Getting Started with Julia Programming

**前言**

julia是一种新的编程语言，在麻省理工学院应用计算小组教授Alan Edelman的监督下开发完成。它的开发始于2009年，并于2012年2月首次公开发布。它仍然是一种相当年轻的语言，但其基础很稳定；自0.1版本以来，核心语言没有向后不兼容的更改。它基于清晰而坚实的原则，并且在技术、数据科学家和高性能计算领域的受欢迎程度正在稳步上升。我们概述了Julia所基于的原理，并将它们与其他语言进行比较。

**涉及内容**

**第一章** julia的平台安装，指导您安装julia环境所需的所有必要组件。它教会您如何使用Julia的控制台(REPL)，并讨论您可以使用的一些更复杂的开发编辑器。

**第二章** 讨论了julia中的基本内置类型：变量、类型和操作，以及可以对它们执行的操作，以便您准备开始使用它们编写代码。

**第三章** 函数，解释了为什么函数是Julia的基本构件，以及如何有效地使用它们。

**第四章** 控件流，显示了Julia优雅的控件结构、如何执行、错误处理以及如何使用coroutines(在Julia中称为Tasks)来构造代码的执行。

**第五章** 集合类型，探索分组单个值的不同类型，如数组和矩阵、元组、字典和集合。

**第六章** 更多关于类型、方法和模块的内容将深入到类型概念中，并解释如何在多个分派中使用它来获得类c的性能。模块，一个更高级的代码组织概念，也被讨论。

**第七章** 在Julia中，元编程触及了Julia的更深层次，比如表达式和反射功能，并演示了宏的强大功能。

**第八章** I/O、网络和并行计算展示了如何使用DataFrames处理文件和数据库中的数据。

**第九章** 运行外部程序，查看Julia如何与命令行和其他语言交互，并讨论性能技巧。

**第十章** 标准库和包深入到标准库中，并演示了数据可视化的重要包。

**附录**，宏和包的列表。为您提供了这本书中使用的宏和包的方便参考列表。

**约定**

在这本书中，您将会发现许多不同类型的信息的文本样式。下面是这些风格的一些例子和解释它们的意义。文本、数据库表名、文件夹名、文件名、文件扩展名、路径名、虚拟url、用户输入和Twitter句柄中的代码如下所示：“例如，我们使用数据文件winequality.csv。包含1599个样本测量值的, 12列数据。”

代码块的设置如下：

using DataFrames

fname = “winequality.csv”

data = readtable(fname,separator = ‘;’)

当我们希望将您的注意力吸引到代码块的某个特定部分时，相关的行或项以粗体显示。

using DataFrames

fname = “winequality.csv”

**data = readtable(fname,separator = ‘;’)**

任何命令行输入或输出如下所示:

julia main.jl

**新术语**和**重要词**会用粗体显示。

如有问题，[请联系question@packtpub.com](mailto:请联系question@packtpub.com)

Julia的基本原理

本文将介绍Julia在技术、数据科学家和高性能计算领域迅速走红的原因。我们将讨论以下主题。

* The scope of Julia
* Julia’s place among other programming languages
* A comparsion with other languages for the data scientist
* Useful links

**The scope of Julia**

Julia的核心设计师和开发人员(Jeff Bezanson、Stefan Karpinski和Viral Shah)已经明确表示，julia的诞生源于对技术计算学科现有软件工具集深深的失望。基本上可以归结为以下几点困境：

* 原型设计是这个领域中的一个问题，它需要一种高级的、易于使用的、灵活的语言，使开发人员能够专注于问题本身，而不是语言和计算的底层细节。
* 问题的实际计算需要最大的性能; 10倍的计算时间会产生很大的差异(想想一天还是十天)，因此生产版本通常必须用C或FORTRAN编写。
* 在Julia之前，实践者必须满足于“速度即方便”的权衡，使用对开发人员友好的、有表现力的、有几十年历史的解释性语言：如MATLAB、R或Python在高层次表达问题。为了对性能敏感的部分进行编程并加速实际的计算，人们不得不使用静态编译语言，如C或FORTRAN，甚至是汇编代码。julia为您提供了编写高性能代码的可能性，这些代码尽可能有效地使用CPU和内存资源，但是从头到尾都使用纯julia，减少了对低级语言的需求。通过这种方式，您可以使用一个简单的编程模型从问题原型到接近C的性能进行快速迭代。

综上所述，他们设计Julia的规格如下:

* Julia是开放源码的，并拥有麻省理工学院(MIT)的许可。
* 通过减少开发时间，它被设计成一种易于使用和学习的、优雅的、清晰的和动态的、交互式的语言。为了达到这个目的，julia看起来就像一个带着明显而熟悉的数学符号的伪代码。

例如，这是一个多项式函数的定义，直接从代码开始：

x -> 7x^3 + 30x^2 + 5x + 42

* 它提供了强大的计算能力和速度，而不需要离开Julia环境。
* 它可以用于一般的编程目的，而不仅仅是在纯计算领域。
* 它具有内置和简单的功能，可以在现在和以后的多核世界中使用并发和并行功能。

Julia把这一切统一在一个环境中，直到现在，大多数研究人员和语言设计师都认为这是不可能的。

**Julia’s place among the other**

**programming languages**

julia协调并将以前被认为是独立的技术集合在一起，也就是:

* 一方面是动态、非类型化和解释语言(Python、Ruby、Perl、MATLAB/Octave、R等)
* 另一种语言的静态类型和编译语言(C、c++、Fortran和Fortress)

Julia怎么能有第一个和第二类速度的灵活性?

Julia没有静态编译步骤。机器代码是由基于llvm的JIT编译器实时生成的。这个编译器与语言的设计一起，帮助Julia实现了数值、技术和科学计算的最大性能。性能的关键是类型信息，它由一个完全自动和智能的类型推断引擎收集，它从变量中包含的数据中推断类型。实际上，因为Julia有一个动态类型系统，所以在代码中声明变量的类型是可选的。指示类型是不必要的，但是可以这样做，以记录代码、改进工具的可能性，或者在某些情况下，向编译器提供提示，以选择更优化的执行路径。这个可选的键入规程作为它与Dart共享的一个方面。无类型Julia是该语言的一个有效且有用的子集，类似于传统的动态语言，但它仍然以静态编译的速度运行。Julia将泛型编程和多态函数应用于极限，只编写了一个算法，并将其应用于广泛的类型。这提供了不同类型的通用功能，例如:size是具有50个具体方法实现的通用函数。一个名为**dynamic multiple dispatch**的系统从几十个方法定义中有效地为函数的所有参数选择最优方法。根据实际的类型，选择或生成非常特定和高效的函数本地代码实现，因此它的类型系统使它更接近原始的机器操作。

**Note**

请记住，类型不是静态检查的。由于类型错误导致的异常可能在运行时发生，因此必须进行彻底地测试。在编程语言中对Julia进行分类时，它包含了多种范式，比如过程、函数、元编程，以及面向对象(但不是全部)。它绝不是一种完全基于类的语言，比如Java、Ruby或c#。然而，它的类型系统提供了一种继承，而且非常强大。数值类型和其他类型的转换和升级是优雅的、友好的和快速的，用户定义类型与内置类型一样快速和紧凑。至于函数式编程，Julia使程序的设计变得非常简单，只使用纯函数，没有副作用;函数是一流的对象，就像数学一样。

Julia还支持基于消息传递模型的多处理环境，允许程序使用分布式数组来运行多个进程(本地或远程)，并基于任何并行编程模型实现分布式程序。

Julia同样适合一般编程，就像python。它具有与Perl或其他语言一样好的、现代的(Unicode能力)字符串处理和正则表达式。此外，它还可以用于shell级别，作为一种胶水语言来同步其他程序的执行或管理其他进程。

Julia有一个用julia自己编写的标准库，以及一个基于**GitHub**(称为**元数据**)的内置包管理器，用来处理不断增加的称为包的**外部库集合**。它是跨平台的，支持GNU/Linux、Darwin/OS X、Windows和FreeBSD(64位)和x86(32位)体系结构。

**A comparison with other languages for the data scientist**

因为速度是Julia的终极目标之一，所以在Julia的网站上可以看到与其他语言的基准比较**(http://julialang.org/)**。这表明，Julia的竞争对手C和Fortran通常只保留了两个完全优化的C代码，而传统的动态语言类别则远远落后。Julia的明确目标之一是要有足够好的表现，这样你就不用去学习C语言了。这与下面的环境形成了对比，在这些环境中(甚至对于NumPy)，您经常需要与C一起工作，以便在迁移到生产环境时获得足够的性能。因此，可以预见一个技术计算的新时代，在这个时代，库可以用高级语言开发，而不是用C或FORTRAN。Julia特别擅长运行MATLAB和rstyle程序。让我们更详细地比较一下。

