# 概述

1、Python三大优点：简单、功能强大、支持面向对象。

2、Python的一些特点：

1）大小写严格区分

2）简单、易学、支持面向对象

3）开源

4）库非常丰富

5）跨平台使用

6）解释性语言

7）高级语言

# 安装

Python的安装主要在于环境变量的设置，主要有如下的几种环境变量：

PYTHONPATH：python探索路径，默认我们import的模块都会从这里查找

PYTHONSTARTUP：Python启动后，先寻找该环境变量，然后执行此变量指定的文件中的代码

PYTHONCASEOK：加入该环境变量，就会使python导入模块的时候不区分大小写

PYTHONHOME：另一种探索路径，通常内嵌于PYTHONSATRTUP或者PYTHONPATH目录中，使得两个模块更加容易切换

# 版本

Python2与Python3的主要区别：

# 语法

## 数据类型

### 基本数据类型

Python定义了一些标准类型，用于存储各种类型的数据，有5个标准的数据类型：

Numbers（数字）

String（字符串）

List（列表）

Tuple（元组）

Dictionary（字典）

### 类型转换

有时候，我们需要对内置的数据类型进行转换，数据类型的转换，你只需要将数据类型作为函数名即可。具体操作函数如下：

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转换为一个浮点数

complex(real[,img]) 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

set(s) 将序列s转换为集合

dict(d) 创建一个字段，d必须是一个序列(key,value)元组

frozenset(s) 转换为不可变集合

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) 将一个字符串转换为它的整数值

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

## 常量/变量

Python中的变量赋值不需要类型声明。

每个变量在内存中创建，都包括变量的标识，名称和数据这些信息。每个变量都在使用前必须赋值，变量赋值以后该变量才会被创建。

等号=用来给变量赋值，等号运算符左边是一个变量名，右边是存储在变量中的值。

### 常量

### 变量

#### 变量命名

在python声明变量的时候变量和值之间可以有空格，但是shell中不能有空格！

#### 变量赋值

#### 使用变量

使用变量的时候直接使用变量名，而shell中使用变量名则需要加$。

## 数字

### 定义

Python Number数据类型用于存储数值，数据类型是不允许改变的，这就意味着如果改变Number数据类型的值，将重新分配内存空间。

以下实例在变量赋值时Number对象被创建：

var1 = 1

var2 = 10

### 删除

可以使用del语句删除一些Number对象引用，语法为：

del var1[,var2[,var3[……,varN]]]

### 类型

Python支持四种不同的数值类型：

整型（int）：通常被称为整型或整数，是整数或者负数，不带小数点

长整型（long intergers）：无限大小的整数，整数最后是一个大写或小写的L

浮点数（floating point real value）：浮点型由整数部分和小数部分组成，也可以使用科学计数法表示（2.5e2=2.5\*102=250）

复数（complex number）：复数由实数部分和虚数部分组成，可以用a+bj或者complex(a,b)表示，复数的实部和虚部都是浮点数

### math/cmath

python中数学运算常用的函数基本都在math模块、cmath模块汇总。

Python math模块提供了许多对浮点数的数学运算函数，cmath模块包含了一些用于复数运算的函数。cmath模块的函数跟math模块函数基本一致，区别是cmath模块运算的是复数，math模块运算的是数学运算。

### 数学函数

abs(x)：返回数字的绝对值

ceil(x)：返回数字的上入整数，比如math.ceil(4.1)返回5

cmp(x,y)：如果x<y返回-1，x=y返回0，x>y返回1

exp(x)：返回e的x次幂

fabs(x)：返回数字的绝对值

floor(x)：返回数字的下舍整数，比如math.cloor(4.9)返回4

log(x)：log运算

log10(x)：返回以10为基数的x的对数

max(x1,x2)：返回给定参数的最大值，参数可以为序列

min(x1,x2)：返回给定参数的最小值，参数可以为序列

mod(fx)：返回x的整数部分与小数部分，两部分的数值符号与x相同，整数部分以浮点型表示

pow(x,y)：x\*\*y运算厚的值

round(x[,n])：返回浮点数x的四舍五入值

sqrt(x)：返回数字x的平方根

### 随机数函数

随机数可以用于数学，游戏，安全等领域中，还经常被嵌入到算法中，用以提高算法效率，并提高程序的安全性。

Python包含以下常用随机函数：

choice(seq)：从序列的元素中随机选取一个元素

randrange([start,]stop[,step])：从指定范围内，按指定基数递增的集合中获取一个随机数，基数缺省值为1

random()：随机生成一个实数，在[0,1)范围内

sedd([x])：改变随机数生成器的种子seed

### 三角函数

### 数学常量

pi 数学常量pi（圆周率，一般用π表示）

e 数学常量e（即自然常数）

## 字符串

字符串是python中最常用的数据类型，我们可以使用引号（单引号或双引号）来创建字符串（shell脚本中单引号、双引号有区别，单引号内部原样输出，双引号内部可以执行脚本）。

Python不支持单字符类型，单字符在python中也是作为一个字符串使用的。

Python访问字符串，可以使用方括号来读取字符串。

### 转移字符

### 字符串运算符

### 字符串格式化

### 三引号

### 内建函数

## 列表

### 定义

### 访问列表中的值

### 更新列表

### 删除列表元素

### 列表脚本操作符

### 列表截取

### 列表函数&方法

## 元组

### 定义

### 创建空元组

### 访问元组

### 修改元组

### 删除元组

### 元组运算符

### 无关闭分隔符

### 元组内置函数

## 字典

### 定义

### 访问字典

### 修改字典

### 删除字典元素

## 标识符

## 关键字

## 对象

## 行与缩进

# 运算符/表达式

## 运算符

## 优先级

## 表达式

# 流程控制

在Python中通常的情况下程序的执行是从上往下执行的，而某些时候我们为了改变程序的执行顺序，故而使用控制流语句控制程序怎么执行，而在Python找那个，有三种控制流类型，一种是顺序结构，就是指按顺序执行的结构，第二种是分支结构，第三种是循环结构。

Python的三种控制流：

分支结构if

循环结构while

循环结构for

break语句

continue语句

## 分支结构if

## 循环结构while

## 循环结构for

## break语句

## continue语句

# 函数与函数式编程

## 函数

函数是用来封装特定功能的，比如python里面，len()是一个函数，len()这个函数实现的功能是返回一个字符串的长度，所以说len()这个函数的特定功能就是返回长度，再比如，我们可以自定义一个函数，然后编写这个函数的功能，之后要使用的时候再调用这个函数。

函数分为两种类型，一种是系统自带的不用我们编写其功能系统自己就有的（BIF=build in function，可以通过命令dir(\_\_builtins\_\_)查看，进而通过help(int)查看帮助），比如len()这种函数，另一种函数是我们自定义的，需要我们编写其功能的，这种函数自由度高，叫做自定义函数，需要使用的时候直接调用该函数。

在python中要想使用自定义函数，就得首先定义一个函数，定义一个函数包括两个部分的含义，第一个含义是申明这个指定的部分是函数，而不是其他的对象，第二个含义是要定义这个函数所包含的功能，也就是要编写这个函数的功能。

## 形参/实参

形参一般发生在函数定义的过程中，形参一般是指参数的名称，而不代表函数的值，他仅仅只是形式上的函数，仅仅只标明一个函数里面，哪个位置有哪个名称的参数而已。

实参与形参刚好互相弥补，实参一般是在函数调用时出现，实参一般指的是参数具体的值。

在python中一个函数中出现多个参数的时候，我们可以通过参数的名称直接给我们的参数赋值，那么这些参数称之为关键参数。

## 全局变量/局部变量

Python中一个变量的是在一定的范围内起作用的，在其起作用的这个范围我们称之为作用域。

在Python中作用域在一定范围内而非全局都起作用的变量，我们将其称之为局部变量。在一个函数中，我们的变量如果没有进行全局变量申明，它默认就是一个局部变量。

在Python中如果我们想让某些变量的作用域为全局，也就是作用在程序的全部的地方，那么我们就要对这个变量进行全局声明，声明后这个变量就成了全局变量（关键字global）。

