



Algoritmos e Lógica de Programação

Aula 03

Expressões Matemáticas e Tipos de Dados

Prof. Dr. Dilermando Piva Jr

1º Semestre - DSM



IDENTIFICADORES

Nome de variáveis, funções, classes e módulos.

Regras...

- Identificadores podem ser uma combinação de letras minúsculas (a-z) e maiúsculas (A-Z), dígitos (0-9) e underscore (_)
- Identificadores não podem iniciar com um dígito.
- Palavras chaves não podem ser utilizadas como identificadores
- Não se pode utilizar espaços e nem caracteres especiais, tais como: !, @, #, \$, % etc.
- Um identificador pode ter qualquer tamanho (não existe limite de caracteres)

Variáveis...

Regras para nomes de identificadores

- Padrões de Projetos:

- camelCase



camelCase

- snake_case



snake_case

- kebab-case



kebab-case

Variáveis...

Regras para nomes de identificadores

- Padrões de Projetos:

- camelCase



camelCase

- snake_case



snake_case

- kebab-case



kebab-case



PEP 8

PALAVRAS-CHAVES

São palavras reservadas (`__builtins__`)

Palavras-chaves em Python

and	as	not
assert	finally	or
break	for	pass
class	from	nonlocal
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield
False	True	None

COMENTÁRIOS...

Linhas que não serão analisadas pelo Interpretador Python

Servem para DOCUMENTAR

Comentários...

- Comentário de uma única linha

```
# Este é um comentário em Python de uma linha
```

- Comentário de várias linhas

```
"""Este é um comentário  
Em Python contendo  
Múltiplas linhas"""
```

O que é uma Expressão Matemática?

Expressão Matemática...

- É uma combinação de números (0-9), operadores (+, - etc.), variáveis (x, y etc.) e símbolos gráficos ([, { etc.) agrupados de forma significativa de modo a permitir a verificação de valores, formas, meios ou fins.

Exemplos:

$$\frac{1}{a^2 - ab} + \frac{1}{ab - b^2}$$

$$\left[\frac{(8 + x^3) \cdot (x^2 - 4)}{(x^2 + 4x + 4) \cdot ((x^2 - 2x + 4)) \cdot (4 - 2x)} \right]^5$$

Expressão Matemática em Programação

- As expressões matemáticas (e lógicas) como conhecemos e utilizamos, não podem ser implementadas no computador no formato que conhecemos.
- Elas devem sofrer um processo conhecido como **linearização**. Além disso, existe um conjunto de operações matemáticas que deve ser implementado para possibilitar que instruções gráficas, como raiz quadrada, possam ser devidamente utilizadas. A maioria dessas operações será implementada como funções.

Linearização de Expressões

- Uma expressão matemática convencional:

$$x = \frac{3y}{5y + 7} + 2y$$

- A mesma expressão linearizada:

$$x = ((3 * y) / (5 * y + 7)) + (2 * y)$$

Operadores Matemáticos...

Operadores matemáticos em Python

Operador	Nome do Operador	Descrição	Exemplo
+	Operador de Adição	Soma dois operandos, resultando na sua soma.	$5+3=8$
-	Operador de Subtração	Subtrai dois operandos, resultando na sua diferença.	$5-3=2$
*	Operador de Multiplicação	Produz o produto de dois operandos	$5*3=15$
/	Operador de Divisão	Produz o quociente da divisão do operando a direita pelo operando a esquerda	$5/3=1.666666666666....$
%	Operador de Módulo	Divide o operando a direita pelo operando à esquerda e retorna o resto dessa divisão	$5\%3=2$
**	Operador de Exponenciação	Produz o exponencial, onde o operando a esquerda é elevado pelo operando a direita	$5**3=125$
//	Operador de Divisão Inteira	Retorna a parte inteira de uma divisão.	$5//3=1$

Para fixar... Como ficaria:

- Uma expressão matemática convencional:

$$y = \frac{x + 3b}{2x + c}$$

- A expressão linearizada:

Para fixar... Como ficaria:

- Uma expressão matemática convencional:

$$y = \frac{x + 3b}{2x + c}$$

- A expressão linearizada:

$$y = (x + 3 * b) / (2 * x + c)$$

O que é uma Expressão Lógica?

Expressão Lógica...