**MATLAB**

Julia对MATLAB用户非常友好;它的语法非常类似于MATLAB，但是Julia的目标是成为一种比MATLAB更通用的语言。Julia中大多数函数的名称都对应于MATLAB/Octave名称，而不是R名称。然而，在表面之下，计算的方式是完全不同的。Julia在线性代数方面也有同样强大的能力，这是MATLAB传统应用的领域。然而，使用julia不会给你带来同样的license。此外，基准测试显示，与Octave (MATLAB的开源版本)相比，根据操作的类型，它的速度要快10到1000倍。Julia用软件包提供了MATLAB语言的接口：MATLAB.jl

<https://github.com/lindahua/MATLAB.jl>

**R**

到目前为止，R是统计领域中所选择的开发语言。Julia在这个领域和R一样有用，但是性能提高了10到1000倍。在MATLAB中做统计是令人沮丧的，就像在R中做线性代数一样，但是Julia符合这两个目的。Julia的类型系统比基于矢量的R类型丰富得多。一些统计专家，如道格拉斯·贝茨，也大力支持并提拔julia。Julia用包提供了R语言的接口Rif.jl

<https://github.com/lgautier/Rif.jl>

**Python**

与Python相比，Julia的性能优势要高出10到30倍。但是，Julia将读起来像Python的代码编译成执行起来像C的机器代码。此外，如果需要，您可以使用PyCall包从Julia内部调用Python函数

<https://github.com/stevengj/PyCall.jl>

由于在所有这些语言中有大量现有的库，任何实用的数据科学家都可以而且需要在手头的问题需要的时候将Julia代码与R或Python混合使用。Julia也可以应用于数据分析和大数据，因为这些通常涉及预测分析、建模问题，这些问题通常可以简化为线性代数算法或图分析技术，所有这些都是julia擅长处理的。

在高性能计算(HPC)领域，像Julia这样的语言一直是缺乏的。使用Julia，各个领域专家可以进行实验，并且可以快速、轻松地表达问题，使他们可以像使用桌面PC一样轻松地使用现代HPC硬件。**换句话说，一种无需了解底层机器架构细节就能让用户快速入门的语言在这个领域非常受欢迎。**

**Useful links**

以下是在使用Julia时有用的链接:

* Julia的主页可以在<http://julialang.org>
* 有关文档，请参考<http://docs.julialang.org/en/latest>
* 查看包<http://pkg.julialang.org/indexorg.html>
* 订阅邮件列表<http://julialang.org/community>
* 获得IRC频道的支持<http://webchat.freenode.net/?channels=julia>

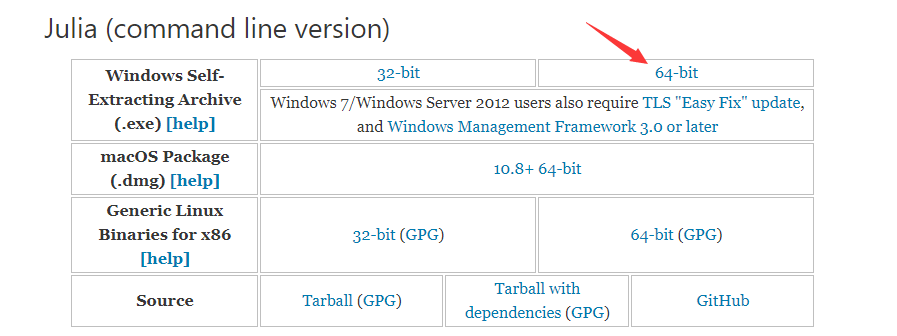
**Summary**

在介绍中，我们对Julia的特点进行了概述，并将其与该领域的现有语言进行了比较。Julia的主要优势在于它能够为不同的输入类型生成专门的代码。当与编译器推断这些类型的能力结合在一起时，就可以在抽象级别上编写Julia代码，同时实现与低级代码相关的效率。Julia已经相当稳定并且具备生产能力。Julia的学习曲线非常温和;这个想法是，那些不关心花哨的语言特性的人也应该能够有效地使用它，并且只有当它们变得有用或需要的时候才能了解新的特性。

