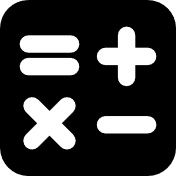
ssssss

**EXPRESSÕES, OPERADORES E ORDEM DE PRECEDÊNCIA**

Segundo Lopes e Garcia [1] uma expressão é um conjunto de variáveis e constantes numéricas que se relacionam por meio de operadores, compondo uma fórmula que, uma vez avaliada, resulta em um valor.

Para montagem de expressões precisamos conhecer os possíveis operadores, são eles:

* Operadores aritméticos;
* Operadores relacionais;
* Operadores lógicos.

O Quadro 1 apresenta os operadores aritméticos e sua ordem de precedência. Porém, deve-se salientar que o usuário no momento da escrita de uma expressão ainda deverá levar em consideração os parênteses internos como os primeiros na ordem de precedência global de uma expressão aritmética.

|  |
| --- |
| Quadro 1 – Operadores aritméticos. |
| |  |  |  | | --- | --- | --- | | **Operador em Python 3** | **Função** | **Hierarquia** | | **+** | Adição | **3º** | | **-** | Subtração | | **\*** | Multiplicação | **2º** | | **/** | Divisão | | **\*\*** | Exponenciação | **1º** | | **sqrt** | Radiciação | | **%** | Resto da divisão | | **//** | Quociente da divisão | |

Para exemplificar os níveis de hierarquia do computador segue um exemplo simples:

|  |
| --- |
| **>>>** X = 8 + 9 \* 2  26 |

Verificamos que nesse exemplo o nível de hierarquia 2° (multiplicação e divisão) será executado antes da adição.

Outro exemplo:

|  |
| --- |
| **>>>** X = 4 / 2 \* 1 + 3  5 |

O Quadro 2 e Quadro 3 apresentam os operadores relacionais e lógicos, respectivamente.

|  |
| --- |
| Quadro 2 – Operadores relacionais. |
| |  |  | | --- | --- | | **Operador em Python 3** | **Função** | | **==** | Igual | | **>** | Maior que | | **<** | Menor que | | **>=** | Maior igual | | **<=** | Menor irgual | | **!=** | Diferente | |

|  |
| --- |
| Quadro 3 – Operadores lógicos. |
| |  |  |  | | --- | --- | --- | | **Operador em Python 3** | **Função** | **Hierarquia** | | **and** | Conjunção | **2º** | | **or** | Disjunção | **3º** | | **not** | Negação | **1º** | |

Dentro das variáveis lógicas é possível construir a chamada Tabela-Verdade, que é o conjunto de todas as possibilidades combinatórias entre valores de diversas variáveis lógicas, as quais encontram em apenas duas condições (V ou F), e um conjunto de operadores lógicos.

|  |
| --- |
| Quadro 4 – Tabela-verdade operação de negação. |
| |  |  | | --- | --- | | **A** | **not A** | | F | V | | V | F | |

|  |
| --- |
| Quadro 5 – Tabela-verdade operação de conjunção. |
| |  |  |  | | --- | --- | --- | | **A** | **B** | **A and B** | | F | F | **F** | | F | V | **F** | | V | F | **F** | | V | V | **V** | |

|  |
| --- |
| Quadro 6 – Tabela-verdade operação de disjunção não-exclusiva. |
| |  |  |  | | --- | --- | --- | | **A** | **B** | **A or B** | | F | F | **F** | | F | V | **V** | | V | F | **V** | | V | V | **V** | |

A ordem de precedência global dos operadores é dada no Quadro 7.

|  |
| --- |
| Quadro 7 – Precedência dos operadores. |
| |  |  | | --- | --- | | **Hierarquia** | **Operadores** | | **1º** | Parênteses mais internos | | **2º** | Operadores aritméticos | | **3º** | Operadores relacionais | | **4º** | Operadores lógicos | |

Além dos operadores é possível citar os comandos de entrada, atribuição e saída para expressões. No Python 3 o comando de atribuição é dado pelo símbolo *=* e em relação a saída o principal comando é a função *print(var)* que poderá ser empregada para escrever todos os tipos de variáveis. Exemplos podem ser vistos a seguir:

|  |
| --- |
| **>>>** H = 50  **>>> print**(“tipo da variável”, **type**(H), ”valor = ”, H)  tipo da variável <class 'int'> valor = 50 |

No Python 3 os valores entre *“ “* representam os caracteres, já a função *type( )* está relacionada ao tipo da variável conforme descrito anteriormente.

Os comandos de entrada basicamente se resumem aos comandos leia ou escreva ([escrita em português estruturado](https://pt.wikipedia.org/wiki/Portugol)). Em Python 3 comandos como o *input( )*, *open( )* e *read( )* são bastante empregados para entrada de variáveis.

**Exercício 1.1:** Utilizando o conceito de tabela verdade construa um algoritmo que permita classificar o tipo de protensão necessário em uma viga a partir de informações da sua classe de agressividade e tipo de protensão. Empregue a tabela 13.4 da NBR 6118 para a aplicação.

**Exercício 1.2 [2]:** Utilizando os conceitos anteriores “calcular” as expressões abaixo:

1. ;
2. ;
3. ;
4. ;
5. ;
6. ;

**REFERÊNCIAS**

[1] Lopes A, Garcia G. Introdução à programação: 500 algoritmos resolvidos. Rio de Janeiro (RJ): Campus; 2002.

[2] Forbellone ALV, Eberspächer HF. Lógica de programação: a construção de algoritmos e estruturas de dados. São Paulo: Pearson Prentice Hall; 2007.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente