Aula 3

**Manipulando Strings**

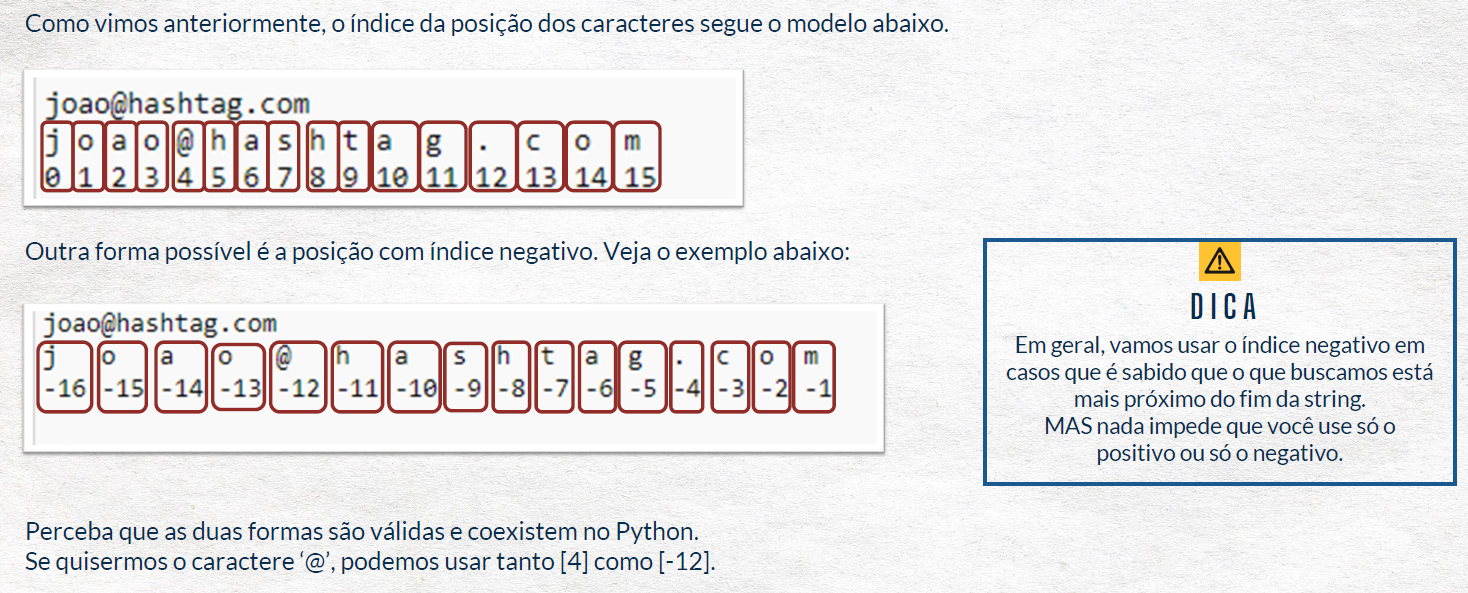
Cadeia de Caracteres == Strings

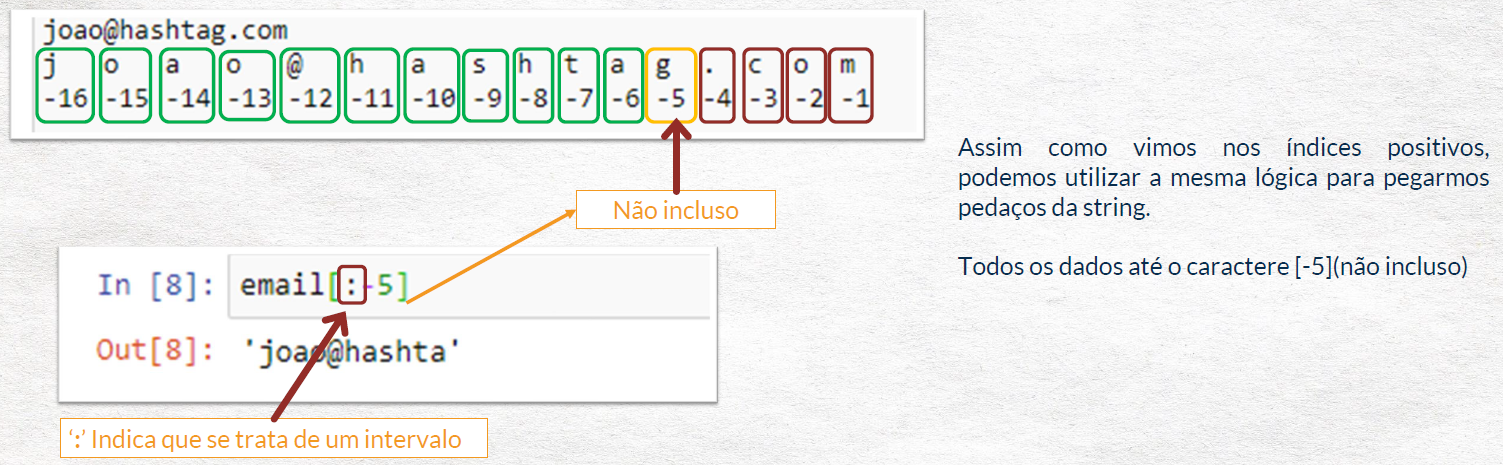
frase = “Mega adrenalina mode on”

