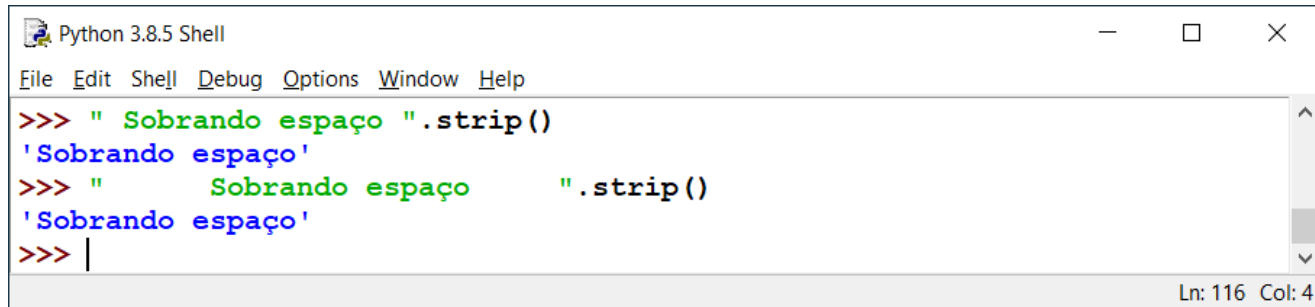
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> A = "Hello, World"
>>> print (A.replace("Hello", "Olá"))
Olá, World
>>>
>>> |
```

Ln: 104 Col: 4

FUNÇÕES COMPLEMENTARES

FUNÇÃO STRIP()

- Retira espaços em branco no começo e no fim



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> " Sobrando espaço ".strip()
'Sobrando espaço'
>>> "      Sobrando espaço      ".strip()
'Sobrando espaço'
>>> |
```

Ln: 116 Col: 4

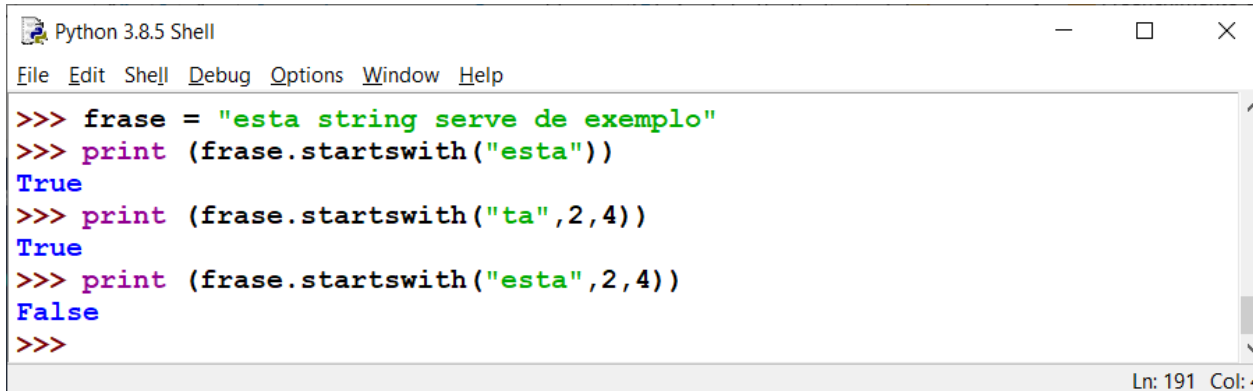
FUNÇÕES COMPLEMENTARES

FUNÇÃO STARTSWITH()

- A função Startswith() é utilizada para verificar se a string é especificado sub-string no início, se ele retorna True, caso contrário, False.
- Se o argumento implorar e terminar o valor especificado, verifique dentro da faixa especificada.

string.startswith(str, beg=0,end=len(string));

- ✓ **str** - caracteres detectados.
- ✓ **beg** - parâmetro opcional é usado para definir a posição inicial da detecção string.
- ✓ **end** - parâmetro opcional é usado para definir a detecção posição final



