Data Science & Big Data

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Operadores e Pacotes

Prof. André Grégio

DSBD

Outros operadores

Além de adição (+), subtração (-), divisão (/), multiplicação (*) e exponenciação (**), temos os operadores de:

- Piso (*floor*): //
- Resto: %
- Raiz: x**(1/n)
- Precedência: ()

Outros operadores - Exemplos

- Piso (*floor*):
  - 3/2 = 1.5 (divisão)
  - 3//2 = 1
  - -3//2 = ?
- Resto:
  - 5%2 = 1
- Raiz:
  - 9**(1/2) = 3.0
- Precedência:
  - 4-2/2 = 3
  - (4-2)/2 = 1

Outros operadores

- Comparação:
  - ==, !=, >, <, >=, <=
- Atribuição:
  - =
  - [+, -, *, /, %, **, //]=
- Lógicos:
  - and, or, not
- Associação e Identidade:
  - in, not in; is, is not

Outros operadores - Exemplos

- Comparação:
  - A == 5
  - B == 3
  - A < B?
- Atribuição:
  - A += B?
  - A *= B?
- Lógicos:
  - X = 0
  - Y = 1
  - X and Y?
  - X or Y?
- Associação/Identidade:
  - X is not Y?
  - X in [0, 2]?

Pacotes e módulos

A linguagem Python possui vários “pacotes” com algoritmos já prontos para inúmeros fins, além dos tradicionais (como sys, math e os):

- Scikit-bio, para bioinformática
- Scikit-learn para data mining e data analysis, construído com:
- NumPy, para vetores N-dimensionais
- SciPy, para computação científica em geral
- Matplotlib, para geração de gráficos 2D
- Dezenas de outros pacotes para machine learning, estatística e big data analytics

Pacotes e módulos

Benefícios:

- **Simplicidade:** módulos se concentram em pequenas porções do problemas em vez de problema todo
- **Manutenção:** são projetados para impor limites lógicos entre diferentes domínios de problemas minimizando interdependência.
- **Reuso de código:** uma funcionalidade definida em um módulo pode ser facilmente reutilizada por outras partes do aplicativo por meio de uma interface adequadamente definida.

Pacotes e módulos

Um **pacote** é uma coleção de **módulos**

Um **módulo** é simplesmente um *arquivo Python*

Módulos em um pacote estão dispostos em uma estrutura hierárquica:

```
:/usr/lib/python3.8$ ls *.py
...
numbers.py
opcode.py
operator.py
optparse.py
os.py
...
```

```
:/usr/lib/python3.8/http$ ls

client.py
cookiejar.py
cookies.py
__init__.py
__pycache__
server.py
```

Como os módulos funcionam

No início do script, deve-se “importar” o módulo a ser utilizado, como feito em outras linguagens

- `#include<stdio.h>` em C
- `uses crt;` em PASCAL
- `import NOME_DO_MÓDULO` em Python

O módulo importado conterá funções que implementam algoritmos de acordo com o objetivo em questão e podem ser utilizados no seu programa!

Funções de um módulo

- Uma função é uma sequência de instruções para realizar uma dada tarefa (similar a uma função matemática)

- Um módulo é geralmente composto por funções relacionadas, que também podem ser chamadas de **métodos**

- É possível selecionar funções de um módulo, por exemplo:
  - Importação completa → import MODULO
  - Importação seletiva → from MODULO import func1, func2

Chamadas de funções de módulos

- Uma função é chamada por seu **nome**
- A entrada de uma função é seu **argumento**
- A saída é o **valor de retorno** ou resultado
- Exemplo, função “type()”:
  - A = 42 → variável com um inteiro atribuído a ela
  - type(A) → *`type`* é o **nome da função**,
    A é o **argumento**
  - <type ‘int’> → **valor de retorno**, i.e., resultado da aplicação da função

Chamadas de funções de módulos

Python provê funções para conversão de tipo:

- `int('42')` → converte o argumento para inteiro, se possível
- `int(3.1415)` → valor de retorno é 3
- `float(42)` → valor de retorno?
- `str(42)` → valor de retorno?

