UFC - UNIVERSIDADE FEDERAL DO CEARÁ
Centro de Ciências - Física e Matemática
Departamento de Computação

FP - Fundamentos de Programação - 2024.1

Utilizando Variável *String* em *Python*

Professor: José Caminha Alencar Araripe Júnior
Fortaleza, 02 de setembro de 2024

Informações: Utilização de Variável "*string*" em *Python*.

PyCharmv

1. O que é uma variável do tipo *string*

É um tipo de variável utilizada em linguagem de programação para guardar as informações no formato de caractere, de texto, que podem ser compreendidos como representações de informações escritas dentro de um código.

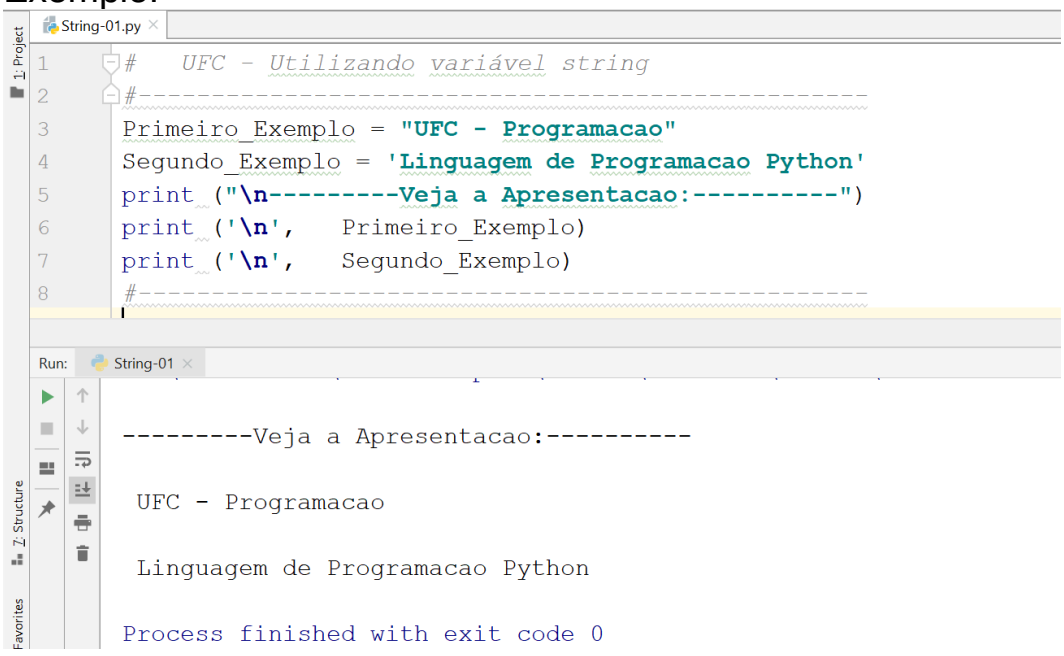
O *Python* tem uma classe de *string* interna chamada **str**.

Portanto, uma *string* é um tipo de dados usado para armazenar informações de texto (inclusive caracteres especiais).

São considerados tipos simples pois seus valores são compostos por uma sequência de caracteres, podendo conter números, letras, caracteres especiais e pontuações.

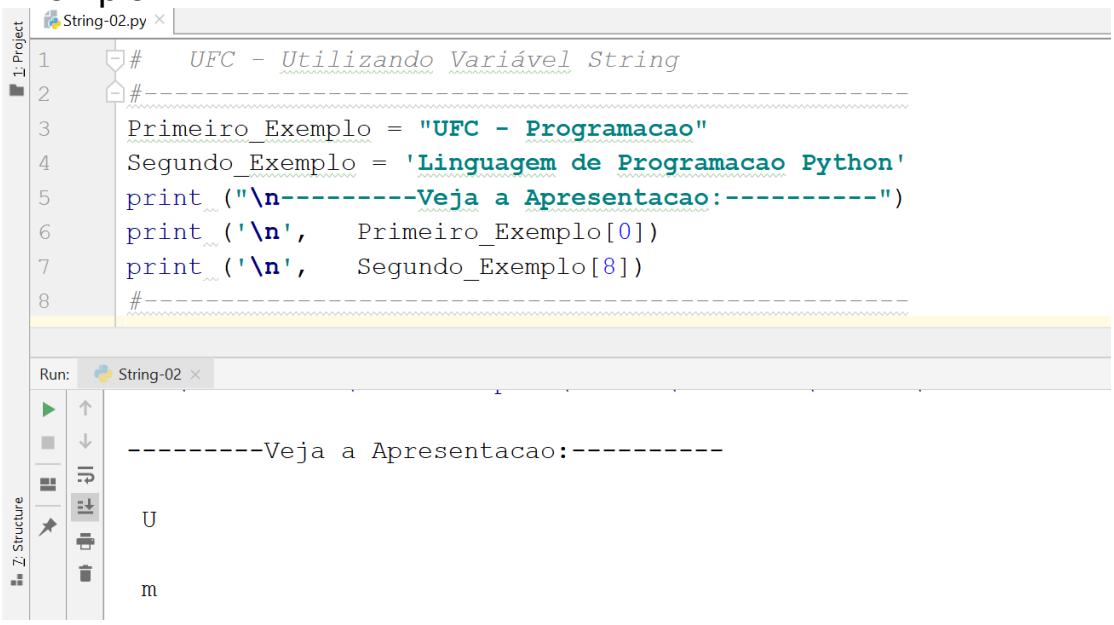
As ***strings*** em *Python* são sempre colocadas entre aspas simples ('...') ou aspas duplas ("...").

