**四种大数据语言**

## 一、Python

**1 Python简介**

**（1） Python是什么**

相信混迹IT界的很多朋友都知道，Python是近年来最火的一个热点，没有之一。从性质上来讲它和我们熟知的C、java、php等没有什么本质的区别，也是一种开发语言，而且已经进阶到主流的二十多种开发语言的top 5（数据源自最新的TIOBE排行榜）。来头不小啊！二十多种主流的开发语言，我该从哪一个开始呢？人生苦短，let‘s python！

**（2）Python的由来和发展趋势**

Python的前世源自鼻祖“龟叔”。1989年，吉多·范罗苏姆（Guido van Rossum）在阿姆斯特丹为了打发无聊的圣诞节，决心开发一个新的脚本解释程序，自此Python和创始人“龟叔”开始进入公众视野。他希望这个新的叫做Python的语言，能符合他的理想：创造一种C和shell之间，功能全面，易学易用，可拓展的语言。

1991年，第一个Python编译器诞生。它基于C语言实现，并能够调用C语言的库文件。后面历经版本的不断换代革新，Python走到了非常具有里程碑意义的一个节点，即2004的2.4版本诞生了目前最流行的WEB框架Django！六年后Python发展到2.7版本，这是目前为止2.x版本中最新且较为广泛使用版本。

2.7版本的诞生不同于以往2.x版本的垂直换代逻辑，它是2.x版本和3.x版本之间过渡的一个桥梁，以便最大程度上继承3.x版本的新特性，同时尽量保持对2.x的兼容性。因此3.x版本在2.7版本之前就已经问世，从2008年的3.0版本开始，python3.x系呈迅猛发展之势，版本更新活跃，一直发展到现在最新的3.5.2版本。3.5版本也是目前3.x系列中主流且广泛使用的版本，后续相关程序的demo，默认均基于3.5版本展开。

**（3）Python 2 VS Python 3**

许多初学者会纠结于从2.x版本还是3.x版本开始python学习之路。这里简要对比下个人学习了解到的两者的差异：

3.x版本中 PRINT IS A FUNCTION，使用起来更简单方便；

3.x版本中默认使用unicode编码，妈妈再也不用担心你的程序编码问题了；

3.x版本中某些库进行了改名，具体还有待在后续学习中深入展开；

目前3.x版本还不能完全支持Twisted module，这恐怕是3.x版本中最大的遗憾。也不要过于纠结这点，据说官方正在弥补这个不足，同时也相信能有办法解决的：）

In summary : Python 2.x is legacy, Python 3.x is the present and future of the language. 长江后浪推前浪，python的未来是属于3.x系列的，官方宣布2020年之后不再支持2.x系列中最nb的2.7版本，2.x版本将逐渐寿终正寝，最终被铭刻在开发语言的丰碑上。

所以学习3.x版本没错的！

**（4）Python的优缺点**

**@@Python优点**：

1）.“优雅”、“明确”、“简单”

这是python的定位，使得python程序看上去简单易懂，初学者容易入门，学习成本更低。但随着学习的不但深入，python一样可以满足胜任复杂场景的开发需求。引用一个说法，Python的哲学是就是简单优雅，尽量写容易看明白的代码，尽量写少的代码。

2）.开发效率高

python作为一种高级语言，具有丰富的第三方库，官方库中也有相应的功能模块支持，覆盖了网络、文件、GUI、数据库、文本等大量内容。因此开发者无需事必躬亲，遇到主流的功能需求时可以直接调用，在基础库的基础上施展拳脚，可以节省你很多功力和时间成本，大大降低开发周期。

3）.无需关注底层细节

Python作为一种高级开发语言，在编程时无需关注底层细节（如内存管理等）。

4）.功能强大

Python是一种前端后端通吃的综合性语言，功能强大，php能胜任的角色它都能做，至于后端如何胜任，需要在后续学习中逐步领悟。

5）.可移植性

Python可以在多种主流的平台上运行，开发程序时只要绕开对系统平台的依赖性，则可以在无需修改的前提下运行在多种系统平台上，具体有待后续学习中深入展开。其他优点有待继续发掘。

**@@@Python缺点：**

1）代码运行速度慢

因为Python是一种高级开发语言，不像c语言一样可以深入底层硬件最大程度上挖掘榨取硬件的性能，因此它的运行速度要远远慢于c语言。另外一个原因是，Python是解释型语言，你的代码在执行时会一行一行地翻译成CPU能理解的机器码，这个翻译过程非常耗时，所以很慢。而C程序是运行前直接编译成CPU能执行的机器码，所以非常快。

但需要注意的是，这种慢对于不需要追求硬件高性能的应用场合来讲根本不是问题，因为它们比较的数量级根本不是用户能直观感受到的！想想程序执行所需的时间数量级？例如开发一个下载MP3的网络应用程序，C程序的运行时间需要0.001秒，而Python程序的运行时间需要0.1秒，慢了100倍，但由于网络更慢，需要等待1秒，用户体验几乎没有差别，除非你用非常精确的计时器来计时。

2）发布程序时必须公开源代码

什么？有没有搞错？是的，发布程序时我们必须公开源代码！还是因为Python是一种解释性语言，没有编译打包的过程（据说最新的python可以打包，但本质上还是把源代码和解释器打在一起，没有太大实际意义）。想想我们的shell脚本是不是也是这个情况，你能在不发布源代码的情况下发布一个黑盒子程序来让别人正常使用么？

这个缺点仅限于你想单纯靠卖开发出来的软件产品挣钱的时候。但在这个开发互联的时代，不靠卖产品本身来赚钱的商业模式越来越主流了，所以问题也不是没法解决。

总结起来，Python的优点多于缺点，而且缺点在多数它能胜任的情况下不是根本性问题，瑕不掩瑜，尽情享受它给我们带来的乐趣和便利吧！

**（5）Python的适用领域**

1）. 云计算开发

云计算领域最火的语言，典型代表openstack。

2）. web开发

众多优秀的web框架，众多优秀的大型web站点（Youtube,instagrm,豆瓣等）均基于python开发。

3）. 系统运维

各种自动化工具的开发，CMDB、监控告警系统、堡垒机、配置管理&批量分发工具等均可以搞定。这是自己需要重点学习的领域。

4）. 科学计算、人工智能

据网上了解用于围棋大战的谷歌阿发狗就使用了python开发（但估计只是部分而已）。

5）. 图形GUI处理

暂时没多少体会，后续领悟吧。

6）. 网络爬虫

不知道给网络爬虫单独归类是否合适，姑且这样吧。这年头网络爬虫无处不在，很多都是基于python开发，包括谷歌的爬虫。估计这是爬虫工程师的必备工具吧。

**（6）Python应用场景**

谷歌：Google app engine、google earth、爬虫、广告等

Youtube: 世界最大的在线视频网站基于python开发，好强大

Instagram：美国最大的图片分享网站，全部基于python开发

facebook：大量的基础库基于python开发

Redhat：yum包管理工具基于python开发

国内知名互联网公司的应用：豆瓣、知乎的网站，还有很多主流的互联网公司（阿里、腾讯、百度、金山、搜狐、盛大、网易、新浪、网易、果壳、土豆等）通过python来完成各种任务。