Exemplo: funções de um módulo

Um módulo básico de Python é o de funções matemáticas (*math*)

- Agora temos funções para calcular:
  - Raiz quadrada → sqrt()
  - Logaritmo → log10()
  - Seno → sin()

- Onde fica esse módulo?
  - Não achei *math.py* no /usr/lib/Python...

Exemplo: funções de um módulo

- Alguns módulos são escritos em C para eficiência, portanto são *built-in* e não possuem um .py no diretório padrão de módulos.

```
import sys
sys.builtin_module_names
('_ast', '_bisect', '_blake2', '_codecs', '_collections', '_datetime', '_elementtree',
'_functools', '_heapq', '_imp', '_io', '_locale', '_md5', '_operator', '_pickle',
'_posixsubprocess', '_random', '_sha1', '_sha256', '_sha3', '_sha512', '_signal',
'_socket', '_sre', '_stat', '_string', '_struct', '_symtable', '_thread', '_tracemalloc',
'_warnings', '_weakref', 'array', 'atexit', 'binascii', 'builtins', 'cmath', 'errno',
'faulthandler', 'fcntl', 'gc', 'grp', 'itertools', 'marshal', 'math', 'posix', 'pwd',
'pyexpat', 'select', 'spwd', 'sys', 'syslog', 'time', 'unicodedata', 'xxsubtype',
'zipimport', 'zlib')
```

Como usar o módulo *math*?

Exemplo: funções de um módulo

Para usar funções de um módulo, aplica-se a Notação “ponto”

- Notação “.” → *módulo.função*
  - Exemplo, math:

```
>>> import math
>>> N = 400
>>> Res = math.sqrt(N)
>>> print(Res)
20.0
```

Exemplo: funções de um módulo

E se eu utilizar importação seletiva?

```
from math import sqrt
```

- Evita notação ".", pois posso chamar apenas `sqrt()` no programa
- Porém, tem menos legibilidade, pois o sqrt() fica "perdido" no código

```
>>> from math import sqrt
>>> N = 400
>>> Res = sqrt(N)
>>> print(Res)
20.0
```

Outras funções de um módulo

- Para descobrir quais funções estão incluídas em um módulo e como utilizá-las, consulte a documentação:

https://docs.python.org/3/library/math.html

Por exemplo:

- Qual função usar para calcular o logaritmo natural de um número?
- E o logaritmo em uma dada base?

Outras funções de um módulo

- Documentação:

https://docs.python.org/3/library/math.html

Exemplo:

- Qual função usar para calcular o logaritmo natural de um número?
  - math.log(x) → x na base “e”

- E o logaritmo em uma dada base?
  - math.log(x, b) → x na base “b”

Listar funções de um módulo

- Outra forma para listar funções de um módulo é pela função “dir()”:

```
>>> import math
>>> x = dir(math)
>>> print(x)
['__doc__', '__loader__', '__name__', '__package__', '__spec__',
'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp',
'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf',
'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf',
'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh',
'tau', 'trunc']
```

Composições básicas

- Uma variável pode receber uma chamada de função
  - A função pode receber como argumento uma composição de funções e outras variáveis
- Exemplos:
  - `v = 4/3*math.pi*r**3`

  ou
  - `v = 4/3*math.pi*math.pow(r,3)`

- Para ilustrar,
  - `v = 4/3*math.pi*math.pow(r,math.sqrt(9))`

Exercícios

1. Refaça o exercício da fórmula de Bhaskara da seguinte forma:
   a. Com quatro raízes de saída (x1_0 e x2_0, com o resultado real, e x1_1 e x2_1 com resultado arredondado para baixo);
   b. Use funções do módulo math, sempre que possível;
   c. Calcule o erro resultante do arredondamento.
2. Escreva uma instrução que calcule a raiz cúbica de uma variável “x”.
3. Qual o resultado da expressão “23+7*50”?
   a. Como obter o resultado 1500 da expressão acima? Escreva essa instrução em Python.
4. Como saber se um número é par ou ímpar? Escreva uma instrução que resolva esse problema.
5. Escreva uma instrução que calcule a área de um círculo (use função pow())