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> frase = "esta string serve de exemplo"
>>> print (frase.startswith("esta"))
True
>>> print (frase.startswith("ta",2,4))
True
>>> print (frase.startswith("esta",2,4))
False
>>>
```

Ln: 191 Col: 4

FUNÇÕES COMPLEMENTARES

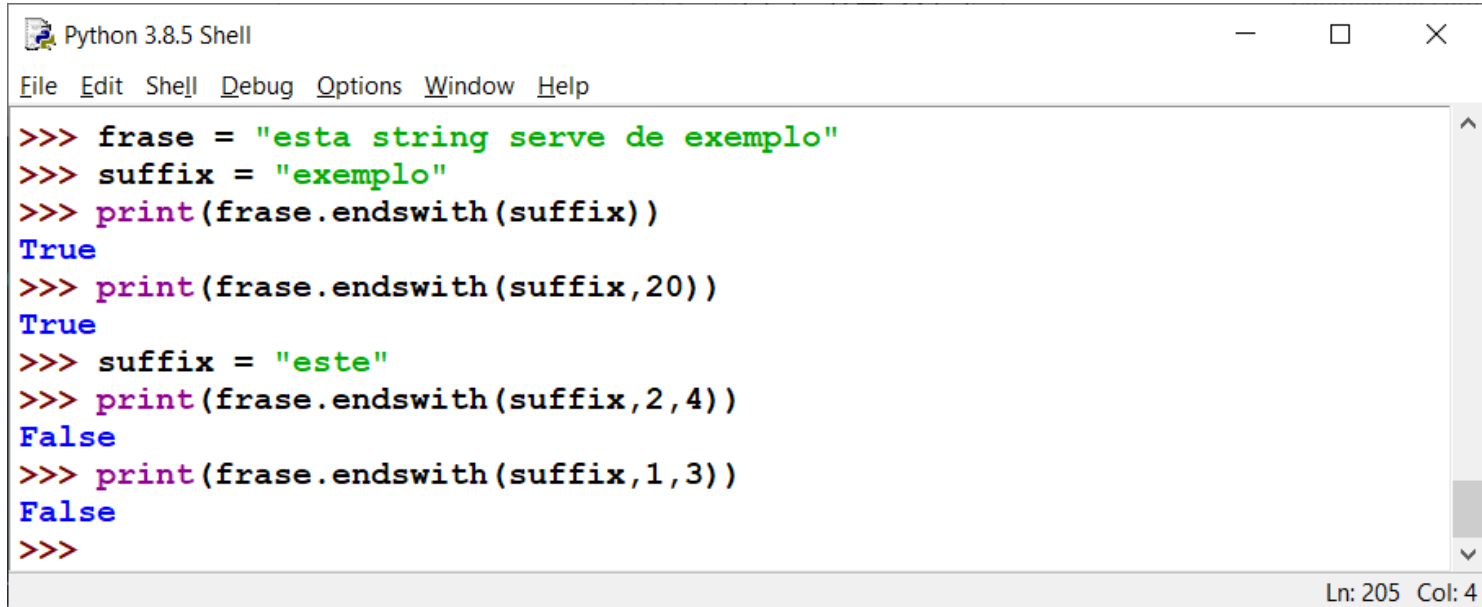
FUNÇÃO ENDSWITH()

- A função Startswith() é usado para determinar se uma sequência termina com o sufixo especificado, se terminam com o sufixo especificado retorna True, caso contrário, False.
- Parâmetros opcionais "start" e "fim" para a posição de início e fim da sequência de pesquisa.