Exemplo:



Uma *String* é uma sequências de caracteres, uma sequência indexada onde podemos **acessar** qualquer caractere a partir de uma **posição desejada**. A primeira posição o valor do índice é igual a 0 (zero).

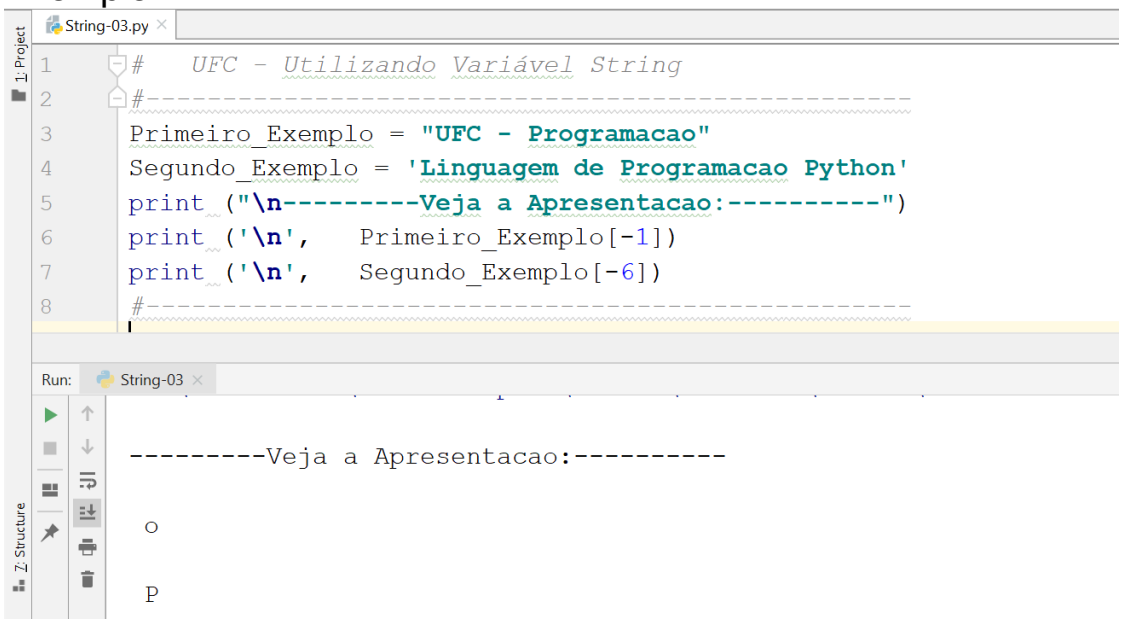
Exemplo:



Nesse exemplo serão impressos as letras **U** e **m**. O **U** é a informação que está na posição 0 (zero) da variável "*Primeiro_Exemplo*" e o **m** é informação que está na posiçao 8 (oito) da variável "*Segundo_Exemplo*".

Quando utilizamos índices com valor **negativo**, a relação indexada é considerada do fim para o início, ou seja, da última posição da *string* para a primeira posição.

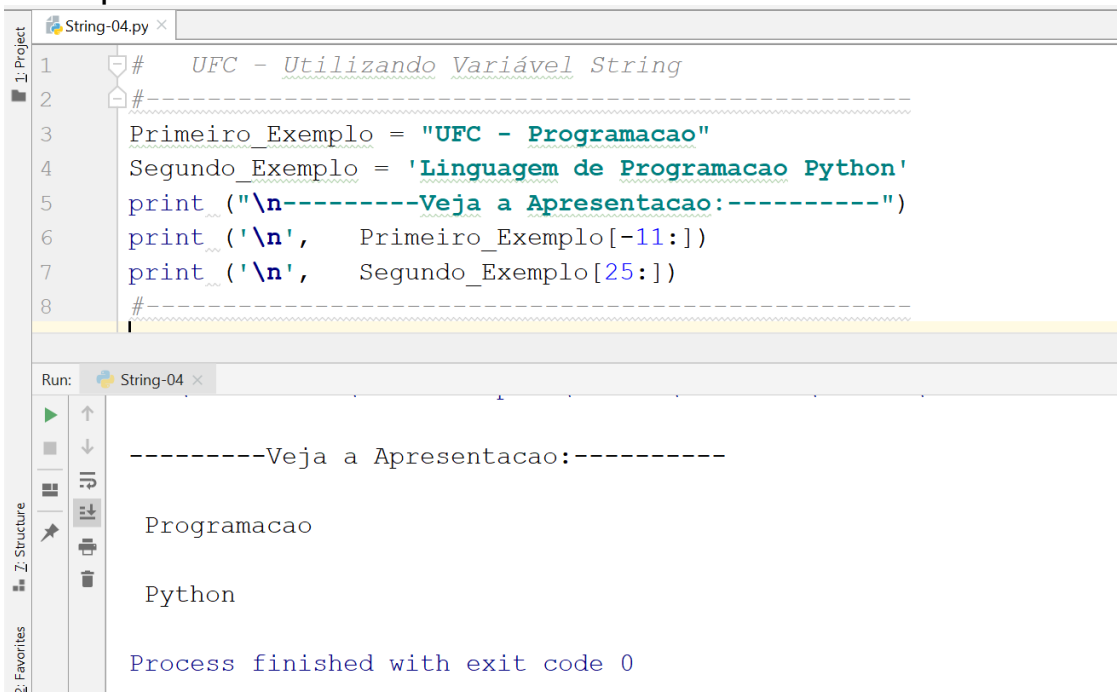
Exemplo:



Nesse exemplo foram impressos as letras **o** e **P**. O **o** é a informação que está na posição -1 (menos um), ou seja, a primeira posição da direita para a esquerda da variável *string* "***Primeiro_Exemplo***" e o **P** é a informação que está na posiçao -6 (menos seis), ou seja, a sexta posição da direita para a esquerda da variável *string* "***Segundo_Exemplo***".

Use dois pontos para obter os caracteres de uma posição informada até uma posição final da *string*:

Exemplo:



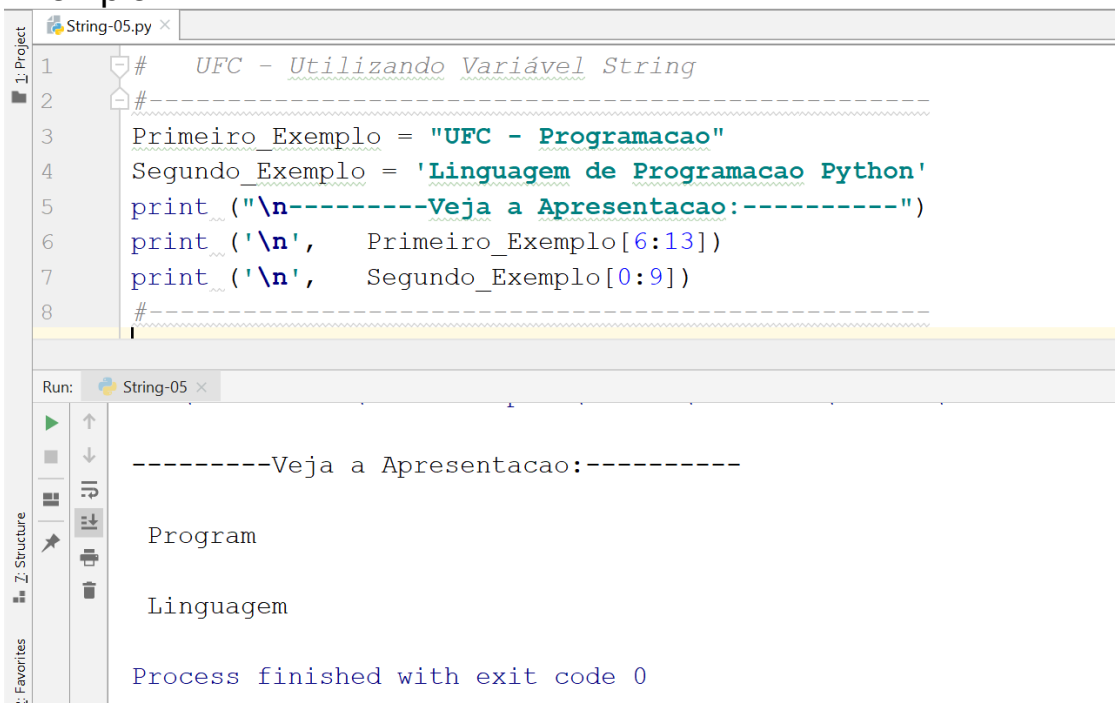
Nesse exemplo foram apresentados **Programacao** e **Python**.

O texto **Programacao** corresponde a informação que está nas 11 últimas posições, -11 (menos onze), ou seja, contempla 11 posições da direita para a esquerda da variável *string* "***Primeiro_Exemplo***".

O texto **Python** corresponde as informações contidas nas posições de 25 até a última posição da *string*, isso da esquerda para a direita da variável "***Segundo_Exemplo***".

Também podemos retornar um intervalo de caracteres, especificando um índice inicial e final, separados por dois pontos.

Exemplo:



Nesse exemplo foram apresentados **Program** e **Linguagem**.

O texto **Program** corresponde a informação que está a partir da posição 6 (seis) até a posição 13 (treze) da variável *string* "***Primeiro_Exemplo***", da esquerda para a direita.