- Diferentemente de uma expressão matemática onde o resultado pode ser um número, ou uma variável ou uma combinação entre número e variável, uma expressão lógica sempre resulta em um de dois valores possíveis:
ou ela é **VERDADEIRA (True)** ou ela é **FALSA (False)**

Exemplos:

Suponha que $x=3$ e $y=5$:

$z = x > y$

$z = ?$

$r = y \geq x$

$r = ?$

Expressão Lógica...

- Diferentemente de uma expressão matemática onde o resultado pode ser um número, ou uma variável ou uma combinação entre número e variável, uma expressão lógica sempre resulta em um de dois valores possíveis:
ou ela é **VERDADEIRA (True)** ou ela é **FALSA (False)**

Exemplos:

Suponha que $x=3$ e $y=5$:

$z = x > y$

$z = \text{False}$

$r = y \geq x$

$r = \text{True}$

Operadores Relacionais...

Operadores relacionais em Python

Operador	Operação	Exemplo	Resultado
<code>==</code>	Igual	<code>a==b</code> (a é igual a b?)	False
<code>!=</code>	Diferente	<code>a!=b</code> (a é diferente de b?)	True
<code>></code>	Maior que	<code>a>b</code> (a é maior que b?)	False
<code><</code>	Menor que	<code>a<b</code> (a é menor que b?)	True
<code>>=</code>	Maior ou igual que	<code>a>=b</code> (a é maior ou igual que b?)	False
<code><=</code>	Menor ou igual que	<code>a<=b</code> (a é menor ou igual que b?)	True

O valor de a = 10 e b = 20

Operadores Lógicos...

Operadores lógicos em Python

Operador	Nome do Operador	Descrição	Exemplo
and	E lógico	Realiza a operação E e o resultado é VERDADEIRO se ambos os operandos forem VERDADEIROS	a and b (resulta em False)
or	OU lógico	Realiza a operação OU e o resultado é VERDADEIRO se pelo menos um dos operandos for VERDADEIRO	a or b (resulta em True)
not	NÃO lógico	Inverte o estado de um operando	not a (resulta em False)

O valor de a = True e b = False

Operadores lógicos em Python

Tabelas Verdade

		and	or	not
P	Q	P and Q	P or Q	Not P
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

Expressões Lógicas...

- Uma expressão lógica:

$x = (p > q) \text{ and } (\text{not}(p == q)) \rightarrow x = ?$

$y = (\text{not}(q < p)) \text{ or } (q != p) \rightarrow y = ?$

Levando em consideração que: $p = 5$ e $q = 7$

Expressões Lógicas...

- Uma expressão lógica:

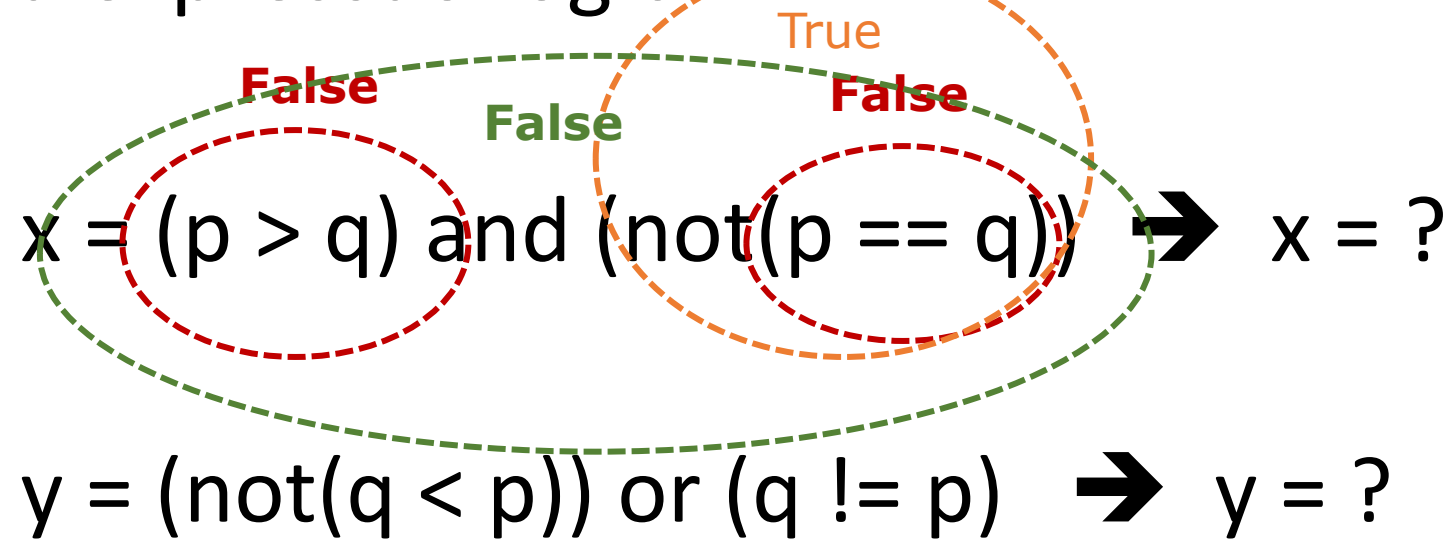
False **False**
 $x = (p > q) \text{ and } (\text{not}(p == q)) \rightarrow x = ?$

