

ALGORITMO

Wesley Spalenza

IFES – Campus Cariacica, ES, Brasil



Introdução e Variáveis

--- Tipos de DADOS ---

Os dados ou variáveis são elementos do mundo exterior, que representam dentro de um computador, ao digitamos ou declararmos as informações manipuladas pelos seres humanos. É alocado (reservado) na memória do computador.

Eles podem ser classificados em três tipos primitivos ou tipos básicos:

- Numéricos (representados por valores numéricos **inteiros** ou **reais**).
- Caracteres (representados por valores **alfabéticos** ou **alfanuméricos**)
- Lógicos (valores dos tipos **falso** e **verdadeiro**).

- **Dados do Tipo Inteiro**

Os dados numéricos positivos e negativos pertencem ao conjunto de números inteiros, excluindo qualquer valor numérico fracionário (que pertence ao conjunto de números reais), por exemplo, os valores 35, 0, 234, -56, -9, entre outros.

Apresentaremos primeiro a representação em português estruturado (Portugol) com o comando ***inteiro***

No Fortran 90 e 95 (2003) é em outros compiladores ***integer***.

No ***Python***, não é necessária a definição/declaração das variáveis, mas elas existem.

- **Dados do Tipo Real**

São reais os dados numéricos positivos e negativos que pertencem ao conjunto de números reais. Incluindo todos os valores fracionários e inteiros, por exemplo, os valores 35, -56, -9, -45.999, 4.5, 3.141592... entre outros.

A apresentação em Portugal é: ***real***.

No Fortran 90 e 95 (2003) e demais compiladores são em inglês: ***real***

*No Fortran ainda existe o tipo de variável complexa- **complex***

- **Dados do Tipo Caractere**

São caracteres delimitados pelos símbolos aspas (“ ... “). Eles são representados por letras (de A até Z), números (de 0 até 9), símbolos (por exemplo, todos os símbolos imprimíveis existentes num teclado) ou palavras contendo esses símbolos.

A apresentação em Portugal é ***caractere***.

No Fortran 90 e 95 (2003) é ***character***

- **Dados do Tipo Lógico**

São lógicos os dados com valores binários do tipo sim e não, verdadeiro e falso, 1 e 0, entre outros, em que apenas um dos valores pode ser escolhido.

O tipo de dado lógico é também conhecido como booleano, devido à contribuição do filósofo e matemático inglês George Boole à área de lógica matemática e à eletrônica digital.

A apresentação em Portugol é ***lógico***.

No Fortran 90 e 95 (2003) é ***logical***

Aritmética Computacional

- Os operadores aritméticos são responsáveis pelas operações matemáticas a serem realizadas em um computador.
- O termo operador é utilizado na área de programação para estabelecer as ferramentas responsáveis por executar algum tipo de ação computacional.
- Os operadores aritméticos são responsáveis pela execução do processamento matemático, exceto o operador de atribuição que pode ser usado também em ações de processamento lógico.
- Os operadores aritméticos são classificados em duas categorias, sendo **binários** ou **unários**.
- São **binários** quando utilizados em operações matemáticas de radiciação, exponenciação, divisão, multiplicação, adição e subtração;
- São **unários** quando atuam na inversão do estado de um valor numérico, que pode ser passado de positivo para negativo ou vice-versa.

Tabela de operadores aritméticos

Operador	Operação	Descrição	Tipo	Prioridade	Resultado
+	" + n " ou " n "	Manutenção de sinal	Unário	-	Positivo
-	- n	Inversão de sinal	Unário	-	Negativo
←	x ← n	Atribuição do valor " n " a " x "	Binário	-	Positivo ou Negativo
↑	x ↑ n	Exponenciação de x ⁿ	Unário	1	Inteiro ou Real ¹⁵
↑ (1 / n)	x ↑ (1 / n)	Radiciação de ⁿ √x	Unário	1	Real
/	x / n	Divisão de " x " por " n "	Binário	2	Real
*	x * n	Multiplicação de " x " por " n "	Binário	2	Inteiro ou Real
+	x + n	Adição de " x " com " n "	Binário	3	Inteiro ou Real
-	x - n	Subtração de " n " de " x "	Binário	3	Inteiro ou Real
div	x div n	Divisão de " x " por " n "	Binário	4	Inteiro

Por que PYTHON?

- Software livre e com uma ampla comunidade de colaboradores e desenvolvedores, graças Python Foundation: www.python.org
- Adaptável a qualquer arquitetura de computador: Linux, Windows, Mac OS X, FreeBSD, Android.
- Interessante como primeira linguagem devido a simplicidade.
- Administra e desenvolve interfaces de pequenos e grandes projetos.
- Possui um grande número de pacotes/bibliotecas para importação como: Math, Numpy, Sympy, pandas, etc.

