IBGE
Instituto Brasileiro de Geografia e Estatística

Escola Nacional de Ciências Estatísticas

Introdução à Programação

Aula 03: Programação em Python:

Operações Aritméticas e Funções Matemáticas

Prof. Eduardo Corrêa

Data 14/03/2024

Tópicos da Aula

- Temas desta aula:
  - Operadores Aritméticos
  - Operadores de String
  - Funções Matemáticas (Módulo math)

Operadores Aritméticos (1/6)

- Um **operador** é um símbolo da linguagem de programação capaz de operar sobre pares de valores.
- Você conhecerá diferentes classes de operadores nesse curso.
- Vamos começar com os **operadores aritméticos**, que são aqueles que **executam operações matemáticas** como adição e multiplicação.

Principais operadores aritméticos

Operador	*Operando*	*Resultado*	*Operador*
Adição	Inteiro ou Real	Inteiro ou Real	+
Subtração	Inteiro ou Real	Inteiro ou Real	-
Multiplicação	Inteiro ou Real	Inteiro ou Real	*
Divisão	Inteiro ou Real	Real	/
Exponenciação	Inteiro ou Real	Inteiro ou Real	**
Quociente da Divisão	Inteiro ou Real	Inteiro ou Real	//
Resto da Divisão	Inteiro ou Real	Inteiro ou Real	%

Operadores Aritméticos (2/6)

- Como na matemática, há uma hierarquia de prioridades:
  - A potenciação tem maior prioridade: **
  - Seguida da multiplicação e divisão: *, /, //, %
  - E por último a adição e subtração: +, -
- E, é claro, você pode usar **parênteses** para modificar a ordem natural dos cálculos.
- **Exemplo**: qual será o valor armazenado em w?

```
w = ((5 * ((25 % 13) + 100) / (2 * 13)) // 2)
```

Operadores Aritméticos (3/6)

- **Exemplo**: qual será o valor armazenado em w?

```
w = ((5 * ((25 % 13) + 100) / (2 * 13)) // 2)
    ((5 * ((25 % 13) + 100) / (2 * 13)) // 2)
    ((5 * (12 + 100) / (2 * 13)) // 2)
    ((5 * 112 / (2 * 13)) // 2)
    ((5 * 112 / 26) // 2)
    ((560 / 26) // 2)
    (21.53846153846154 // 2)
    10.0
```

Operadores Aritméticos (4/6)

- Observações importantes:

(1). Toda divisão realizada com o operador **/** resultará em um **valor float**, mesmo que a divisão seja exata.

Ex.: 4 / 2 = 2.0

Veja que o resultado é 2.0 (float) e não 2 (int)

(2). Ao contrário das outras operações, a exponenciação é resolvida da direita para esquerda ← (caso contrário não daria certo...)

Ex.: 2 ** 2 ** 3 = 256

```
2 ** 2 ** 3
2 ** 8
256
```

Operadores Aritméticos (5/6)

- Observações importantes (*continuação*):

(3). Você **não pode** realizar operações aritméticas entre um número e uma string, mesmo que o conteúdo da string seja um número:

```
1 + "1"
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

- Para fazer a conta, você precisa converter a string para um número usando a função int() ou a função float().

```
1 + int("1")
```

```
2
```

Operadores Aritméticos (6/6)

- Mais alguns exemplos:

```
print((-1 / 2), (1 / 2), (1 // 2), (1 % 2))
```

```
-0.5   0.5   0   1
```

```
print((2 ** 4), (2 * 4.0), (2 * 4))
```

```
16   8.0   8
```

Operadores de String (1/3)

- Quando os operadores **+** e ***** são aplicados sobre dados do tipo int ou float, eles realizam a adição e multiplicação, respectivamente.
- Porém, esses operadores exercem **outra função** quando aplicados sobre pares de strings (dados do tipo str) !!!
  - Para strings:
    - **+** é o operador de **concatenação**.
    - ***** é o operador de **repetição**.

Operadores de String (2/3)

- **+** quando aplicado a strings é o operador de **concatenação**.
  - Ele "cola" (concatena) duas strings uma na outra.
  - Veja os exemplos:

```
"AB" + "CDE"
```

```
'ABCDE'
```

```
n = "Antonio"
s1 = "Carlos"
s2 = "Jobim"
nome_completo = n + " " + s1 + " " + s2
print(nome_completo)
```

```
' Antonio Carlos Jobim'
```

Operadores de String (3/3)

- ***** quando aplicado a uma string e um número inteiro *n* atua como operador de **repetição**.
  - Ele repetirá a string *n* vezes.
  - Veja os exemplos:

```
"AB" * 3

'ABABAB'