## 函数返回值

在Python中有的函数是有返回值的，有的函数是没有返回值的。而有返回值的函数，我们让函数可以返回一个值，也可以让函数返回多个值。

## 闭包

## Lambda表达式

使用Lambda表达式好处：

1、Python写一些执行脚本的时候，使用lambda表达式可以省下定义函数过程，比如说我们只是需要写个简单的脚本来管理服务器时间，我们就不需要专门定义一个函数然后再写调用，使用Lambda就可以使得代码更加精简。

2、对于一些比较抽象并且整个程序执行下来只需要调用一两次的函数，有时候给函数起个名字也是比较头疼的问题，使用Lambda就不需要考虑命名的问题了。

3、简化代码的可读性，由于普通的函数阅读起来经常要跳到开头def定义部分，使用Lambda函数可以省去这样的步骤。

# 模块/包/分发

## 模块

模块是对函数功能的扩展（函数是可以实现一项或多项功能的一段程序），模块是可以实现一项或多项功能的程序块。从定义可以看出，函数是一段程序，模块是一项程序块，也就是说函数和模块都是用来实现功能的，但是模块的范围比函数要广，在模块里面，可以重用多个函数。

Python中在一个程序中要使用某个模块，不能直接使用模块里面的功能。在使用模块之前，必须先导入指定模块，只要导入了一个模块，才能使用一个模块。

## 字节编译

### .pyc文件

我们所说的.pyc文件是指以.pyc为后缀名的这一类文件。由于计算机只认识二进制的机器语言，所以在执行Python模块的时候，有两种执行方式：一种是先将模块里面的内容编译成二进制语言，然后执行这些二进制语言，另一种是直接执行对应模块的二进制语言程序。第二种方式省略了编译这一步，所以执行速度相对来说要快一些。而把模块编程成二进制语言程序的这个过程叫做字节编译，这个过程会产生一个与编译的模块对应的.pyc文件。.pyc文件就是经过编译后的模块对应的二进制文件。

### 字节编译与编译

把模块编译成二进制语言程序的这个过程叫做字节编译。有人认为，python中出现字节编译，那么python应该是一种编译型语言。但是，这种观点是不正确的，python中虽然出现了编译这个过程，但是python中编译的过程是在python解释器中发生的。其实每种程序要想让计算机执行，都只能变成二进制的形式，编译型语言是指在软件中就有一个独立的编译模块去将程序编译，而python中字节编译这部分功能是由解释器完成的，所以python仍然是解释型语言。

### .pyc文件产生

.pyc两种产生方式：

### .pyc文件的使用

在python中，.pyc文件最大的一个作用就是加快了模块的运行速度。.pyc还可以做反编译等高级功能。

## from…import

python中导入一个模块的方法是使用import，但是import只是导入了这个模块，而并没有导入我们模块的某个属性或方法。

我们要不仅导入一个模块，还有导入模块中对应的一个功能，我们可以使用from…import语句。在python中使用from…import语句只能一次导入一个模块的一个功能，我们如果要想一次性把这个模块的所有功能，也就是所有属性与方法都导入的话，可以使用from…import\*语句。

## \_\_name\_\_属性

在python函数中，如果一个函数调用其他函数完成一项功能，我们称这个函数为主函数，如果一个函数没有调用其他函数，这种函数叫做非主函数。模块与之类似，如果一个模块是被直接使用的，而没有被别人调用，我们称这个模块为主模块，如果一个模块被别人调用，我们这种模块叫做非主模块。