- Python vem crescendo em várias áreas como: física teórica e experimental, Matemática, Química, Engenharias, Biotecnologias, animações, aplicativos móveis, jogos, plataforma web, inteligência artificial, ciência de dados, etc.
- Legibilidade e simplicidade na sintaxe de códigos/scripts, não precisando de “”, “”, “:”, “{ }”, etc , no início e fim de blocos.
- Linguagem de rápida compilação e boa velocidade de comunicação entre bibliotecas e banco de dados

Instalação

Existem várias formas, ambientes e Sistemas Operacionais para usar-se o python.

- **Windows:**

- - Ambiente IDLE / Shell : www.python.org
- - Anaconda / jupyter: www.anaconda.org, <https://jupyter.org/>
- - Google Colab: <https://colab.research.google.com/>
- - Pycharm: <https://www.jetbrains.com/pt-br/pycharm/download/>

Detalhamentos a seguir...

- **IDE (Integrated Development Environment) ou Ambiente de Desenvolvimento Integrado,**

é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo, em geral é um EDITOR para qualquer tipo de compilador. Edita o código fonte do programa escrito na(s) linguagem(ns) suportada(s) pela IDE.

- **Exemplos**

- [IntelliJ IDEA](#) - IDE da JetBrains para desenvolvimento em diversas linguagens, principalmente JAVA. Também fornece recursos para produtividade de desenvolvimento em tecnologias web como HTML5, AngularJS, NodeJS, GWT entre outros;
- [Android Studio](#) - IDE oficial da Google para desenvolvimento na plataforma Android;
- Arduino - IDE para microcontroladores linguagem wiring com bibliotecas em C.
- [Delphi](#) - Trabalha originalmente com a linguagem [Object Pascal](#) / [Pascal](#), agregando na suite Delphi Studio 2005, a linguagem [C#](#) e a extensão da *Object Pascal* para [.NET](#);

- [Eclipse](#) - Gera código [Java](#) (através de plugins, o Eclipse suporta muitas outras linguagens como [Python](#) e [C](#) / [C++](#));
- [Netbeans](#) - Gera código [Java](#) (e suporta muitas outras linguagens como PHP, [Python](#) e [C](#) / [C++](#));
- [Sun Studio](#)- Linguagens C, C++ e Fortran;
- [SharpDevelop](#) - Gera código [C#](#);
- [DEV-C++](#), [Code::Blocks](#), [Turbo C](#) - Geram código para C e C++;
- [Source Insight](#) - Suporta: C, C++, VHDL, para Ambiente [Windows](#)
- [Force \(IDE\)](#) - Gera código [Fortran](#), para Ambiente [Windows](#)
- [OutSystems](#)- OutSystems Platform;
- [Access \(IDE\)](#) - Gera código VBA - Visual Basic for Applications, para Ambiente [Windows](#)
- [Visual Basic](#) - Gera código [Basic](#);

- **IDLE (Integrated Development and Learning Environment - PYTHON)** ou Ambiente integrado de aprendizado e desenvolvimento.

é um ambiente de desenvolvimento integrado para Python, que vem junto com a implementação padrão da linguagem desde 1.5. 2b1. ... IDLE pretende ser um IDE simples e adequado para iniciantes, especialmente em um ambiente educacional.

Como usaremos o python, após a instalação do ANACONDA

www.anaconda.com

1ª)

- **desinstalar todas as versões python**
- **instalar anaconda com o path, baixando o anaconda**

2ª) abrir o prompt:

>python --version

3ª) Entrar no IDEL (SHELL) do python

Abrir o prompt e digitar

>idle

Vamos aos testes:

```
>>> 1+1
```

```
>>> print("Testando o IDLE")
```

```
>>> a = 1; b=2
```

```
>>> a+b
```

Aqui no IDLE, se quisermos voltar a instrução anterior, basta apertar **ALT+P**, volta até duas instruções. Se quisermos buscar duas anteriores **ALT+N**.

Pode-se configurar as configurações do IDLE. Basta ir em

-> Options/configure IDLE

Pode-se abrir várias janelas do IDLE, e cada instrução será guardada na memória do PC. Se quisermos reiniciar as instruções de um IDLE, basta clicar em **CTRL+F6**

Vai aparecer RESTART: Shell

Para escrever um SCRIPT MULTILINE (código de múltiplas linhas), ou simplesmente código ou script, basta clicarmos em

FILE -> NEW FILE (CTRL+N)

OBS: Não iremos usar a IDLE pois possui algumas limitações, mas serve para testar algumas instruções mais simples.

