Python基础

1. 认识python和基础知识

2. 判断语句和循环语句

3. 字符串、列表、元组、字典

4. 函数

5. 文件操作、综合应用

6. 面向对象1

7. 面向对象2

8. 面向对象3、异常、模块

9. 强化练习

10. 应用:打飞机

1. 认识python和基础知识

1.1. 认识python(了解)

认识python(了解)

1. Python发展历史

起源

Python的作者，Guido von Rossum，荷兰人。1982年，Guido从阿姆斯特丹大学获得了数学和计算机硕士学位。然而，尽管他算得上是一位数学家，但他更加享受计算机带来的乐趣。用他的话说，尽管拥有数学和计算机双料资质，他总趋向于做计算机相关的工作，并热衷于做任何和编程相关的活儿。

在那个时候，Guido接触并使用过诸如Pascal、C、Fortran等语言。这些语言的基本设计原则是让机器能更快运行。在80年代，虽然IBM和苹果已经掀起了个人电脑浪潮，但这些个人电脑的配置很低。比如早期的Macintosh，只有8MHz的CPU主频和128KB的RAM，一个大的数组就能占满内存。所有的编译器的核心是做优化，以便让程序能够运行。为了增进效率，语言也迫使程序员像计算机一样思考，以便能写出更符合机器口味的程序。在那个时代，程序员恨不得用手榨取计算机每一寸的能力。有人甚至认为C语言的指针是在浪费内存。至于动态类型，内存自动管理，面向对象…… 别想了，那会让你的电脑陷入瘫痪。

这种编程方式让Guido感到苦恼。Guido知道如何用C语言写出一个功能，但整个编写过程需要耗费大量的时间，即使他已经准确的知道了如何实现。他的另一个选择是shell。Bourne Shell作为UNIX系统的解释器已经长期存在。UNIX的管理员们常常用shell去写一些简单的脚本，以进行一些系统维护的工作，比如定期备份、文件系统管理等等。shell可以像胶水一样，将UNIX下的许多功能连接在一起。许多C语言下上百行的程序，在shell下只用几行就可以完成。然而，shell的本质是调用命令。它并不是一个真正的语言。比如说，shell没有数值型的数据类型，加法运算都很复杂。总之，shell不能全面的调动计算机的功能。

Guido希望有一种语言，这种语言能够像C语言那样，能够全面调用计算机的功能接口，又可以像shell那样，可以轻松的编程。ABC语言让Guido看到希望。ABC是由荷兰的数学和计算机研究所开发的。Guido在该研究所工作，并参与到ABC语言的开发。ABC语言以教学为目的。与当时的大部分语言不同，ABC语言的目标是“让用户感觉更好”。ABC语言希望让语言变得容易阅读，容易使用，容易记忆，容易学习，并以此来激发人们学习编程的兴趣。比如下面是一段来自Wikipedia的ABC程序，这个程序用于统计文本中出现的词的总数：

HOW TO RETURN words document:

PUT {} IN collection

FOR line IN document:

FOR word IN split line:

IF word not.in collection:

INSERT word IN collection

RETURN collection

HOW TO用于定义一个函数。一个Python程序员应该很容易理解这段程序。ABC语言使用冒号和缩进来表示程序块。行 尾没有分号。for和if结构中也没有括号() 。赋值采用的是PUT，而不是更常见的等号。这些改动让ABC程序读起来像一段文字。 尽管已经具备了良好的可读性和易用性，ABC语言最终没有流行起来。在当时，ABC语言编译器需要比较高配置的电脑才能运行。而这些电脑的使用者通常精通计算机，他们更多考虑程序的效率，而非它的学习难度。除了硬件上的困难外，ABC语言的设计也存在一些致命的问题： 可拓展性差。ABC语言不是模块化语言。如果想在ABC语言中增加功能，比如对图形化的支持，就必须改动很多地方。 不能直接进行IO。ABC语言不能直接操作文件系统。尽管你可以通过诸如文本流的方式导入数据，但ABC无法直接读写文 件。输入输出的困难对于计算机语言来说是致命的。你能想像一个打不开车门的跑车么？ 过度革新。ABC用自然语言的方式来表达程序的意义，比如上面程序中的HOW TO 。然而对于程序员来说，他们更习惯 用function或者define来定义一个函数。同样，程序员更习惯用等号来分配变量。尽管ABC语言很特别，但学习难度 也很大。 传播困难。ABC编译器很大，必须被保存在磁带上。当时Guido在访问的时候，就必须有一个大磁带来给别人安装ABC编 译器。 这样，ABC语言就很难快速传播。 1989年，为了打发圣诞节假期，Guido开始写Python语言的编译器。Python这个名字，来自Guido所挚爱的电视剧Monty Python's Flying Circus。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。Guido作为一个语言设计爱好者，已经有过设计语言的尝试。这一次，也不过是一次纯粹的hacking行为。

一门语言的诞生

1991年，第一个Python编译器诞生。它是用C语言实现的，并能够调用C语言的库文件。从一出生，Python已经具有了 ：类，函数，异常处理，包含表和词典在内的核心数据类型，以及模块为基础的拓展系统。 Python语法很多来自C，但又受到ABC语言的强烈影响。来自ABC语言的一些规定直到今天还富有争议，比如强制缩进。 但这些语法规定让Python容易读。另一方面，Python聪明的选择服从一些惯例，特别是C语言的惯例，比如回归等号赋值。Guido认为，如果“常识”上确立的东西，没有必要过度纠结。 Python从一开始就特别在意可拓展性。Python可以在多个层次上拓展。从高层上，你可以直接引入. py文件。在底层，你可以引用C语言的库。Python程序员可以快速的使用Python写. py文件作为拓展模块。但当性能是考虑的重要因素时，Python程序员可以深入底层，写C程序，编译为.so文件引入到Python中使用。Python就好像是使用钢构建房一样，先规定好大的框架。而程序员可以在此框架下相当自由的拓展或更 改。 最初的Python完全由Guido本人开发。Python得到Guido同事的欢迎。他们迅速的反馈使用意见，并参与到Python的改进。Guido和一些同事构成Python的核心团队。他们将自己大部分的业余时间用于hack Python。随后，Python拓 展到研究所之外。Python将许多机器层面上的细节隐藏，交给编译器处理，并凸显出逻辑层面的编程思考。Python程 序员可以花更多的时间用于思考程序的逻辑，而不是具体的实现细节。这一特征吸引了广大的程序员。Python开始流行。

人生苦短，我用python

时势造英雄

我们不得不暂停我们的Python时间，转而看一看瞬息万变的计算机行业。1990年代初，个人计算机开始进入普通家庭。Intel发布了486处理器，windows发布window 3.0开始的一系列视窗系统。计算机的性能大大提高。程序员开始关注计算机的易用性，比如图形化界面。

Windows 3.0

由于计算机性能的提高，软件的世界也开始随之改变。硬件足以满足许多个人电脑的需要。硬件厂商甚至渴望高需求软 件的出现，以带动硬件的更新换代。C++和Java相继流行。C++和Java提供了面向对象的编程范式，以及丰富的对象库。在牺牲了一定的性能的代价下，C++和Java大大提高了程序的产量。语言的易用性被提到一个新的高度。我们还记得 ，ABC失败的一个重要原因是硬件的性能限制。从这方面说，Python要比ABC幸运许多。 另一个悄然发生的改变是Internet。1990年代还是个人电脑的时代，windows和Intel挟PC以令天下，盛极一时。尽管Internet为主体的信息革命尚未到来，但许多程序员以及资深计算机用户已经在频繁使用Internet进行交流，比如 使用email和newsgroup。Internet让信息交流成本大大下降。一种新的软件开发模式开始流行：开源。程序员利用 业余时间进行软件开发，并开放源代码。1991年，Linus在comp.os.minix新闻组上发布了Linux内核源代码，吸引大批hacker的加入。Linux和GNU相互合作，最终构成了一个充满活力的开源平台。 硬件性能不是瓶颈，Python又容易使用，所以许多人开始转向Python。Guido维护了一个maillist，Python用户就通过邮件进行交流。Python用户来自许多领域，有不同的背景，对Python也有不同的需求。Python相当的开放，又容 易拓展，所以当用户不满足于现有功能，很容易对Python进行拓展或改造。随后，这些用户将改动发给Guido，并由Gu ido决定是否将新的特征加入到Python或者标准库中。如果代码能被纳入Python自身或者标准库，这将极大的荣誉。由于Guido至高无上的决定权，他因此被称为“终身的仁慈独裁者”。 Python被称为“Battery Included”，是说它以及其标准库的功能强大。这些是整个社区的贡献。Python的开发者来自不同领域，他们将不同领域的优点带给Python。比如Python标准库中的正则表达是参考Perl，而lambda, map, filter, reduce等函数参考了Lisp。Python本身的一些功能以及大部分的标准库来自于社区。Python的社 区不断扩大，进而拥有了自己的newsgroup，网站，以及基金。从Python 2.0开始，Python也从maillist的开发方式，转为完全开源的开发方式。社区气氛已经形成，工作被整个社区分担，Python也获得了更加高速的发展。 到今天，Python的框架已经确立。Python语言以对象为核心组织代码，支持多种编程范式，采用动态类型，自动进行内存回收。Python支持解释运行，并能调用C库进行拓展。Python有强大的标准库。由于标准库的体系已经稳定，所以Python的生态系统开始拓展到第三方包。这些包，如Django、web.py、wxpython、numpy、matplotlib、PIL，将Python升级成了物种丰富的热带雨林。

启示录

Python崇尚优美、清晰、简单，是一个优秀并广泛使用的语言。Python在TIOBE排行榜中排行第八，它是Google的第三大开发语言，Dropbox的基础语言，豆瓣的服务器语言。Python的发展史可以作为一个代表，带给我许多启示。 在Python的开发过程中，社区起到了重要的作用。Guido自认为自己不是全能型的程序员，所以他只负责制订框架。如果问题太复杂，他会选择绕过去，也就是cut the corner。这些问题最终由社区中的其他人解决。社区中的人才是异常丰富的，就连创建网站，筹集基金这样与开发稍远的事情，也有人乐意于处理。如今的项目开发越来越复杂，越来越庞大，合作以及开放的心态成为项目最终成功的关键。 Python从其他语言中学到了很多，无论是已经进入历史的ABC，还是依然在使用的C和Perl，以及许多没有列出的其他 语言。可以说，Python的成功代表了它所有借鉴的语言的成功。同样，Ruby借鉴了Python，它的成功也代表了Python某些方面的成功。每个语言都是混合体，都有它优秀的地方，但也有各种各样的缺陷。同时，一个语言“好与不好”的评 判，往往受制于平台、硬件、时代等等外部原因。程序员经历过许多语言之争。其实，以开放的心态来接受各个语言，说不定哪一天，程序员也可以如Guido那样，混合出自己的语言。

关键点常识

Python的发音与拼写

Python的意思是蟒蛇，源于作者喜欢的一部电视剧 (C呢？)

Python的作者是Guido van Rossum（龟叔）

Python是龟叔在1989年圣诞节期间，为了打发无聊的圣诞节而用C编写的一个编程语言

Python正式诞生于1991年

Python的解释器如今有多个语言实现，我们常用的是CPython（官方版本的C语言实现），其他还有Jython（可以运行在Java平台）、IronPython（可以运行在.NET和Mono平台）、PyPy（Python实现的，支持JIT即时编译）

Python目前有两个版本，Python2和Python3，最新版分别为2.7.12和3.5.2，现阶段大部分公司用的是Python2

Life is shot, you need Python. 人生苦短，我用Python。

2017年1月份 编程语言流行排行榜 编程语言排行榜

2. Python优缺点

优点

简单————Python是一种代表简单主义思想的语言。阅读一个良好的Python程序就感觉像是在读英语一样，尽管这个英语的要求非常严格！Python的这种伪代码本质是它最大的优点之一。它使你能够专注于解决问题而不是去搞明白语言本身。

易学————就如同你即将看到的一样，Python极其容易上手。前面已经提到了，Python有极其简单的语法。

免费、开源————Python是FLOSS（自由/开放源码软件）之一。简单地说，你可以自由地发布这个软件的拷贝、阅读它的源代码、对它做改动、把它的一部分用于新的自由软件中。FLOSS是基于一个团体分享知识的概念。这是为什么Python如此优秀的原因之一——它是由一群希望看到一个更加优秀的Python的人创造并经常改进着的。

高层语言————当你用Python语言编写程序的时候，你无需考虑诸如如何管理你的程序使用的内存一类的底层细节。

可移植性————由于它的开源本质，Python已经被移植在许多平台上（经过改动使它能够工作在不同平台上）。如果你小心地避免使用依赖于系统的特性，那么你的所有Python程序无需修改就可以在下述任何平台上面运行。这些平台包括Linux、Windows、FreeBSD、Macintosh、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、Acom RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE甚至还有PocketPC、Symbian以及Google基于linux开发的Android平台！

解释性————这一点需要一些解释。一个用编译性语言比如C或C++写的程序可以从源文件（即C或C++语言）转换到一个你的计算机使用的语言（二进制代码，即0和1）。这个过程通过编译器和不同的标记、选项完成。当你运行你的程序的时候，连接/转载器软件把你的程序从硬盘复制到内存中并且运行。而Python语言写的程序不需要编译成二进制代码。你可以直接从源代码运行程序。在计算机内部，Python解释器把源代码转换成称为字节码的中间形式，然后再把它翻译成计算机使用的机器语言并运行。事实上，由于你不再需要担心如何编译程序，如何确保连接转载正确的库等等，所有这一切使得使用Python更加简单。由于你只需要把你的Python程序拷贝到另外一台计算机上，它就可以工作了，这也使得你的Python程序更加易于移植。

面向对象————Python既支持面向过程的编程也支持面向对象的编程。在“面向过程”的语言中，程序是由过程或仅仅是可重用代码的函数构建起来的。在“面向对象”的语言中，程序是由数据和功能组合而成的对象构建起来的。与其他主要的语言如C++和Java相比，Python以一种非常强大又简单的方式实现面向对象编程。

可扩展性————如果你需要你的一段关键代码运行得更快或者希望某些算法不公开，你可以把你的部分程序用C或C++编写，然后在你的Python程序中使用它们。

丰富的库————Python标准库确实很庞大。它可以帮助你处理各种工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV文件、密码系统、GUI（图形用户界面）、Tk和其他与系统有关的操作。记住，只要安装了Python，所有这些功能都是可用的。这被称作Python的“功能齐全”理念。除了标准库以外，还有许多其他高质量的库，如wxPython、Twisted和Python图像库等等。

规范的代码————Python采用强制缩进的方式使得代码具有极佳的可读性。

缺点

运行速度，有速度要求的话，用C++改写关键部分吧。

国内市场较小（国内以python来做主要开发的，目前只有一些web2.0公司）。但时间推移，目前很多国内软件公司，尤其是游戏公司，也开始规模使用他。

中文资料匮乏（好的python中文资料屈指可数）。托社区的福，有几本优秀的教材已经被翻译了，但入门级教材多，高级内容还是只能看英语版。

构架选择太多（没有像C#这样的官方.net构架，也没有像ruby由于历史较短，构架开发的相对集中。Ruby on Rails 构架开发中小型web程序天下无敌）。不过这也从另一个侧面说明，python比较优秀，吸引的人才多，项目也多。

3. Python应用场景

Web应用开发

Python经常被用于Web开发。比如，通过mod\_wsgi模块，Apache可以运行用Python编写的Web程序。Python定义了WSGI标准应用接口来协调Http服务器与基于Python的Web程序之间的通信。一些Web框架，如Django,TurboGears,web2py,Zope等，可以让程序员轻松地开发和管理复杂的Web程序。

操作系统管理、服务器运维的自动化脚本

在很多操作系统里，Python是标准的系统组件。 大多数Linux发行版以及NetBSD、OpenBSD和Mac OS X都集成了Python，可以在终端下直接运行Python。有一些Linux发行版的安装器使用Python语言编写，比如Ubuntu的Ubiquity安装器,Red Hat Linux和Fedora的Anaconda安装器。Gentoo Linux使用Python来编写它的Portage包管理系统。Python标准库包含了多个调用操作系统功能的库。通过pywin32这个第三方软件 包，Python能够访问Windows的COM服务及其它Windows API。使用IronPython，Python程序能够直接调用.Net Framework。一般说来，Python编写的系统管理脚本在可读性、性能、代码重用度、扩展性几方面都优于普通的shell脚本。

科学计算

NumPy,SciPy,Matplotlib可以让Python程序员编写科学计算程序。

桌面软件

PyQt、PySide、wxPython、PyGTK是Python快速开发桌面应用程序的利器。

服务器软件（网络软件）

Python对于各种网络协议的支持很完善，因此经常被用于编写服务器软件、网络爬虫。第三方库Twisted支持异步网络编程和多数标准的网络协议(包含客户端和服务器)，并且提供了多种工具，被广泛用于编写高性能的服务器软件。

游戏

很多游戏使用C++编写图形显示等高性能模块，而使用Python或者Lua编写游戏的逻辑、服务器。相较于Python，Lua的功能更简单、体积更小；而Python则支持更多的特性和数据类型。

构思实现，产品早期原型和迭代

YouTube、Google、Yahoo!、NASA都在内部大量地使用Python。

1.2. 第一个python程序

<1>编写python程序方法1

打开“超级终端”

输入python3 ，输入python3表示用的python这门编程语言的第3个版本，如果只输入python的话表示用的是python的第2个版本

输入以下代码

print('hello world')

<2>编写python程序方法2

打开编辑软件sublime

把以下代码，编写如下代码

保存代码

运行程序

<3>另外一种运行python的程序的方法

在代码第一行写入执行时的python解释器路径，编辑完后需要对此python文件添加'x'权限

<4>练一练

要求：编写一个程序，输出itcast.cn

<5>小总结

对于编写python程序，上面有3种方法，那到实际开发中哪一种用的比较多呢？一般是用第2或者第3种，即保存在xxx.py文件中，这样可以直接下一次执行运行；而如果用第一种方法的话，每一次运行程序都需要重新进行输入，费时费力

1.3. 注释编写第一个python程序

1. 注释的引入

<1> 看以下程序示例（未使用注释）

使用注释

<2> 看以下程序示例（使用注释）

使用注释

<3> 小总结（注释的作用）

通过用自己熟悉的语言，在程序中对某些代码进行标注说明，这就是注释的作用，能够大大增强程序的可读性

2. 注释的分类

<1> 单行注释

以#开头，#右边的所有东西当做说明，而不是真正要执行的程序，起辅助说明作用

# 我是注释，可以在里写一些功能说明之类的哦

print('hello world')

<2> 多行注释

'''我是多行注释，可以写很多很多行的功能说明

这就是我牛X指出

哈哈哈。。。

'''

'''

下面的代码完成 ，打印一首诗

名字叫做：春江花月夜

作者，忘了

'''

print('╔═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╤═╗')

print('║　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│ ║')

print('║　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　║')

print('║　│春│滟│江│空│江│江│人│不│白│谁│可│玉│此│鸿│昨│江│斜│不│　║')

print('║　│江│滟│流│里│天│畔│生│知│云│家│怜│户│时│雁│夜│水│月│知│　║')

print('║　│潮│随│宛│流│一│何│代│江│一│今│楼│帘│相│长│闲│流│沉│乘│　║')

print('║　│水│波│转│霜│色│人│代│月│片│夜│上│中│望│飞│潭│春│沉│月│　║')

print('║　│连│千│绕│不│无│初│无│待│去│扁│月│卷│不│光│梦│去│藏│几│　║')

print('║春│海│万│芳│觉│纤│见│穷│何│悠│舟│徘│不│相│不│落│欲│海│人│　║')

print('║江│平│里│甸│飞│尘│月│已│人│悠│子│徊│去│闻│度│花│尽│雾│归│　║')

print('║花│，│，│，│，│，│，│，│，│，│，│，│，│，│，│，│，│，│，│　║')

print('║月│海│何│月│汀│皎│江│江│但│青│何│应│捣│愿│鱼│可│江│碣│落│　║')

print('║夜│上│处│照│上│皎│月│月│见│枫│处│照│衣│逐│龙│怜│潭│石│月│　║')

print('║　│明│春│花│白│空│何│年│长│浦│相│离│砧│月│潜│春│落│潇│摇│　║')

print('║　│月│江│林│沙│中│年│年│江│上│思│人│上│华│跃│半│月│湘│情│　║')

print('║　│共│无│皆│看│孤│初│望│送│不│明│妆│拂│流│水│不│复│无│满│　║')

print('║　│潮│月│似│不│月│照│相│流│胜│月│镜│还│照│成│还│西│限│江│　║')

print('║　│生│明│霰│见│轮│人│似│水│愁│楼│台│来│君│文│家│斜│路│树│　║')

print('║　│。│。│。│。│。│？│。│。│。│？│。│。│。│。│。│。│。│。│　║')

print('║　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　║')

print('║　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　│　║')

print('╚═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╧═╝')

3. python程序中，中文支持

如果直接在程序中用到了中文，比如

print('你好')

如果直接运行输出，程序会出错：

中文出错

解决的办法为：在程序的开头写入如下代码，这就是中文注释

#coding=utf-8

修改之后的程序:

#coding=utf-8

print('你好')

运行结果:

你好

注意：

在python的语法规范中推荐使用的方式：

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

1.4. 变量以及类型

变量以及类型

<1>变量的定义

在程序中，有时我们需要对2个数据进行求和，那么该怎样做呢？

大家类比一下现实生活中，比如去超市买东西，往往咱们需要一个菜篮子，用来进行存储物品，等到所有的物品都购买完成后，在收银台进行结账即可

如果在程序中，需要把2个数据，或者多个数据进行求和的话，那么就需要把这些数据先存储起来，然后把它们累加起来即可

在Python中，存储一个数据，需要一个叫做变量的东西，如下示例:

num1 = 100 #num1就是一个变量，就好一个小菜篮子

num2 = 87 #num2也是一个变量

result = num1 + num2 #把num1和num2这两个"菜篮子"中的数据进行累加，然后放到 result变量中

说明:

所谓变量，可以理解为菜篮子，如果需要存储多个数据，最简单的方式是有多个变量，当然了也可以使用一个

程序就是用来处理数据的，而变量就是用来存储数据的

想一想：我们应该让变量占用多大的空间，保存什么样的数据？

<2>变量的类型

生活中的“类型”的例子:

程序中:

为了更充分的利用内存空间以及更有效率的管理内存，变量是有不同的类型的，如下所示:

怎样知道一个变量的类型呢？

在python中，只要定义了一个变量，而且它有数据，那么它的类型就已经确定了，不需要咱们开发者主动的去说明它的类型，系统会自动辨别

可以使用type(变量的名字)，来查看变量的类型

1.5. 标示符和关键字

标示符和关键字

<1>标示符

什么是标示符，看下图:

标示符

开发人员在程序中自定义的一些符号和名称

标示符是自己定义的,如变量名 、函数名等

<2>标示符的规则

标示符由字母、下划线和数字组成，且数字不能开头

思考：下面的标示符哪些是正确的，哪些不正确为什么

fromNo12

from#12

my\_Boolean

my-Boolean

Obj2

2ndObj

myInt

test1

Mike2jack

My\_tExt

\_test

test!32

haha(da)tt

int

jack\_rose

jack&rose

GUI

G.U.I

python中的标识符是区分大小写的

大小写

<3>命名规则

见名知意

起一个有意义的名字，尽量做到看一眼就知道是什么意思(提高代码可 读性) 比如: 名字 就定义为 name , 定义学生 用 student

驼峰命名法

驼峰法 小驼峰式命名法（lower camel case）： 第一个单词以小写字母开始；第二个单词的首字母大写，例如：myName、aDog

大驼峰式命名法（upper camel case）： 每一个单字的首字母都采用大写字母，例如：FirstName、LastName

不过在程序员中还有一种命名法比较流行，就是用下划线“\_”来连接所有的单词，比如send\_buf

<4>关键字

什么是关键字

python一些具有特殊功能的标示符，这就是所谓的关键字

关键字，是python已经使用的了，所以不允许开发者自己定义和关键字相同的名字的标示符

查看关键字:

and as assert break class continue def del

elif else except exec finally for from global

if in import is lambda not or pass

print raise return try while with yield

可以通过以下命令进行查看当前系统中python的关键字

1.6. 输出输出

1. 普通的输出

生活中的“输出”

软件中的“输出”

python中变量的输出

# 打印提示

print('hello world')

print('给我的卡---印度语，你好的意思')

2. 格式化输出

<1>格式化操作的目的

比如有以下代码:

pirnt("我今年10岁")

pirnt("我今年11岁")

pirnt("我今年12岁")

...

想一想:

在输出年龄的时候，用了多次"我今年xx岁"，能否简化一下程序呢？？？

答:

字符串格式化

<2>什么是格式化

看如下代码:

age = 10

print("我今年%d岁"%age)

age += 1

print("我今年%d岁"%age)

age += 1

print("我今年%d岁"%age)

...

在程序中，看到了%这样的操作符，这就是Python中格式化输出。

age = 18

name = "xiaohua"

print("我的姓名是%s,年龄是%d"%(name,age))

<3>常用的格式符号

下面是完整的，它可以与％符号使用列表:

格式符号 转换

%c 字符

%s 通过str() 字符串转换来格式化

%i 有符号十进制整数

%d 有符号十进制整数

%u 无符号十进制整数

%o 八进制整数

%x 十六进制整数（小写字母）

%X 十六进制整数（大写字母）

%e 索引符号（小写'e'）

%E 索引符号（大写“E”）

%f 浮点实数

%g ％f和％e 的简写

%G ％f和％E的简写

3. 换行输出

在输出的时候，如果有\n那么，此时\n后的内容会在另外一行显示

print("1234567890-------") # 会在一行显示

print("1234567890\n-------") # 一行显示1234567890，另外一行显示-------

4. 练一练

编写代码完成以下名片的显示

==================================

姓名: dongGe

QQ:xxxxxxx

手机号:131xxxxxx

公司地址:北京市xxxx

==================================

1.7. 输入输入

1. python2版本中

咱们在银行ATM机器前取钱时，肯定需要输入密码，对不？

那么怎样才能让程序知道咱们刚刚输入的是什么呢？？

大家应该知道了，如果要完成ATM机取钱这件事情，需要先从键盘中输入一个数据，然后用一个变量来保存，是不是很好理解啊

1.1 raw\_input()

在Python中，获取键盘输入的数据的方法是采用 raw\_input 函数（至于什么是函数，咱们以后的章节中讲解），那么这个 raw\_input 怎么用呢?

看如下示例:

password = raw\_input("请输入密码:")

print '您刚刚输入的密码是:', password

运行结果:

注意:

raw\_input()的小括号中放入的是，提示信息，用来在获取数据之前给用户的一个简单提示

raw\_input()在从键盘获取了数据以后，会存放到等号右边的变量中

raw\_input()会把用户输入的任何值都作为字符串来对待

1.2 input()

input()函数与raw\_input()类似，但其接受的输入必须是表达式。

>>> a = input()

123

>>> a

123

>>> type(a)

<type 'int'>

>>> a = input()

abc

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<string>", line 1, in <module>

NameError: name 'abc' is not defined

>>> a = input()

"abc"

>>> a

'abc'

>>> type(a)

<type 'str'>

>>> a = input()

1+3

>>> a

4

>>> a = input()

"abc"+"def"

>>> a

'abcdef'

>>> value = 100

>>> a = input()

value

>>> a

100

input()接受表达式输入，并把表达式的结果赋值给等号左边的变量

2. python3版本中

没有raw\_input()函数，只有input()

并且 python3中的input与python2中的raw\_input()功能一样

1.8. 运算符

运算符

python支持以下几种运算符

算术运算符

下面以a=10 ,b=20为例进行计算

运算符 描述 实例

+ 加 两个对象相加 a + b 输出结果 30

- 减 得到负数或是一个数减去另一个数 a - b 输出结果 -10

\* 乘 两个数相乘或是返回一个被重复若干次的字符串 a \* b 输出结果 200

/ 除 x除以y b / a 输出结果 2

// 取整除 返回商的整数部分 9//2 输出结果 4 , 9.0//2.0 输出结果 4.0

% 取余 返回除法的余数 b % a 输出结果 0

\*\* 幂 返回x的y次幂 a\*\*b 为10的20次方， 输出结果 100000000000000000000

>>> 9/2.0

4.5

>>> 9//2.0

4.0

赋值运算符

运算符 描述 实例

= 赋值运算符 把=号右边的结果给左边的变量 num=1+2\*3 结果num的值为7

>>> a, b = 1, 2

>>> a

1

>>> b

2

复合赋值运算符

运算符 描述 实例

+= 加法赋值运算符 c += a 等效于 c = c + a

-= 减法赋值运算符 c -= a 等效于 c = c - a

\*= 乘法赋值运算符 c \*= a 等效于 c = c \* a

/= 除法赋值运算符 c /= a 等效于 c = c / a

%= 取模赋值运算符 c %= a 等效于 c = c % a

\*\*= 幂赋值运算符 c \*\*= a 等效于 c = c \*\* a

//= 取整除赋值运算符 c //= a 等效于 c = c // a

1.9. 数据类型转换常用的数据类型转换

函数 说明

int(x [,base ]) 将x转换为一个整数

long(x [,base ]) 将x转换为一个长整数

float(x ) 将x转换到一个浮点数

complex(real [,imag ]) 创建一个复数

str(x ) 将对象 x 转换为字符串

repr(x ) 将对象 x 转换为表达式字符串

eval(str ) 用来计算在字符串中的有效Python表达式,并返回一个对象

tuple(s ) 将序列 s 转换为一个元组

list(s ) 将序列 s 转换为一个列表

chr(x ) 将一个整数转换为一个字符

unichr(x ) 将一个整数转换为Unicode字符

ord(x ) 将一个字符转换为它的整数值

hex(x ) 将一个整数转换为一个十六进制字符串

oct(x ) 将一个整数转换为一个八进制字符串

举例

a = '100' # 此时a的类型是一个字符串，里面存放了100这3个字符

b = int(a) # 此时b的类型是整型，里面存放的是数字100

1.10. 判断语句介绍判断语句介绍

<1>生活中的判断场景

火车站安检进火车站

上网吧

上网吧

<2>开发中的判断场景

密码判断

速度与激情5

重要日期判断

if 今天是周六或者周日:

约妹子

if 今天是情人节:

买玫瑰

if 今天发工资:

先还信用卡的钱

if 有剩余:

又可以happy了，O(∩\_∩)O哈哈~

else:

噢，no。。。还的等30天

小总结：

如果某些条件满足，才能做某件事情，而不满足时不允许做，这就是所谓的判断

不仅生活中有，在软件开发中“判断”功能也经常会用到

1.11. if语句

f判断语句

<1>if判断语句介绍

if语句是用来进行判断的，其使用格式如下：

if 要判断的条件:

条件成立时，要做的事情

demo1:

age = 30

print "------if判断开始------"

if age>=18:

print "我已经成年了"

print "------if判断结束------"

运行结果:

------if判断开始------

我已经成年了

------if判断结束------

demo2:

age = 16

print "------if判断开始------"

if age>=18:

print "我已经成年了"

print "------if判断结束------"

运行结果:

------if判断开始------

------if判断结束------

小总结：

以上2个demo仅仅是age变量的值不一样，结果却不同；能够看得出if判断语句的作用：就是当满足一定条件时才会执行那块代码，否则就不执行那块代码

注意：

代码的缩进为一个tab键，或者4个空格

<2>练一练

要求：从键盘获取自己的年龄，判断是否大于或者等于18岁，如果满足就输出“哥，已成年，网吧可以去了”

使用input从键盘中获取数据，并且存入到一个变量中

使用if语句，来判断 age>=18是否成立

<3>想一想

判断age大于或者等于18岁，使用的是 >=，还有哪

1.12. 比较、关系运算符

<1> 比较(即关系)运算符

python中的比较运算符如下表

运算符 描述 示例

== 检查两个操作数的值是否相等，如果是则条件变为真。 如a=3,b=3则（a == b) 为 true.

!= 检查两个操作数的值是否相等，如果值不相等，则条件变为真。 如a=1,b=3则(a != b) 为 true.

<> 检查两个操作数的值是否相等，如果值不相等，则条件变为真。 如a=1,b=3则(a <> b) 为 true。这个类似于 != 运算符

> 检查左操作数的值是否大于右操作数的值，如果是，则条件成立。 如a=7,b=3则(a > b) 为 true.

< 检查左操作数的值是否小于右操作数的值，如果是，则条件成立。 如a=7,b=3则(a < b) 为 false.

>= 检查左操作数的值是否大于或等于右操作数的值，如果是，则条件成立。 如a=3,b=3则(a >= b) 为 true.

<= 检查左操作数的值是否小于或等于右操作数的值，如果是，则条件成立。 如a=3,b=3则(a <= b) 为 true.

<2> 逻辑运算符

运算符 逻辑表达式 描述 实例

and x and y 布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。 (a and b) 返回 20。

or x or y 布尔"或" - 如果 x 是 True，它返回 True，否则它返回 y 的计算值。 (a or b) 返回 10。

not not x 布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。 not(a and b) 返回 False

1.13. 作业

作业

必做题：

1. 说出变量名字，可以由哪些字符组成

2. 写出变量命名时的规则

3. 说出什么是驼峰法（大驼峰、小驼峰）

4. 编写程序，完成以下要求：

提示用户进行输入数据

获取用户的数据数据（需要获取2个）

对获取的两个数字进行求和运行，并输出相应的结果

5. 编写程序，完成以下要求：

提示用户进行输入数据

获取用户的数据数据（需要获取2个）

对获取的两个数字进行减法运行，并输出相应的结果

6. 编写程序，完成以下信息的显示:

==================================

= 欢迎进入到身份认证系统V1.0

= 1. 登录

= 2. 退出

= 3. 认证

= 4. 修改密码

==================================

7. 编写程序，从键盘获取一个人的信息，然后按照下面格式显示

==================================

姓名: dongGe

QQ:xxxxxxx

手机号:131xxxxxx

公司地址:北京市xxxx

==================================

8. 编写程序，从键盘获取用户名和密码，然后判断，如果正确就输出以下信息

亲爱的xxx，欢迎登陆 爱学习管理系统

1.14. 附录-推荐的python电子书

附录-推荐的python电子书

python学习路线图

优先级

入门:python核心编程

提高:python cookbook

其他

2. 判断语句和循环语句

2.1. if-else

if-else

想一想：在使用if的时候，它只能做到满足条件时要做的事情。那万一需要在不满足条件的时候，做某些事，该怎么办呢？

答：else

<1>if-else的使用格式

if 条件:

满足条件时要做的事情1

满足条件时要做的事情2

满足条件时要做的事情3

...(省略)...

else:

不满足条件时要做的事情1

不满足条件时要做的事情2

不满足条件时要做的事情3

...(省略)...

demo1

chePiao = 1 # 用1代表有车票，0代表没有车票

if chePiao == 1:

print("有车票，可以上火车")

print("终于可以见到Ta了，美滋滋~~~")

else:

print("没有车票，不能上车")

print("亲爱的，那就下次见了，一票难求啊~~~~(>\_<)~~~~")

结果1：有车票的情况

有车票，可以上火车

终于可以见到Ta了，美滋滋~~~

结果2：没有车票的情况

没有车票，不能上课

亲爱的，那就下次见了，一票难求啊~~~~(>\_<)~~~~

<2>练一练

要求：从键盘输入刀子的长度，如果刀子长度没有超过10cm，则允许上火车，否则不允许上火车

2.2. elif

elif

想一想:

if能完成当xxx时做事情

if-else能完成当xxx时做事情1，否则做事情2

如果有这样一种情况：当xxx1时做事情1，当xxx2时做事情2，当xxx3时做事情3，那该怎么实现呢？

答:

elif

<1> elif的功能

elif的使用格式如下:

if xxx1:

事情1

elif xxx2:

事情2

elif xxx3:

事情3

说明:

当xxx1满足时，执行事情1，然后整个if结束

当xxx1不满足时，那么判断xxx2，如果xxx2满足，则执行事情2，然后整个if结束

当xxx1不满足时，xxx2也不满足，如果xxx3满足，则执行事情3，然后整个if结束

demo:

score = 77

if score>=90 and score<=100:

print('本次考试，等级为A')

elif score>=80 and score<90:

print('本次考试，等级为B')

elif score>=70 and score<80:

print('本次考试，等级为C')

elif score>=60 and score<70:

print('本次考试，等级为D')

elif score>=0 and score<60:

print('本次考试，等级为E')

<2> 注意点

可以和else一起使用

if 性别为男性:

输出男性的特征

...

elif 性别为女性:

输出女性的特征

...

else:

第三种性别的特征

...

说明:

当 “性别为男性” 满足时，执行 “输出男性的特征”的相关代码

当 “性别为男性” 不满足时，如果 “性别为女性”满足，则执行 “输出女性的特征”的相关代码

当 “性别为男性” 不满足，“性别为女性”也不满足，那么久默认执行else后面的代码，即 “第三种性别的特征”相关代码

elif必须和if一起使用，否则出错

2.3. if嵌套

if嵌套

通过学习if的基本用法，已经知道了

当需要满足条件去做事情的这种情况需要使用if

当满足条件时做事情A，不满足条件做事情B的这种情况使用if-else

想一想：

坐火车或者地铁的实际情况是：先进行安检如果安检通过才会判断是否有车票，或者是先检查是否有车票之后才会进行安检，即实际的情况某个判断是再另外一个判断成立的基础上进行的，这样的情况该怎样解决呢？

答：

if嵌套

<1>if嵌套的格式

if 条件1:

满足条件1 做的事情1

满足条件1 做的事情2

...(省略)...

if 条件2:

满足条件2 做的事情1

满足条件2 做的事情2

...(省略)...