* Lembreando que toda a cadeia de caracteres pode estar entre aspas simples e aspas duplas. Caso use 3 aspas dentro do print, normalmente serve para escrever textos longos na tela.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |

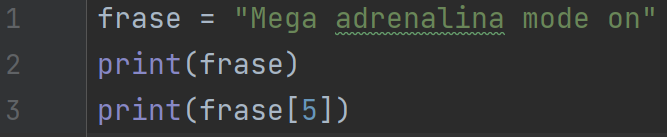
* Como ao colocar a nossa frase dentro de um espaço de memória nós enumeramos essese espaço de 0 até 22. Ou seja, nós temos 23 caracteres. Em outras palavras, colocamos cada caracter dentro de uma **lista**.
* **Spoiler**: **Listas** são imutáveis. Falaremos mais sobre isso em outro momento.



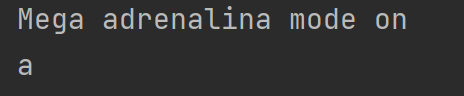


**Operações com Strings**

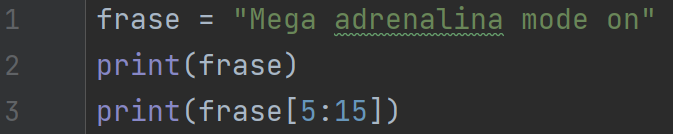
1. **Fatiamento:** 
   1. Pegando apenas um character:



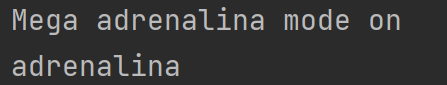
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |



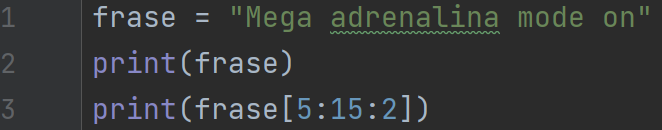
* 1. Pegando mais de um character em ordem:



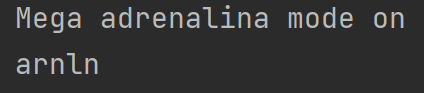
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |



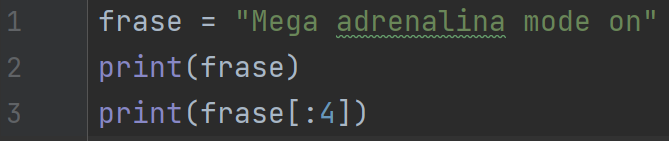
* Os dois valores server para varrer uma string;
* Notem que o segundo valor é, na verdade, um valor acima da última “casa” lida;
  1. Pegando mais de um caracter mas com um salto:



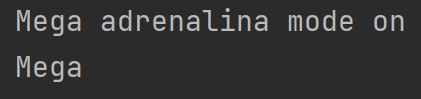
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |



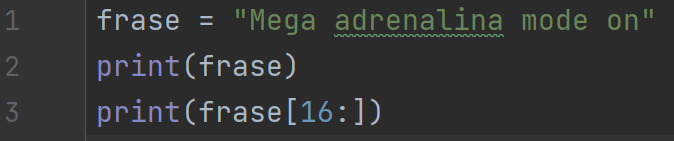
* O último valor nos informa o salto que queremos dar;
  1. Varrendo uma string do início até onde quisermos parar:



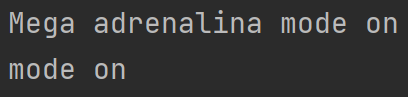
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |



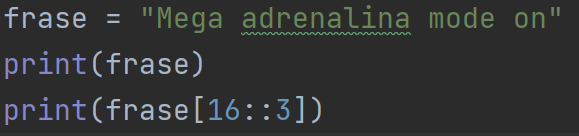
* Não informar o valor inicial, indica ao Python que, por padrão, ele deverá varrer a string do começo, ou seja, do zero, até o valor anterior ao informado. No nosso caso, de 0 até 3.
  1. Varrendo um string de um ponto até o final:



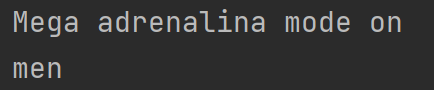
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |



* Informando o primeiro valor, mas não o último, o Python entenderá como ler essa string partindo do valor informado até o final.
  1. Varrendo uma string de um ponto até o final com salto:

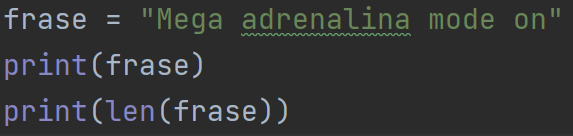


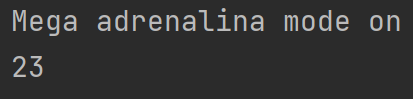
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |



1. **Análise:**

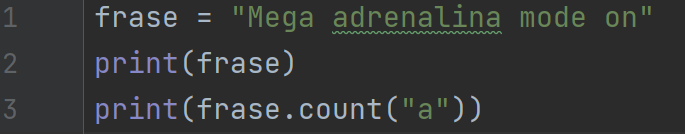
2.1) Comprimento de uma string:



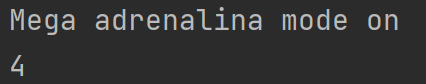


* Função: **len(<string>)**

2.2) Contagem direcionada:

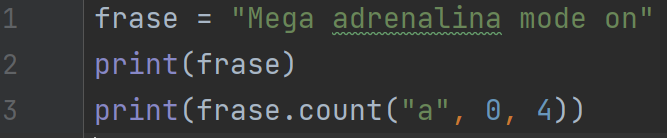


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |

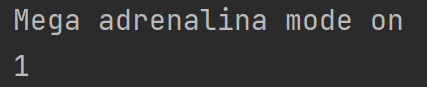


* Função: **<string>.count(“<caracter>”)**

2.3) Contagem direcionada com fatiamento:

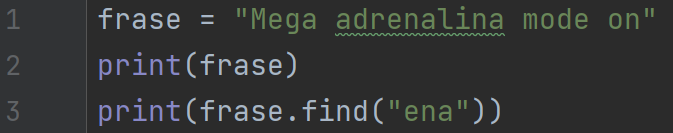


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |

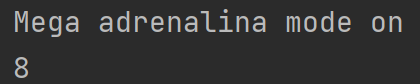


* Os parêmetros dentro da função serão trabalhados da mesma forma como anteriormente.
* Função: **<string>.count(“<caracter>”, <início>, <fim + 1>)**

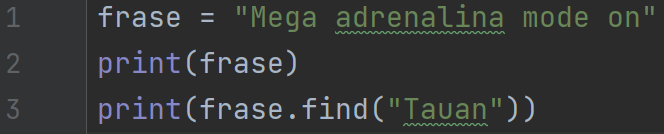
2.4) Busca direcionada:

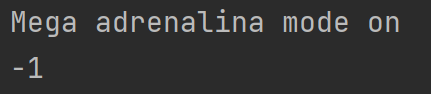


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |

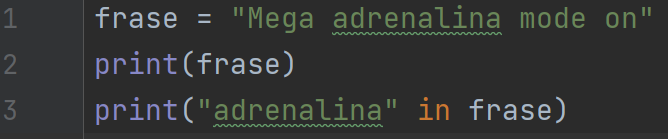


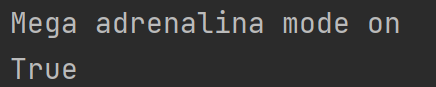
* Função: **<string>.find(“<pedaço>”)**
* Essa função retorna se ele encontrou ou não o trecho buscado E onde começou esse trecho.
* Se essa função retornar “-1”, isso quer dizer que o valor passado para função NÃO EXISTE dentro da string que estamos procurando.