$y = (\text{not}(q < p)) \text{ or } (q != p) \rightarrow y = ?$

Levando em consideração que: $p = 5$ e $q = 7$

Expressões Lógicas...

- Uma expressão lógica:



Levando em consideração que: $p = 5$ e $q = 7$

Expressões Lógicas...

- Uma expressão lógica:

$x = (p > q) \text{ and } (\text{not}(p == q)) \rightarrow x = ?$

$y = (\text{not}(\text{False})) \text{ or } (\text{True}) \rightarrow y = ?$

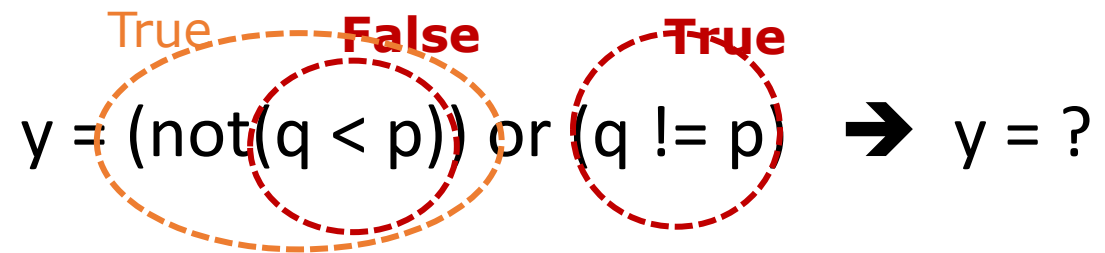
Levando em consideração que: $p = 5$ e $q = 7$

Expressões Lógicas...

- Uma expressão lógica:

$x = (p > q) \text{ and } (\text{not}(p == q)) \rightarrow x = ?$

$y = (\text{not}(q < p)) \text{ or } (q != p) \rightarrow y = ?$



Levando em consideração que: $p = 5$ e $q = 7$

Expressões Lógicas...

- Uma expressão lógica:

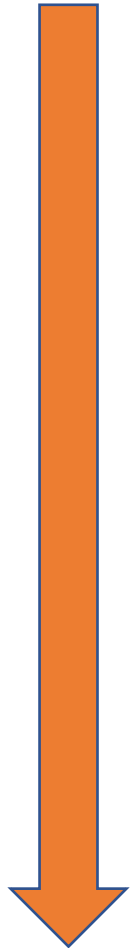
$x = (p > q) \text{ and } (\text{not}(p == q)) \rightarrow x = ?$