**2 Python基础**

**（1）编程开发规范**

1）. 代码开头

python中代码必须顶格编写,除非被作为语句嵌套在条件判断或条件循环中。

2）. 代码层级

代码与代码之间的逻辑层级关系（涉及逻辑判断、包含关系等情况）通过空格来界定，第一层顶格编写，第二层（被包含或者逻辑判断后的执行动作）开头预留4个空格（如果不考虑跨平台可以使用tab）

3）. 对齐方式

同一个层级的代码必须严格对齐，如第一层都是顶格，第二层都是4个空格开头，第三层则是8个空格，以此类推。

4）. 注释

单行注释用#开头，单独一行作为注释或者在代码后面通过#跟上注释均可，多行注释在首尾处用成对的三引号引用即可，可以是成对的三个单引号或者三个双引号。

5）. 代码块

目前来看if逻辑判断和for循环条件后面均以冒号结尾，然后通过4个空格开始下一层级的实际执行动作语句编写。这部分在冒号后面接上的语句称为代码块。

6）. 程序命名

编写的python程序必须以.py结尾。

7）. 项目规范

启动一个项目开发时先在开发工具中创建project，然后根据需要创建目录，最后在目录下创建当前活动的py程序。

**（2）变量**

python中的变量也用于存储临时值以便后续灵活调用，在定义时只能使用字母、数字、下划线的任意组合，且不能以数字开头，不能以python内置的关键字来命名变量（如and，import，global，finally等，有待后续学习加深印象）。对变量进行字符串类型的赋值时，字符串需要用成对的引号引起来，可以是单引号、双引号或者三引号。

建议通过以下两种形式来定义较为复杂的变量，便于阅读：

1）. 驼峰体

即每个单词的首字母大写，其他小写，如MappingTable

2）. 下划线

不同单词之间通过下划线分开，如mappting\_table

**（3）字符编码**

字符编码经历了ASCII码->unicode->UTF-8的演变过程，python 2.x默认使用ASCII编码，而3.x默认使用UTF-8编码。由于字符编码事关重大，一言不合就会导致乱码，因此强烈建议统一使用万用的utf-8编码。怎么使用呢？

1）. 程序开头明确显式定义

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

windows下编程可忽略第一行，但为了保留移植性建议始终声明变量。

2）. 在程序解释器中选择utf-8字符编码

以上两个条件缺一不可，虽然python 3.x版本默认使用utf-8编码，但还是建议在程序中声明编码格式。

以上是结果，下面谈谈为什么强烈建议使用urf-8编码，以及为什么utf-8编码就可以通吃。

众所周知计算机是美国人发明的，因此最早就只有包括所有字母的大小写、数字、必要的特殊符号等合计才127个字符被编码到计算机里，这个编码就是ASCII编码。由于需要进行编码的字符才有127个，因此用7个二进制位就足以搞定了，但8个二进制位才能构成一个完整的字节，因此索性用8个二进制位来进行编码，剩余的一位只好浪费掉。所以ASCCI码只占用一个字符。

随着计算机技术的推广和普及，其他不以英语为官方语言的国家也开始使用计算机了。那么现在问题来了，一个字节的8个bit位只能表示非常有限的字符（设计时只包括所有字母的大小写、数字、必要的特殊符号，其中整数还不能超过255），既有的ASCII编码表里没有包含其他国家的语言字符，但显然也没法强迫全球都使用英语，于是各个国家开始进行本土化改良，设计能满足自己国家需要的编码，如国内的GB2312。这里的改良涉及两个要点，一是原有的一个字节的编码空间要扩充为两个字节甚至更多，二是新的编码还不能与既有的ASCII编码冲突，否则老美要跟你捉急了！你可以想象，中国搞定了一套，日本也搞定了一套，接着是韩国、越南等巴拉巴拉，大家都有自己的编码标准，各自为政，遇到多语言混合使用的场景这编码表还不炸开锅，连同计算机一起傻掉！这时Unicode应运而生。Unicode把所有语言都统一到一套编码里，用两个甚至更多的字节来进行编码（通常用两个字节，搞不定时再喊人加砝码呀），这样就不会再有乱码问题了。

Unicode搞定了多国语言乱码和编码统一的问题，但你会发现如果需要处理的内容全部是英文的话，用Unicode编码比ASCII编码需要多一倍的存储空间，在存储和传输上就不太经济划算。OK，Unicode的演进改良版utf-8就是来解决这个问题的。utf-8的优势在于它使用“可变长编码”，在继续沿用ASCII编码基础上来对1个字节不能搞定的那些语言字符进行编码设计，这样英文还是用1个字节，遇到中文就用3个或者更多的字节来处理。这样既可以保障空间和性能的最大化，还能兼顾老美的老巢和其他非英文的编码兼容问题，一举三得！所以你会发现ASCII其实是utf-8的子集，utf-8向下兼容。

**（4）Python解释器**

python是一门解释性语言，自然不能缺了解释器（类似于JVM）。官方支持Cpython、Jpython、Ironpython和pypy等，默认使用基于C语言实现的Cpython。它也是最为广泛使用的解释器。程序运行时会把.py文件转换成.pyc字节码文件，然后运行在python虚拟机上。

**（5）构建Python的开发环境**

构建好包括解释器（虚拟机）和开发工具IDE在内的Python的开发环境才便于展开程序开发之旅。其中解释器（虚拟机）从python官方下载安装即可，需要注意的是建议安装3.5版本，windows下安装后需要把python的bin目录和scripts目录追加到系统环境变量path和用户变量path中。我的win 7用户不是以administor，补充追加了用户变量后才能在cmd下傻瓜式地输入python。linux下系统默认是2.x的版本，安装3.5版本时需要先将旧版本重命名为其他目录（如python2.6）后再安装，安装完毕无需额外设置环境变量。好的程序猿都得有自己好使的IDE，python程序建议使用pycharm工具来进行开发。从官网下载安装社区版即可，需要注意的是选择后本地的解释器和项目的默认字符编码。其他一些好玩的特性、快捷键等只有后续逐步体会了。

**（6）模块初识**

python内置了很多模块，同时也可以自定义编写一些好玩的模块，使用模块时需要通过关键字import导入。导入后python会从系统变量的路径中从前往后一次查找，首次成功查找匹配后不再进行后续查找。因此如果需要使用自定义模块，需要特别关注自定义模块的存放位置，底线是只存放在系统变量的一个路径下，否则可能得到意外的结果。

系统变量的路径可通过以下方式查看：

以上是windows下3.5版本的输出结果，linux平台或者其他版本的输出有所差别，以实际为准。

**（7）用户输入和格式化输出**

用户输入通过input来实现，示例如下：

YourName=input("Please input you name here: ")

简单的输出通过print来实现，示例如下：

print("Hi,", YourName) #调用输出上面用户输入的变量YourName

注意：

1）. 3.x版本中input默认接收用户输入的类型为字符串，如果需要使用其他的数据类型则需要进行转换，如 UserAge=int(input("Please input your name:"))

2）. print输出中可以接上转义符号\来实现更多的输出显示效果，如 \n表示换行，\t为制表符。

print的使用进阶——字符串的格式化输出：

python中，对字符串进行格式化输出与c语言是相同的，即借助%来实现。%s表示与字符串替换，%d表示用整数替换，%f表示用浮点数替换，如果输出的字符串中本身就需要%（比如输出比率），则需要用连续的两个%%来输出%。

示例程序如下：

#!/usr/bin/env python

#-\*- coding:utf-8 -\*-

print ("Your name is %s" %"Bob")

print ("Your age is %d" %21)

print ("Your name is %s,age is %d" %("Bob",21))

从以上示例程序可看出，利用print进行格式化输出的基本格式是在需要替换输出的位置用%s,%d等占位符，然后在后面通过%来接上实际替换后需要输出的内容，如果需要替换输出的元素不止一个，则需要在%后面通过一对圆括号把实际替换后需要输出的内容包含进来，同时在每个元素之间用逗号隔开。

**（8）Python的运行逻辑**

Python的运行工作逻辑如下：

内存加载代码->语法分析->解释器加载->生成字节码->执行字节码->生成机器码->CPU执行

**3 开始你的Python开发之旅**

**（1）第一个Python程序--HelloWorld**

python的第一个程序也从hello world开始把：

#!/usr/bin/env python

#! -\*- coding:utf-8 -\*-

print("Hello world!")

执行结果：

"C:\Program Files\Python35\python.exe" D:/python/Day1/test/HelloWorld.py

Hello world!

Process finished with exit code 0

**（2）If-else判断**

1）If-else语法规范

以下是if-else判断的语法结构规范

if condition1:

command\_layer1\_1

if condition2:

command\_layer2\_2

else:

command\_layer2\_2

else:

command\_layer1\_2

以下为一个演示两层if-else循环的程序：

#!/usr/bin/env python #顶格编写

#! -\*- coding:utf-8 -\*-

user\_input = input("Please input you username:")

if user\_input == "Bob": #注意这里的冒号结尾

passwd\_input = input("Please input your password:") #注意从这里开始，第一个if条件为真时需要执行的动作语句均需要左缩进4个空格

if passwd\_input == "password": #第一个if下的第二个if，仍然要左缩进4个空格，同时冒号结尾

print("Welcome login,%s!" %user\_input) #第二层if条件为真时执行的动作语句，需要在第一层语句基础上再缩进4个空格，因此需要缩进8个空格

else: #第二层if-else中的else，因此需要与第二层if对齐，缩进4个空格

print("Invalid username or password, please check your input") #第二层if-else条件为假时执行的动作语句，同样需要与第二层if一样缩进8个空格

else: #第一层if-else中的else关键字，顶格冒号结尾

print("Invalid username or password, please check your input") #第一层if-else判断条件为假时执行的动作，与第一层if一样需要缩进4个空格

说明：该示例程序仅为演示多层if-else的语法结构，程序本身的设计存在漏洞；空格缩进在pycharm IDE环境中会被自动处理，但在普通文件编辑器中需要手动设置。

以下为改良版示例程序，通过引入对if的多条件判断来避免上述程序的漏洞：

1 #!/usr/bin/env python

2 #! -\*- coding:utf-8 -\*-

3 username=input("Please input you username:\n")

4 passwd=input("Please input you password:\n")

5 if username == "Bob" and passwd == "password":

6 print("Welcome login, %s!" %username)

7 else:

8 print("Invalid username or password, please check your input!")

此时只有用户名和密码同时输入正确了才会给出相应提示，否则均提示口令无效，避免暴力破解。

2）if-elif-else扩展

上述判断均为单一式的if-else判断，以下为if-elif-else的判断扩展：

语法结构：

if condition1:

command1

elif condition2:

command2

elif condition3:

command3

else condition4:

command4

不过这种结构仅仅适用于单一条件存在多种case情况下，语法结构看起来还是比较简单，当然顶格、左缩进4个空格和冒号这些规范一样要遵循。

还是来一个示例程序加深理解：

#!/usr/bin/env python

#! -\*- coding:utf-8 -\*-

age=int(input("Please input your age\n"))

if age >= 18:

print("Oh, you're an adult\n")

elif age >=6:

print("Ha, you're a teenager\n")

else:

print("Come on, little kid!\n")

（3）For循环

1）for循环语法规范

For循环的基本语法规范是：

for variable in XXX:

loop command

其中variable表示命名的变量，一般程序中使用i，j等等，XXX表示变化的范围，可以是list列表，一般会考虑使用range函数，来表示一个整数序列，如range（5）就表示小于5的整数序列，即0-4。 语法规范中同样需要for语句后面的结尾冒号，以及循环体中的4个空格的左缩进。

2）示例程序

猜数字游戏，通过系统生成一个随机数作为预设年龄，对用户提供3次猜的机会，前两次如果没有猜中给出数字范围大小的提示，如果第3次还没有猜中则给予鼓励提示，同时打印出这个预设的数字，当然三次当中有任何一次猜中会给用户猜中提示的：

1 #!/usr/bin/env python

2 #! -\*- coding:utf-8 -\*-

3 import random #导入随机数模块

4 Age=random.randrange(10)#随机生成一个小于10的整数（0-9，不包括负数），并赋值给Age

6 for i in range(3):

7 if i < 2:

8 guess\_number=int(input("Please input the age of my dog you guess:\n"))

9 if guess\_number > Age:

10 print("The age you guess is a little big, think smaller!\n")

11 elif guess\_number < Age:

12 print("The age you guess is a little small, think bigger!\n")

13 else:

14 print("Bingo, you got the number,congratulations!\n")

15 break

16 else:

17 guess\_number=int(input("Please input the age of my dog you guess:\n"))

18 if guess\_number == Age:

19 print("Bingo, you got the number,congratulations!\n")

20 else:

21 print("Oh,you just got bad luck, come to try again, you can do it! The actual age of my dog is %d...\n"% Age)

注意：

1.为便于程序调试可先对Age赋值固定的数字

2.前两次如果猜中了就需要退出循环，这里使用了break关键字，意思是跳出整个循环，与之相对应的还有continue关键字，用于跳出当次循环，二者的使用区别还有待进一步深入学习

OK，现在来改进下程序， 当用户连续三次猜错后继续给机会，让用户选择是继续猜还是直接退出，如果继续则再一次获得三次猜的机会，如此循环下去。还是用for循环吧：

1 #!/usr/bin/env python

2 #! -\*- coding:utf-8 -\*-

3 Age=22

4 counter=0

5 for i in range(10):

6 if counter < 3:

7 guess\_number=int(input("Plese input your guess number:\n"))

8 if guess\_number == Age:

9 print("You got the number, congratulations!")