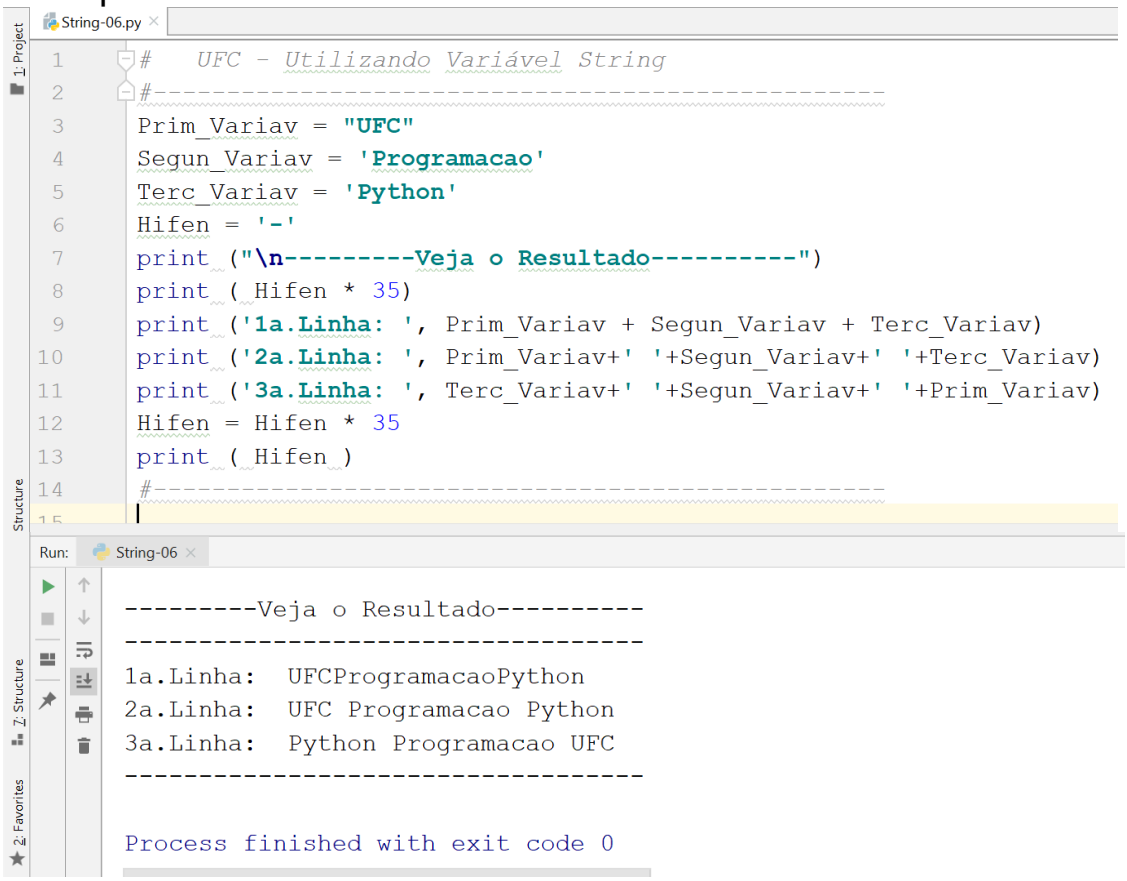
O texto **Linguagem** corresponde a informação que está a partir da posição 0 (zero) até a posição 9 (nove) da variável *string* "***Segundo_Exemplo***", da esquerda para a direita.

1.1 Operações com *strings*

Juntar *string* pode ser algo essencial e saber como realizar essa operação é muito importante!

Em *python* podemos concatenar *strings*, ou seja, juntar duas ou mais *strings* em uma única, utilizando o operador **+**, desta forma:

Exemplo:



Nesse exemplo foram utilizadas 3 (três) variáveis *string* conforme apresentado a seguir:

Prim_Variav = “UFC”
Segun_Variav = ‘Programacao’
Terc_Variav = ‘Python’

O exemplo também utiliza uma quarta variável de nome **Hifen** onde é guardado a informação “**-**“.

O programa imprime 6 (seis) linhas.

A segunda linha apresentada no resultado corresponde a execução da linha 8 (oito) do programa. O valor da variável *string* **Hifen** é multiplicado por 35, ou seja, a informação dessa variável é repetida por 35 vezes e corresponde a:

As linhas terceira até a quinta são apresentados os valores das variáveis **Prim_Variav**, **Segun_Variav** e **Terc_Variav** juntadas conforme o disposto nas linhas 9, 10 e 11 do programa. As informações dessas variáveis são concatenadas, juntadas, com a utilização da operação adição "**+**".

Na linha 12 do programa a informação contida no variável *string* de nome **Hifen**, que um "**-**", é repetida por 35 vezes e guardada na propria variável *string* **Hifen**.

O *Python* só permite a concatenação utilizando o operador + quando os operandos são *strings*.