如何区分主模块与非主模块？如果一个模块的\_\_name\_\_属性的值是\_\_main\_\_，那么就说明这个模块是主模块，反之亦然。其实我们可以把\_\_name\_\_看作一个变量，这个变量是系统给出的，这个变量的功能是判断一个模块是否是主模块。

## 自定义模块

Python中模块有非常多，有的模块是不需要用户自己去定义的去编写的，Python在安装的时候就自带的模块，这些模块是系统自带模块。而有一些模块刚好跟这种模块不同，他是需要我们自己去定义，自己去编写的模块，这些模块我们把其称之为自定义模块。

# 数据结构/算法

## 栈

栈是一种数据结构，这种数据结构在Python中不是内置数据结构，属于拓展数据结构。

这种数据结构的特点：首先，栈相当于一端开口一端封闭的容器，数据A可以存储在栈里面，把数据A移动到栈里面这个过程叫做进栈，也叫做压栈、入栈，数据A进入到栈里面之后，就到了栈顶，同时占了栈的一个位置。当再进入一个数据B的时候，也就是再将一个数据入栈的时候，这个时候，新的数据就占据了栈顶的位置，原来的数据就被新的数据压入到了栈顶的下一个位置里。栈只能对其栈顶的数据进行操作，所以这个时候原来的数据就不能被操作，此时只能对新数据进行操作，可以将其出栈或删除等。等数据B出栈后，方可对A进行操作。

## 队列

队列是一种数据结构，是拓展的数据结构。

其特点：首先，队列相当于两端都开的容器，但是一端只能进行删除操作，不能进行插入操作，而另一端只能进行插入操作而不能进行删除操作，进行插入操作的这端叫做队尾，进行删除操作的这端叫做队首。数据是从队尾进队首出的。

## 链表

链表是一种非连续，非顺序的存储方式。链表由一系列节点组成，每个节点包括两部分，一部分是数据域，另一部分是指向下一节点的指针域。链表可以分为单向链表，单向循环链表，双向链表，双向循环链表。

## bitmap

bitmap是一种数据结构，bit指的是位，map指的是图，bitmap也叫做位图。这种数据结构的存储简单来说就是把原来的数，转化为二进制来存储，每个位占一个存储单元。我们操作bitmap中的数据，也就是相当于操作一个位。Bitmap数据结构的优点是可以实现很好的排序。

## 树

树是一种非线性的数据结构，树具有非常高的层次性。利用树来存储数据，能够使用公有元素进行存储，能够很大的程度上节约存储空间。

树的定义是首先有且只有一个根节点，其次他有N个不相交子集，每个子集为一颗子树。

二叉树是一种特殊的树，二叉树要么为空树，要么为左、右两个不相交的子树组成。二叉树是有序树，即使只有一个子树，也需要区分该子树是左子树还是右子树。二叉树每个节点的度不能大于2，可以取0，1，2。二叉树的存储方式有两种，一种是顺序存储，一种是链式存储。顺序存储中采用一维数组的存储方式，链式存储中，采用链表的存储方式，通常分为三部分，数据域，左孩子链域和右孩子链域。

## 图

图是一种数据结构，图可以简单的理解为是一个关系网络，该网络中有N多结点，每个结点上存储着一个数据，数据之间的关联我们可以用线把关联的结点连起来的方式表示。其中，有的数据关系是有方向的，比如数据A->数据B，其关系只能从A到B，不能从B到A，如果数据之间的关系是有方向的，这个数据关系是用弧线表示。有的数据关系是没有方向的，A-B表示既可以A到B关联，也可以B到A关联，这种没有方向的关系用线段表示。