10 break

11 elif guess\_number > Age:

12 print("The number you guessed is too big, guess a smaller one\n")

13 else:

14 print("The number you guessed is too small, guess a bigger one\n")

15 counter += 1

16 elif counter == 3:

17 continue\_flag=input("Do you want to continue? Please type Y to continue or N to quit:\n ")

18 if continue\_flag == "Y":

19 counter = 0

20 else:

21 print("Bye")

22 break

23 else:

24 print("You've tried too many times.")

这里为了保证每一个轮回中的第四次（上述程序中的第四次和第八次）能让程序继续循环，引入了另外一个变量来进行计数并重置。把for循环换作while循环看起来差不多：

1 #!/usr/bin/env python

2 #! -\*- coding:utf-8 -\*-

3 Age=22

4 i=0

5 counter=0

6 while counter < 10:

7 if i < 3:

8 guess\_number=int(input("Plese input your guess number:\n"))

9 if guess\_number == Age:

10 print("You got the number, congratulations!")

11 break

12 elif guess\_number > Age:

13 print("The number you guessed is too big, guess a smaller one\n")

14 else:

15 print("The number you guessed is too small, guess a bigger one\n")

16 i += 1

17 else:

18 continue\_flag=input("Do you want to continue? Please type Y to continue or N to quit:\n ")

19 if continue\_flag == "Y":

20 i = 0

21 else:

22 print("Bye")

23 break

24 counter +=1

**4 搭建开发环境**

1) 可以到www.python.org下载安装包，然后通过configure、make、make install进行安装。

2) 也可以到www.activestate.com去下载ActivePython组件包。(ActivePython是对Python核心和常用模块的二进制包装，它是ActiveState公司发布的Python开发环境。ActivePython使得Python的安装更加容易，并且可以应用在各种操作系统上。ActivePython包含了一些常用的Python扩展，以及Windows环境的编程接口)。对ActivePython来说，如果你是windows用户，下载msi包安装即可；如果你是Unix用户，下载tar.gz包直接解压即可。

3) Python的IDE，包括PythonWin、Eclipse+PyDev插件、Komodo、EditPlus

**5 python小结**

1）. Python是一门主流且发展迅猛的开发语言，目前存在2.x和3.x两个版本分支，前端后端都能胜任，功能强大，在互利网公司应用广泛，值得托付；

2）. Python程序的开发需要安装解释器和IDE，建议使用pycharm，2.x和3.x下的默认字符编码不同，永远推荐使用utf-8编码，包括程序编码和解释的编码两个层面；

3）. 系统自带模块和自定义编写的模块均可以通过import导入使用；

4）. 在进行python程序开发时，注意代码需要分层体现，第一层代码顶格编写，第二层（用在第一次出现的流程控制语句中的代码块）左缩进4个空格；

5）. input用于接收用户输入，默认输入的数据类型是字符串，可通过int关键字转换为整数。print可进行简单输出，如要格式化输出则需要借助%s %d %f等来实现，具体视数据类型而定；

6）. 流程控制语句if-else判断和for、while循环都需要注意代码书写格式，流程控制语句关键字一定是以冒号结尾，具体的流程控制代码块则需要左缩进处理，第一层左缩进4个空格，第二层8个空格，以此类推；

7）.continue和break可分别用于退出当前（当次）流程控制和整个循环，课堂上演示了在for循环中的使用，在if-else判断中不能使用。

## 二、R语言

**1、什么是R语言？**

R是用于统计分析、绘图的语言和操作环境。R是属于GNU系统的一个自由、免费、开源的软件，它是一个用于统计计算和统计制图的优秀工具。R语言是主要用于统计分析、绘图的语言和操作环境。 R本来是由来自新西兰奥克兰大学的Ross Ihaka和Robert Gentleman 开发。 （也因此称为R）现在由“R开发核心团队”负责开发。 R是基于S语言的一个GNU项目，所以也可以当作S语言的一种实现，通常用S语言编写的代码都可以不作修改的在R环境下运行。 R的语法是来自Scheme。

**2、R语言能干什么？**

1）、R是科学计算的强大工具包。

2）、R主要擅长统计分析方面工作。

3）、R具有顶尖的绘图功能。

4）、R的交互式数据分析功能强大且灵活。

5）、R可以轻松地从多个数据源导入数据。

6）、金融领域数据统计与计算的强大工具。

7）、数据挖掘和机器学习领域的必备工具。

**3、R语言有什么优势？**

1）、R是免费的。很多商业统计软件价格不菲，投入成千上万美元都是有可能的。可能国内一些人感觉不会很明显，因为很多人是使用单位或学校集体购买的软件，或者使用破解过的统计软件，而不是花自己钱。对于前者，并不是每个人都有机会享受这种福利。而对于破解过的统计软件，其计算结果的准确性很可能会被质疑。

2）、R主要擅长统计分析方面工作。R最初是由两位统计学家开发的，其主要优势也在于统计分析方面。它提供了各种各样的数据处理和分析技术，几乎任何数据分析过程都可以在R中完成。与此相比较，SPSS、MINITAB、MATLAB等数据分析软件更加适合于已经处理好的、规范的数据，而对于还未完成处理过程，或者在分析中仍需大量与处理过程的数据而言，它们可能会显得繁琐一些。

3）、R具有顶尖的绘图功能。尤其对于复杂数据的可视化问题，R的优势更加明显。一方面，R中各种绘图函数和绘图参数的综合使用，可以得到各式各样的图形结果，无论对于常用的直方图、饼图、条形图等，还是复杂的组合图、地图、热图、动画，以及自己脑子里突然想到的其他图形展现方式，都可以采用R语言实现。另一方面，从数值计算到得到图形结果的过程灵活，一旦程序写好后，如果需要修改数据或者调整图形，只需要修改几个参数或者直接替换原始数据即可，不用重复劳动。这对需要绘制大量同类图形的用户比较适用。例如，某用户需要观察某一因素与其他100个因素的交互关系，可以绘制100个条形图或走势图。配合R的循环和条形图功能，可以很快得到jpg、png、bmp、tiff、gif或postscript等格式的图形结果。

4）、R的交互式数据分析功能强大且灵活。一个完整的数据分析过程可以大体包括以下几个步骤：（1）导入数据；（2）数据准备、探索和清洗；（3）拟合一个统计模型；（4）得到结果并进行评估；（5）如果结果的评估不理想，重新完成（3）步骤；（6）得到多个模型的结果，并进行交叉检验；（7）根据模型结果进行预测、分析等；（8）形成报告。R中每个步骤的所有“输出”都可以直接作为下一个步骤的“输入”，可以批量完成以上所有的（8）个步骤。这个优点主要与STATA、SPSS等统计软件相比而言的，STATA和SPSS的统计分析结果是一大串图表或统计量。如果只是一个或少数几个步骤的分析，这种出现一大串结果的批处理更加直观方便。但如果分析中涉及很多连串的模型，而且一些模型需要反复调用前面模型的结果，那么R的灵活性会更加凸显。

