[GIM] Curso de Férias - Python

Luiz Eduardo Barros e Victor Matheus Castro

Conteúdo do Dia

- Funções Recursivas
- Strings
- Manipulação de Arquivos

Recursão e Funções Recursivas

- Uma função recursiva é uma função que, em seu código, invoca a si mesma;
- A ideia central da recursão é utilizar resultados anteriores ou precedentes para se chegar a um novo resultado.

Função Fatorial

- Matematicamente, a função Fatorial é definida da seguinte maneira:

$$n! = 1, \text{ se } n = 0 \text{ ou } n = 1$$
$$n! = n(n\text{-}1) \text{ se } n > 1$$

- Poderíamos criar uma função fat() que recebe um parâmetro n;
- Temos de testar qual o valor de n para retornar o resultado correto:

$$n = 0 \text{ ou } n = 1, \text{ fat}(n) = 1$$
$$\text{se } n > 1, \text{ fat}(n) = n*\text{fat}(n\text{-}1)$$

Recursão

- Note que, na definição da função fatorial, verificamos duas condições:

  * A primeira condição é chamada de **caso base** da recursão: é uma condição que fará a recursão parar de se "aprofundar";

  * A segunda condição é chamada **caso recursivo**

- Funções recursivas podem ter mais de um caso base e mais de um caso recursivo;
- **IMPORTANTE:** é de extrema importância garantir que em algum momento a função pare de fazer novas chamadas recursivas, caso contrário ela se prolongará indefinidamente!

Strings

- É um tipo usado para representar palavras/textos;
- Acesso de valores por indexação (similar às listas);
- O índice da primeira letra inicia em 0;
- Índice negativo: o último símbolo de uma string tem índice -1, o penúltimo -2...;
- No fatiamento nome[x:y], selecionamos os símbolos do x ao y-1;
- A conversão para string pode ser feita usando str(variavel).

Operadores

Nome	Operador	Descrição
Concatenação	+	Gera uma nova string com os elementos das strings usadas como argumento
Repetição	*	Gera uma nova string que é a repetição do argumento n vezes seguidas
Comparações	$>, <, >=, <=, ==$	Compara lexicograficamente as strings

Substituição em Strings: o operador %

Operador	Descrição	Exemplo
%d	Substitui um valor inteiro dentro de uma string	a = "Tenho %d anos"
%f	Substitui um valor em ponto flutuante dentro de uma string	a = "Peso %f kg"
%s	Substitui uma string dentro de outra string	a = "Me chamo %s"

Substituição em Strings: o operador %

- Formatação de tipos: **m.n**
  - **m**: indica a quantidade de caracteres reservados;
    - **para floats**:
      - **n**: indica o número em casas decimais;
    - **para inteiros**:
      - **n**: indica o tamanho total do número, preenchido com zeros à esquerda.

Manipulações e Caracteres Especiais

- Updating Strings (criando novas Strings a partir de outras já existentes).

```
>>> nome = "Victor"
>>> nome[:6] + "ta cansado"
'Victor ta cansado'
```

- Caracteres não impressos (funcionam de maneira especial).

Nome	Operador	Descrição	Exemplo
Nova linha	\n	Acrescenta uma nova linha	"cet\ncet"
Tab	\t	Acrescenta um tab	"\tcet"
Tab vertical	\v	Acrescenta um tab verticla	"\vcet"

Métodos de str:

Nome	Descrição	Exemplo
capitalize	Primeiro caracter maiusculo	"cet".capitalize()
title	Todas as palavras em CamelCase	"cet manha".title()
center	Retorna a string centrada na largura passada por argumento	"cet".center(40)
count	Retorna o numero de ocorrencias de uma substring	"cet noite".count("te")
expandtabs	\t por um número de espaços dado por argumento	"cet\t".expandtabs(3)
find	Retorna o menor index onde a substring ocorre	"cet".find("e")

Nome	Descrição	Exemplo
Index	Igual ao find, porém retorna erro se não existir a substring	"tecnologia".index("te")
É alfanumérico	Retorna verdadeiro caso todos os caracteres sejam alfanuméricos	"ect123".isalnum()
É lower	Retorna verdadeiro caso todos os caracteres sejam lower case	"ecT".islower()
split	Separa uma string em várias substrings conforme um parâmetro	s.split()
Está/não Está contido	Verifica se o caracter 'x' está na string	'i' in "luiz"
Comprimento	Retorna o comprimento da string	len("ect")

Strings: imutáveis

- É importante ressaltar que não é possível mudar o valor de uma parte da string, ao invés disso, é recomendável criar novas strings a partir de anteriores.

```
>>> exemplo = "rio grande do norte"
>>> exemplo2 = 'p' + exemplo[1:]
>>> exemplo2
'pio grande do norte'
```

String: while e for

- Podemos utilizar laços para percorrer caracteres de uma String.

```
>>> exemplo = "universidade"
>>> i = 0
>>> while i < len(exemplo):
>>>         letra = exemplo[i]
>>>         print letra
>>>         i += 1
```

- Podemos fazer o mesmo utilizando um laço FOR:

```
>>> for letra in exemplo:
>>>         print letra
```

O Módulo String

- Possui um conjunto de constantes e funções úteis para manipular as strings;
- Utilização: "import string"

Constantes do Módulo String

Nome	Definição
string.lowercase	Contém todas as letras minúsculas
string.uppercase	Contém todas as letras maiúsculas
string.digits	Contém todos os dígitos
string.hexdigits	Contém todos os dígitos em hexadecimais

Manipulação de Arquivos

- Para manipular arquivos utilizamos a função open, na seguinte estrutura:

```
>>> variavel = open("file", "modo")
```

Onde:

variavel Nome da variável que receberá o conteúdo da função open (receber o arquivo propriamente dito).

file Nome do arquivo que você quer ler ou escrever.

modo Indica o que você quer fazer com o arquivo, seja ler "r"(read) ou escrever "w"(write).

Métodos de manipulação

Descrição	Exemplo
Retorna uma string única com todo o conteúdo do arquivo	arquivo.read()
Retorna a próxima linha do arquivo, e incrementa a posição atual	arquivo.readline()
Retorna todo o conteúdo do arquivo em uma lista, uma linha do arquivo por elemento da lista	arquivo.readlines()
Escreve a string data para o arquivo, na posição atual ou ao final, dependendo do modo.	arquivo.write(data)
Muda a posição atual do arquivo para o valor indicado em n	arquivo.seek(n)
Fecha o arquivo. Sempre utilizado no final da manipulação do arquivo	arquivo.close()

Escrevendo em um arquivo

```
>>> arquivo.write("Este texto será inserido no novo arquivo que acabamos de criar através do método write do objeto arquivo que acabamos de criar ")
>>>arquivo.close()
```

Lendo em um arquivo

```
>>> conteudo = arquivo.read()
>>> print conteudo
```