1.2 Formatação de *Strings* utilizando format()

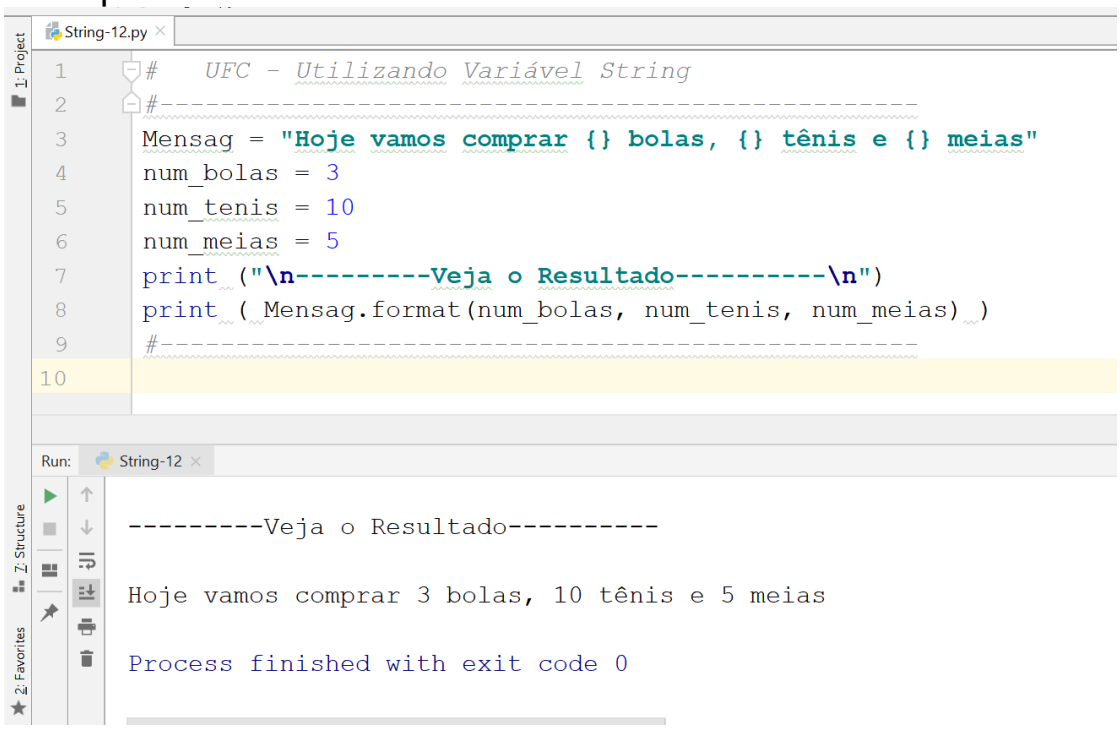
A formatação de string utilizando o **.format( )** pode ser realizado inserindo-se dentro da *string* os marcadores **{ }** (chaves) nas posições onde desejamos que os argumentos sejam inseridos.

Imagina que queremos utilizar o texto:

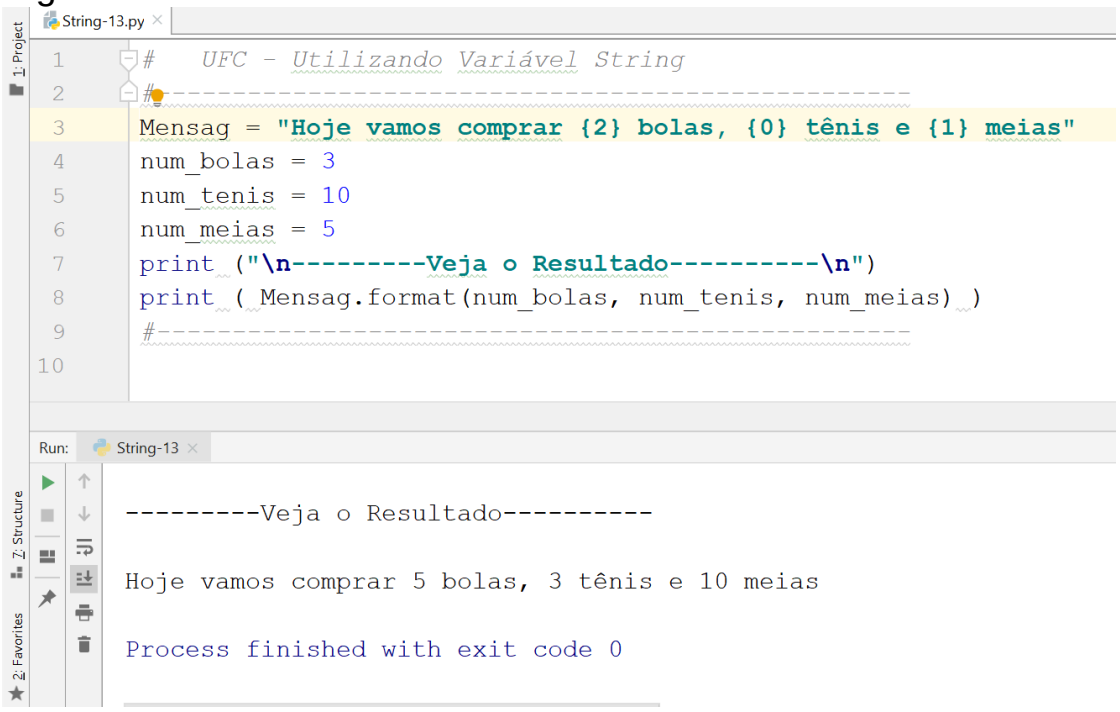
Mensag = “Hoje vamos comprar { } bolas, { } tênis e { } meias.”