$y = (\text{not}(q < p)) \text{ or } (q != p) \rightarrow y = ?$

True False True

Levando em consideração que: $p = 5$ e $q = 7$

Prioridade em Python



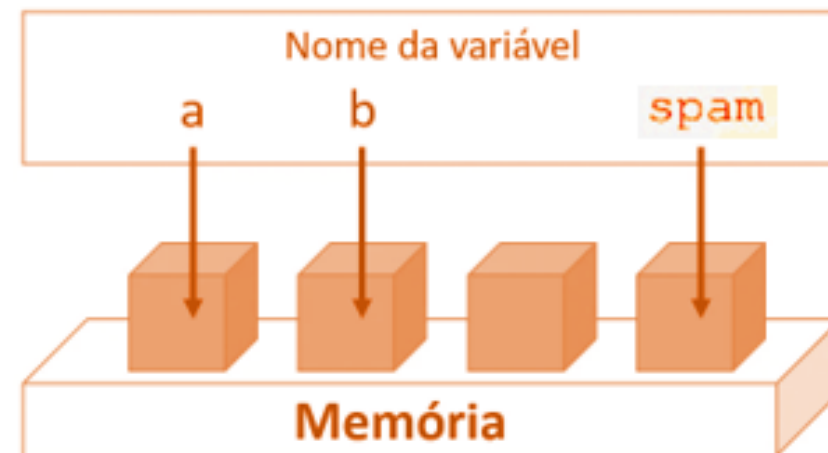
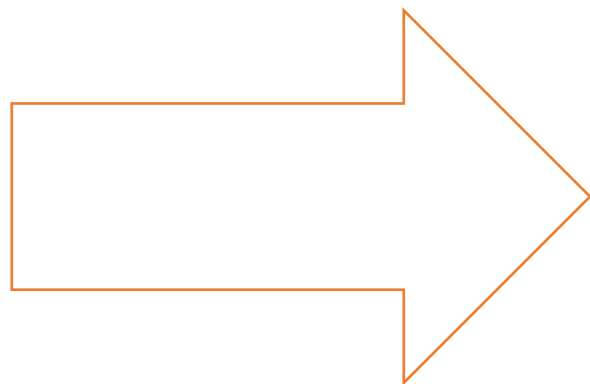
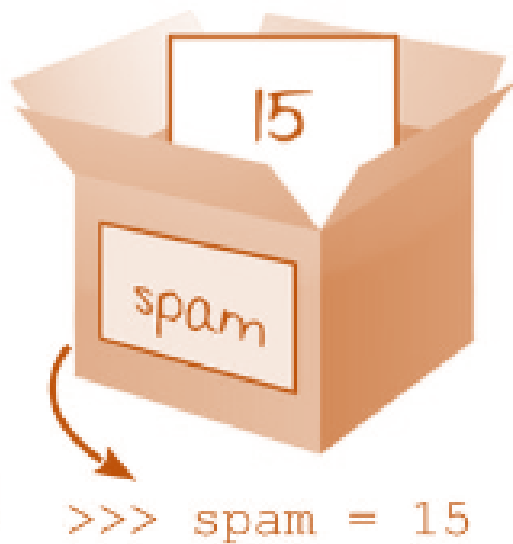
OPERADORES	SIGNIFICADO
()	Parênteses
**	Exponente
+X, -X	Soma ou subtração unária (definição do sinal)
*, /, //, %	Multiplicação, Divisão, Divisão Inteira e Módulo
+, -	Adição e subtração
==, !=, >, >=, <, <=	Operadores de comparação
not	Não Lógico
and	E lógico
or	OU lógico

O que é uma variável?

Uma variável...

- “... que pode assumir qualquer um dos valores em um conjunto de valores”. Dicionário Oxford
- Em COMPUTAÇÃO:
 - “... um espaço na memória do computador destinado a um dado que pode ser alterado durante a execução de um programa (algoritmo)”.
 - Este espaço de memória tem um endereço.
 - Existe um rótulo (nome da variável) que vai associar esse nome a tal endereço da memória.

Variáveis são espaços em memória, utilizados para “guardar” uma determinada informação



Uma variável...

Tabela

Rótulo	Endereço
X	0x00234DA1
Y	0x00234DA5
...	

Memória

Endereço	Valor
00234DA1	00000000
00234DA2	00000000
00234DA3	00000000
00234DA4	00000101
00234DA5	00000000
00234DA6	00000000
00234DA7	00000000
00234DA8	00000110
00234DA9	

Inteiro de 32 bits

00000000000000000000
0000000000000000000101 b = 5 d

00000000000000000000
0000000000000000000110 b = 6 d

Uma variável...

Tabela

Rótulo	Endereço
X	0x00234DA1
Y	0x00234DA5
...	

Memória

Endereço	Valor
00234DA1	00110101
00234DA2	00000000
00234DA3	00000000
00234DA4	00000000
00234DA5	00110110
00234DA6	00000000
00234DA7	00000000
00234DA8	00000000
00234DA9	

CARACTER (8bits)

00110101 b = "5"

00110110 b = "6"

Tabela ASCII

Decimal	Hexadecimal	Binary	Octal	Char
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:

Tipos de Dados em Python

PRIMITIVOS

- Caracteres ou String – str – 'A', '%', 'c', 'pyPRO'
- Inteiros – int – 0, 2, 45, -56
- Reais ou decimal – float – 3.1416, 0.456, -34.45
- Complexos – complex – 3j, 45j
- Booleanos (Lógicos) – bool – True ou False

OUTRAS ESTRUTURAS DE DADOS

- Listas (list)
- Tuplas (tuple)
- Dicionário (dic)
- ...