string.endswith(suffix[, start[, end]])

- ✓ **sufixo** - O parâmetro pode ser uma string ou um elemento.
- ✓ **start** - A posição de início da cadeia
- ✓ **end** - Posição final dos personagens

FUNÇÕES COMPLEMENTARES



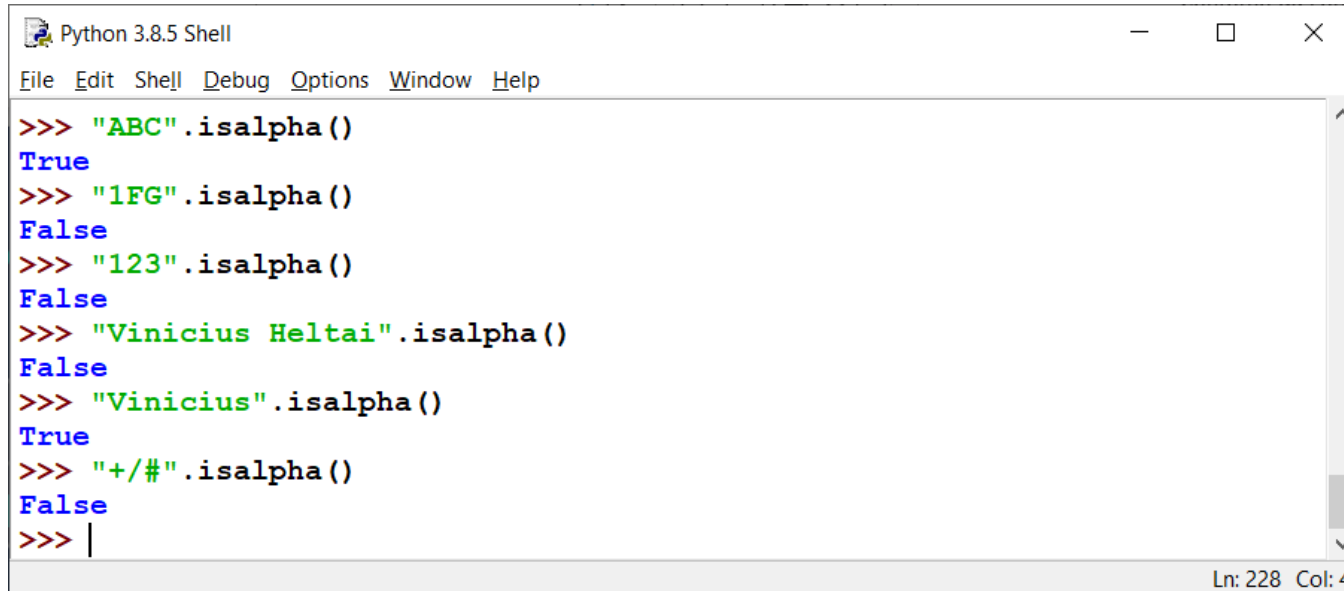
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> frase = "esta string serve de exemplo"
>>> suffix = "exemplo"
>>> print(frase.endswith(suffix))
True
>>> print(frase.endswith(suffix,20))
True
>>> suffix = "este"
>>> print(frase.endswith(suffix,2,4))
False
>>> print(frase.endswith(suffix,1,3))
False
>>>
```