A variável *string* de nome Mensag tem definido os espaços onde serão inseridas as informações quando da impressão.

Exemplo 1:



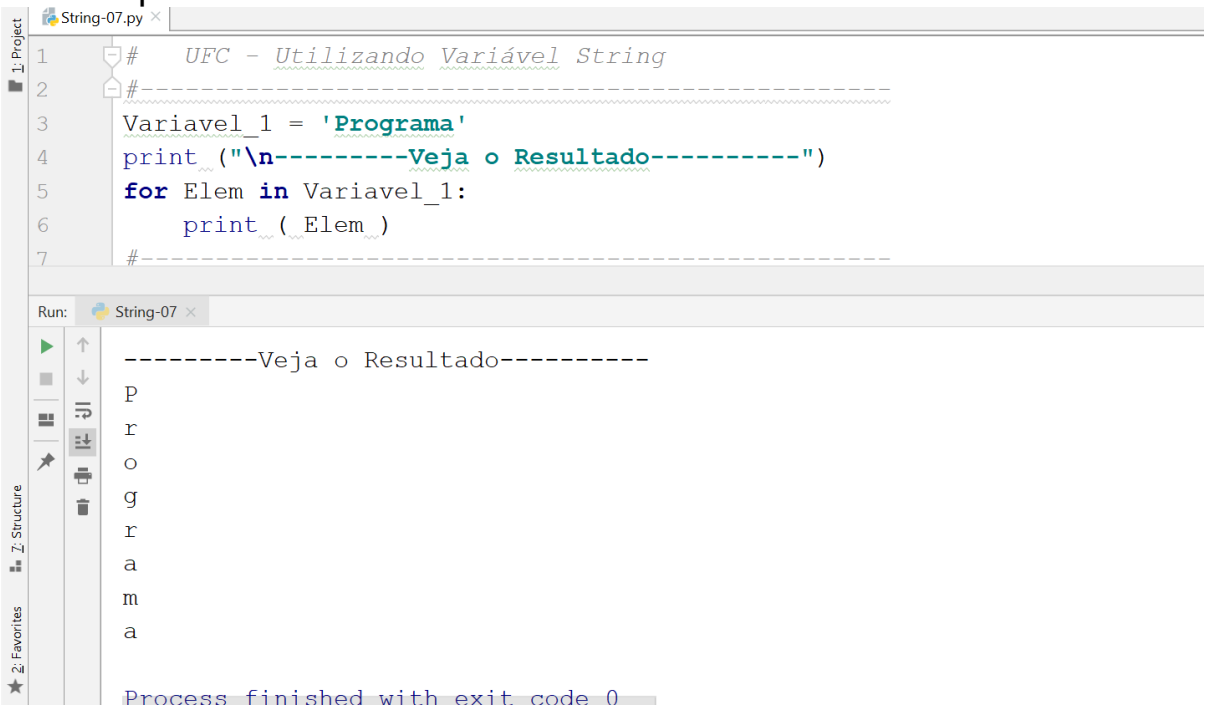
Exemplo 2: Também podemos definir as posições em que os argumentos serão inseridos:



1.3 *Strings* em Estruturas de repetição

Sabemos que as *strings* são iteráveis e por isso, podemos utilizar as estruturas de repetição **for** e **while** para percorrer seus itens, para percorrer cada uma das posições, desde a posição 0 (zero) até a última posição da *string*.

1º exemplo com o **for**:



2º exemplo com o **for**:

String-08.py

```
1  #    UFC - Utilizando Variável String
2  #-------------------------------------------------
3  Variavel_1 = 'Python'
4  NumElem = len( Variavel_1 )
5  print ("\n---------Veja o Resultado----------")
6  for I in range(NumElem):
7      print ( Variavel_1[I] )
8  #-------------------------------------------------
```

Run: String-08

```
---------Veja o Resultado----------
P
y
t
h
o
n

Process finished with exit code 0
```

3º exemplo com o **while**:

String-09.py

```
1   #   UFC - Utilizando Variável String
2   #---------------------------------------------------
3   I = 0
4   Variavel_1 = 'Computacao'
5   NumElem = len( Variavel_1 )
6   print ("\n---------Veja o Resultado----------")
7   while I < NumElem:
8       print ( Variavel_1[I] )
9       I += 1
10  #---------------------------------------------------
```

Run: String-09

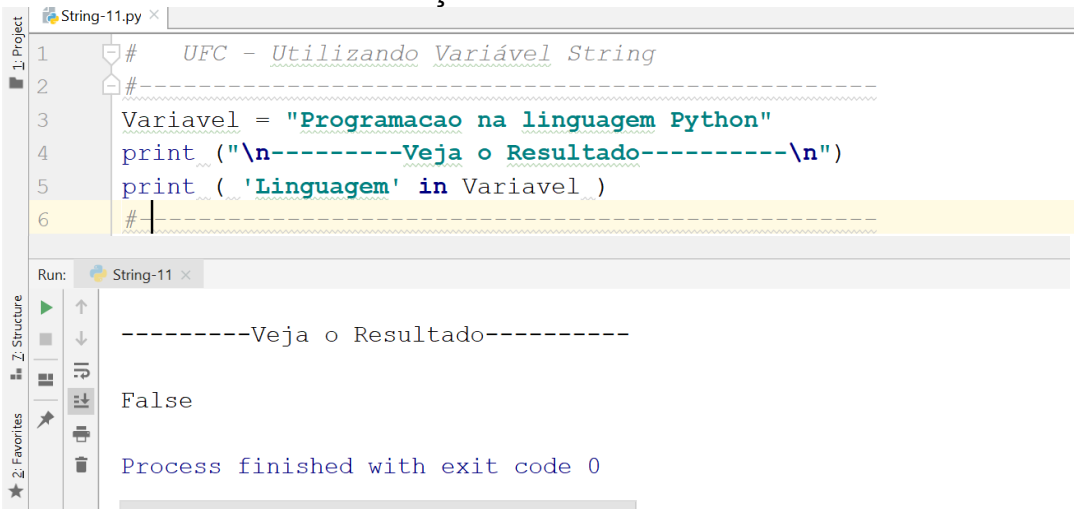
```
---------Veja o Resultado----------
C
o
m
p
u
t
a
c
a
o

Process finished with exit code 0
```

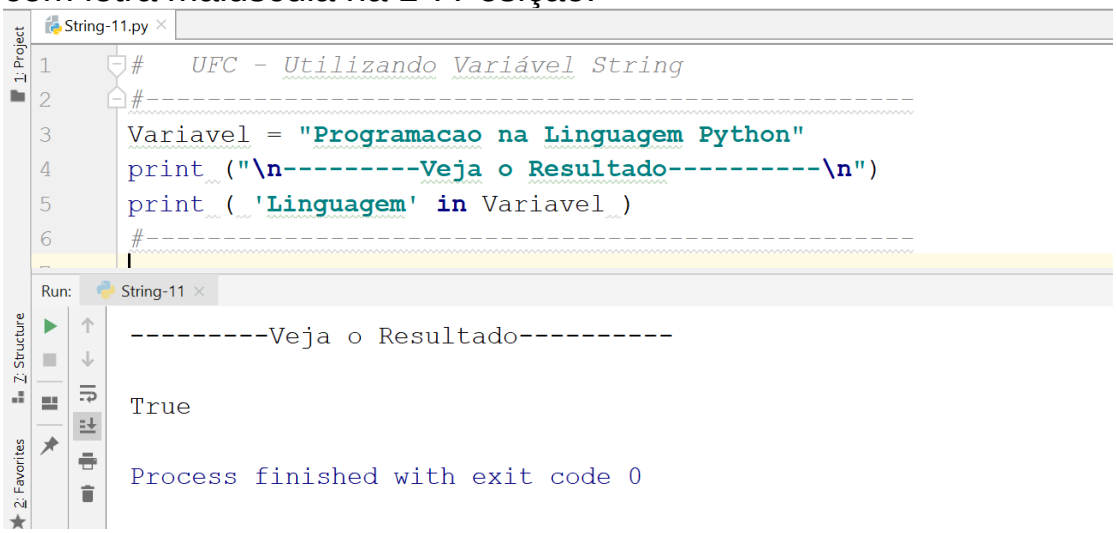
1.4 Verificar se uma *String* pertence à outra *String*

Para verificar se uma determinada sequência de caracteres existe em uma *string*, basta usar o operador in.

Exemplo 1: O resultado é "**False**" pois a palavra "**linguagem**" está com letra minúscula na 1ª. Posição.



Exemplo 2: O resultado é "**True**" pois a palavra "**Linguagem**" está com letra maiúscula na 1ª. Posição.



1.5 Caracteres de saída (Escape Characters)

Existem alguns caracteres que não podem ser inseridos em *strings*, pois geram algum tipo de erro, como por exemplo, tentar adicionar aspas duplas em uma *string* que foi criada com aspas duplas.

Para evitar o erro, é necessário utilizar um ***caractere de escape***, ou ***caracter de saída***, utilizado para gerar uma interpretação alternativa ao texto que foi definido.

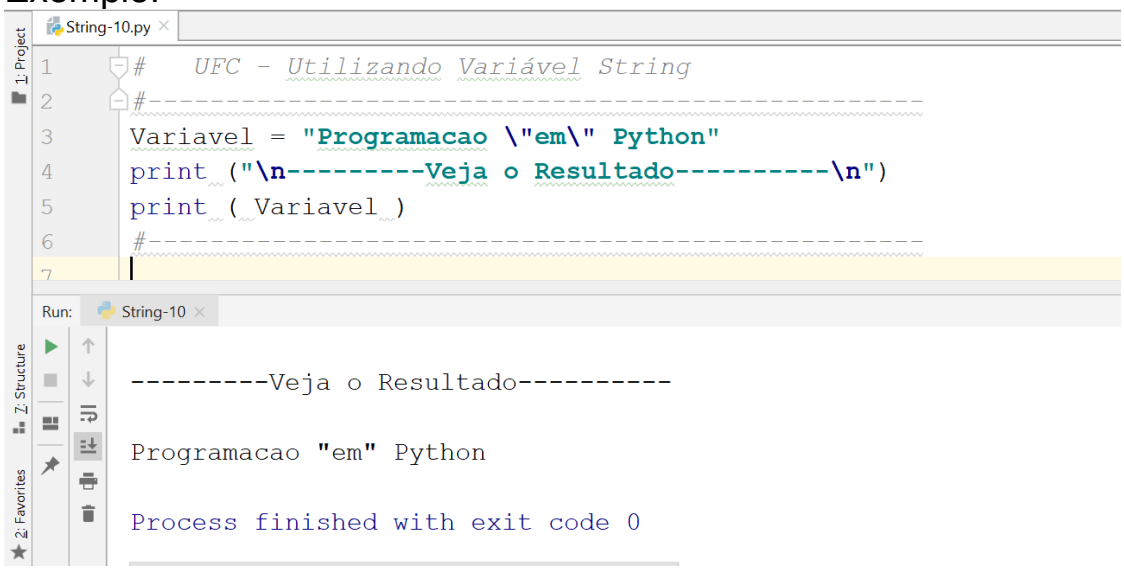
Em *Python*, eles são caracterizados pela barra inversa **** seguido por outro caractere.