5）、R可以轻松地从多个数据源导入数据，包括文本文件、数据库、其他统计软件等。这一点很多其他软件还难以做到。如readLines()、read.table（）等函数可以导入文本数据，foreign包的read.spss()可以导入SPSS软件的sav格式数据，foreign包的read.dta()可以导入STATA软件的dta格式数据等等。其他有关数据导入导出的函数可以参见R帮助文档《R Data Import/Export》。

6）、R的更新速度很快，包含最新的大量统计方法和案例。R社区由全球大量维护者共同维护，再加上R包的扩展性很强，几乎每天都有人为R社区贡献新的方法，提供其他软件中尚不可用的、新颖的统计计算案例。在本材料“R资源”部分会介绍一些常用的R资源获取途径。

7）、R也可以很美。目前最基本的R软件（R2.15.2）只有一个编程窗口，看起来挺丑的。但它穿上衣服以后也可以很漂亮，它的衣服就是各式各样的GUI（Graphical User Interface，图形用户界面）工具。目前比较常用的GUI工具包括Rstudio、Rattle、Rattle、Red-R、Deducer、RKWard、JGR、R Commander、Tinn-R等。其中Rstudio可能是最受欢迎的。

**4、R语言编程工具**

R语言安装时会自带一个GUI工具，当然还有其他众多集成开发工具，如：Rstudio、Rattle、Rattle、Red-R、Deducer、RKWard、JGR、R Commander、Tinn-R等。其中Rstudio可能是最受欢迎的。

**5、R语言学习路线图**

1）.初级入门

《An Introduction to R》，这是官方的入门小册子。其有中文版，由丁国徽翻译，译名为《R导论》。《R4Beginners》，这本小册子有中文版应该叫《R入门》。除此之外，还可以去读刘思喆的《153分钟学会R》。这本书收集了R初学者提问频率最高的153个问题。为什么叫153分钟呢？因为最初作者写了153个问题，阅读一个问题花费1分钟时间，全局下来也就是153分钟了。有了这些基础之后，要去读一些经典书籍比较全面的入门书籍，比如《统计建模与R软件》，国外还有《R Cookbook》和《R in action》。

2）.高级入门

读了上述书籍之后，你就可以去高级入门阶段了。这时候要读的书有两本很经典的。《Statistics with R》和《The R book》。之所以说这两本书高级，是因为这两本书已经不再限于R基础了，而是结合了数据分析的各种常见方法来写就的，比较系统的介绍了R在线性回归、方差分析、多元统计、R绘图、时间序列分析、数据挖掘等各方面的内容，看完之后你会发现，哇，原来R能做的事情这么多而且如此简洁。

3）.绘图与可视化

亚里斯多德说，“较其他感觉而言，人类更喜欢观看”。因此，绘图和可视化得到很多人的关注和重视。那么，如何学习R画图和数据可视化呢？再简单些，如何画直方图？如何往直方图上添加密度曲线呢？我想读完下面这几本书你就大致会明白了。

首先，画图入门可以读《R Graphics》，个人认为这本是比较经典的，全面介绍了R中绘图系统。该书对应的有一个网站，google之就可以了。更深入的可以读《Lattice：Multivariate Data Visualization with R》。上面这些都是比较普通的。当然，有比较文艺和优雅的——ggplot2系统，看《ggplot2：Elegant Graphics for Data Analysis》。还有数据挖掘方面的书：《Data Mining with Rattle and R》，主要是用Rattle软件，个人比较喜欢Rattle!当然，Rattle不是最好的，Rweka也很棒！再有就是交互图形的书了，著名的交互系统是ggobi，这个我已经喜欢两年多了，关于ggobi的书有《Interactive and Dynamic Graphics for Data Analysis With R and GGobi》，不过，也只是适宜入门，更多更全面的还是去ggobi的主页吧，上面有各种资料以及包的更新信息！

4）.计量经济学

关于计量经济学，首先推荐一本很薄的小册子:《Econometrics In R》，做入门用。然后，是《Applied Econometrics with R》，该书对应的R包是AER，可以安装之后配合使用，效果甚佳。计量经济学中很大一部分是关于时间序列分析的，这一块内容在下面的地方说。

5）.时间序列分析

时间序列书籍的书籍分两类，一种是比较普适的书籍，典型的代表是：《Time Series Analysis and Its Applications ：with R examples》。该书介绍了各种时间序列分析的经典方法及实现各种经典方法的R代码，该书有中文版。如果不想买的话，建议去作者主页直接下载，英文版其实读起来很简单。时间序列分析中有一大块儿是关于金融时间序列分析的。这方面比较流行的书有两本《Analysis of financial time series》，这本书的最初是用的S-plus代码，不过新版已经以R代码为主了。这本书适合有时间序列分析基础和金融基础的人来看，因为书中关于时间序列分析的理论以及各种金融知识讲解的不是特别清楚，将极值理论计算VaR的部分就比较难看懂。另外一个比较有意思的是Rmetrics推出的《TimeSeriesFAQ》，这本书是金融时间序列入门的东西，讲的很基础，但是很难懂。对应的中文版有《金融时间序列分析常见问题集》，当然，目前还没有发出来。经济领域的时间序列有一种特殊的情况叫协整，很多人很关注这方面的理论，关心这个的可以看《Analysis of Integrated and Cointegrated Time Series with R》。最后，比较高级的一本书是关于小波分析的，看《Wavelet Methods in Statistics with R》。

6）.金融

金融的领域很广泛，如果是大金融的话，保险也要被纳入此间。用R做金融更多地需要掌握的是金融知识，只会数据分析技术意义寥寥。我觉得这些书对于懂金融、不同数据分析技术的人比较有用，只懂数据分析技术而不动金融知识的人看起来肯定如雾里看花，甚至有人会觉得金融分析比较低级。这方面比较经典的书籍有：《Advanced Topics in Analysis of Economic and Financial Data Using R》以及《Modelling Financial Time Series With S-plus》。金融产品定价之类的常常要用到随机微分方程，有一本叫《Simulation Inference Stochastic Differential Equations：with R examples》的书是关于这方面的内容的，有实例，内容还算详实!此外，是风险度量与管理类。比较经典的有《Simulation Techniques in Financial Risk Management》、《Modern Actuarial Risk Theory Using R》和《Quantitative Risk Management：Concepts, Techniques and Tools》。投资组合分析类和期权定价类可以分别看《Portfolio Optimization with R》和《Option Pricing and Estimation of Financial Models with R》。

7）.数据挖掘

这方面的书不多，只有《Data Mining with R:learing with case studies》。不过，R中数据挖掘方面的包已经足够多了，参考包中的帮助文档就足够了。

**6 R语言入门之Hello world**

**（1）环境搭建**

注：以win7环境为例

[1] 先下载R语言安装包，可以从R语言官网https://www.r-project.org/ 上去下载，选择一个镜像站点，当前的下载包为R-3.3.2-win.exe，大约72M，下载完之后直接安装;

[2] 安装完之后，在桌面会有R x64 3.3.2和R i386 3.3.2两个图标，分别表示64位和32位的环境。这里选择64位的环境，点击进入。

注：这里使用的R语言自带的解释环境，也可以使用R语言集成开发环境RStudio。

**（2）Hello world代码示例**

从上面搭建好的环境，点击进入提示环境如下：

>

然后直接输入print ("Hello world!"); ，如下：

> print ("Hello world!");

[1] "Hello world!"

