Python简单教程

# 1. Python综述

## 1.1 python是什么

Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。

* Python的设计具有很强的可读性，相比其他语言经常使用英文关键字，其他语言的一些标点符号，它具有比其他语言更有特色语法结构。
* Python是一种解释型语言： 这意味着开发过程中没有了编译这个环节。类似于PHP和Perl语言。
* Python是交互式语言： 这意味着，我们可以在一个Python提示符后面直接互动执行写自己的程序。
* Python是面向对象语言: 这意味着Python支持面向对象的风格或代码封装在对象的编程技术。
* Python是初学者的语言：Python简单易学，对初级程序员而言，是一种伟大的语言，它支持广泛的应用程序开发，从简单的文字处理到 WWW 浏览器再到游戏。

## 1.2 python的发展

 Python 是由 Guido van Rossum（龟叔） 在八十年代末和九十年代初，在荷兰国家数学和计算机科学研究所设计出来的。

 Python 本身也是由诸多其他语言发展而来的,这包括 ABC、Modula-3、C、C++、Algol-68、SmallTalk、Unix shell 和其他的脚本语言等等。像 Perl 语言一样，Python 源代码同样遵循 GPL(GNU General Public License)协议。现在 Python 是由一个核心开发团队在维护，Guido van Rossum 仍然占据着至关重要的作用，指导其进展。

## 1.3 python的特点

* 易于学习：Python有相对较少的关键字，结构简单，和一个明确定义的语法，学习起来更加简单。
* 易于阅读：Python代码定义的更清晰。
* 易于维护：Python的成功在于它的源代码是相当容易维护的。
* 一个广泛的标准库：Python的最大的优势之一是丰富的库，跨平台的，在UNIX，Windows和Macintosh兼容很好。
* 互动模式：互动模式的支持，您可以从终端输入执行代码并获得结果的语言，互动的测试和调试代码片断。
* 可移植：基于其开放源代码的特性，Python已经被移植（也就是使其工作）到许多平台。
* 可扩展：如果你需要一段运行很快的关键代码，或者是想要编写一些不愿开放的算法，你可以使用C或C++完成那部分程序，然后从你的Python程序中调用。
* 数据库：Python提供所有主要的商业数据库的接口。
* GUI编程：Python支持GUI可以创建和移植到许多系统调用。
* 可嵌入: 你可以将Python嵌入到C/C++程序，让你的程序的用户获得"脚本化"的能力。

# Python3安装

## 2.1 python3 和 python2的区别

Python的3​​.0版本，常被称为Python 3000，或简称Py3k。相对于Python的早期版本，这是一个较大的升级。

为了不带入过多的累赘，Python 3.0在设计的时候没有考虑向下相容。

许多针对早期Python版本设计的程式都无法在Python 3.0上正常执行。

为了照顾现有程式，Python 2.6作为一个过渡版本，基本使用了Python 2.x的语法和库，同时考虑了向Python 3.0的迁移，允许使用部分Python 3.0的语法与函数。

新的Python程式建议使用Python 3.0版本的语法。

除非执行环境无法安装Python 3.0或者程式本身使用了不支援Python 3.0的第三方库。目前不支援Python 3.0的第三方库有Twisted, py2exe, PIL等。

Python 3.0 在 print函数、Unicode编码、除法运算、数据类型和异常等方面都与2.X版本有所变化。

## 2.2 python3环境的安装（以windows为例）

第一步，下载python环境安装包

官方地址：<https://www.python.org/>

第二步，安装python环境











第三步，测试是否安装成功

打开cmd命令终端，输入python，显示版本信息表示安装成功



## 2.3 Anaconda 安装（可选）

如果是在windows系统中安装，为了更简单地使用python中丰富的库资源，可以直接安装一个python“全家桶”——Anaconda。

Anaconda 是一个python的发行版，包括了python和很多常见的软件库, 和一个包管理器conda。常见的科学计算类的库都包含在里面，使得安装比常规python安装要容易。注意，装了Anaconda就不需要再装python了。

Anaconda不仅可以方便地安装、更新、卸载工具包，而且安装时能自动安装相应的依赖包，同时还能使用不同的虚拟环境隔离不同要求的项目；从而大大简化了工作流程。

下载地址：<https://www.anaconda.com/distribution/>

下载需要的对应版本，安装非常简单，只要跟着引导一步步做就可以了。

# 3. Python基本语法

## 3.1 编码

 默认情况下，python3源文件以UTF-8编码，所有字符串都是unicode字符串。同时可以指定源文件的不同编码

 文件开头加上

 # -\*- coding: UTF-8 -\*-

 # coding=utf-8(等号两边不能有空格)

 允许在源文件中使用utf-8字符集中的字符编码，对应的适合语言为中文等。

## 3.2 标识符

* 第一个字符必须是字母表中的字母或下划线\_
* 标识符中的其他部分由字母、数字和下划线组成
* 标识符对大小写敏感
* 在python3中，非ASCII标识符 (如中文字符) 也是允许的

## 3.3 注释

 单行注释：井号 #

 多行注释：三个单引号'''，或者三个双引号**"""**

## 3.4 关键子和保留字

 import keyword

 print(len(keyword.kwlist)) #33

 print(keyword.kwlist) #打印关键字

 ['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

 简介：

 1.and：表示逻辑‘与’

 2.del：用于list列表操作，删除一个或者连续几个元素。

 3.from：导入相应的模块，用import或者from...import

 4.not：表示逻辑‘非’

 5.while：while循环，允许重复执行一块语句，一般无限循环的情况下用它

 6.as：as单独没有意思，是这样使用：with....as用来代替传统的try...finally语法的。

 7.elif:和if配合使用的，if语句中的一个分支用elif表示。

 8.global :定义全局变量

 例如：

 name = 10

 age = 20

 def test():

 global name

 age = 30

 name = 'aa'

 #print(name)

 test()

 print(name) # 'aa'

 print(age) # 20

 9.or：表示逻辑“或”

 10.with：和as一起用，使用的方法请看as，见with

 11.assert：表示断言（断言一个条件就是真的，如果断言出错则抛出异常）用于声明某个条件为真，如果该条件不是真的，则抛出异常：AssertionError

 v1 = 10

 v2 = 20

 assert(v1 > v2)

 12.else：参考下面if的解释

 13.if：if语句用于选择分支，依据条件选择执行那个语句块。(if语句中最好不要嵌套if语句，建议把嵌套的if语句写在另一个函数中)

 14.pass：pass的意思就是什么都不做

 15.yield：用起来和return很像，但它返回的是一个生成器

 16.break：作用是终止循环，程序走到break的地方就是循环结束的时候。

 17.except：和try一起使用，用来捕获异常。

 18.import：用来导入模块，有时这样用from....import

 19.class：定义类

 20.in：查找列表中是否包含某个元素，或者字符串a是否包含字符串b。

 21.raise：raise可以显示地引发异常。一旦执行raise语句，后面的代码就不执行了

 22.continue：跳过continue后面循环块中的语句，继续进行下一轮循环。

 23.finally:看到finally语句，必然执行finally语句的代码块。

 24.is：Python中的对象包含三要素：id、type、value,用来判断对象是否相等

 25.return：用于跳出函数，也可以在跳出的同时返回一个值。

 26.def：用于定义方法

 27.for：for....in 一起使用：它在一序列的对象上递归，就是遍历队列中的每个项目

 28.lambda:即匿名函数

 29.try：出现在异常处理中，使用格式为：try...except，try中放想要执行的语句，except捕获异常

 30.nonlocal：nonlocal关键字用来在函数或其他作用域中使用外层(非全局)变量

 例如：

 def make\_counter():

 count = 0

 def counter():

 nonlocal count

 count += 1

 return count

 return counter

 def make\_counter\_test():

 mc = make\_counter()

 print(mc())

 print(mc())

 print(mc())

 make\_counter\_test()

## 3.5 行和缩进

 学习 Python 与其他语言最大的区别就是，Python的代码块不使用大括号{}来控制类，函数以及其他逻辑判断。python 最具特色的就是用缩进来写模块。

 缩进的空白数量是可变的，但是所有代码块语句必须包含相同的缩进空白数量，这个必须严格执行。

 例如：

 if True:

 print "True"

 else:

 print "False"

## 3.6 多行语句

 Python语句中一般以新行作为语句的结束符。但是我们可以使用斜杠（ \）将一行的语句分为多行显示，如下所示：

 total = item\_one + \

 item\_two + \

 item\_three

 print('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\

 aaaaaaaaaaaaaaaaaaaa')

 语句中包含 [], {} 或 () 括号就不需要使用多行连接符。如下实例：

 days = ['Monday', 'Tuesday', 'Wednesday',

 'Thursday', 'Friday']

## 3.7 python的引号

 Python 可以使用引号( ' )、双引号( " )、三引号( ''' 或 """ )表示字符串，引号的开始与结束必须的相同类型的。

 其中三引号可以由多行组成，编写多行文本的快捷语法，常用于文档字符串，在文件的特定地点，被当做注释。

 word = 'word'

 sentence = "这是一个句子。"

 paragraph = """这是一个段落。

 包含了多个语句"""

## 3.8 Python空行

 函数之间或类的方法之间用空行分隔，表示一段新的代码的开始。类和函数入口之间也用一行空行分隔，以突出函数入口的开始。空行与代码缩进不同，空行并不是Python语法的一部分。书写时不插入空行，Python解释器运行也不会出错。但是空行的作用在于分隔两段不同功能或含义的代码，便于日后代码的维护或重构。

 记住：空行也是程序代码的一部分。

## 3.9 等待用户输入

 执行下面的程序在按回车键后就会等待用户输入：

 input("\n\n按下 enter 键后退出。")

## 3.10 同一行显示多条语句

 Python可以在同一行中使用多条语句，语句之间使用分号(;)分割，以下是一个简单的实例：

 x = 'runoob';print(x + '\n')

## 3.11 多个语句构成代码组

 缩进相同的一组语句构成一个代码块，我们称之代码组。像if、while、def和class这样的复合语句，首行以关键字开始，以冒号( : )结束，该行之后的一行或多行代码构成代码组。

 我们将首行及后面的代码组称为一个子句(clause)。

 例如：

 if expression :

 suite

 elif expression :

 suite

 else :

 suite

## 3.12 Print 输出

 同类型才可以相加

 print 默认输出是换行的，如果要实现不换行需要在变量末尾加上 end=""：

 x="a"

 y="b"

 # 换行输出

 print( x )

 print( y )

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

 # 不换行输出

 print( x, end=" " )

 print( y, end=" " )

 print()

## 3.13 import 与 from...import

 在 python 用 import 或者 from...import 来导入相应的模块。

 a、将整个模块(somemodule)导入，格式为： import somemodule

 b、从某个模块中导入某个函数,格式为： from somemodule import somefunction

 c、从某个模块中导入多个函数,格式为： from somemodule import firstfunc, secondfunc, thirdfunc

 d、将某个模块中的全部函数导入，格式为： from somemodule import \*

1. 导入 sys 模块

 import sys

 print('================Python import mode==================');

 print ('命令行参数为:')

 for i in sys.argv:

 print (i)

 print ('\n python 路径为',sys.path)

1. 导入 sys 模块的 argv,path 成员

 from sys import argv,path # 导入特定的成员

 print('================python from import=====================')

 print('path:',path) # 因为已经导入path成员，所以引用不需要加sys.path

## 3.14 命令行参数

 很多程序可以执行一些操作来查看一些基本信息，Python可以使用-h参数查看各参数帮助信息：

 $ python -h

usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...

 Options and arguments (and corresponding environment variables):

 -c cmd : program passed in as string (terminates option list)

 -d : debug output from parser (also PYTHONDEBUG=x)

 -E : ignore environment variables (such as PYTHONPATH)

 -h : print this help message and exit

# 4. 基本数据类型

## 4.1 变量赋值

 counter = 100 # 整型变量

 miles = 1000.0 # 浮点型变量

 name = "runoob" # 字符串

 print (counter)

 print (miles)

 print (name)

## 4.2 多变量赋值

 Python允许你同时为多个变量赋值。例如：

 a = b = c = 1

 以上实例，创建一个整型对象，值为 1，从后向前赋值，三个变量被赋予相同的数值。

 可以为多个对象指定多个变量。例如：

 a, b, c = 1, 2, "runoob"

 a,b = b,a # 变量的交换

 以上实例，两个整型对象 1 和 2 的分配给变量 a 和 b，字符串对象 "runoob" 分配给变量 c。

 注意：

 a = 10

 b = 20

 a,b = b,a+5

 print(a,b)

 结果：a = 20,b=15

## 4.3 标准数据类型

 Number（数字）、String（字符串）、List（列表）、Tuple（元组）、Set（集合）、Dictionary（字典）

 a、不可变数据（3 个）：Number（数字）、String（字符串）、Tuple（元组）

 b、可变数据（3 个）：List（列表）、Dictionary（字典）、Set（集合）

## 4.4 Number

 int、float、bool、complex(复数)

 例如：

 a,b,c,d = 20,5.5,True,5+4j

 print(type(a),type(b),type(c),type(d))

### 4.1.1 函数

 type(a) # 判断数据类型

 isinstance(a,int) # 判断数据是否属于某类型

 del var1,var2 # 手动GC

区别:

 (1) type()不会认为子类是一种父类类型

 (2) isinstance()会认为子类是一种父类类型

 例如：

 class A:

 pass

 class B(A):

 pass

 print(type(A()) == A)

 print(type(B()) == A)

 print(isinstance(B(),A))

 print(isinstance(B(),A))

### 4.1.2 进制

 二进制：使用0b开头 例如：0b1010

 八进制：使用0o开头 例如：0o555

 十六进制：0x开头 例如：0x52A74（大小写都OK）

python中没有数字的大小限制，可以使用任意大的数字；python可以保证整数运算的精确，但是浮点数运算时可能会得到一个不精确的结果。

### 4.1.3 数学函数

 import math

 (1) 基本数学函数

 **函数 返回值 ( 描述 )**

 abs(x) 返回数字的绝对值，如abs(-10) 返回 10

 ceil(x) 返回数字的上入整数，如math.ceil(4.1) 返回 5

 (x>y)-(x<y) 如果 x < y 返回 -1, 如果 x == y 返回 0,

如果 x > y 返回 1

 exp(x) 返回e的x次幂(ex),

如math.exp(1) 返回2.718281828459045

 fabs(x) 返回数字的绝对值，如math.fabs(-10) 返回10.0

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

 log(x) 如math.log(math.e)返回1.0,math.log(100,10)返回2.0

 log10(x) 返回以10为基数的x的对数，

如math.log10(100)返回 2.0

 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。

 modf(x) 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 运算后的值。

 round(x [,n]) 返回浮点数x的四舍五入值，如给出n值，

则代表舍入到小数点后的位数。

 sqrt(x) 返回数字x的平方根。

 (2) 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。

 **函数 描述**

 choice(seq) 从序列的元素中随机挑选一个元素，

比如random.choice(range(10))，

从0到9中随机挑选一个整数。

 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() 随机生成下一个实数，它在[0,1)范围内。

 seed([x]) 改变随机数生成器的种子seed。

如果你不了解其原理，你不必特别去设定seed，

Python会帮你选择seed。

 shuffle(lst) 将序列的所有元素随机排序

 uniform(x, y) 随机生成下一个实数，它在[x,y]范围内。

 (3) 三角函数

 **函数 描述**

acos(x) 返回x的反余弦弧度值。

 asin(x) 返回x的反正弦弧度值。

 atan(x) 返回x的反正切弧度值。

 atan2(y, x) 返回给定的 X 及 Y 坐标值的反正切值。

 cos(x) 返回x的弧度的余弦值。

 hypot(x, y) 返回欧几里德范数 sqrt(x\*x + y\*y)。

 sin(x) 返回的x弧度的正弦值。

 tan(x) 返回x弧度的正切值。

 degrees(x) 将弧度转换为角度,如degrees(math.pi/2) ， 返回90.0

 radians(x) 将角度转换为弧度

 (4) 数学常量

 **常量 描述**

 pi 数学常量 pi（圆周率，一般以π来表示）

 e 数学常量 e，e即自然常数（自然常数）。

## 4.5 String

Python中的字符串用单引号 ' 或双引号 " 括起来，同时使用反斜杠 \ 转义特殊字符。下标从0开始。

 加号 + 是字符串的连接符， 星号 \* 表示复制当前字符串，紧跟的数字为复制的次数。

1. 字符串截取

 变量[头下标:尾下标:步长)

 -6 -5 -4 -3 -2 -1

 0 1 2 3 4 5

 +---+---+---+---+---+---+

 | a | b | c | d | e | f |

 +---+---+---+---+---+---+

 例如：

 str = 'hello world'

 l = str[0:1:1]

 print(l)

1. 字符串打印

 Python 使用反斜杠(\)转义特殊字符，如果你不想让反斜杠发生转义，可以在字符串前面添加一个 r，表示原始字符串：

 print('Ru\noob')

 Ru

 oob

 print(r'Ru\noob')

 Ru\noob

1. 字符串获取

 print(str[0])

 注意：

 1、反斜杠可以用来转义，使用r可以让反斜杠不发生转义。

 2、字符串可以用+运算符连接在一起，用\*运算符重复。

 3、Python中的字符串有两种索引方式，从左往右以0开始，从右往左以-1开始。

 4、Python中的字符串不能改变。

 str = 'abcdef'

 str[0] = 's' # 报错

 5、Python 没有单独的字符类型，一个字符就是长度为1的字符串。

## 4.6 List

 List（列表）是Python 中使用最频繁的数据类型。列表可以完成大多数集合类的数据结构实现。列表中元素的类型可以不相同，它支持数字，字符串甚至可以包含列表(所谓嵌套)。列表是写在方括号[]之间、用逗号分隔开的元素列表。和字符串一样，列表同样可以被索引和截取，列表被截取后返回一个包含所需元素的新列表。