Ao invés do IDLE, podemos usar também o prompt de comando (CMD do Windows) para fazermos estas instruções.

-> Digitar CMD em “Pesquisar na aba do Windows”

Abrirá o mesmo ambiente que a IDLE, porém no prompt de comando do Windows.

VARIÁVEIS EM PYTHON

- **Inteiras,**
- **Reais ou flutuantes,**
- **Strings ou caracteres,**

string não tem uma tradução em computação em PT

- **Lógicas ou booleanas**

Características de todas as variáveis

- **NOME** – declaração;
- **TIPO** – um dos tipos, citado anteriormente;
- **TAMANHO** – quantos BYTES ele ocupa na memória;
- **VALOR** - qual a atribuição.

- **INTEIRA**

Vamos abrir a IDLE no prompt de comando.

```
>>>a = 1; b = 2
```

```
>>>a+b
```

```
>>>c          #sem declarar dá um erro
```

Para testar:

```
>>>type(a)
```

- **REAL OU FLUTUANTE**

```
>> d = 2.5; e = 2;
```

```
>> d*e
```

```
>>>type(d)
```

- **STRING OU CARACTERE**

```
>>> texto = "texto é uma string"
```

Representada sempre com aspas simples ou duplas. Se agora instruírmos

```
>>> print(texto)
```

Diferença na escrita no código (script). Na primeira instrução aparecerá a instrução entre aspas, usando o print, não!

```
>>> type(texto)
```

- **LÓGICA OU BOOLEANA**

Valor binário, 0 ou 1, verdadeiro ou falso, ou qualquer outra atribuição binária.

```
>>> a = True ou False
```

Impressão de valores

Vamos declarar variáveis:

```
variavel_int = 5  
variavel_real = 3.1  
variavel_str = "um texto qualquer"
```

Escrever ou no IDLE no Script

```
#INTEIRO  
print("o valor é: ", variavel_int)  
  
#outra forma  
print("o valor é: %i" %variavel_int)  
#fazendo assim não é necessário colocar a virgula
```

Impressão de valores

Continuação ...

```
#STRING
```

```
print("o valor é: ", variavel_str)
```

```
#convertendo o número em String
```

```
#ou ainda
```

```
print("o valor é: %s" + variavel_str)
```

```
#concatenando
```

```
print("o valor é: " + str(variavel_int))
```

```
#não precisa mais da virgula pois estamos concatenando duas strings
```

```
#não mais separando dois tipos de variaveis
```

```
#REAL
```

```
print("o valor é: ", variavel_real)
```

```
#outra forma
```

```
print("o valor é: %f" %variavel_real)
```

```
#fazendo asssim não é necessário colocar a virgula
```

```
#podemos definir o número de casas decimais a serem impressos
```

```
print("o valor é: %.2f" %variavel_real)
```

```
print("o valor é: %.20f" %variavel_real)
```

Impressão de valores

Continuação ...

```
#REAL
```

```
print("o valor é: ", variavel_real)
```

```
#outra forma
```

```
print("o valor é: %f" %variavel_real)
```

```
#fazendo assim não é necessário colocar a virgula
```

```
#podemos definir o número de casas decimais a serem impressos
```

```
print("o valor é: %.2f" %variavel_real)
```

```
print("o valor é: %.20f" %variavel_real)
```


EXEMPLOS E ENTRADA DE DADOS

Exemplo: Input / output

```
login = input("Entrar com seu Login: ")  
senha = input("Entrar com sua Senha: ")  
  
print("O seu login é: %s, e a senha é: %s " %(login, senha))
```

EXEMPLOS E ENTRADA DE DADOS

Exemplo: Operações matemáticas

```
a = 10; b = 6
```

```
c = a/b          #divisão normal
```

```
d = a//b         #divisão inteira
```

```
e = a%b          #módulo da divisão/ resto ==>      10/4 = 1 + 6/4
```

```
print("multiplicação: ", a*b)
```

```
print("divisão: ", c)
```

```
print(type(c))
```

```
print("divisão inteira: ", d)
```

```
print(type(d))
```

```
print("resto da divisão 6/2 módulo): ", 6%2)
```

```
print("resto da divisão (módulo): ", e)
```