Caso se queira a impressão da mensagem:

Programando "em" Python

A maneira correta para se conseguir que esse código tenha a mesma saída sem erro, será necessário usar o ***caractere de saída*** ****, a barra invertida, antes do caracter especial que se deseja inserir.

Exemplo:



Dessa forma, o *Python* vai entender que você quer realmente definir aspas no texto, e não fechar uma *string*.

1.6 Conclusão

Nesse texto você viu vários conceitos importantes quando estamos utilizando variável do tipo *strings* em *Python*.

Claro que existem outras informações sobre *strings* que poderão complementadas.

A seguir alguns métodos do *Python* para trabalhar com *strings*:

- **find(***subStringProcurada***)**
  - Retorna a posição do índice da primeira ocorrência da *subStringProcurada* na *String* sendo consultada. Caso não seja encontrada, retorna menor um (-1)
- **replace(** *subStringProcurada*, *subStringNova* **)**
  - Retorna uma cópia da *string* sendo consultada, substituindo
- **count(subStringProcurada)**
  - Retorna a quantidade de ocorrências
- **upper()**
  - Retorna uma cópia da *String*, convertendo tudo para maiúsculo
- **lower()**
  - retorna uma cópia da *String*, convertendo tudo para minúsculo
- **strip()**
  - Retorna uma cópia da *String*, removendo todos os caracteres brancos do início e do final
- **split()**
  - Retorna uma lista de todas as palavras *String*
- **split(** *subStringSeparadora* **)**
  - Retorna uma lista de todas as palavras *String*, sendo o delimitador procurado entre palavras aquele especificado em *subStringSeparadora*.

2. Métodos do *Python* para utilizar nas listas

A seguir alguns métodos do *Python* como:

Para adicionar ou excluir item na lista:

- **.append()**: adiciona o item ao final da lista;
- **.insert()**: insere um item na lista na posição indicada;
- **del**: remove um item da lista baseado na posição indicada;

  Ex: .....

  ```
  del lista_exemplo[ 2 ]
  ```

  .....

  Exclui o item da posição 2 da "*lista_exemplo*".

- **.remove()**: remove um item baseado no seu valor e não na sua posição;
- **.pop()**: remove o último item da **Lista;**
- **.sort()**: ordena os valores do maior para o menor;

  Ex: .....

  ```
  lista_exemplo.sort()
  ```

  .....

  Os dados da "*lista_exemplo*" serão colocados em ordem crescente, do menor para o maior.

- **+**: Unindo Listas, utilizado para unir duas listas, concatenar listas;

  Ex: .....

  ```
  lista_exemplo3 = list_exemplo1 + lista_exemplo2
  ```

  .....

  A "*lista_exemplo3*" terá os valores da "*lista_exemplo1*" acrescidos dos valores da "*lista_exemplo2*".

- ***:** Esse operador repete (replica) uma lista diversas vezes;
- **len()**: Essa função retorna a **quantidade de elementos** da lista, o comprimento de uma lista, ou o número de itens que a compõem;

  Ex: .....

  num_elementos = len(*nome_exemplo1*)

  .....

- **in**: Verifica se um "***valor***" existe nos itens de uma lista. Percorre todos os itens e retorna "*True*", se encontrar, e "*False*", se não encontrar;

  Ex: .....

  if 'vv' in lista_exemplo1:

  .....

- **min()**: Identifica o menor valor dentre todos os itens da lista;

  Ex: .....

  menor_valor = min(lista_exemplo1)

  .....

- **max()**: Identifica o maior valor dentre todos os itens da lista;

  Ex: .....

  maior_valor = max(lista_exemplo1)

  .....

- **sum()**: Realiza a soma de todos os itens da lista;

  Ex: .....

  somatorio = sum(lista_exemplo1)

  .....

- **count**: retorna o número de vezes em que um elemento aparece na lista;

  Ex: .....

  quant = lista_exemplo1.count(45)

  .....

- **reverse**: inverte a ordem dos elementos na lista.

  Ex: .....

  lista_exemplo1.reverse()

  .....

x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x