O que é uma constante?

Uma constante...

“Algo que não muda!”.

Uma constante...

- “Algo que não muda!”.
- PYTHON:
 - “uma linguagem dinâmica!”
 - Não existe em Python recursos como em outras linguagens:
 - `#define`
 - `int final...`
 - Python é feito para adultos...
 - UTILIZE NOME DE VARIÁVEIS EM MAIÚSCULA
 - Ex: `PI = 3.1415`
 - ... `area = PI * (raio*raio)`

TIPO INTEIRO

Tipo Inteiro

```
10
num = 10
print(num)
10+20
7/2
int(7/2)
7//2
7%2
7**2
type(10)
type(num)
#tamanho: 32 ou 64 bits? Em python não tem limite!!
2**32
2**64
2**2000
#para limpar o console: CTRL + L
1000000 --> 1_000_000
#PARA FACILITAR A VISUALIZAÇÃO
num = num+1
num+=1
#(-, *, /)
dir(num)
num.__add__(8)
```

TIPO FLOAT

Tipo Float

TIPO FLOAT (reais ou decimais):

ATENÇÃO: O separador de casas decimais é o PONTO e não a VIRGULA

10.0

num = 10.0

print(num)

10.44 + 20.56

7/2

int(7/2)

inteiro = 5

num = float(inteiro)

print(num)

type(10.44)

type(num)

#tamanho: 64 bits? Em python não tem limite!!

2.45**32

2.45**64

2.45**2000

#para limpar o console: CTRL + L

1000000.00000000 -->

1_000_000.000_000_00

#PARA FACILITAR A VISUALIZAÇÃO

num = num+1.8

print(int(num))

cuidado com a perda da precisão no processo de conversão

dir(num)

num.__add__(8)

#Numeros complexos

n acompanhado de i // n acompanhado de j

5j

6+5j

5*6j

5**6j

TIPO LÓGICO ou BOOLEANO

Tipo Lógico

TIPO BOOLEANO (LÓGICO)

ATENÇÃO: True False

True

flag = False

print(flag)

type(flag)

dir(flag)

flag.real

flag = False

flag.real

num = 2

dir(num)

num

TIPO STRING

Tipo String

TIPO STRING (CHAR ou character)

ATENÇÃO: 1 ou mais caracteres = STRING

ATENÇÃO 2: s = 'a' ou "a" -> sem quebra de linha

s = '''a''' ou """a""" -> com quebra de linha

```
letra='a'
```

```
palavra='pyPRO'
```

```
type(letra)
```

```
type(palavra)
```

```
frase="Seja um profissional Python!"
```

```
print(frase)
```

```
print(frase[1])
```

**ESCOPO DE UMA
VARIÁVEL**

E CONVERSÕES

ESCOPO

- Global
- Local

```
x = 0
y = 0
...
```

```
print("Gy - ", y)
print("Gx - ", x)
...
```

```
def funcao():
    y=4
    print("Fy - ", y)
    print("Fx - ", x)
...
```

```
funcao()
print("Gy - ", y)
print("Gx - ", x)
```

```
C:\python>type global_local.py
x=0
y=0

print("Gy - ", y)
print("Gx - ", x)

def funcao():
    y=4
    print("Fy - ", y)
    print("Fy - ", x)

funcao()
print("Gy - ", y)
print("Gx - ", x)
```

```
C:\python>python global_local.py
Gy - 0
Gx - 0
Fy - 4
Fy - 0
Gy - 0
Gx - 0
```

Conversões entre tipos...

- Inteiro → Decimal
`x = float(y)`
- Decimal → Inteiro
`x = int(y)` e `x = round(y)`
- String → Inteiro
`x = int(num)`
- String → Decimal
`x = float(num)`
- Inteiro → String
`num = str(x)` → só inteiros!!
- Decimal → String
`num = str(x)`

Agora é com você...