2.5) Busca com “in”:

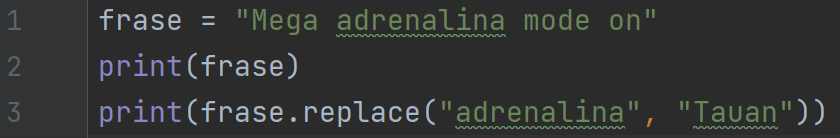


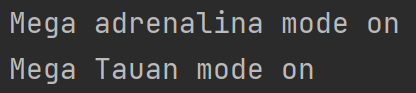


* Função: **“<pedaço>” in <string>**
* Essa função retorna True ou False. Apenas no informando se há ou não aquele trecho dentro da string que estamos analisando.

1. **Transformações:**

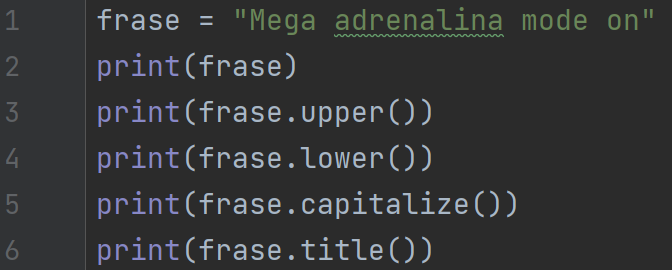
3.1) Substituição:

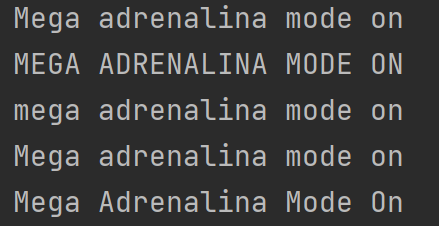




* Função: **<string>.replace(“<antigo>”, “<novo>”)**
* Essa função substitui a sequência de caracteres procurada, pela nova sequência escolhida. Essa realocação altera o tamanho da string.
* Se usarmos, por exemplo, frase = frase.replace(“adrenalina”, “Tauan”) o que acontecerá?

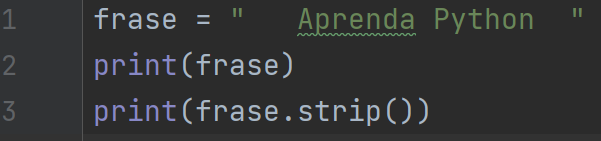
3.2) Métodos:



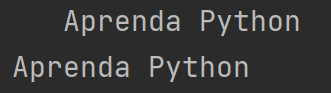


3.3) Removendo espaços inúteis:









* Remove os espaços vazios que o user possa vir a deixar, porém, não remove os espaços entre as palavras.
* Existe também algumas variações dessas funções que removem os espaços vazios a direita e a esquerda apenas.





