COMPUTAÇÃO I

Período 2020-2, turmas EN1 e EPT

Professor: Aloísio Pina

Expressões Aritméticas

São formadas por operadores aritméticos e operandos numéricos e têm resultado numérico.

Operadores aritméticos:

- soma: +
- subtração: -
- divisão: /
- multiplicação: *
- potenciação: **
- quociente inteiro da divisão: //
- resto inteiro da divisão: %

Exemplos:

In [1]:
```
5 + 2
```

Out[1]: 7

In [2]:
```
5 - 2
```

Out[2]: 3

In [3]:
```
5 / 2
```

Out[3]: 2.5

In [4]:
```
5 * 2
```

Out[4]: 10

In [5]:
```
5 ** 2
```

Out[5]: 25

In [6]:
```
5 // 2
```

Out[6]: 2

In [7]:
```
5 % 2
```

Out[7]: 1

ATENÇÃO: não use ponto ou x para multiplicação, não use barra invertida para divisão e não use acento circunflexo para potenciação!
Exemplos:

In [8]:
```
5 . 2
```

```
  File "<ipython-input-8-d948a991d7ec>", line 1
    5 . 2
      ^
SyntaxError: invalid syntax
```

In [9]:
```
5 x 2
```

```
  File "<ipython-input-9-993d2c48fc26>", line 1
    5 x 2
      ^
SyntaxError: invalid syntax
```

In [10]:
```
5 \ 2
```

```
  File "<ipython-input-10-3d62a96c7823>", line 1
    5 \ 2
        ^
SyntaxError: unexpected character after line continuation character
```

In [11]:
```
5 ^ 2
```

Out[11]:
```
7
```

Observação: em python e na maioria das linguagens de programação usa-se ponto decimal em vez de vírgula. A vírgula é usada para separar elementos de uma sequência.
Exemplo:

In [12]:
```
5.2
```

Out[12]:
```
5.2
```

In [13]:
```
5,2
```

Out[13]:
```
(5, 2)
```

Precedência de operações:

- 1) Operações entre parênteses
- 2) **
- 3) *, /, //, %
- 4) +, -

Obs. 1: operações de mesma precedência são feitas da esquerda para a direita.
Obs. 2: não se usam colchetes e chaves em expressões em python, usa-se hierarquia de parênteses.

Exemplo:

In [14]:
```
((5 * 2) // (4 - 1)) / 0.5
```

Out[14]:
```
6.0
```

Funções

Basicamente, são de 3 tipos:

1 - Funções Intrínsecas

Estão disponíveis para qualquer programa em python, sem a necessidade de importação.
Exemplo: função abs

In [15]:
```
abs(-7)
```

Out[15]: 7

2 - Funções em Módulos da Biblioteca Padrão

Necessitam ser importadas para serem usadas no seu programa.

É possível importar o módulo inteiro e usar só as funções que desejar.
A importação do módulo é feita usando "import" seguido do nome do módulo:

import modulo

Após importar o módulo, para usar as funções, deve-se especificar em que módulo estão seguido de ponto e da função com seus argumentos:

modulo.funcao(argumentos)

Exemplo: Importando e usando o módulo math, que contém funções e constantes matemáticas:

In [16]:
```
import math
math.sqrt(25)
```

Out[16]: 5.0

Outra possibilidade é importar apenas as funções que vai usar.
A importação é feita usando "from" seguido do nome do módulo, depois "import" e por fim os nomes das funções que deseja importar:

from modulo import f1, f2, f3...

Para usar as funções importadas, basta usar o nome da função e seus argumentos:

funcao(argumentos)

Exemplo: Importando a função sqrt (raiz quadrada) do módulo math:

In [17]:
```
from math import sqrt
sqrt(25)
```

Out[17]: 5.0

DICA: Para saber quais são as funções e constantes de um determinado módulo, basta importá-lo e usar a função intrínseca "help".
Exemplo:

In [18]:
```
import math
help(math)
```

```
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considered.

    atanh(x, /)
        Return the inverse hyperbolic tangent of x.

    ceil(x, /)
        Return the ceiling of x as an Integral.

        This is the smallest integer >= x.

    comb(n, k, /)
        Number of ways to choose k items from n items without repetition and without
order.

        Evaluates to n! / (k! * (n - k)!) when k <= n and evaluates
        to zero when k > n.

        Also called the binomial coefficient because it is equivalent
        to the coefficient of k-th term in polynomial expansion of the
        expression (1 + x)**n.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    copysign(x, y, /)
        Return a float with the magnitude (absolute value) of x but the sign of y.

        On platforms that support signed zeros, copysign(1.0, -0.0)
        returns -1.0.

    cos(x, /)
        Return the cosine of x (measured in radians).

    cosh(x, /)
        Return the hyperbolic cosine of x.
```

```
    degrees(x, /)
        Convert angle x from radians to degrees.

    dist(p, q, /)
        Return the Euclidean distance between two points p and q.

        The points should be specified as sequences (or iterables) of
        coordinates.  Both inputs must have the same dimension.

        Roughly equivalent to:
            sqrt(sum((px - qx) ** 2.0 for px, qx in zip(p, q)))

    erf(x, /)
        Error function at x.

    erfc(x, /)
        Complementary error function at x.

    exp(x, /)
        Return e raised to the power of x.

    expm1(x, /)
        Return exp(x)-1.

        This function avoids the loss of precision involved in the direct evaluation
of exp(x)-1 for small x.

    fabs(x, /)
        Return the absolute value of the float x.

    factorial(x, /)
        Find x!.

        Raise a ValueError if x is negative or non-integral.

    floor(x, /)
        Return the floor of x as an Integral.

        This is the largest integer <= x.

    fmod(x, y, /)
        Return fmod(x, y), according to platform C.

        x % y may differ.

    frexp(x, /)
        Return the mantissa and exponent of x, as pair (m, e).

        m is a float and e is an int, such that x = m * 2.**e.
        If x is 0, m and e are both 0.  Else 0.5 <= abs(m) < 1.0.

    fsum(seq, /)
        Return an accurate floating point sum of values in the iterable seq.

        Assumes IEEE-754 floating point arithmetic.

    gamma(x, /)
        Gamma function at x.

    gcd(x, y, /)
        greatest common divisor of x and y

    hypot(...)
        hypot(*coordinates) -> value

        Multidimensional Euclidean distance from the origin to a point.

        Roughly equivalent to:
```

```
        sqrt(sum(x**2 for x in coordinates))

    For a two dimensional point (x, y), gives the hypotenuse
    using the Pythagorean theorem:  sqrt(x*x + y*y).

    For example, the hypotenuse of a 3/4/5 right triangle is:

        >>> hypot(3.0, 4.0)
        5.0

isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
    Determine whether two floating point numbers are close in value.

      rel_tol
        maximum difference for being considered "close", relative to the
        magnitude of the input values
      abs_tol
        maximum difference for being considered "close", regardless of the
        magnitude of the input values

    Return True if a is close in value to b, and False otherwise.

    For the values to be considered close, the difference between them
    must be smaller than at least one of the tolerances.

    -inf, inf and NaN behave similarly to the IEEE 754 Standard.  That
    is, NaN is not close to anything, even itself.  inf and -inf are
    only close to themselves.

isfinite(x, /)
    Return True if x is neither an infinity nor a NaN, and False otherwise.

isinf(x, /)
    Return True if x is a positive or negative infinity, and False otherwise.

isnan(x, /)
    Return True if x is a NaN (not a number), and False otherwise.

isqrt(n, /)
    Return the integer part of the square root of the input.

ldexp(x, i, /)
    Return x * (2**i).

    This is essentially the inverse of frexp().

lgamma(x, /)
    Natural logarithm of absolute value of Gamma function at x.

log(...)
    log(x, [base=math.e])
    Return the logarithm of x to the given base.

    If the base not specified, returns the natural logarithm (base e) of x.

log10(x, /)
    Return the base 10 logarithm of x.

log1p(x, /)
    Return the natural logarithm of 1+x (base e).

    The result is computed in a way which is accurate for x near zero.

log2(x, /)
    Return the base 2 logarithm of x.

modf(x, /)
    Return the fractional and integer parts of x.
```

```
        Both results carry the sign of x and are floats.

    perm(n, k=None, /)
        Number of ways to choose k items from n items without repetition and with or
der.

        Evaluates to n! / (n - k)! when k <= n and evaluates
        to zero when k > n.

        If k is not specified or is None, then k defaults to n
        and the function returns n!.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    pow(x, y, /)
        Return x**y (x to the power of y).

    prod(iterable, /, *, start=1)
        Calculate the product of all the elements in the input iterable.

        The default start value for the product is 1.

        When the iterable is empty, return the start value.  This function is
        intended specifically for use with numeric values and may reject
        non-numeric types.

    radians(x, /)
        Convert angle x from degrees to radians.

    remainder(x, y, /)
        Difference between x and the closest integer multiple of y.

        Return x - n*y where n*y is the closest integer multiple of y.
        In the case where x is exactly halfway between two multiples of
        y, the nearest even value of n is used. The result is always exact.

    sin(x, /)
        Return the sine of x (measured in radians).

    sinh(x, /)
        Return the hyperbolic sine of x.

    sqrt(x, /)
        Return the square root of x.

    tan(x, /)
        Return the tangent of x (measured in radians).

    tanh(x, /)
        Return the hyperbolic tangent of x.

    trunc(x, /)
        Truncates the Real x to the nearest Integral toward 0.

        Uses the __trunc__ magic method.

DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
    tau = 6.283185307179586

FILE
    (built-in)
```

3 - Funções Definidas pelo Programador

É preciso que sejam definidas antes de serem usadas. A definição é semelhante a definição matemática de uma função:

Definição matemática: Funcao(parâmetros) = cálculos com os parâmetros

Chamada da função: Funcao(argumentos)

Os argumentos servem de valores para os parâmetros da função, que são usados nos cálculos e a função "retorna" o resultado.

Em python, a definição fica assim:

In [ ]:
```
def Funcao(parâmetros):
    return cálculos com os parâmetros
```

Exemplo: função que recebe como entrada dois valores e retorna a soma.

Definição matemática: Soma(x, y) = x + y

Chamada da função: Soma(5, 2) => 7

Definição em python:

In [19]:
```
def Soma(x, y):
    return x + y
```

Chamada da função:

In [20]:
```
Soma(5, 2)
```

Out[20]: 7