Ln: 205 Col: 4

FUNÇÕES COMPLEMENTARES

FUNÇÃO ISALPHA()

- Retorna False se a string contiver algum caractere que não seja letras



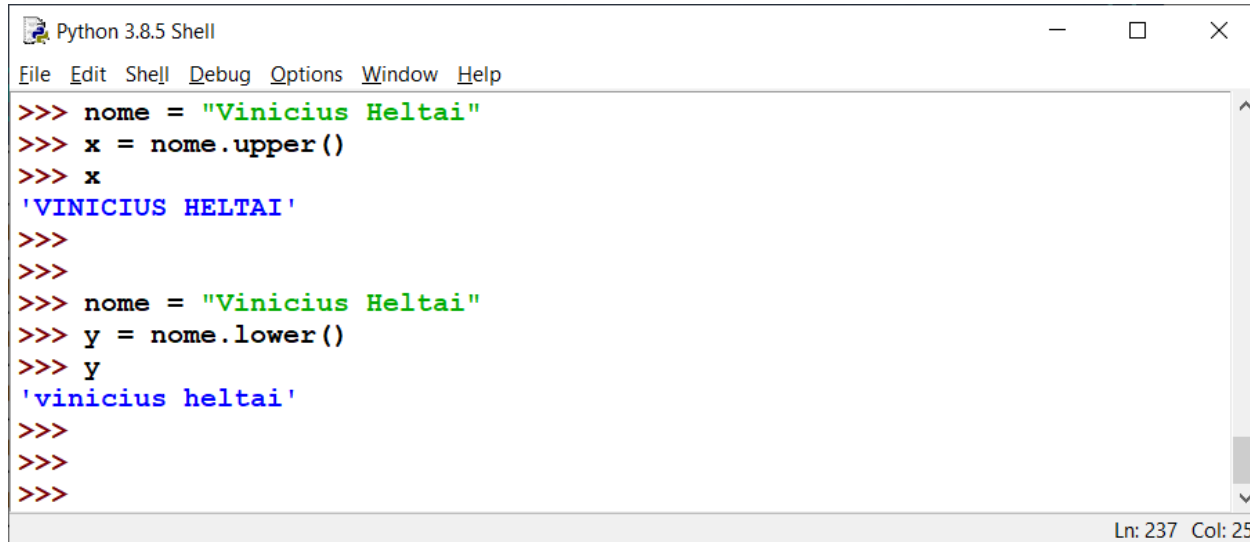
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> "ABC".isalpha()
True
>>> "1FG".isalpha()
False
>>> "123".isalpha()
False
>>> "Vinicius Heltai".isalpha()
False
>>> "Vinicius".isalpha()
True
>>> "+/#".isalpha()
False
>>> |
```

Ln: 228 Col: 4

FUNÇÕES COMPLEMENTARES

FUNÇÕES UPPER() / LOWER()

- Transforma a string em maiúscula / minúscula

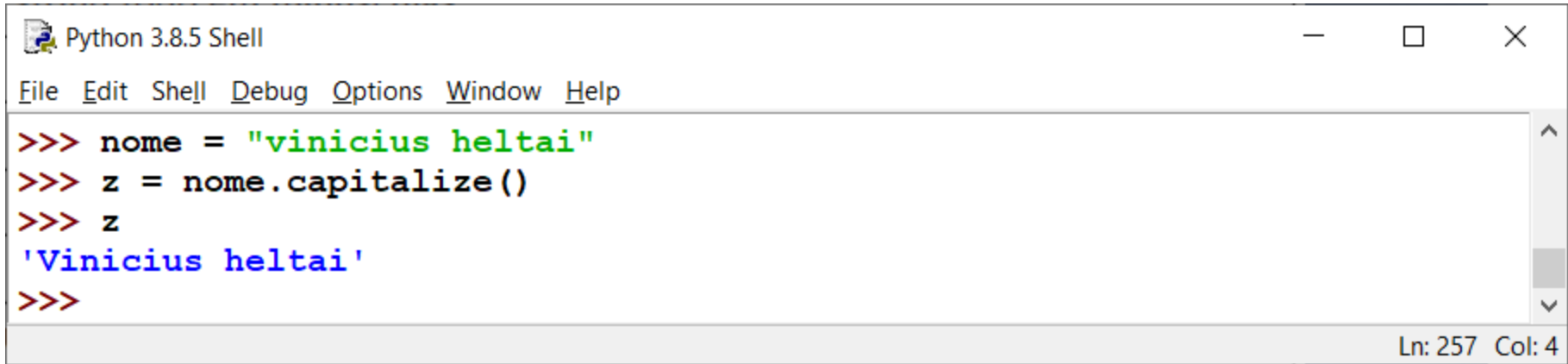