 定义：

 list = [0,1,2,3,'c']

 list[0] # 0

 len(list) # 长度5

 list[0:3] # [0,1,2]

 注意：

 1、List写在方括号之间，元素用逗号隔开。

 2、和字符串一样，list可以被索引和切片。

 3、List可以使用+操作符进行拼接。

 4、List中的元素是可以改变的。

 5、不支持与或非运算

## 4.7 Tuple

 元组（tuple）与列表类似，不同之处在于元组的元素不能修改。元组写在小括号 () 里，元素之间用逗号隔开。

 定义：

 t = (0,1,2,3,'c')

 t[0] # 0

 len(list) # 长度5

 t[0:3] # (0,1,2)

 t[0] = 2 # 报错

 注意：

 1、与字符串一样，元组的元素不能修改。虽然tuple的元素不可改变，但它可以包含可变的对象，比如list列表。

 list = [0,2,1,3]

 t = (0,1,2,3,list)

 t[0] = 1 # 报错

 t[4][0] = 1

 print(t[4]) # ok

 2、元组也可以被索引和切片，方法一样。

 3、注意构造包含0或1个元素的元组的特殊语法规则。

 tup1 = () # 空元组

 tup2 = (20,) # 一个元素，需要在元素后添加逗号

 4、元组也可以使用 + 或 \* 操作符进行拼接。

## 4.8 Set

 集合（set）是由一个。合的事物或对象称作元素或是成员。基本功能是进行成员关系测试和删除重复元素。

 可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。

 创建格式：

 parame = {value01,value02,...} 或者 set(value)

 s = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}

 print(s) # {'Mary', 'Jack', 'Rose', 'Tom', 'Jim'}

 s = set('1b1b2b3b2b') # {'2', '3', '1', 'b'}

 set操作：

 # set可以进行集合运算

 a = set('abracadabra')

 b = set('alacazam')

 print(a)

 >>> {'b', 'a', 'c', 'r', 'd'}

 print(a - b) # a和b的差集

 >>> {'b', 'd', 'r'}

 print(a | b) # a和b的并集

 >>> {'l', 'r', 'a', 'c', 'z', 'm', 'b', 'd'}

 print(a & b) # a和b的交集

 >>> {'a', 'c'}

 print(a ^ b) # a和b中不同时存在的元素

 >>> {'l', 'r', 'z', 'm', 'b', 'd'}

## 4.9 Dictionary（字典）

 字典（dictionary）是Python中另一个非常有用的内置数据类型。

 列表是有序的对象集合，字典是无序的对象集合。两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。

 字典是一种映射类型，字典用"{ }"标识，它是一个无序的键(key) : 值(value)对集合。

 键(key)必须使用不可变类型。

 在同一个字典中，键(key)必须是唯一的。

 字典创建：

 (1) d = {"a":1,"b":2,"c":3}

 (2) d = dict([('Runoob', 1), ('Google', 2), ('Taobao', 3)])

 (3) d = {x:x\*\*2 for x in (2, 4, 6)}

 (4) d = dict(Runoob=1, Google=2, Taobao=3)

 字典操作：

 tinydict = {'name': 'guigu','code':1, 'site': 'www.atguigu.com'}

 print (dict['name']) # 输出键为 'name' 的值

 print (dict['code']) # 输出键为 'code' 的值

 print (tinydict) # 输出完整的字典

 print (tinydict.keys()) # 输出所有键 dict\_keys(['name', 'code', 'site'])

 print (tinydict.values()) # 输出所有值 dict\_values(['guigu', 1, 'www.atguigu.com'])

## 4.10 python类型转换

 函数 描述

 int(x [,base]) 将x转换为一个整数

 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

 set(s) 转换为可变集合

 dict(d) 创建一个字典，

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（ASCII码）

 ord(x) 将一个字符转换为它的ASCII码值

 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

# 5. python解释器

 Linux/Unix的系统上，一般默认的 python 版本为 2.x，我们可以将 python3.x 安装在 /usr/local/python3 目录中。

 安装完成后，我们可以将路径 /usr/local/python3/bin 添加到您的 Linux/Unix 操作系统的环境变量中，这样您就可以通过 shell 终端输入下面的命令来启动 Python3 。

 $ PATH=$PATH:/usr/local/python3/bin/python3 # 设置环境变量

 $ python3 --version

 Python 3.4.0

## 5.1 环境变量设置

在Window系统下你可以通过以下命令来设置Python的环境变量，假设你的Python安装在 C:\Python34 下:

 set path=%path%;C:\python34

## 5.2 交互式编程

 我们可以在命令提示符中输入"python"（或者"python3"，具体视安装时的命令名称而定）命令来启动Python解释器：

 $ python3

1. 执行以上命令后，出现如下窗口信息：

 $ python3

 Python 3.4.0 (default, Apr 11 2014, 13:05:11)

 [GCC 4.8.2] on linux

 Type "help", "copyright", "credits" or "license" for more information.

