数据挖掘实践

丁兆云

第一章 Python基础知识

1.1 Python基础知识

1.2 Python的特点

1.3 安装和运行Python

1.4 标识符和语法

1.5 内存管理

Python的起源

- 荷兰人Guido van Rossum发明了Python，并以BBC在1960-1970年间播放的室内情景幽默剧Monty Python’s Flying Circus来命名
- 1991年， 第一个Python编译器（也是解释器）诞生，用C语言实现，拥有类、函数、异常和包括列表和字典在内的核心数据类型
- Python现有两个版本：Python 2和Python 3。后者是更新的版本，且不与前者向后兼容；Python提供代码转换工具

- 为什么我们需要Python?
  - **软件质量：在与其他脚本语言相比，Python对可读性、一致性和软件质量更为关注**
    - 可读性意味着Python代码易于重用和维护
    - Python具有对更为高级软件重用机制的深层支持，这些机制包括面向对象和函数式编程（function programming, FP*）

* FP是一种编程范式，其中计算是通过对函数的求值来组织的，它尽量避免涉及状态与可变数据。FP强调的是函数的施用，与之相对的命令式编程则强调状态的改变。

- **软件开发者的生产率：与编译型语言或静态类型语言C、C++、和Java相比，Python可以数倍地提高软件开发者的生产率**
  - Python代码长度通常只是等价C++或Java代码的1/5至1/3
  - 类型检查、调试和维护工作更少
  - Python程序在完成后可以立即执行，无需其他工具所需要的冗长的编译和链接步骤

- **程序的可移植性：大部分的Python程序可以不经修改直接在大部分主流平台上运行**
  - 在Linux和Windows之间进行Python代码移植，通常仅需要在机器之间进行脚本代码的拷贝
  - Python为编写可移植图形用户界面、数据库访问程序和基于Web的系统的代码编写提供多种操作
  - 包括用于程序启动和目录处理的操作系统接口在Python中也具有很好的可移植性

- **支持标准库：Python拥有大量预建和可移植的函数，也称为标准库（standard library）**
  - 可以使用自行开发软件和大量的第三方应用支持软件对Python进行扩展
  - Python的第三方领域提供了用于网站建设、数字编程、串行端口访问、游戏开发等多种工具
  - 举例来说，NumPy就是一个用于科学计算的、免费的、比数字编程系统Matlab功能更为强大的Python扩展包

– **组件集成：Python脚本可以通过大量的集成机制与应用中的其他部分进行通信**

- Python在集成中是进行产品定制和扩展的工具
- Python代码可以调用C和C++的库，可以从C和C++程序中被调用，可以与Java和.NET组件集成，可以通过串行端口与其他设备接口，还可以利用包括SOAP（Simple Object Access Protocol）、XML-RPC（Remote Procedure Call）和CORBA（Common Object Request Broker Architecture）在内的接口在网络上进行交互

- **乐趣：Python易于使用和具有大量内建工具集的特点，使得编程行为更具有乐趣而非令人讨厌的琐事**

第一章 Python基础知识

1.1 Python基础知识

1.2 Python的特点

1.3 安装和运行Python

1.4 标识符和语法

1.5 内存管理

- Python的不足之处
  - **Python唯一的主要不足之处是它的执行速度不如编译型语言（诸如C和C++）那样快**
  - 目前Python的标准实现中将源代码语句编译成一种称为字节码的中间形式（而非二进制机器码），然后解释执行字节码
  - 对执行速度的关注取决于你的程序类型，无论是处理文件还是创建GUI，你的程序运行将与使用C一样快，因为这样的任务将发送给解释器中编译的C代码执行
  - 执行速度的损失由开发速度的提高来弥补

- Python的技术优势
  - **面向对象**：
    - Python的类模式（class model）支持多态性、操作符重载和多重继承
    - Python拥有简单的语法和类型，便于面向对象编程的应用
    - Python的面向对象的特性使其成为适合其它面向对象程序设计语言的脚本工具，例如，Python程序可以继承C++、Java和C#程序中的类，从而成为其子类
    - 对Python程序设计而言，面向对象是一种选择（option）；和C++一样，Python同时支持过程化编程和面向对象编程

– 免费：

- 和其它开源软件，例如Tcl、Perl和Linux一样，我们可以直接从Internet上免费下载Python的系统源代码
- “免费”并非意味着“没有技术支持”，事实上，Python在线社区回复用户的问题非常迅速；此外，由于Python自带完整的源代码，开发者可以自行研究和修改该编程语言的实现
- Python的开发由一个专门的社区来协调完成，这个社区包括了Python的创建者—Guido van Rossum、Benevolent Dictator for Life（BDFL）和数以千计的支持者；语言改变过程严格

– 可移植：

- Python的标准实现是用ANSI C写的，因此可以在几乎所有的主流平台上编译和运行，例如
  - Linux和Unix
  - Microsoft Windows
  - Mac OS
  - 运行Symbian OS和Windows Mobile的移动电话
  - 游戏控制台和iPods
  - 运行Google的Android和Apple的iOS的平板计算机和智能电话
- Python程序自动编译成可移植的二进制代码，然后在安装了兼容的Python解释器的平台上运行

– 功能强大：

- Python是一种混合物：介于传统脚本语言和系统开发语言之间，提供了脚本语言的简单和易用，以及编译语言中高级的软件工程工具
- 支持动态类型：Python代码不需要复杂的类型和大小声明
- 自动内存管理：自动进行对象的存储位置分配和垃圾回收
- 支持大型程序开发：在开发大型的系统时，Python提供模块、类和异常等工具，因此支持将系统组织成组件，使用面向对象编程来重用和定制代码，以及优雅地处理时间和错误

– 功能强大（继续）：

- 内建对象类型：Python提供常用的数据结构，包括列表、字典和字符串；内建对象可以根据需要伸展和收缩，同时还可以任意嵌套
- 内建工具：为了处理上述对象类型，Python提供了强大的操作，其中有合并、分片、排序和映射等等
- 软件库工具：Python拥有大量的预编码软件库，可以用来完成正则表达式匹配和网络编程等各项任务
- 第三方工具：因为Python是开源的，开发者可以自行编写程序来完成Python自身所不能支持的任务

– 可以和其它编程语言混搭：

- Python程序可以通过“胶水”与使用其它语言编写的组件“粘”在一起，例如通过Python的C API可以从C程序中调用Python程序（或相反）
- 这种“混搭”，即将Python与使用其它语言编写的软件库混合使用，使其成为一种适合快速原型开发的重要工具：利用Python进行快速开发，然后转移到C中进行发布

– 易于学习和使用：

- 作为稍有经验的程序员，可以花费几小时掌握该语言中的特定部分，或者花上几天时间编写某个小型的Python程序
- 当然，成为Python专家没有如此之快！大多数人认为Python学习曲线与其它编程工具相比更为温和
- 运行Python程序意味着你仅需将代码写出来然后运行即可；没有编译和链接的环节
- 交互式编程体验：程序发生改变后几乎可以立刻看到这种变化带来的影响
- Python提供的语法特别简单，内建工具功能强大

第一章 Python基础知识

1.1 Python基础知识

1.2 Python的特点

1.3 安装和运行Python

1.4 标识符和语法

1.5 内存管理

安装Python

- 在Windows系统中安装Python，可以参照以下步骤
  - 打开浏览器，访问www.python.org
  - 点击Download链接
  - 在靠近最下方的Files列表中选择Windows x86 MSI installer，点击该链接下载安装文件
  - 将下载的python-x.x.x.msi（x.x.x是版本号）文件保存在计算机中的合适位置
  - 在Windows的资源管理器中双击运行该文件，就能根据安装向导的指示开始安装过程，结束后即可食用Python

第一章 Python基础知识

1.1 Python基础知识

1.2 Python的特点

1.3 安装和运行Python

1.4 标识符和语法

1.5 内存管理

标识符

- 什么是标识符？— **标识符是用来识别包括变量和函数等元素的名字**
- 标识符是由字母、数字和下划线组成的**任意长度**的字符序列
- 标识符的开始字符可以是字母和下划线，但**不能是数字**，例如number1, number2, area, Area和AREA
- **标识符不能是关键字** — 关键字，也叫保留字，在Python具有特殊的含义，例如import

标识符

- 描述性的标识符能够增加程序的可读性，因此应该尽量避免使用缩略语，例如numberofTeachers就比numTea, numOfTea或numOfTeachers更好一些

- **标识符命名规则**：
  - 变量名中使用小写字母，例如radius和area
  - 如果一个名字由多个字组成，那么把所有字合并，将一个字所有字母小写，其后所有字的首字母大写，例如numberOfTeachers

变量

- **变量是指向（或引用）内存中存储的值的名字**，之所以称之为“变量”，是因为它们可能会指向不同的值

```
>>>
>>>
>>> radius = 1.0
>>> area = radius * radius * 3.14159
>>> print("The area is", area, "for radius", radius)
The area is 3.14159 for radius 1.0
>>>
>>> #compute another area
>>> radius = 2.0
>>> area = radius * radius * 3.14159
>>> print("The area is", area, "for radius", radius)
The area is 12.56636 for radius 2.0
>>>
```

赋值语句

- 将值赋给变量的语句称为赋值语句，在Python中“=”号是赋值操作符

- 赋值操作的语法为

  variable = expression

- 其中expression为表达式，是由数值、变量和操作符组成的、可以求得数值的有意义的组合，例如

```
y = 1
radius = 1.0
x = 6 * (6 / 2) + 4
x = x + y
```

同时赋值

- Python支持**同时赋值**（simultaneous assignment），其语法为

  var1, var2, …, varn = exp1, exp2, …, expn

这个语句告诉Python将“=”号右侧所有表达式求值，然后依次赋值给“=”号左侧的变量

- 考虑交换变量x和y中的值的一般做法

```
>>> x = 1
>>> y = 2
>>> temp = x
>>> x = y
>>> y = temp
>>> print("x = ", x, "y = ", y)
x =  2 y =  1
>>>
```

同时赋值

- 在Python中我们只需要

```
>>> x = 1
>>> y = 2
>>> x, y = y, x
>>> print("x = ", x, "y = ", y)
x =  2 y =  1
>>>
```

- 同时赋值还可以用来从一条语句中获取多个输入的值，例如

```
# Prompt the user to enter three numbers
number1, number2, number3 = eval(input("Enter three numbers separated by commas: "))

# Compute average
average = (number1 + number2 + number3) / 3

# Display result
print("The average of", number1, number2, number3, "is", average)
```

数据类型

- **数字数据**（numeric data）包括两种：整型和实型，在Python中带有小数点的数字即为实数，例如3.5和1.0，而3和1都是整数

- 在Python中，下列对象都相当于False： [], (), {}, 0, None, 0.0, ‘’可以使用内置函数bool()来检测

操作符

- 用于数字数据的操作符包括标准的算术操作符：

+	加法
–	减法
*	乘法
/	实数除法
//	整数除法
**	求幂
%	取余

操作符

- 操作符“–”可以是一元操作符，也可以是二元操作符。当其作为一元操作符使用时，例如-5，表示取数字5的负值

- 操作符“/”执行实数除法操作，得到的结果为实数，例如

```
>>>
>>> 4 / 2
2.0
>>> 2 / 4
0.5
```

操作符

- 操作符“//”执行整数除法操作，得到的结果为整数

```
>>>
>>> 4 // 2
2
>>> 2 // 4
0
>>>
```

- 操作符“%”执行取余数操作，得到除法之后的余数，该操作在编程中的用处很大
  - 例如判断奇、偶数（奇数%2为1，偶数%2为0）
  - 还可以判断类似10天之后是周几这样的问题

第一章 Python基础知识

1.1 Python基础知识

1.2 Python的特点

1.3 安装和运行Python

1.4 标识符和语法

1.5 内存管理

动态类型语言

- Python是**动态类型语言**（dynamically typed language），其特点是变量内存地址的分配和类型的检查是在运行（而非编译时）时进行的

- 动态类型语言的核心是**对象和引用分离**。例如

  a = 5

  a = “hello”