说明

外层的if判断，也可以是if-else

内层的if判断，也可以是if-else

根据实际开发的情况，进行选择

<2>if嵌套的应用

demo：

chePiao = 1 # 用1代表有车票，0代表没有车票

daoLenght = 9 # 刀子的长度，单位为cm

if chePiao == 1:

print("有车票，可以进站")

if daoLenght < 10:

print("通过安检")

print("终于可以见到Ta了，美滋滋~~~")

else:

print("没有通过安检")

print("刀子的长度超过规定，等待警察处理...")

else:

print("没有车票，不能进站")

print("亲爱的，那就下次见了，一票难求啊~~~~(>\_<)~~~~")

结果1：chePiao = 1;daoLenght = 9

有车票，可以进站

通过安检

终于可以见到Ta了，美滋滋~~~

结果2：chePiao = 1;daoLenght = 20

有车票，可以进站

没有通过安检

刀子的长度超过规定，等待警察处理...

结果3：chePiao = 0;daoLenght = 9

没有车票，不能进站

亲爱的，那就下次见了，一票难求啊~~~~(>\_<)~~~~

结果4：chePiao = 0;daoLenght = 20

没有车票，不能进站

亲爱的，那就下次见了，一票难求啊~~~~(>\_<)~~~~

想一想:为什么结果3和结果4相同？？？

<3>练一练

情节描述：上公交车，并且可以有座位坐下

要求：输入公交卡当前的余额，只要超过2元，就可以上公交车；如果空座位的数量大于0，就可以坐下

2.4. if应用:猜拳游戏

应用:猜拳游戏

<1>运行效果:

<2>参考代码:

import random

player = input('请输入：剪刀(0) 石头(1) 布(2):')

player = int(player)

computer = random.randint(0,2)

# 用来进行测试

#print('player=%d,computer=%d',(player,computer))

if ((player == 0) and (computer == 2)) or ((player ==1) and (computer == 0)) or ((player == 2) and (computer == 1)):

print('获胜，哈哈，你太厉害了')

elif player == computer:

print('平局，要不再来一局')

else:

print('输了，不要走，洗洗手接着来，决战到天亮')

2.5. 循环语句介绍

循环介绍

<1>生活中的循环场景

跑道

跑道

风扇

跑道

CF加特林

跑道

<2>软件开发中循环的使用场景

跟媳妇承认错误，说一万遍"媳妇儿，我错了"

print("媳妇儿，我错了")

print("媳妇儿，我错了")

print("媳妇儿，我错了")

...(还有99997遍)...

使用循环语句一句话搞定

i = 0

while i<10000:

print("媳妇儿，我错了")

i+=1

<3>小总结

一般情况下，需要多次重复执行的代码，都可以用循环的方式来完成

循环不是必须要使用的，但是为了提高代码的重复使用率，所以有经验的开发者都会采用循环

2.6. while循环

while循环

<1>while循环的格式

while 条件:

条件满足时，做的事情1

条件满足时，做的事情2

条件满足时，做的事情3

...(省略)...

demo

i = 0

while i<5:

print("当前是第%d次执行循环"%(i+1))

print("i=%d"%i)

i+=1

结果:

当前是第1次执行循环

i=0

当前是第2次执行循环

i=1

当前是第3次执行循环

i=2

当前是第4次执行循环

i=3

当前是第5次执行循环

i=4

2.7. while循环应用

while循环应用

1. 计算1~100的累积和（包含1和100）

参考代码如下:

#encoding=utf-8

i = 1

sum = 0

while i<=100:

sum = sum + i

i += 1

print("1~100的累积和为:%d"%sum)

2. 计算1~100之间偶数的累积和（包含1和100）

参考代码如下:

#encoding=utf-8

i = 1

sum = 0

while i<=100:

if i%2 == 0:

sum = sum + i

i+=1

print("1~100的累积和为:%d"%sum)

2.8. while循环的嵌套以及应用

while循环嵌套

前面学习过if的嵌套了，想一想if嵌套是什么样子的？

类似if的嵌套，while嵌套就是：while里面还有while

<1>while嵌套的格式

while 条件1:

条件1满足时，做的事情1

条件1满足时，做的事情2

条件1满足时，做的事情3

...(省略)...

while 条件2:

条件2满足时，做的事情1

条件2满足时，做的事情2

条件2满足时，做的事情3

...(省略)...

<2>while嵌套应用一

要求：打印如下图形：

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

参考代码：

i = 1

while i<=5:

j = 1

while j<=i:

print("\* ",end='')

j+=1

print("\n")

i+=1

<3>while嵌套应用二：九九乘法表

参考代码：

i = 1

while i<=9:

j=1

while j<=i:

print("%d\*%d=%-2d "%(j,i,i\*j),end='')

j+=1

print('\n')

i+=1

2.9. for循环

for循环

像while循环一样，for可以完成循环的功能。

在Python中 for循环可以遍历任何序列的项目，如一个列表或者一个字符串等。

for循环的格式

for 临时变量 in 列表或者字符串等:

循环满足条件时执行的代码

else:

循环不满足条件时执行的代码

demo1

name = 'dongGe'

for x in name:

print(x)

运行结果如下:

demo2

name = ''

for x in name:

print(x)

else:

print("没有数据")

运行结果如下:

2.10. break和continue

break和continue

1. break

<1> for循环

普通的循环示例如下：

name = 'dongGe'

for x in name:

print('----')

print(x)

运行结果:

带有break的循环示例如下:

name = 'dongGe'

for x in name:

print('----')

if x == 'g':

break

print(x)

运行结果:

<2> while循环

普通的循环示例如下：

i = 0

while i<10:

i = i+1

print('----')

print(i)

运行结果:

带有break的循环示例如下:

i = 0

while i<10:

i = i+1

print('----')

if i==5:

break

print(i)

运行结果:

小总结:

break的作用：用来结束整个循环

2. continue

<1> for循环

带有continue的循环示例如下:

name = 'dongGe'

for x in name:

print('----')

if x == 'g':

continue

print(x)

运行结果:

<2> while循环

带有continue的循环示例如下:

i = 0

while i<10:

i = i+1

print('----')

if i==5:

continue

print(i)

运行结果:

小总结:

continue的作用：用来结束本次循环，紧接着执行下一次的循环

3. 注意点

break/continue只能用在循环中，除此以外不能单独使用

break/continue在嵌套循环中，只对最近的一层循环起作用

2.11. 总结

总结

if往往用来对条件是否满足进行判断

if有4中基本的使用方法：

基本方法

if 条件:

满足时做的事情

满足与否执行不同的事情

if 条件:

满足时做的事情

else:

不满足时做的事情

多个条件的判断

if 条件:

满足时做的事情

elif 条件2:

满足条件2时做的事情

elif 条件3:

满足条件3时做的事情

else:

条件都不满足时做的事情

嵌套

if 条件:

满足时做的事情

这里还可以放入其他任何形式的if判断语句

while循环一般通过数值是否满足来确定循环的条件

i = 0

while i<10:

print("hello")

i+=1

for循环一般是对能保存多个数据的变量，进行便利

name = 'dongGe'

for x in name:

print(x)

if、while、for等其他语句可以随意组合，这样往往就完成了复杂的功能

2.12. 作业

作业

必做题：

1. 使用if，编写程序，实现以下功能：

从键盘获取用户名、密码

如果用户名和密码都正确（预先设定一个用户名和密码），那么就显示“欢迎进入xxx的世界”，否则提示密码或者用户名错误

2. 使用while，完成以下图形的输出

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

选做题：

1. 根据以下信息提示，请帮小明计算，他每月乘坐地铁支出的总费用

提示信息:

北京公交地铁新票价确定

据北京市发改委网站消息称，北京市将从2015年12月28起实施公共交通新票价：地铁6公里(含)内3元，公交车10公里(含)内2元，使用市政交通一卡通刷卡乘公交车普通卡5折，学生卡2.5折。

　　具体实施方案如下：

　　一、城市公共电汽车价格调整为：10公里(含)内2元，10公里以上部分，每增加1元可乘坐5公里。使用市政交通一卡通刷卡乘坐城市公共电汽车，市域内路段给予普通卡5折，学生卡2.5折优惠;市域外路段维持现行折扣优惠不变。享受公交政策的郊区客运价格，由各区、县政府按照城市公共电汽车价格制定。

　　二、轨道交通价格调整为：6公里(含)内3元;6公里至12公里(含)4元;12公里至22公里(含)5元;22公里至32公里(含)6元;32公里以上部分，每增加1元可乘坐20公里。使用市政交通一卡通刷卡乘坐轨道交通，每自然月内每张卡支出累计满100元以后的乘次，价格给予8折优惠;满150元以后的乘次，价格给予5折优惠;支出累计达到400元以后的乘次，不再享受打折优惠。

要求：

假设每个月，小明都需要上20天班，每次上班需要来回1次，即每天需要乘坐2次同样路线的地铁；每月月初小明第一次刷公交卡时，扣款5元；编写程序，帮小明完成每月乘坐地铁需要的总费用

3. 字符串、列表、元组、字典

3.1. 字符串介绍

字符串介绍

想一想：

当打来浏览器登录某些网站的时候，需要输入密码，浏览器把密码传送到服务器后，服务器会对密码进行验证，其验证过程是把之前保存的密码与本次传递过去的密码进行对比，如果相等，那么就认为密码正确，否则就认为不对；服务器既然想要存储这些密码可以用数据库（比如MySQL），当然为了简单起见，咱们可以先找个变量把密码存储起来即可；那么怎样存储带有字母的密码呢？

答：

字符串

<1>python中字符串的格式

如下定义的变量a，存储的是数字类型的值

a = 100

如下定义的变量b，存储的是字符串类型的值

b = "hello itcast.cn"

或者

b = 'hello itcast.cn'

小总结：

双引号或者单引号中的数据，就是字符串

3.2. 字符串输出

字符串输出

demo

name = 'xiaoming'

position = '讲师'

address = '北京市昌平区建材城西路金燕龙办公楼1层'

print('--------------------------------------------------')

print("姓名：%s"%name)

print("职位：%s"%position)

print("公司地址：%s"%address)

print('--------------------------------------------------')

结果:

--------------------------------------------------

姓名： xiaoming

职位： 讲师

公司地址： 北京市昌平区建材城西路金燕龙办公楼1层

--------------------------------------------------

3.3. 字符串输入

字符串输入

之前在学习input的时候，通过它能够完成从键盘获取数据，然后保存到指定的变量中；

注意：input获取的数据，都以字符串的方式进行保存，即使输入的是数字，那么也是以字符串方式保存

demo:

userName = input('请输入用户名:')

print("用户名为：%s"%userName)

password = input('请输入密码:')

print("密码为：%s"%password)

结果：（根据输入的不同结果也不同）

请输入用户名:dongGe

用户名为： dongGe

请输入密码:haohaoxuexitiantianxiangshang

密码为： haohaoxuexitiantianxiangshang

3.4. 下标和切片

下标和切片

1. 下标索引

所谓“下标”，就是编号，就好比超市中的存储柜的编号，通过这个编号就能找到相应的存储空间

生活中的 "下标"

超市储物柜

高铁二等座

高铁一等座

绿皮车

字符串中"下标"的使用

列表与元组支持下标索引好理解，字符串实际上就是字符的数组，所以也支持下标索引。

如果有字符串:name = 'abcdef'，在内存中的实际存储如下:

如果想取出部分字符，那么可以通过下标的方法，（注意python中下标从 0 开始）

name = 'abcdef'

print(name[0])

print(name[1])

print(name[2])

运行结果:

2. 切片

切片是指对操作的对象截取其中一部分的操作。字符串、列表、元组都支持切片操作。

切片的语法：[起始:结束:步长]

注意：选取的区间属于左闭右开型，即从"起始"位开始，到"结束"位的前一位结束（不包含结束位本身)。

我们以字符串为例讲解。

如果取出一部分，则可以在中括号[]中，使用:

name = 'abcdef'

print(name[0:3]) # 取 下标0~2 的字符

运行结果:

name = 'abcdef'

print(name[0:5]) # 取 下标为0~4 的字符

运行结果:

name = 'abcdef'

print(name[3:5]) # 取 下标为3、4 的字符

运行结果:

name = 'abcdef'

print(name[2:]) # 取 下标为2开始到最后的字符

运行结果:

name = 'abcdef'

print(name[1:-1]) # 取 下标为1开始 到 最后第2个 之间的字符

运行结果:

>>> a = "abcdef"

>>> a[:3]

'abc'

>>> a[::2]

'ace'

>>> a[5:1:2]

''

>>> a[1:5:2]

'bd'

>>> a[::-2]

'fdb'

>>> a[5:1:-2]

'fd'

想一想

（面试题）给定一个字符串aStr, 请反转字符串

3.5. 字符串常见操作

3.6. 列表介绍

3.7. 列表的循环遍历

3.8. 列表的常见操作

3.9. 列表的嵌套

3.10. 元组

3.11. 字典介绍

3.12. 字典的常见操作1

3.13. 字典的常见操作2

3.14. 字典的遍历

3.15. 公共方法

3.16. 引用

3.17. 作业

本書使用 GitBook 釋出