 >>>

1. 在 python 提示符中输入以下语句，然后按回车键查看运行效果：

 print ("Hello, Python!");

1. 以上命令执行结果如下：

 Hello, Python!

1. 当键入一个多行结构时，续行是必须的。我们可以看下如下 if 语句：

 >>> flag = True

 >>> if flag :

 ... print("flag 条件为 True!")

 ...

 flag 条件为 True!

## 5.3 脚本式编程

 (1) 将如下代码拷贝至 hello.py文件中：

 print ("Hello, Python!");

1. 通过以下命令执行该脚本：

 python3 hello.py

1. 输出结果为：

 Hello, Python!

1. 在Linux/Unix系统中，你可以在脚本顶部添加以下命令让Python脚本可以像SHELL脚本一样可直接执行：

 #! /usr/bin/env python3

1. 然后修改脚本权限，使其有执行权限，命令如下：

 $ chmod +x hello.py

1. 执行以下命令：

 ./hello.py

1. 输出结果为：

 Hello, Python!

# 6. 运算符

## 6.1 算术运算符

 a = 10 b = 21

 运算符 描述 实例

 + 加 两个对象相加；

a + b 输出结果 31

 - 减 得到负数或是一个数减去另一个数；

a - b 输出结果 -11

 \* 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 210

 / 除 x 除以 y；

b / a 输出结果 2.1

 % 取模 返回除法的余数；

b % a 输出结果 1

 \*\* 幂 返回x的y次幂 ；

a\*\*b 为10的21次方

 // 取整除 向下取接近除数的整数 ；

9//2 # 4

 -9//2 #-5

## 6.2 比较运算符

 a = 10 b = 20

 运算符 描述 实例

 == 等于 比较对象是否相等

(a == b) 返回 False

 != 不等于 比较两个对象是否不相等

(a != b) 返回 True

 > 大于 返回x是否大于y

(a > b) 返回 False

 < 小于 返回x是否小于y

(a < b) 返回 True

 >= 大于等于 返回x是否大于等于y

(a >= b) 返回 False

 <= 小于等于 返回x是否小于等于y

(a <= b) 返回 True

 注意：所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。

## 6.3 赋值运算符

 运算符 描述 实例

 = 简单的赋值运算符 c = a + b将a + b的运算结果赋值为 c += 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

## 6.4 位运算符

 a = 60, b = 13

 二进制形式：a = 0011 1100, b = 0000 1101

运算符 描述 实例

 & 按位与运算符 (a & b) 输出结果 12 ，

二进制解释： 0000 1100

 | 按位或运算符 (a | b) 输出结果 61 ，

二进制解释： 0011 1101

 ^ 按位异或运算符 (a ^ b) 输出结果 49 ，

二进制解释： 0011 0001

 ~ 按位取反运算符 (~a ) 输出结果 -61 ，

二进制解释： 1100 0011

 << 左移动运算符 a << 2 输出结果 240 ，

二进制解释： 1111 0000

 >> 右移动运算符 a >> 2 输出结果 15 ，

二进制解释：0000 1111

## 6.5 逻辑运算符(bool)

 a = 10, b = 20

 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x的值，

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

## 6.6 成员运算符

 除了以上的一些运算符之外，Python还支持成员运算符，测试实例中包含了一系列的成员，包括字符串，列表或元组。

 运算符 描述 实例

 in 如果在指定的序列中找到值返回 True，否则返回 False

x在y序列中,如果x在y序列中返回True

 not in 如果在指定的序列中没有找到值返回 True，否则返回 False

x不在y序列中,如果x不在y序列中返回True

## 6.7 身份运算符

 身份运算符用于比较两个对象的存储单元

 运算符 描述 实例

 is is是判断两个标识符是不是引用自一个对象

 x is y, 类似 id(x) == id(y) , 如果引用的是同一个对象则返回 True，否则返回 False

 is not is not是判断两个标识符是不是引用自不同对象

 x is not y ， 类似id(a)!=id(b)。如果引用的不是同一个对象则返回结果True，否则返回False。

## 6.8 Python运算符优先级

 以下表格列出了从最高到最低优先级的所有运算符：

 运算符 描述

 \*\* 指数 (最高优先级)

 ~ + - 按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@)