# 正则表达式

## 常用符号

. : 匹配任意字符，换行符\n除外

\* ：匹配前一个字符0次或无限次

? ：匹配前一个字符0次或1次

.\*：贪心算法

.\*?：非贪心算法

()：括号内的数据作为结果返回

## 常用方法

findall：匹配所有符合规律的内容，返回包含结果的列表

search：匹配并提取第一个符合规律的内容，返回一个正则表达式对象（object)

sub：替换符合规律的内容，返回替换后的值

## 使用

import re

from re import \*

from re import findall,search,sub,S

不需要complie

使用\d+匹配纯数字

# 输入/输出

# 执行环境

# 测试/调试

# 面向对象

## 类

## 方法

### 魔法方法\_\_init\_\_

## 继承

如果一个子类只继承一个父类，那么这叫做单继承。如果一个子类继承了两个或两个以上的父类，那么这叫做多继承。

假如一个子类A继承了多个父类B和C，并且此时B父类与C父类中有同名的方法，此时子类A到底应该选择父类B的方法使用还是应该选父类C的方法使用呢？这就叫做多继承方法冲突。在Python中，我们可以这样解决，我们可以看子类A先继承哪个父类，出现这种继承方法冲突时，我们选择先继承的父类中的方法去使用。也就是从左往右去看，左边的父类优先。

# 异常处理

## 异常

当Python程序出现某些异常状况的时候，我们将这种状态称为Python的异常。比如当我们没有正确缩进的时候，会出现异常错误，这种状态就是异常的一种情形。当发生的Python异常的时候，会把异常的类型和异常的位置打印出来，便于我们程序员去处理。

在编程的时候，直到某些语句可能会导致某种错误的发生，而现在我们想在编程的时候就把发生的这种错误解决掉，这个时候，我们可以用except语句对可能出错的部分进行处理。

## 异常引发

当某些程序出现异常的时候，执行时会自动将该异常类型和异常发生的位置打印出来。但是，这些错误的类型，是系统已经定义好的，比如系统规定，python必须严格缩进，那么系统就会自己定义一个异常类型，当用户不按这个规定来的时候，就会引发该异常。其实，除了系统自定义异常外，我们也是可以自己规定异常的，比如我们可以做一个字符串长度规定，当字符串长度不符合要求时就属于异常，然后再具体规定这种情况的时候的异常的名称，该异常怎么显示等等。那么，如果要自定义异常，就有一个关键的步骤：某种情况下才引发某种异常，比如字符串长度小3，引发自定义的A异常。这个过程就叫做异常的引发。

当某些程序出现某个异常的时候，执行时会自动将该异常类型和异常发生的位置打印出来。但是，这些错误的类型，是系统已经定义好的，比如系统规定，Python必须严格缩进，那么系统就会自己定义一个异常类型，当用户不按这个规定来的时候，就引发该异常。其实，除了系统自定义异常外，我们也是可以自己规定异常的，比如我们可以做一个字符串长度规定，当字符串长度不符合要求时就属于异常，然后再具体规定这种情况的时候的异常的名称，该异常怎么显示等等。那么，如要自定义异常，就有一个关键的步骤：某种情况下才引发某种异常，比如字符串长度小于3，引发自定义A异常。这个过程就叫做异常的引发。

在Python中，要想实现让某种情况下引发某种自定义的异常这个功能，可以使用raise语句实现。

## try…finally

当一段程序出现异常的时候，就不会继续执行下去了。但是有时，我们希望，不管某段程序发没发生异常，都得执行某些操作的时候，我们就可以使用try…finally语句实现。

# Python库

## 内置函数

## 内置异常

## 运行时服务

### atexit

### copy

### gc

### inspect

### marshal

### pickle

### SYS

### traceback

### types

### warnings

### weakref

## 数学运算

## 字符串/文本处理

### codecs

### re

### string

### struct

### unicodedata

## 数据库访问

## 文件目录处理

## 操作系统服务

### Commands

### ConfigParser

### datetime

### errno

### fcntl

### io

### logging

### mmap

### msvcrt

### optparse

### os

### signal

### subprocess

### time

### winreg

# 多线程编程

# 网络编程