```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> nome = "Vinicius Heltai"
>>> x = nome.upper()
>>> x
'VINICIUS HELTAI'
>>>
>>>
>>> nome = "Vinicius Heltai"
>>> y = nome.lower()
>>> y
'vinicius heltai'
>>>
>>>
>>>
Ln: 237 Col: 25
```

FUNÇÕES COMPLEMENTARES

FUNÇÕES CAPITALIZE()

- Retorna uma string com o primeiro caractere em maiúscula, e o resto em minúsculas.



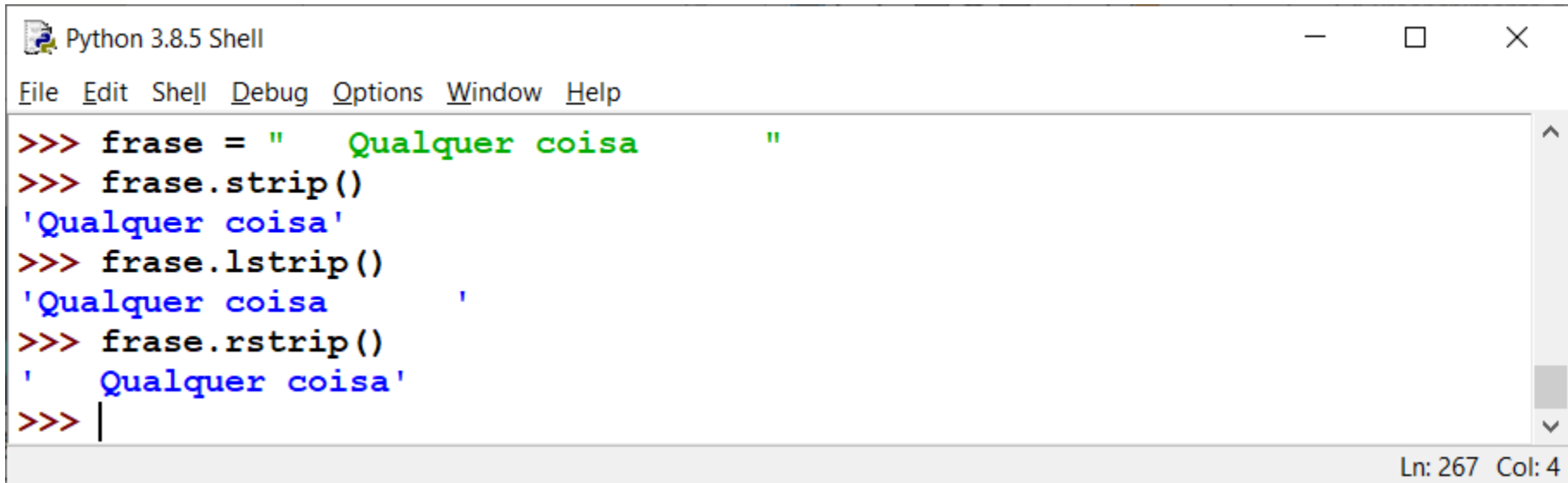
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> nome = "vinicius heltai"
>>> z = nome.capitalize()
>>> z
'Vinicius heltai'
>>>
```

Ln: 257 Col: 4

FUNÇÕES COMPLEMENTARES

FUNÇÕES STRIP() / LSTRIP() / RSTRIP()

- **STRIP()** – Retorna uma string removendo caracteres em branco do início e do fim
- **LSTRIP()** – Retorna uma string removendo caracteres em branco no início
- **RSTRIP()** – Retorna uma string removendo caracteres em branco no fim.