>

**（3）统计分析代码示例**

下面是一个婴儿月龄与体重的统计分析示例：

> age<-c(1,3,5,2,11,9,3,9,12,3) #使用组合函数c()创建向量age

> weight<-c(4.4,5.3,7.2,5.2,8.5,7.3,6.0,10.4,10.2,6.1) #使用组合函数c()创建体重向量weight

> mean(weight) #调用mean求平均值的函数，求体重的平均值

[1] 7.06

> sd(weight) #调用sd()求标准差的函数，得到体重的标准差

[1] 2.077498

> cor(age, weight) #调用cor()函数，得到月龄与体重的相关度

[1] 0.9075655

> plot(age, weight) #使用plot()函数，用图形展示月龄和体重的关系

>

**（4）运行R语言脚本的方法**

> source("C:\\Users\\dhu\\Desktop\\hello.R")

**（5）ubuntu环境部署**

**[1] 安装R语言(Ubuntu默认未安装R语言包)**

$ sudo apt-get update

$ sudo apt-get install r-base-core

**[2] 安装依赖包**

$ sudo R

$ install.packages("DBI")

$ install.packages("Rserve")

$ install.packages("RMySQL")

注意：镜像选择的时候部分镜像不可用，可以选择39，即厦门大学的可用。

**[3] 配置MySQL访问组**

在 /etc/mysql/my.cnf中添加如下代码，然重启MySQL

[RF7]

user=xxx

password=xx

host=192.168.2.80

database=xxx

default-character-set=utf8

注意：配置好之后需要执行service mysql restart重启MySQL

**[4] 代码编写及测试数据**

略

**[5] 运行reg.R**

进入R解释环境，执行如下：

> source("/home/dennis/reg.R")

**[6] 从命令行运行R脚本文件**

$ Rscript reg.R

附：R语言脚本示例

#! /usr/bin/env Rscript

print ("Hello world!");

运行方法

> source ("/home/dennis/hello.R")

[1] "Hello world!"

## 三、Ruby

**1 Ruby简介**

Ruby，一种为简单快捷的面向对象编程（面向对象程序设计）而创的脚本语言，在20世纪90年代由日本人松本行弘(Yukihiro Matsumoto)开发，遵守GPL协议和Ruby License。它的灵感与特性来自于 Perl、Smalltalk、Eiffel、Ada以及 Lisp 语言。由 Ruby 语言本身还发展出了JRuby（Java平台）、IronRuby（.NET平台）等其他平台的 Ruby 语言替代品。Ruby的作者于1993年2月24日开始编写Ruby，直至1995年12月才正式公开发布于fj（新闻组）。因为Perl发音与6月诞生石pearl（珍珠）相同，因此Ruby以7月诞生石ruby（红宝石）命名。

Ruby是一种纯粹的面向对象编程语言。它由日本的松本行弘（まつもとゆきひろ/Yukihiro Matsumoto）创建于1993年。您可以在 www.ruby-lang.org 的 Ruby 邮件列表上找到松本行弘（まつもとゆきひろ/Yukihiro Matsumoto）的名字。在 Ruby 社区，松本也被称为马茨（Matz）。Ruby 是"程序员的最佳朋友"。Ruby 的特性与 Smalltalk、Perl 和 Python 类似。Perl、Python 和 Smalltalk 是脚本语言。Smalltalk 是一个真正的面向对象语言。Ruby，与 Smalltalk 一样，是一个完美的面向对象语言。使用 Ruby 的语法比使用 Smalltalk 的语法要容易得多。

**2 Ruby 的特性**

Ruby 是开源的，在 Web 上免费提供，但需要一个许可证。

Ruby 是一种通用的、解释的编程语言。

Ruby 是一种真正的面向对象编程语言。

Ruby 是一种类似于 Python 和 Perl 的服务器端脚本语言。

Ruby 可以用来编写通用网关接口（CGI）脚本。

Ruby 可以被嵌入到超文本标记语言（HTML）。

Ruby 语法简单，这使得新的开发人员能够快速轻松地学习 Ruby。

Ruby 与 C++ 和 Perl 等许多编程语言有着类似的语法。

Ruby 可扩展性强，用 Ruby 编写的大程序易于维护。

Ruby 可用于开发的 Internet 和 Intranet 应用程序。

Ruby 可以安装在 Windows 和 POSIX 环境中。

Ruby 支持许多 GUI 工具，比如 Tcl/Tk、GTK 和 OpenGL。

Ruby 可以很容易地连接到 DB2、MySQL、Oracle 和 Sybase。

Ruby 有丰富的内置函数，可以直接在 Ruby 脚本中使用。

**3 工具**

为了执行本教程中讨论的实例，您需要 RAM 至少为 2GB（推荐为 4GB）的 Intel Core i3 或 i5 的计算机。您还需要以下软件：

Linux 或 Windows 95/98/2000/NT 或 Windows 7 操作系统

Apache 1.3.19-5 Web 服务器

Internet Explorer 5.0 或以上的 Web 浏览器

Ruby 1.8.5

**4 Ruby 安装 - Linux**

**（1）源码安装**

下载 Ruby 之后，解压到新创建的目录下：

$ tar -xvzf ruby-2.2.3.tgz

$ cd ruby-2.2.3

现在，配置并编译源代码，如下所示：

$ ./configure

$ make

$ sudo make install

安装后，通过在命令行中输入以下命令来确保一切工作正常：

$ruby -v

ruby 2.2.3……

如果一切工作正常，将会输出所安装的 Ruby 解释器的版本，如上所示。如果您安装了其他版本，则会显示其他不同的版本。

**（2）自动安装 Ruby**

如果您的计算机已经连接到 Internet，那么最简单安装 Ruby 的方式是使用 yum 或 apt-get。在命令提示符中输入以下的命令，即可在您的计算机上安装 Ruby。

$ sudo yum install ruby # CentOS, Fedora, 或 RHEL 系统

或

sudo apt-get install ruby-full # Debian 或 Ubuntu 系统

如果你是苹果系统，可以使用 brew 命令安装：

$ brew install ruby

**5 运行**

通过Ruby命令

ruby HelloRuby.rb

通过irb

irb>

irb(main):001:0>print "Hello World!\n"

Hello World!