第一章 Julia平台安装

本章指导您下载和安装Julia的所有必要组件。本章所涉及的主题如下：

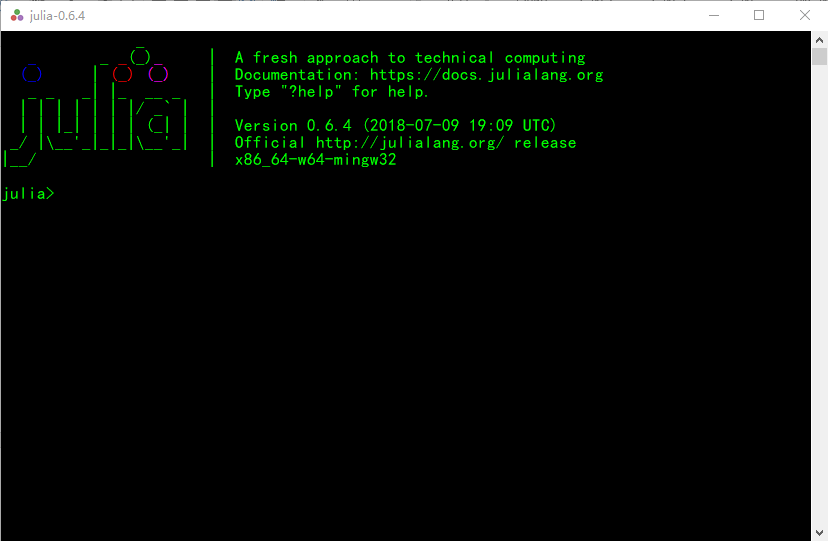
* 安装Julia
* 使用Julia shell 工作
* 程序包
* 安装程序包
* 其他编辑器以及IDE
* 总结

在这一章的结尾，你将有一个运行Julia的平台。此外，您将能够使用Julia的shell，以及编辑器或集成开发环境，这些环境具有许多内置特性，从而使开发更加舒适。

在官网<http://julialang.org/downloads>下载，如下图

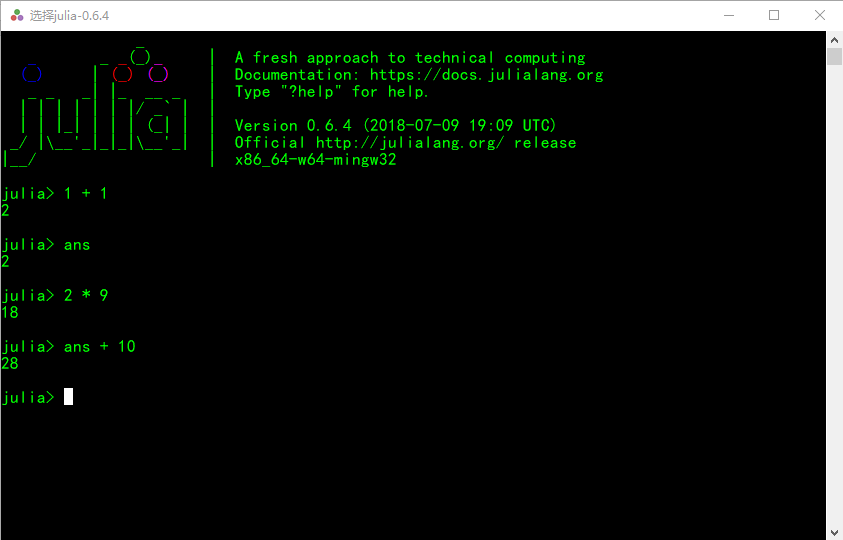
本人下载的是64位。下载完成之后，双击进行安装。为了以后方便使用，可把安装路径写入path环境变量中。

安装完成后的画面如下

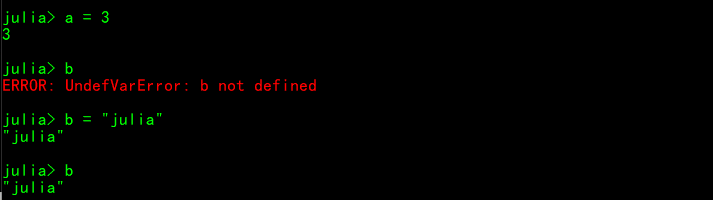


**Working with Julia’s shell**

我们从上一节中的Julia shell开始(参考前面的两个屏幕截图)，通过在终端会话中发出Julia命令来验证安装的正确性。shell或REPL是Julia的工作环境，您可以在其中与Just in Time(JIT)编译器交互，以测试代码片段。如果测试通过，可以将此代码复制并粘贴到扩展名为.jl的文件中，例如program.jl。或者，您可以在文本编辑器或IDE中继续处理这段代码，比如我们将在本章后面指出的那些代码。在有Julia标志的横幅出现后，您会得到Julia >提示输入。按CTRL+D推出，或输入quit()或exit()之后会按回车键退出。要计算一个表达式，输入它并按ENTER显示结果，如下图所示:



如果不想显示结果的话，可以在表达式后面输入分号“;”。 在这种情况下，产生的值都存储在一个名为ans的变量中，该变量可以在表达式中使用，但只能在REPL中使用。可以通过输入为a = 3赋值。Julia是动态的，我们不需要输入a类型，但是我们需要为变量输入一个值，这样Julia就可以推断出它的类型。使用不绑定值的变量b，会导致错误:b未定义消息。字符串用双引号("")来描述，如在b = "Julia"中。下面的截图用REPL说明了这些工作:



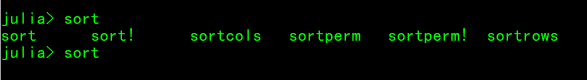
通过使用上下箭头键，可以在同一个会话中检索以前的表达式。以下键绑定也很方便:

* 要清除或中断当前命令，请按CTRL + C
* 要清除屏幕(但变量保存在内存中)，请按CTRL + L
* 要重置会话以便清除变量，请在REPL中输入命令workspace()

输入?启动帮助模式(help?>)以快速访问Julia的文档。另外，要获取关

于变量a的更多信息、输入?之后输入a，并获取关于函数的更多信息，如sort、输入?之后输入sort。

REPL还有自动补齐的特点。比如输入so之后连续按两次TAB键，会有下图所示



可以在REPL中通过include来执行Julia脚本。例如脚本hello.jl,里面的内容为println("Hello, Julia World!")。如下所示

C:\Users\luk\AppData\Roaming\Tencent\Users\735343320\QQ\WinTemp\RichOle\IR@[9C1]6L)3@(@2@W8CYRT.png

Note

更多信息请见

http://docs.julialang.org/en/latest/manual/interactingwith-

julia/#key-bindings.

Startup options and Julia scripts

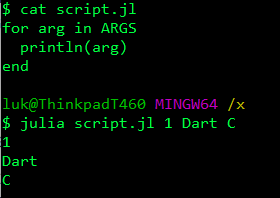
检查环境的一个有用的选项是julia -v，显示当前的版本号。还有一个有趣的选项为-e。如下图所示

C:\Users\luk\AppData\Roaming\Tencent\Users\735343320\QQ\WinTemp\RichOle\MXB79P6O)%F)GG7}CA3ONXF.png

julia -h可以显示所有的选项命令。可以使用下面的命令行执行脚本

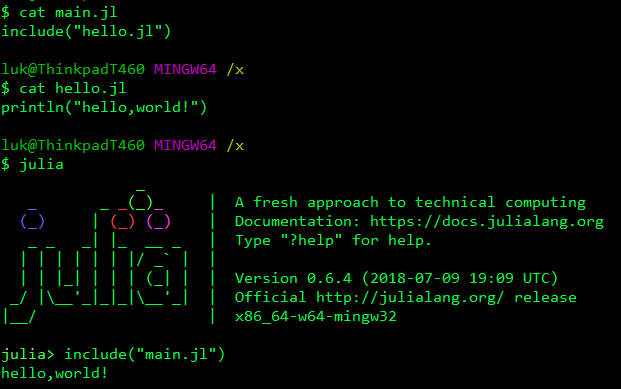
julia script.jl arg1 arg2 arg3

其中arg1,arg2,arg3为选项参数。



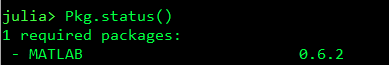
体验上述命令。

脚本文件还可以通过在REPL中来执行包含其他的源文件。例如main.jl中的内容为include(“hello.jl”)。也可以在cmd中执行，julia main.jl