 \* / % // 乘，除，取模和取整除

 + - 加法减法

 >> << 右移，左移运算符

 & 位 'AND'

 ^ | 位运算符

 <= < > >= 比较运算符

 <> == != 等于运算符

 = %= /= //= -= += \*= \*\*= 赋值运算符

 is is not 身份运算符

 in not in 成员运算符

 and or not 逻辑运算符

# 7. 字符串（String）

## 7.1 字符串创建

 s = 'abcd'

 s = "abcd"

 s = """

 abcdefg

 """

## 7.2 字符串访问

 a = s[0] # 访问弟0个元素

 l = len(s) # 字符串的长度

## 7.3 字符串运算

a = “Hello”, b = “Python”

 操作符 描述 实例

 + 字符串连接 a + b 输出结果： HelloPython

 \* 重复输出字符串 a\*2 输出结果：HelloHello

 [] 通过索引获取字符串中字符 a[1] 输出结果 e

 [ : ] 截取字符串中的一部分，遵循左闭右开原则 str[0,2] 是不包含第 3 个字符的

 in 成员运算符 'H' in a 输出结果 True

 not in 成员运算符 'M' not in a 输出结果 True

 r/R 原始字符串 print(r'a\nb') # a\nb

 % 格式字符串 a=10;print("今年我%s岁"%a)

# 今年我10岁

## 7.4 Python字符串格式化

 Python 支持格式化字符串的输出 。尽管这样可能会用到非常复杂的表达式，但最基本的用法是将一个值插入到一个有字符串格式符 %s 的字符串中。

 print ("我叫%s今年%d岁!" %('小明', 10)) # 我叫小明今年10岁!

 符号 描述

 %c 格式化字符及其ASCII码

 %s 格式化字符串

 %d 格式化整数

 %u 格式化无符号整型

 %o 格式化无符号八进制数

 %x 格式化无符号十六进制数

 %X 格式化无符号十六进制数（大写）

 %f 格式化浮点数字，可指定小数点后的精度

 %e 用科学计数法格式化浮点数

 %E 作用同%e，用科学计数法格式化浮点数

 %g %f和%e的简写

 %G %f 和 %E 的简写

 %p 用十六进制数格式化变量的地址

## 7.5 内建函数

 方法 描述

 capitalize() 将字符串的第一个字符转换为大写

 endswith(suffix,beg=0,end=len(string))检查字符串是否以 obj 结束

 expandtabs(tabsize=8) 把字符串 string 中的 tab 符号转为空格，tab 符号默认的空格数是 8 。

 find(str, beg=0 end=len(string)) 检测 str 是否包含在字符串中,如果包含返回开始的索引值，否则返回-1

 index(str, beg=0, end=len(string)) 跟find()方法一样，只不过如果str不在字符串中会报一个异常.

 isalnum() 如果字符串至少有一个字符并且所有字符都是字母或数字则返 回 True,否则返回 False

 isdigit() 如果字符串只包含数字则返回 True 否则返回 False..

 isnumeric() 如果字符串中只包含数字字符，则返回 True，否则返回 False

 isspace() 如果字符串中只包含空白，则返回 True，否则返回 False.

 join(seq) 以指定字符串作为分隔符，将 seq 中所有的元素(的字符串表示)合并为一个新的字符串

 len(string) 返回字符串长度

 lower() 转换字符串中所有大写字符为小写.

 lstrip() 截掉字符串左边的空格或指定字符。

 max(str) 返回字符串 str 中最大的字母。

 min(str) 返回字符串 str 中最小的字母。

 replace(old, new [, max]) 把 将字符串中的 str1 替换成 str2,如果 max 指定，则替换不超过 max 次。

 rfind(str, beg=0,end=len(string)) 类似于 find()函数，不过是从右边开始查找.

 rindex( str, beg=0, end=len(string))类似于 index()，不过是从右边开始.

 rstrip() 删除字符串字符串末尾的空格

 split(str="",num=string.count(str)) num=string.count(str))以str为分隔符截取字符串，如果num有指定值，则仅截取 num个子字符串

 splitlines([keepends]) 按照行('\r', '\r\n', \n')分隔

 startswith(str,beg=0,end=len(string))检查字符串是否是以 obj 开头

 strip([chars]) 在字符串上执行 lstrip()和 rstrip()

 upper() 转换字符串中的小写字母为大写

# 8. 列表

## 8.1 列表创建

 list = [1,2,3,4,5,'atguigu']

 [x+1 for x in range(10)] 或 [x+1 for x in (1,1,2,3)]