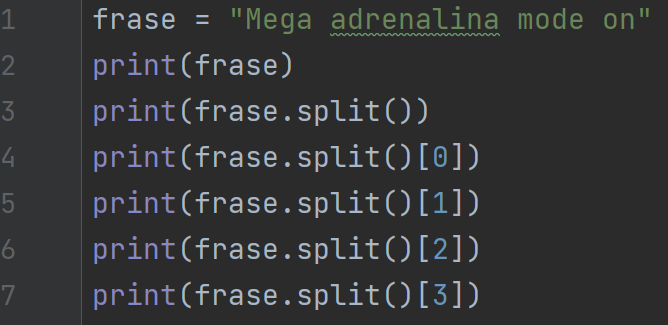


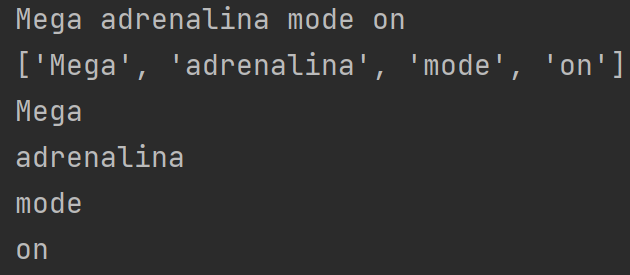


1. **Divisão:**

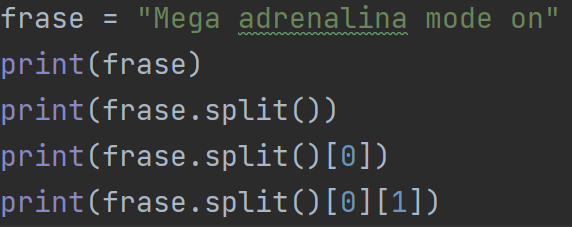
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | **e** | **g** | **a** |  | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** |  | **m** | **o** | **d** | **e** |  | **o** | **n** |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |

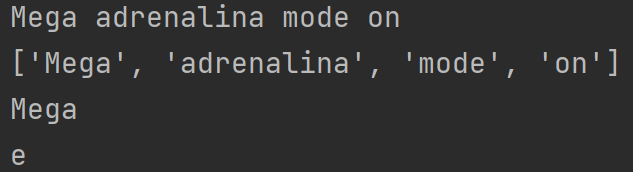
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **M** | **e** | **g** | **a** | | 0 | 1 | 2 | 3 | | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **a** | **d** | **r** | **e** | **n** | **a** | **l** | **i** | **n** | **a** | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | |  |  |  |  | | --- | --- | --- | --- | | **m** | **o** | **d** | **e** | | 0 | 1 | 2 | 3 | | |  |  | | --- | --- | | **o** | **n** | | 0 | 1 | |



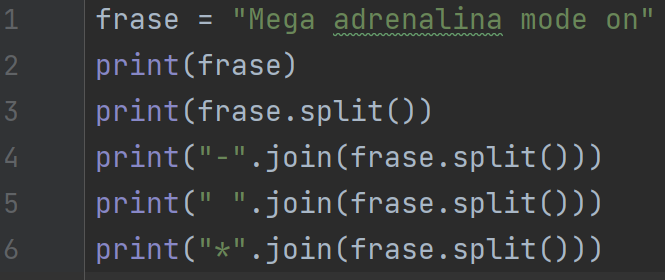


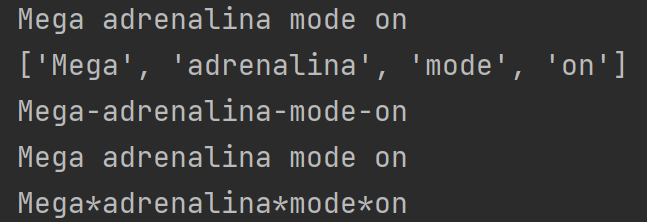
* Além de separar a frase em sublistas, nós também podemos acessar letras dentro das sublistas.





1. **Junção:**



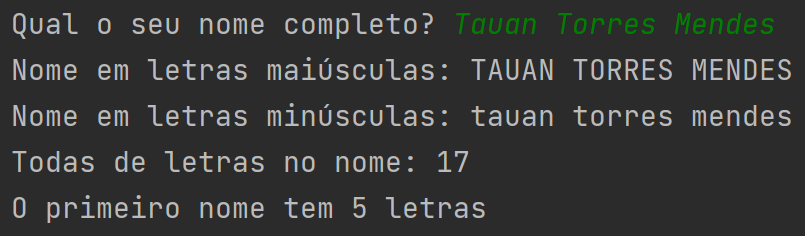


**Prática**

**[Exercício 1]** Escreva um programa que leia o nome completo de uma pessoa e mostre:

1. O nome com todas as letras maiúsculas;
2. O nome com todas as letras minúsculas;
3. Quantas letras ao todo (sem considerar os espaços);
4. Quantas letras tem o primeiro nome.

**Input/Output:**



**[Exercício 2]** Escreva um programa que leia um número entre 0 e 9999 e mostre cada um dos dígitos separados mostrando quantas unidades, dezenas, centenas e milhares há nesse número.