Packages

Julia的大多数标准库(可以在/share/Julia/base中找到)是用Julia自己写的。Julia的代码生态系统的其余部分包含在简单的Git存储库的包中。它们通常是由外部贡献者编写的，并且已经为生物信息学、化学、宇宙学、金融、语言学、机器学习、数学、统计和高性能计算等多种学科提供了功能。可以在**http://pkg.julialang.org**/找到可搜索的包列表。Julia的安装包含一个内置的软件包管理器Pkg，用于在Julia中安装额外的Julia包。在REPL中使用命令行**Pkg.status()**可以查看已安装的包以及它们的版本。



**Pkg.installed()**命令提供了相同的信息，但是是**字典类型**的，在代码中是可用的。

Adding a new package

在添加新包之前，最好使用**Pkg.update()**命令更新已安装包的包数据库。然后，通过发出**Pkg.add(“PackageName”)**命令添加一个新的包，并在代码或REPL中**using** **PackageName**执行。

这里补充一点：因为julia画图调用的是PyPlot库，所以需提前安装好python环境。安装PyPlot库经常会出现问题，如果是第一次安装，特别要注意按照安装说明去操作。一般都是网络不通畅导致安装中断，中断之后把中断的命令重新执行即可。

这里简单介绍一下安装PyPlot的方法。

第一步：ENV[“PYTHON”]=”pythoncommand”； //->非常重要，不能省，往往会忽略。

第二步：Pkg.build(“PyCall”)

第三步：Pkg.add(“PyPlot”)

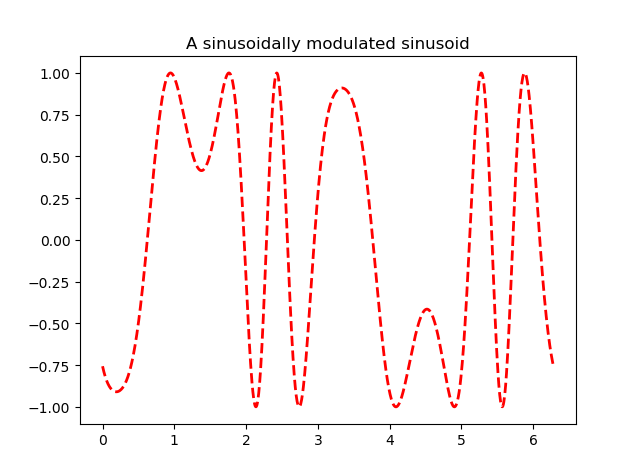
这三步执行过程中非常有可能因为网络问题而中断，中断后重新执行即可。

using PyPlot

x = linspace(0,2\*pi,1000); y = sin.(3 \* x + 4 \* cos.(2 \* x));

plot(x, y, color="red", linewidth=2.0, linestyle="--")

title("A sinusoidally modulated sinusoid")

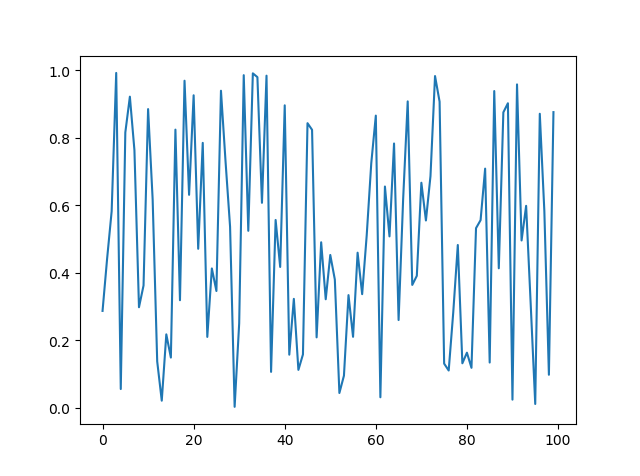


例如，要添加2D绘图功能，请使用**Pkg.add(“Winston”)**安装**Winston**包。要绘制0到1之间100个随机数的图形，请执行以下命令:

using PyPlot

using Winston

plot(rand(100))



接下来，再给出一个例子

关于package更多的更多信息，请见下面的连接

<http://docs.julialang.org/en/latest/manual/packages/>

Other editors and IDEs

对于终端用户，可用的编辑器如下:

* **Vim** <https://github.com/JuliaLang/julia-vim>
* **Emacs** <https://github.com/JuliaLang/julia/tree/master/contrib>

在linux上，**gedit**是一款很好的编辑器。**Notepad++**也是一款不错的编辑器。**SageMath**项目(https://cloud.sagemath.com/)让您使用IPython笔记本，在一个终端的云中运行Julia。