"2" * 5

'22222'
```

veja que não é 10 !!!

Funções - Conceito (1/2)

- Nas linguagens de programação, também existem **funções**.
  - Uma função recebe um ou mais **argumentos** (dados) como entrada e **retorna uma saída**.
  - Por exemplo, já conhecemos a função int():
    - Recebe como entrada uma string
    - Como saída, retorna a string convertida para um inteiro.
      - int('10') → 10

Funções - Conceito (2/2)

- Existem milhares de funções na linguagem Python.
- Elas costumam estar reunidas em **módulos** (também chamados de bibliotecas) de acordo com algum tema.
- Alguns exemplos:
  - módulo math → contém funções matemáticas.
  - módulo statistics → contém funções estatísticas.
  - módulo sqlite3 → contém funções para trabalhar com o banco de dados SQLite.
  - *... existem milhares de outros módulos !!!!*
  - *... alguns criados pela própria equipe do Python e outros por terceiros*

Módulo math (1/8)

- Nessa aula, vamos conhecer as funções do módulo **math**.
- Ele fornece uma série de funções e constantes matemáticas úteis, tais como funções de arredondamento, trigonométricas, logarítmicas, etc.
- Antes de utilizá-lo em um programa, você precisa realizar a sua importação da seguinte maneira:

```
import math
```

  - É assim com qualquer módulo !
  - Se você quiser usar as funções que estão “dentro” de um módulo precisará importa-lo.

Módulo math (2/8)

- A seguir, são relacionadas as principais constantes e funções deste módulo.
- Você **não precisa decorar** essas constantes e funções, mas deve saber como utilizá-las em seus programas.

Constantes do Módulo Math
• **math.e**: constante matemática 2.718281... (número de Euler);
• **math.tau**: constante matemática 6.283185… (equivalente à 2π);
• **math.inf**: valor float que representa +∞. Para –∞ basta usar -math.inf;
• **math.nan**: trata-se do famoso NaN, valor float que representa “not a number”. Este valor é gerado pelo Python sempre que o resultado de um cálculo não puder ser expresso por um número. Qualquer cálculo envolvendo NaN sempre resultará em NaN (ex.: 1 + NaN = NaN).
• **math.pi**: constante matemática 3.14159…

Módulo math (3/8)

Funções de Arredondamento
• **math.ceil**(x): arredondamento “pra cima”, ou seja, retorna o menor inteiro com valor igual ou superior a x;
• **math.floor**(x): arredondamento “pra baixo”, ou seja, retorna o maior inteiro com valor igual ou inferior a x.
• **math.trunc**(x): truncamento, o que significa limitar o número de dígitos de x.

Funções Logarítmicas / Exponenciais
• **math.exp(x)**: retorna e elevado a x;
• **math.log2(x)**: retorna o logaritmo de x na base 2;
• **math.log10(x)**: retorna o logaritmo de x na base 10;
• **math.log(x, b)**: retorna o logaritmo de x na base b (de acordo com a documentação do Python, deve ser utilizada apenas quando b é diferente de 2 e 10).
• **math.pow(x, y)**: retorna x elevado a y;
• **math.sqrt(x)**: retorna a raiz quadrada de x;

Módulo math (4/8)

Funções Trigonométricas (em todas elas, x é um ângulo que deve ser fornecido em radianos)
• **math.acos(x)**: retorna o arco cosseno de x;
• **math.asin(x)**: retorna o arco seno de x;
• **math.atan(x)**: retorna o arco tangente de x;
• **math.cos(x)**: retorna o cosseno de x;
• **math.sin(x)**: retorna o seno de x;
• **math.tan(x)**: retorna a tangente de x;
• **math.degrees(x)**: converte o ângulo x de radianos para graus;
• **math.radians(g)**: converte o ângulo g de graus para radianos.
Também existem as **funções hiperbólicas** análogas, cujos os nomes sempre terminam com letra “h”. Ex.: **math.tanh(x)** retorna a tangente hiperbólica de x.

Módulo math (5/8)

Funções para Teste de Valores
• **math.isnan(x):** retorna True caso x seja NaN; caso contrário False é retornado;
• **math.isfinite(x):** retorna True caso x não seja infinito e nem NaN; caso contrário, False é retornado;
• **math.isinf(x):** retorna True caso x seja infinito (positivo ou negativo); caso contrário, False é retornado,

Módulo math (6/8)

```
# módulo math (Exemplo 1)
import math

# constante PI
print('PI=',math.pi)

# funções de arredondamento
x1 = 5.8
print(x1)
print('ceil',math.ceil(x1))
print('floor',math.floor(x1))
```

Saída:

```
PI= 3.141592653589793
5.8
ceil 6
floor 5
```

Módulo math (7/8)

```
# módulo math (Exemplo 2)
import math

# logaritmo
x2 = 1024
print('log de',x2,'na base 2:', math.log2(x2))

# raiz quadrada
x3 = 81
print('raiz quadrada de',x3,':', math.sqrt(x3))
```

Saída:

```
log de 1024 na base 2: 10.0
raiz quadrada de 81 : 9.0
```

Módulo math (8/8)

```
# módulo math (Exemplo 3 - Funções Trigonométricas)
import math

angulo_graus  = 30
angulo_radianos = math.radians(angulo_graus)
print('\n* * * Angulo=',angulo_graus, ' graus')
print('SENO =',round(math.sin(angulo_radianos),2))
print('COSSENO =',round(math.cos(angulo_radianos),2))
print('TANGENTE =',round(math.tan(angulo_radianos),2))
```

Saída:

```
* * * Angulo= 30  graus
SENO = 0.5
COSSENO = 0.87
TANGENTE = 0.58
```

- *Não esqueça que as funções trigonométricas exigem que o ângulo seja passado em radianos! Você pode converter um valor em graus para radianos usando a função math.radians()*