## 四、Scala

**1 Scala 简介**

Scala 是 Scalable Language 的简写，是一门多范式的编程语言联邦理工学院洛桑（EPFL）的Martin Odersky于2001年基于Funnel的工作开始设计Scala。Funnel是把函数式编程思想和Petri网相结合的一种编程语言。Odersky先前的工作是Generic Java和javac（Sun Java编译器）。Java平台的Scala于2003年底/2004年初发布。.NET平台的Scala发布于2004年6月。该语言第二个版本，v2.0，发布于2006年3月。

截至2009年9月，最新版本是版本2.7.6 。Scala 2.8预计的特性包括重写的Scala类库（Scala collections library）、方法的命名参数和默认参数、包对象（package object），以及Continuation。

2009年4月，Twitter宣布他们已经把大部分后端程序从Ruby迁移到Scala，其余部分也打算要迁移。此外， Wattzon已经公开宣称，其整个平台都已经是基于Scala基础设施编写的。

**2 Scala 特性**

**（1）面向对象特性**

Scala是一种纯面向对象的语言，每个值都是对象。对象的数据类型以及行为由类和特质描述。类抽象机制的扩展有两种途径：一种途径是子类继承，另一种途径是灵活的混入机制。这两种途径能避免多重继承的种种问题。

**（2）函数式编程**

Scala也是一种函数式语言，其函数也能当成值来使用。Scala提供了轻量级的语法用以定义匿名函数，支持高阶函数，允许嵌套多层函数，并支持柯里化。Scala的case class及其内置的模式匹配相当于函数式编程语言中常用的代数类型。更进一步，程序员可以利用Scala的模式匹配，编写类似正则表达式的代码处理XML数据。

**（3）静态类型**

Scala具备类型系统，通过编译时检查，保证代码的安全性和一致性。类型系统具体支持以下特性：

泛型类

协变和逆变

标注

类型参数的上下限约束

把类别和抽象类型作为对象成员

复合类型

引用自己时显式指定类型

视图

多态方法

**（4）扩展性**

Scala的设计秉承一项事实，即在实践中，某个领域特定的应用程序开发往往需要特定于该领域的语言扩展。Scala提供了许多独特的语言机制，可以以库的形式轻易无缝添加新的语言结构：

任何方法可用作前缀或后缀操作符

可以根据预期类型自动构造闭包。

**（5）并发性**

Scala使用Actor作为其并发模型，Actor是类似线程的实体，通过邮箱发收消息。Actor可以复用线程，因此可以在程序中可以使用数百万个Actor,而线程只能创建数千个。在2.10之后的版本中，使用Akka作为其默认Actor实现。

**3 Linux 上安装 Scala**

**（1）Java 设置**

确保你本地以及安装了 JDK 1.5 以上版本，并且设置了 JAVA\_HOME 环境变量及 JDK 的bin目录。我们可以使用以下命令查看是否安装了 Java：

$ java -version

java version "1.8.0\_31"

Java(TM) SE Runtime Environment (build 1.8.0\_31-b13)

Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)

$

接着，我们可以查看是否安装了 Java 编译器。输入以下命令查看：

$ javac -version

javac 1.8.0\_31

$

**（2） 安装Scala**

http://www.scala-lang.org/downloads 下载 Scala 二进制包，本教程我们将下载 2.11.7版本，如下图所示：

解压缩文件包，可将其移动至/usr/local/share下：

mv scala-2.11.7 scala # 重命名 Scala 目录

mv /download/scalapath /usr/local/share # 下载目录需要按你实际的下载路径

修改环境变量，如果不是管理员可使用 sudo 进入管理员权限，修改配置文件profile:

vim /etc/profile

或

sudo vim /etc/profile

在文件的末尾加入:

export PATH="$PATH:/usr/local/share/scala/bin"

:wq!保存退出，重启终端，执行 scala 命令，输出以下信息，表示安装成功：

$ scala

Welcome to Scala version 2.11.7 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0\_31).

Type in expressions to have them evaluated.

Type :help for more information.

**4 Scala 基础语法**

如果你之前是一名 Java 程序员，并了解 Java 语言的基础知识，那么你能很快学会 Scala 的基础语法。Scala 与 Java 的最大区别是：Scala 语句末尾的分号 ; 是可选的。我们可以认为 Scala 程序是对象的集合，通过调用彼此的方法来实现消息传递。接下来我们来理解下，类，对象，方法，实例变量的概念：

（1）对象 - 对象有属性和行为。例如：一只狗的状属性有：颜色，名字，行为有：叫、跑、吃等。对象是一个类的实例。

（2）类 - 类是对象的抽象，而对象是类的具体实例。

（3）方法 - 方法描述的基本的行为，一个类可以包含多个方法。

（4）字段 - 每个对象都有它唯一的实例变量集合，即字段。对象的属性通过给字段赋值来创建。

**5 第一个 Scala 程序**

**（1）交互式编程**

交互式编程不需要创建脚本文件，可以通过以下命令调用：

$ scala

Welcome to Scala version 2.11.7 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0\_31).

Type in expressions to have them evaluated.

Type :help for more information.

scala> 1 + 1

res0: Int = 2

scala> println("Hello World!")

Hello World!

scala>

**（2）脚本形式**

我们也可以通过创建一个 HelloWorld.scala 的文件来执行代码，HelloWorld.scala 代码如下所示：

object HelloWorld {

/\* 这是我的第一个 Scala 程序

\* 以下程序将输出'Hello World!'

\*/

def main(args: Array[String]) {

println("Hello, world!") // 输出 Hello World

}

}

接下来我们使用 scalac 命令编译它：

$ scalac HelloWorld.scala

$ ls

HelloWorld$.class HelloWorld.scala

HelloWorld.class

编译后我们可以看到目录下生成了 HelloWorld.class 文件，该文件可以在Java Virtual Machine (JVM)上运行。编译后，我们可以使用以下命令来执行程序：

$ scala HelloWorld

Hello, world!

**6 Scala Collection**

Scala提供了一套很好的集合实现，提供了一些集合类型的抽象。

Scala 集合分为可变的和不可变的集合。可变集合可以在适当的地方被更新或扩展。这意味着你可以修改，添加，移除一个集合的元素。而不可变集合类，相比之下，永远不会改变。不过，你仍然可以模拟添加，移除或更新操作。但是这些操作将在每一种情况下都返回一个新的集合，同时使原来的集合不发生改变。接下来我们将为大家介绍几种常用集合类型的应

|  |  |
| --- | --- |
| **序号** | **集合及描述** |
| 1 | [Scala List(列表)](http://www.runoob.com/scala/scala-lists.html" \t "http://www.runoob.com/scala/_blank)  List的特征是其元素以线性方式存储，集合中可以存放重复对象。  参考 [API文档](http://www.scala-lang.org/api/current/scala/collection/immutable/List.html) |
| 2 | [Scala Set(集合)](http://www.runoob.com/scala/scala-sets.html)  Set是最简单的一种集合。集合中的对象不按特定的方式排序，并且没有重复对象。  参考 [API文档](http://www.scala-lang.org/api/current/scala/collection/immutable/Set.html" \t "http://www.runoob.com/scala/_blank) |
| 3 | [Scala Map(映射)](http://www.runoob.com/scala/scala-maps.html)  Map 是一种把键对象和值对象映射的集合，它的每一个元素都包含一对键对象和值对象。  参考 [API文档](http://www.scala-lang.org/api/current/scala/collection/immutable/Map.html" \t "http://www.runoob.com/scala/_blank) |
| 4 | [Scala 元组](http://www.runoob.com/scala/scala-tuples.html" \t "http://www.runoob.com/scala/_blank)  元组是不同类型的值的集合 |
| 5 | [Scala Option](http://www.runoob.com/scala/scala-options.html)  Option[T] 表示有可能包含值的容器，也可能不包含值。 |
| 6 | [Scala Iterator（迭代器）](http://www.runoob.com/scala/scala-iterators.html)  迭代器不是一个容器，更确切的说是逐一访问容器内元素的方法。 |