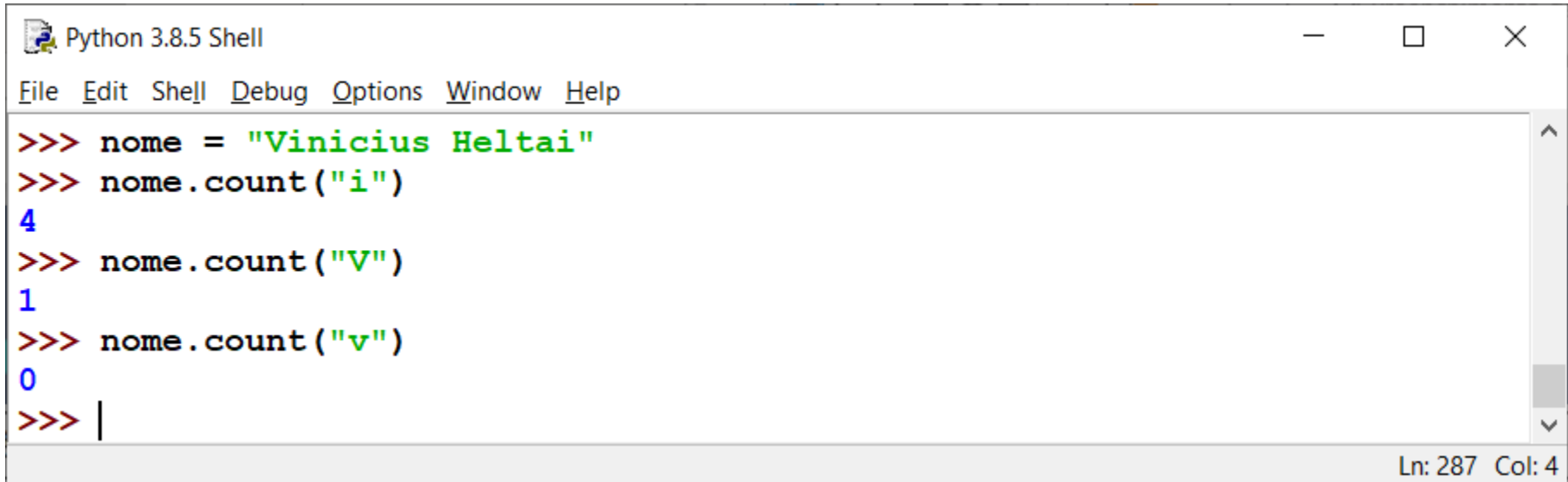
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> frase = "    Qualquer coisa    "
>>> frase.strip()
'Qualquer coisa'
>>> frase.lstrip()
'Qualquer coisa    '
>>> frase.rstrip()
'    Qualquer coisa'
>>> |
```

Ln: 267 Col: 4

FUNÇÕES COMPLEMENTARES

FUNÇÕES COUNT()

- **COUNT()** – Retorna o numero de ocorrências de item



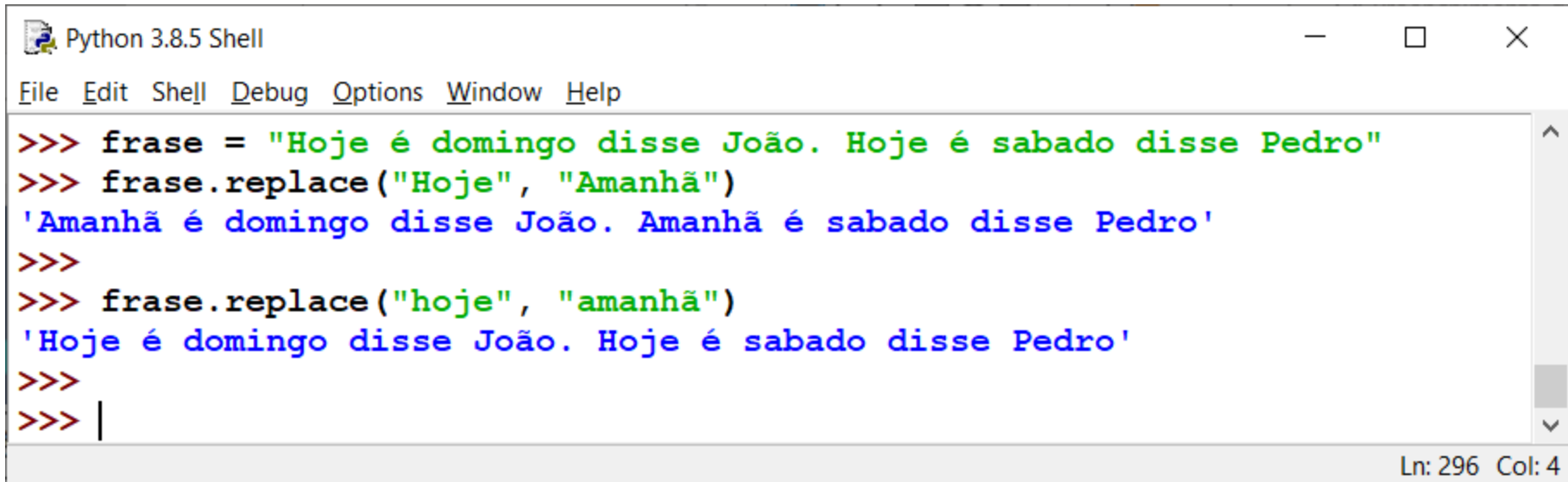
```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> nome = "Vinicius Heltai"
>>> nome.count("i")
4
>>> nome.count("V")
1
>>> nome.count("v")
0
>>> |
```

Ln: 287 Col: 4

FUNÇÕES COMPLEMENTARES

FUNÇÕES REPLACE()

- **REPLACE ()** – Substitui todas as ocorrências do subtring old por new



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
>>> frase = "Hoje é domingo disse João. Hoje é sabado disse Pedro"
>>> frase.replace("Hoje", "Amanhã")
'Amanhã é domingo disse João. Amanhã é sabado disse Pedro'
>>>
>>> frase.replace("hoje", "amanhã")
'Hoje é domingo disse João. Hoje é sabado disse Pedro'
>>>
>>> |
```

Ln: 296 Col: 4



PROF. VINICIUS HELTAI

vinicius.pacheco@docente.unip.br

www.heltai.com.br

(11) 98200-3932



/vheltai



@vheltai



/vheltai



@Vinicius Heltai



@vheltai