  – 5是内存中的一个对象，a是指向这个对象的一个引用

  – 当引用a指向另外一个字符串对象时，对象5就没有引用指向它，此时Python会将这个没有引用指向的对象销毁，释放相应的内存

```
a = 5
b = a
a = 7
```

- 前两句让a指向内存中的对象5，然后让引用b也指向同一对象
- 第三句对引用a重新赋值，此时a指向另外一个对象7，引用a和b指向不同的内存对象

```
a = [1,2,3,4,5]
b = a
a[0] = 7
```

print(a)和print(b)分别是什么？

```
>>>
>>> a = [1,2,3,4,5]
>>> b = a
>>> a[0] = 7
>>> print(a)
[7, 2, 3, 4, 5]
>>> print(b)
[7, 2, 3, 4, 5]
>>>
```

- a[0] = 7 这一语句改变的并不是引用a的指向，而是列表a中的元素a[0]，因此所有指向该列表的引用都受到影响

- 列表可以通过引用其元素，改变对象自身，这类对象称为**可变数据对象**（mutable object）

- 数字和字符串，不能改变对象本身，只能改变引用的指向，这类对象称为**不可变数据对象**（immutable object）

动态类型与函数传递

```
>>> def f(x):
        x = 100
        print(x)
```

```
>>> a = 1
>>> f(a)
100
>>> print(a)
1
```

- 参数x是一个引用,指向a所指向的对象
- 如果参数是不可变的数据对象,则x和a相互独立,对参数x的操作不会影响引用a

```
>>> def f(x):
        x[0] = 100
        print(x)
```

```
>>> a = [1,2,3]
>>> f(a)
[100, 2, 3]
>>>
```

- 如果参数是可变的数据对象,改变参数,有可能改变原对象,因此所有指向该对象的引用都会受影响

引用计数

- Python中，每个对象都存有指向该对象的引用总数，即**引用计数**。如果需要查看某个对象的引用计数，可以使用sys包中的getrefcount()

```
>>> from sys import getrefcount
>>> a = (1,2,3)
>>> print(getrefcount(a))
2
>>>
>>> b = a
>>> print(getrefcount(b))
3
>>>
```

**需要注意的是，当某个引用作为参数传递给getrefcount()时，会创建一个临时引用

垃圾回收

- 当Python中某个对象的引用计数降为0时，说明没有任何引用指向该对象，该对象将作为垃圾被回收
- 频繁的垃圾回收将大大降低Python的工作效率，所以Python只会在特定条件下，自动启动垃圾回收
- 垃圾回收启动的条件是Python中分配对象（object allocation）和取消分配对象（object deallocation）的次数的差值高于某个阈值（如下所示，此处为700）

```
>>> import gc
>>> print(gc.get_threshold())
(700, 10, 10)
>>>
```

分代回收

- 分代回收策略的基本假设是：**存活时间越久的对象，越不可能在后面的程序中变成垃圾**

- 回收策略：
  - Python将所有对象分为0、1和2代，新建对象为第0代
  - 某代对象经历过垃圾回收依然存活，即被归入下一代
  - 垃圾回收时启动对0代对象扫描，0代经过一定次数垃圾回收则会启动对1代对象的垃圾回收；1代经过一定次数垃圾回收则会启动对2代对象的垃圾回收

内存池

- Python中引入内存池机制，用于对小块内存的分配和释放，即Pymalloc机制
  - 当申请的内存小于256字节时，PyObject_Malloc会在内存池中申请内存
  - 当申请的内存大于256字节时，PyObject_Malloc的行为将转化为malloc的行为
- Python对象都有其独立的私有内存池，对象间不共享内存池
- 不用的内存放到内存池而不是返回给操作系统

小结

- Python的技术优势：面向对象、免费、可移植、功能强大、可以与其他编程语言混搭、易于学习和使用
- Python唯一的不足之处是执行速度不如编译型语言（诸如C和C++）那样快
- Python是一种动态类型的语言，对象和引用分离
- Python的内存管理包括垃圾回收、分代回收和内存池管理