EXEMPLOS E ENTRADA DE DADOS

Exemplo: Operações matemáticas

```
a = 2; b = 3
```

```
c = a**b                # potencialização
```

```
d = a**(1/2)            # raiz quadrada . pode ser a**0.5
```

```
e = 10**a                # base 10
```

```
print(c)
```

```
print()
```

```
print(d)
```

```
print()
```

```
print(e)
```

EXEMPLOS E ENTRADA DE DADOS

No modo interativo (IDLE)

```
>>>import math
```

```
>>>dir(math)
```

Exemplo: Operações matemáticas com pacote MATH

```
import math
```

```
PI = math.pi
```

```
print(PI)
```

```
#ou simplesmente math.pi
```

```
cos = math.cos(0)
```

```
print(cos)
```

Contiuação ...

```
expon = math.exp(1)
print(expon)

print(math.sqrt(4))

print(math.factorial(5))

print(math.atan(1))                #pi/4
print(math.atan(1)*180/PI)         # 45 graus

print(math.radians(45))            #conversão
print(math.degrees(PI/4))          #conversão

print(math.log(expon))

print(math.log10(10))

print(math.log(3,9))               # log de 3 na base 9
print(math.log(9,3))               # log de 9 na base 3

print(math.perm(2))                #permutação de 2 números
print(math.perm(3))                #permutação de 3 números
```

EXEMPLOS E OPERADORES RELACIONAIS

= representa atribuição

== representa igualdade

!= representa diferença

Exemplo: Problema 1.5 da lista. Verifique as relações

#No IDLE ou no script

#problema 1.5 lista

```
print('(a): ', 2 < 3)
print('(b): ', 2 > 3)
print('(c): ', 5 - 2 > 3 + 4)
print('(d): ', 3 == 3)
print('(e): ', 3 == 4)
print('(f): ', 3 == 5 - 3)
print('(g): ', 3 <= 4)
print('(h): ', 3 >= 4)
print('(i): ', 3 != 4)
```

Continuação ... Exemplo: Problema 1.5 da lista

```
print('(j): ', 1 == 1, 2 == 1)
print('(k): ', True == True)
print('(l): ', True == False)
print('(m): ', False == False)
```

Exemplo: com variáveis tipo String e Lógicas

#No IDLE ou no script

```
print('a' == 'a')          #temos que colocar as variáveis como string  
print('a' != 'a')  
print('a' == 'b')  
print('a' != 'b')  
print('a' > 'b')  
print('a' < 'b')
```

```
print()  
A = True; B = False  
print(A == A)  
print(A != A)  
print(A == B)  
print(A != B)
```

```
x=(1!=1)  
print(x)
```

```
y=('x' == 'x')  
print(y)  
print("Testando as variáveis booleanas acima: ",x == y)
```


Operadores de Atribuição

São simplificações que o Python e algumas linguagens proporcionam

$x = x + y$	~	$x += y$
$x = x - y$	~	$x -= y$
$x = x * y$	~	$x *= y$
$x = x / y$	~	$x /= y$
$x = x \% y$	~	$x \% = y$

- Exemplo: Realize os testes

$x = 9; y = 3$		
$x = x + y$	~	$x += y$
$x = x - y$	~	$x -= y$
$x = x * y$	~	$x *= y$
$x = x / y$	~	$x /= y$
$x = x \% y$	~	$x \% = y$

#Use o IDLE ou Script

```
x = 9; y = 3
```

```
# soma : x = x + y
x += y
print(x)
```

```
# subtração: x = x - y
x -= y
print(x)
```

```
# multiplicação: x = x * y
x *= y
print(x)
```

```
# divisão: x = x /y
x /= y
print(x)
```

```
# módulo ou resto: x = x%y
x %= y
print(x)
```

Atribuição múltipla ou sequencial

- **Exemplo:** Redefina os valores entre si: $x = 9$; $y = 3$

Em outras linguagens é necessária a introdução de uma terceira variável para armazenamento:

```
z = x
```

```
x = y
```

```
y = z    ->>> x = 3 e y = 9
```

Em python, dados:

```
x = 9; y = 3
```

```
x, y = y, x    ->>> x = 3 e y = 9
```

- Exemplo: Redefina os três valores:

```
>>> x, y, z = 2, 3, 4
>>> x, y, z = x**2, x+y+z, z/x
>>> x, y, z                                # imprimindo na tela os valores
modificados
(4, 9, 2.0)                                # os valores são procesados ao
mesmo tempo
```

- Exemplo: Redefina os valores:

```
#Escrever no IDLE ou Script
```

```
nome, sobrenome, profissão = 'Wesley ', ' Spalenza ', '
Físico'
print(nome + ', ' + sobrenome + ', ' + profissão)
```