2) Escreva o comando de atribuição e resolva a expressão das seguintes expressões matemáticas (implemente o comando de atribuição em todas as linguagens vistas: VisuAlg, Pascal, C, Java e PHP).

$$\text{a) } X = \frac{A + \frac{B}{C}}{D - \frac{E}{F}} \text{ onde } A=2, B=6, C=3, D=4, E=8, F=4$$

$$\text{b) } Y = \frac{\frac{2X^2 - 3X^{(X+1)}}{2} + \frac{\sqrt{X+1}}{X}}{2^X} \text{ onde } X=2$$

Funções Embutidas

- Além dos operadores, é possível usar funções para computar valores
- As funções podem ser definidas:
 - Pelo programador (veremos + tarde)
 - Em módulos da biblioteca padrão
 - Por *default*: são as funções *embutidas* (*built-in*)
 - Na verdade, fazem parte do módulo `__builtins__`, que é sempre importado em toda aplicação
- Ex.:
 - **abs(x)** retorna o valor absoluto do número x
 - **chr(x)** retorna uma string com um único caractere cujo código ASCII é x
 - **ord(s)** retorna o código ASCII do caractere s

Funções Embutidas

```
>>> abs (10)
```

```
10
```

```
>>> abs (-19)
```

```
19
```

```
>>> chr (95)
```

```
'_'
```

```
>>> chr (99)
```

```
'c'
```

```
>>> ord ('a')
```

```
97
```

Importando módulos

- Muitas funções importantes são disponibilizadas em módulos da biblioteca padrão
 - Ex.: o módulo **math** tem funções transcendentais como **sin**, **cos**, **exp** e outras
- Um módulo pode conter não só funções mas também variáveis ou classes
 - Por exemplo, o módulo **math** define a constante **pi**
- Para usar os elementos de um módulo, pode-se usar o comando **import**
 - Formatos:
 - **import** *modulo*
 - **from** modulo **import** *nome,...,nome*
 - **from** modulo **import** *

Importando módulos

- Por exemplo:

```
from math import *  
# importa todos os elementos do módulo math  
from math import sin  
# importa apenas a função sin  
import math  
# importa o módulo math como um todo  
# (todos os elementos têm que ser citados  
# precedidos por math.)
```

Importando módulos

```
>>> import math
>>> a = sin(30)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'sin' is not defined
>>> a = math.sin(30)
>>> from math import sin
>>> a = sin(30)
>>> print (a)
-0.988031624093
>>> a = sin(radians(30))
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'radians' is not defined
>>> from math import *
>>> a = sin(radians(30))
>>> a
0.49999999999999994
```

Explorando Módulos

```
>>> import math
>>> help(math.cos)
Help on built-in function cos in module math:

cos(...)
    cos(x)

        Return the cosine of x (measured in radians).
(END)
```

Pressiona-se “q” para retornar ao interpretador.

Agora é com você...

2) Escreva o comando de atribuição e resolva a expressão das seguintes expressões matemáticas (implemente o comando de atribuição em todas as linguagens vistas: VisuAlg, Pascal, C, Java e PHP).

$$\text{a) } X = \frac{A + \frac{B}{C}}{D - \frac{E}{F}} \text{ onde } A=2, B=6, C=3, D=4, E=8, F=4$$

$$\text{b) } Y = \frac{\frac{2X^2 - 3X^{(X+1)}}{2} + \frac{\sqrt{X+1}}{X}}{2^X} \text{ onde } X=2$$

Agora é com você...

4) Escreva os comandos de atribuição (em todas as linguagens vistas) para as seguintes expressões matemáticas linearização.

$$\text{a) } X = \frac{\sqrt{2B - 4A} + 2F^{-3}}{3 - 2A}$$

$$\text{b) } Y = 2H - \left[\frac{45}{3X} - 4H(3 - H) \right]^{2H}$$

Agora é com você...

(FORBELLONE; EBERSPÄCHER, 2000 - pág. 18) Assinale os identificadores válidos:

- | | |
|--------------------------------------|-------------------------------------|
| a) ☐ (X) | i) ☐ AB*C |
| b) ☐ U2 | j) ☐ O&O |
| c) ☐ AH! | l) ☐ P{0} |
| d) ☐ "ALUNO" | m) ☐ B52 |
| e) ☐ #55 | n) ☐ Rua |
| f) ☐ KM/L | o) ☐ CEP |
| g) ☐ UYT | p) ☐ dia/mês |
| h) ☐ ASDRUBAL | |