Ex:

Número = 8734

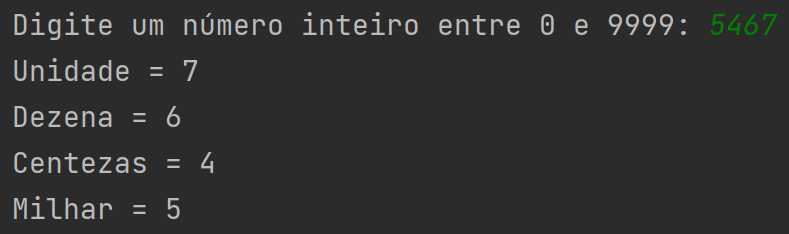
Milhares = 8

Centenas = 7

Dezenas = 3

Unidades = 3

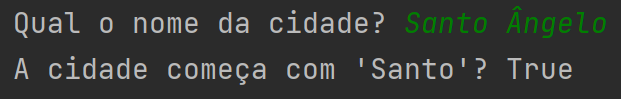
**Input/Output:**



P.S.: Essa questão pode ser resolvida usando apenas operações matemáticas ou strings

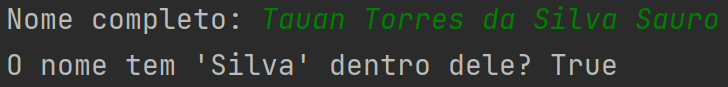
**[Exercício 3]** Escreva um programa que leia o nome de uma cidade e informe se ela começa com a palavra “Santo”.

**Input/Output:**



**[Exercício 4]** Escreva um programa que leia o nome completo de uma pessoa e diga se ela tem “Silva” no nome.

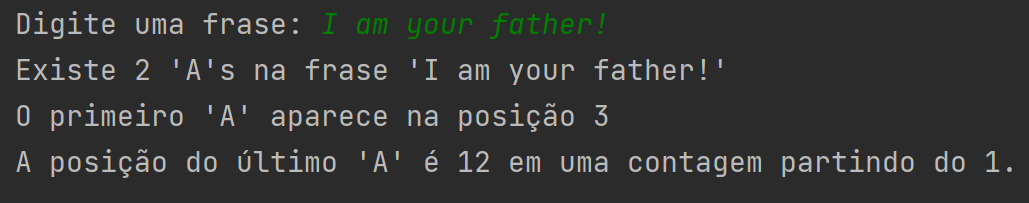
**Input/Output:**



**[Exercício 5]** Escreva um programa que leia um frase e mostre:

1. Quantas vezes aparece a letra “A”;
2. Em que posição ela aparece pela primeira vez;
3. Em que posição aparece pela última vez.

**Input/Output:**



**[Exercício 6]** Escreva um programa que leia o nome completo de uma pessoa e mostre o primeiro e o último nome separadamente.

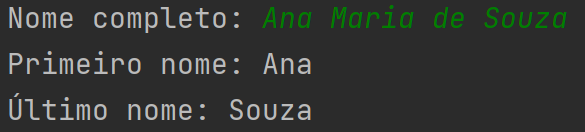
Ex:

Nome = Ana Maria de Souza

Primeiro = Ana

Último = Souza

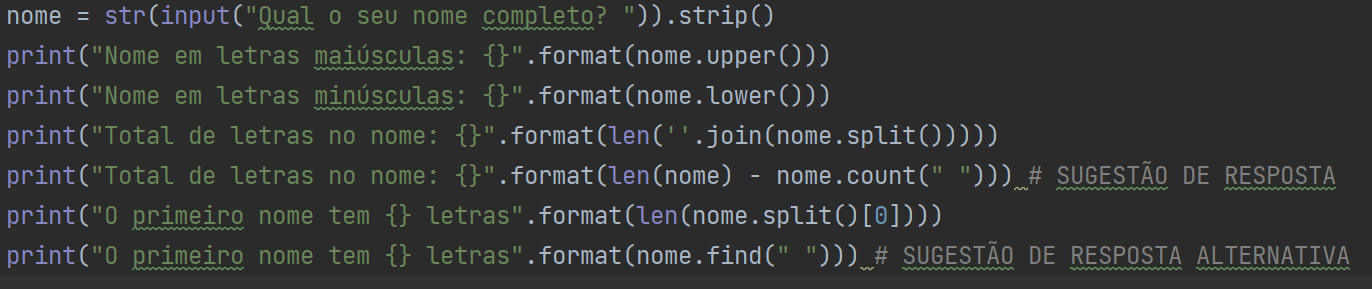
**Input/Output:**



**Respostas:**

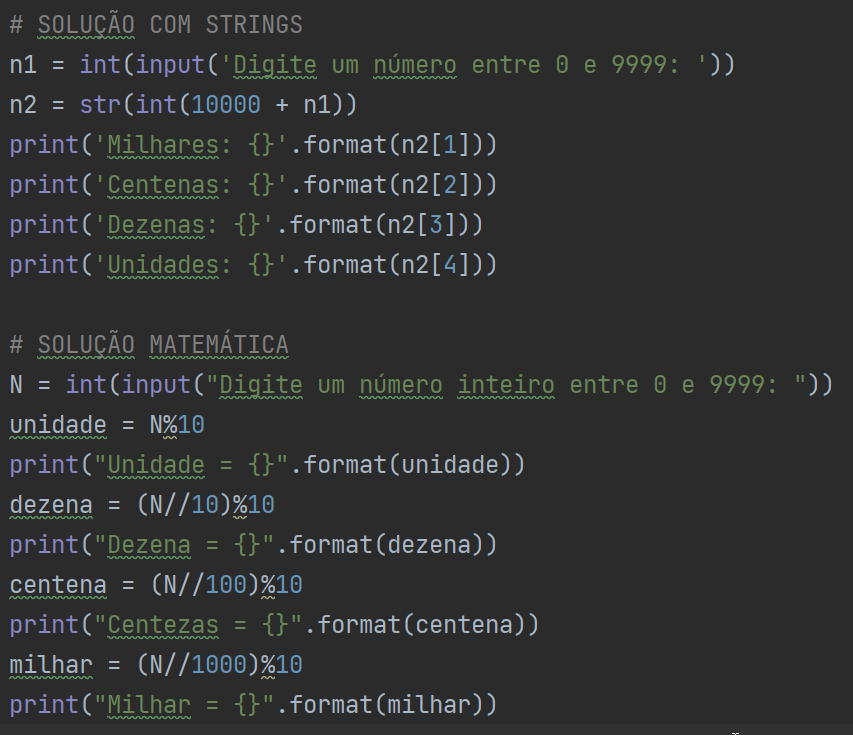
**[Exercício 1]**

**Resposta:**



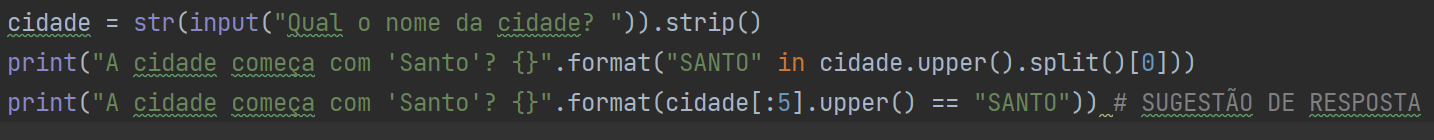
**[Exercício 2]**

**Resposta:**



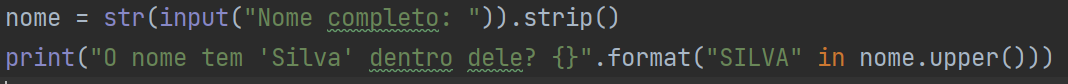
**[Exercício 3]**

**Resposta:**



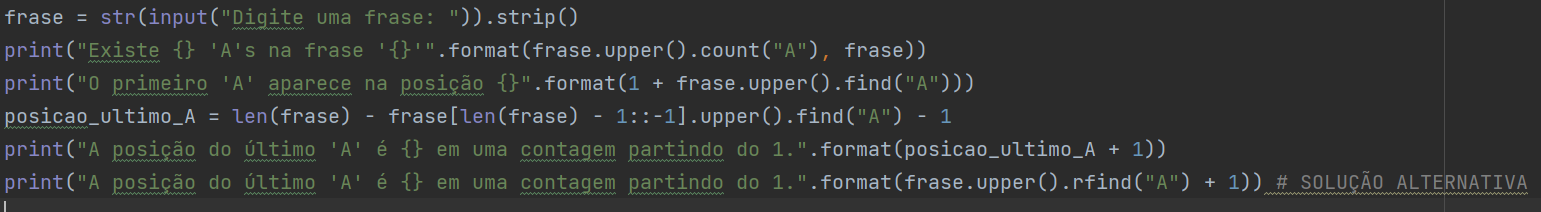
**[Exercício 4]**

**Resposta:**



**[Exercício 5]**

**Resposta:**



**[Exercício 6]**

**Resposta:**

