Big Data & Data Science

Funções em Python

Operadores

Comparação:

- ==, !=, <>, >, <, >=, <=

Atribuição:

- =, [+, -, *, /, %, **, //]=

Lógicos:

- and, or, not

Associação e Identidade:

- in, not in; is, is not

Operador de concatenação

Dois operadores aritméticos podem ser usados em strings

- +
  - Var1 = “bom”
  - Var2 = “dia”
  - Var1+Var2
    - bomdia
- *
  - ‘Ai’*3
    - ‘AiAiAi’

Usando o interpretador

Usar o interpretador Python como calculadora:

- Se o volume de uma esfera com raio *r* é $\frac{4}{3}\pi r^3$, qual o volume de uma esfera com raio 3?
- Suponha que o preço de um livro é R$ 24,95 com desconto de 40% para livrarias. O frete é de R$ 3,00 para a primeira cópia e R$ 0,75 para cada cópia adicional. Qual o custo total para 60 cópias encomendadas por uma livraria?
- Uma pessoa sai de casa às 6:52 AM e corre 1 milha em ritmo leve (8’15’’ por milha), 3 milhas em ritmo rápido (7’12’’ por milha) e 1 milha em ritmo leve novamente. Que horas essa pessoa está de volta em sua casa? Se 1 milha é aproximadamente 1,61 km, qual a velocidade dessa pessoa em média em km/h?

Scripts em Python

Além do modo “interpretador” (interativo), podemos trabalhar em um modo que salva o código em arquivos, para depois o executarmos no **modo script**.

- Extensão dos arquivos: .py

- Para executar em modo *script*:
  - $ python arquivo.py

Anaconda

- Plataforma para *data science* e *machine learning* em Python
- *Open source,* inclui pacotes populares para ciência dos dados e facilita o gerenciamento de ambientes diversos
  - Scikit-learn, TensorFlow, SciPy, etc.
- https://www.anaconda.com

Instalando o Anaconda

- Download para sistemas operacionais Linux, Windows, OS X
  - *Script* sh com os binários embutidos
  - Para instalar, vá em ~/Downloads na máquina disponibilizada para o curso e execute o seguinte comando:

```
sh Anaconda3-5.1.0-Linux-x86_64.sh
```

  - Preste atenção nas instruções!

Editores para programação

- Uso para criação de código em modo *script*
- Sabores:
  - vi/vim
  - nano
  - Emacs
  - Sublime
  - Spyder (vem junto com Anaconda)
  - Qualquer outro de sua preferência… ;-)

Exercícios

- Criar *scripts* para:
  - Calcular o fatorial de um dado “n”
  - Mostrar a sequência de Fibonacci até o “n-ésimo” número
  - Com o que foi aprendido até aqui, calcular a raiz quadrada de “n”
  - Dados “base” e “altura”, calcular a área do triângulo retângulo, armazenar em uma variável “area” e mostrar os tipos de cada variável ao final do cálculo
    - Deu certo?
  - **CUIDADO COM SEUS DADOS CIENTÍFICOS!!!**

Divisão com resultado “real”

Python 3:

- >>> area = 3*5/2
- >>> area
- 7.5

Python 2:

- >>> area = 3*5/2
- >>> area
- 7

Divisão com resultado “real”

Como “consertar” a divisão em Python 2?

- >>> area = 3*5/**2.0**
- >>> area
- 7.5

Compatibilidade com Python 3:

- >>> **from __future__ import division**
- >>> area = 3*5/2
- >>> area
- 7.5

E se eu quiser usar um algoritmo pronto?

Tipos de pacotes/módulos

Como mencionado, Anaconda possui vários “pacotes” com algoritmos já prontos:

- 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*

O que é um módulo?

- Um módulo é simplesmente um arquivo Python
- Um pacote é uma coleção de módulos
  - Módulos em um pacote estão dispostos em uma estrutura hierárquica de diretórios

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

- import NOME_DO_MÓDULO em Python

O módulo conterá **funções** que implementam algoritmos de acordo com o objetivo em questão!

Funções

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

Chamadas de funções

- 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:
  - 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

- 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

- Para usá-lo, `import math`
- Agora temos funções para calcular:
  - Raiz quadrada → `sqrt()`
  - Logaritmo → `log10()`
  - Seno → `sin()`
- Como usar?

Exemplo: funções de um módulo

Notação “.”

- MODULO.funcao
- Exemplo:
  - `import math`
  - `N = 400`
  - `Res = math.sqrt(N)`
  - `print(Res)`

Exemplo: funções de um módulo

Como descobrir as funções de um módulo?

- https://docs.python.org/2/library/math.html

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

Exemplo: funções de um módulo

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”

Composição de funções

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)

Definição de funções

Podemos adicionar novas funções em Python criando-as!

- Definição de função: especifica o nome da função e as instruções a serem executadas quando a função é chamada
- Palavra reservada: **def**
- Exemplo:
  - def funcao():
    - Instrucao1
    - ...
    - InstrucaoN

Definição de funções

- def funcao(): → cabeçalho!
  - Instrucao1 … InstrucaoN → corpo da função

Lembrar:

- Colocar “:” após nome da função no cabeçalho
- Identar o corpo da função corretamente

Crie uma função simples

Função imprimeRefrao()

- Deve conter o refrão de uma música
- Cada instrução imprime uma sentença

Função repeteRefrao()

- Chama imprimeRefrao() “n” vezes

Chame a função imprimeRefrao no modo *script*

Exercícios

Passe os códigos feitos para funções.

- Crie uma função que converta de graus para radianos
- Use funções do módulo *math* e funções que você já implementou para criar uma função que calcule o SENO de um dado ângulo
- Crie uma função em que, dado um número inteiro entre 0 e 10, mostre a tabuada deste número
  - Crie um programa que mostre de maneira formatada as tabuadas do 0 a 10, sem interação do usuário

Exercícios

Comparação de tempos

- Por que é importante?
  - Aplicações científicas → complexidade
- Compare os tempos de execução:
  - Sua função SENO com a função `sin()` do módulo `math`
  - Sua função FATORIAL com a `factorial()` do módulo `math`

Dicas

Conversão de graus para radianos:

- Rad = graus/180 * pi

Seno:

- Série de Taylor:
  - seno(xRad) = $x - x^3/3! + x^5/5! - x^7/7! + x^9/9! - ...$