您还可以使用**JuliaBox**平台(https://juliabox.org/)在云中与Julia一起工作和教学。

Summary

到目前为止，您应该已经能够在您喜欢的工作环境中安装Julia了。你也应该有在REPL工作的经验。从下一章开始，我们将充分利用这一点，我们将通过在REPL中测试所有内容来满足Julia的基本数据类型。

第二章 Variables, Types, and

Operations

Julia是一种可选类型语言，这意味着用户可以**选择指定**传递给函数的参数类型，和函数中使用的变量类型。julia的类型系统是其性能的关键。本章讨论了Julia中基本内置类型、可以对它们执行的操作以及类型和范围的重要概念。

本章涉及以下主题：

* 变量、命名规则和注释
* Types
* 整型
* 浮点型
* 基本的数学函数和运算
* 有理数和复数
* 字符型
* 字符串
* 正则表达式
* Range和数组
* 日期和时间
* Scope和常量

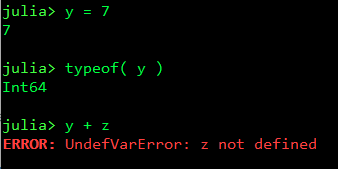
Variables, naming conventions, and

comments

数据存储在诸如1、3.14、“Julia”等值中，每个值都有一个类型，例如3.14的类型是Float64。其他一些基本数据及其数据类型包括Int64类型的42个、Bool类型的true和false以及Char类型的'X'。

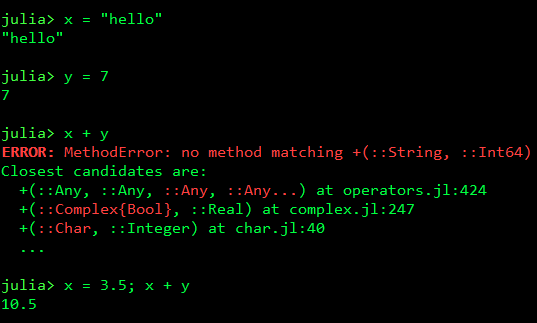
与许多现代编程语言不同，Julia区分了单个字符和字符串。字符串可以包含任意数量的字符，并使用双引号指定，单引号仅用于字符文本。变量是由赋值(如x = 42)绑定到值的名称。这种类型由typeof函数给出。例如，typeof(x)返回Int64。

变量的类型可以改变，例如输入x =“I am Julia”，typeof(x)的返回值为ascii字符串。在Julia中，我们不必声明一个变量(声明它的类型)，例如在C或Java中，但是必须初始化一个变量(它绑定到一个值)，这样Julia就可以推断出它的类型。



在上面的例子中，因为z没有指定数值就被使用，所以出现错误。通过操作符和函数(如+运算符)组合变量，我们得到表达式。与许多其他语言相反，Julia中的所有东西都是一个表达式，因此它返回一个值。这就是为什么在REPL中工作非常棒，因为您可以看到每个步骤的值。

变量的类型决定了你能用它们做什么，也就是说，它们可以用来组合的运算符，在这个意义上，Julia是一种**强类型**语言。在下面的例子中，x仍然是一个字符串值，所以不能用类型为y的y求和Int64，但是如果我们给x一个浮点数，总和可以被计算，如下例所示:



在这里，分号(;)结束第一个表达式并抑制其输出。Julia中变量的名字区分大小写。**按照惯例，小写都用下划线隔开多个单词。**它们从一个字母开始，然后，你可以用字母，数字，下划线，和感叹号。还可以使用Unicode字符。以下是一些有效的变量名：mass,moon\_velocity,current\_time,pos3和。

在代码行首加上字符#表示此行被注释。

# Calculate the gravitational acceleration grav\_acc:

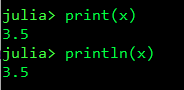
gc = 6.67e-11 # gravitational constant in m3/kg s2

mass\_earth = 5.98e24 # in kg

radius\_earth = 6378100 # in m

grav\_acc = gc \* mass\_earth / radius\_earth^2 # 9.8049 m/s2

多行注释对于编写跨多行或注释代码很有帮助。julia会把所有的文字都放在#=和=#之间。对于打印出的值，使用print或println函数.



如果希望打印输出是彩色的，请使用**print\_with\_color**(:red，“I love Julia!”)第一个参数所指示字符串的颜色。只能在REPL中有作用。

Types