迭代器不是一个容器，更确切的说是逐一访问容器内元素的方法。

实例

// 定义整型 List

val x = List(1,2,3,4)

// 定义 Set

val x = Set(1,3,5,7)

// 定义 Map

val x = Map("one" -> 1, "two" -> 2, "three" -> 3)

// 创建两个不同类型元素的元组

val x = (10, "Runoob")

// 定义 Option

val x:Option[Int] = Some(5)

**7 Scala 文件 I/O**

**（1）Scala 进行文件写操作，直接用的都是 java中 的 I/O 类 （java.io.File)：**

import java.io.\_

object Test {

def main(args: Array[String]) {

val writer = new PrintWriter(new File("test.txt" ))

writer.write("菜鸟教程")

writer.close()

}

}

执行以上代码，会在你的当前目录下生产一个 test.txt 文件，文件内容为"菜鸟教程":

$ scalac Test.scala

$ scala Test

$ cat test.txt

菜鸟教程

**（2）从屏幕上读取用户输入**

有时候我们需要接收用户在屏幕输入的指令来处理程序。实例如下：

import scala.io.\_

object Test {

def main(args: Array[String]) {

print("请输入菜鸟教程官网 : " )

val line = StdIn.readLine()

println("谢谢，你输入的是: " + line)

}

}

Scala2.11 后的版本 Console.readLine 已废弃，使用 scala.io.StdIn.readLine() 方法代替。

执行以上代码，屏幕上会显示如下信息:

$ scalac Test.scala

$ scala Test

请输入菜鸟教程官网 : www.runoob.com

谢谢，你输入的是: www.runoob.com

**（3）从文件上读取内容**

从文件读取内容非常简单。我们可以使用 Scala 的 Source 类及伴生对象来读取文件。以下实例演示了从 "test.txt"(之前已创建过) 文件中读取内容:

import scala.io.Source

object Test {

def main(args: Array[String]) {

println("文件内容为:" )

Source.fromFile("test.txt" ).foreach{

print

}

}

}

执行以上代码，输出结果为:

$ scalac Test.scala

$ scala Test

文件内容为:

菜鸟教程

**8 Scala 提取器(Extractor)**

提取器是从传递给它的对象中提取出构造该对象的参数。Scala 标准库包含了一些预定义的提取器，我们会大致的了解一下它们。Scala 提取器是一个带有unapply方法的对象。unapply方法算是apply方法的反向操作：unapply接受一个对象，然后从对象中提取值，提取的值通常是用来构造该对象的值。

object Test {

def main(args: Array[String]) {

println ("Apply 方法 : " + apply("Zara", "gmail.com"));

println ("Unapply 方法 : " + unapply("Zara@gmail.com"));

println ("Unapply 方法 : " + unapply("Zara Ali"));

}

// 注入方法 (可选)

def apply(user: String, domain: String) = {

user +"@"+ domain

}

// 提取方法（必选）

def unapply(str: String): Option[(String, String)] = {

val parts = str split "@"

if (parts.length == 2){

Some(parts(0), parts(1))

}else{

None

}

}

}

输出结果

$ scalac Test.scala

$ scala Test

Apply 方法 : Zara@gmail.com

Unapply 方法 : Some((Zara,gmail.com))

Unapply 方法 : None

**9 Scala 异常处理**

异常捕捉的机制与其他语言中一样，如果有异常发生，catch字句是按次序捕捉的。因此，在catch字句中，越具体的异常越要靠前，越普遍的异常越靠后。 如果抛出的异常不在catch字句中，该异常则无法处理，会被升级到调用者处。捕捉异常的catch子句，语法与其他语言中不太一样。在Scala里，借用了模式匹配的思想来做异常的匹配，因此，在catch的代码里，是一系列case字句，如下例所示：

import java.io.FileReader

import java.io.FileNotFoundException

import java.io.IOException

object Test {

def main(args: Array[String]) {

try {

val f = new FileReader("input.txt")

} catch {

case ex: FileNotFoundException =>{

println("Missing file exception")

}

case ex: IOException => {

println("IO Exception")

}

}

}

}

**10 Scala 正则表达式**

实例中使用 String 类的 r() 方法构造了一个Regex对象。

然后使用 findFirstIn 方法找到首个匹配项。

如果需要查看所有的匹配项可以使用 findAllIn 方法。

你可以使用 mkString( ) 方法来连接正则表达式匹配结果的字符串，并可以使用管道(|)来设置不同的模式：

import scala.util.matching.Regex

object Test {

def main(args: Array[String]) {

val pattern = new Regex("(S|s)cala") // 首字母可以是大写 S 或小写 s

val str = "Scala is scalable and cool"

println((pattern findAllIn str).mkString(",")) // 使用逗号 , 连接返回结果

}

}

执行以上代码，输出结果为：

$ scalac Test.scala

$ scala Test

Scala,scala

**11 Scala 类和对象**

类是对象的抽象，而对象是类的具体实例。类是抽象的，不占用内存，而对象是具体的，占用存储空间。类是用于创建对象的蓝图，它是一个定义包括在特定类型的对象中的方法和变量的软件模板。

我们可以使用 new 关键字来创建类的对象，实例如下：

class Point(xc: Int, yc: Int) {

var x: Int = xc

var y: Int = yc

def move(dx: Int, dy: Int) {

x = x + dx

y = y + dy

println ("x 的坐标点: " + x);

println ("y 的坐标点: " + y);

}

}

import java.io.\_

class Point(xc: Int, yc: Int) {

var x: Int = xc

var y: Int = yc

def move(dx: Int, dy: Int) {

x = x + dx

y = y + dy

println ("x 的坐标点: " + x);

println ("y 的坐标点: " + y);

}

}

object Test {

def main(args: Array[String]) {

val pt = new Point(10, 20);

// 移到一个新的位置

pt.move(10, 10);

}

}

执行以上代码，输出结果为：

$ scalac Test.scala

$ scala Test

x 的坐标点: 20

y 的坐标点: 30

**12 Scala Iterator（迭代器）**

Scala Iterator（迭代器）不是一个集合，它是一种用于访问集合的方法。

迭代器 it 的两个基本操作是 next 和 hasNext。

调用 it.next() 会返回迭代器的下一个元素，并且更新迭代器的状态。

调用 it.hasNext() 用于检测集合中是否还有元素。

让迭代器 it 逐个返回所有元素最简单的方法是使用 while 循环：

object Test {

def main(args: Array[String]) {

val it = Iterator("Baidu", "Google", "Runoob", "Taobao")

while (it.hasNext){

println(it.next())

}

}

}

执行以上代码，输出结果为：

$ scalac Test.scala

$ scala Test

Baidu

Google

Runoob

Taobao