## 8.2 列表值获取

 l = list[0]

 l1 = list2[1:5]

 len(list) #长度查看

## 8.3 列表更新

 list[0] = 'agg'

## 8.4 删除列表元素

 del list[0]

## 8.5 其它列表操作

 表达式 结果 描述

 len([1, 2, 3]) 3 长度

 [1, 2, 3] + [4, 5, 6] [1, 2, 3, 4, 5, 6] 组合

 ['Hi!']\*4 ['Hi!', 'Hi!', 'Hi!', 'Hi!'] 重复

 3 in [1, 2, 3] True 元素是否存在于列表中

 for x in [1, 2, 3]: print(x, end=" ") 1 2 3 迭代

## 8.6 列表嵌套

 使用嵌套列表即在列表里创建其它列表，例如：

 a = ['a', 'b', 'c']

 n = [1, 2, 3]

 x = [a, n]

 # x = [['a', 'b', 'c'], [1, 2, 3]]

 # x[0] = ['a', 'b', 'c']

 # x[0][1] = 'b'

## 8.7 Python列表函数&方法

 函数 描述

 len(list) 列表元素个数

 max(list) 返回列表元素最大值

 min(list) 返回列表元素最小值

 list(seq) 将元组转换为列表

 list.append(obj) 在列表末尾添加新的对象

 list.count(obj) 统计某个元素在列表中出现的次数

 list.extend(seq) 在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）

 list.index(obj) 从列表中找出某个值第一个匹配项的索引位置

 list.insert(index, obj) 将对象插入列表

 list.pop([index=-1]) 移除列表中的一个元素（默认最后一个元素），并且返回该元素的值

 list.remove(obj) 移除列表中某个值的第一个匹配项

 list.reverse() 反向列表中元素

 list.sort(cmp=None, key=None, reverse=False) 对原列表进行排序

 list.clear() 清空列表

 list.copy() 复制列表

# 9. 元组

## 9.1 元组创建

 tup1 = ('Google', 'atguigu', 1997, 2000);

 tup2 = (1, 2, 3, 4, 5 );

 tup3 = "a", "b", "c", "d"; # 不需要括号也可以

## 9.2 元组值获取

 tup1[1]

 tup1[1:5]

## 9.3 元组更新

 元组不允许更新

## 9.4 删除元组元素

 del tup[0]

## 9.5 元组运算符

 表达式 结果 描述

 len((1, 2, 3)) 3 计算元素个数

 (1, 2, 3) + (4, 5, 6) (1, 2, 3, 4, 5, 6) 连接

 ('Hi!',) \* 4 ('Hi!', 'Hi!', 'Hi!', 'Hi!') 复制

 3 in (1, 2, 3) True 元素是否存在

 for x in (1, 2, 3): print (x,) 1 2 3 迭代(生成器)

## 9.6元组内置函数

 方法 描述 实例

 len(tuple) 计算元组元素个数 len(tuple1)

 max(tuple) 返回元组中元素最大值 max(tuple2)

 min(tuple) 返回元组中元素最小值 min(tuple2)

 tuple(seq) 将列表转换为元组 list1= ['Google', 'Taobao', 'Runoob', 'Baidu']

 tuple1=tuple(list1)

 tuple1 = ('Google', 'Taobao', 'Runoob', 'Baidu')

# 10. 字典

 在字典中，键必须不可变，所以可以用数字，字符串或元组充当，而用列表就不行。

## 10.1 字典创建

 dict = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}

 dict = {x:x+1 for x in range(10)}

## 10.2 字典值获取

 dict['Alice']

## 10.3 更新字典

 dict['Alice'] = 10

## 10.4 删除字典元素

 del dict['Alice']

 del dict

## 10.5 字典内置函数&方法

 函数 描述 实例

 len(dict) 计算字典元素个数，即键的总数。 dict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}

 len(dict)

 str(dict) 输出字典，以可打印的字符串表示。 dict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}

 str(dict)

 "{'Name': 'Runoob', 'Class': 'First', 'Age': 7}"

 type(variable) 返回输入的变量类型 dict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}

 type(dict)

 <class 'dict'>

 radiansdict.clear() 删除字典内所有元素

 radiansdict.copy() 返回一个字典的浅复制

 radiansdict.fromkeys() 创建一个新字典，以序列seq中元素做字典的键，val为字典所有键对应的初始值

 radiansdict.get(key, default=None) 返回指定键的值，如果值不在字典中返回default值

 key in dict 如果键在字典dict里返回true，否则返回false

 radiansdict.items() 以列表返回可遍历的(键, 值) 元组数组

 radiansdict.keys() 返回一个迭代器，可以使用 list() 来转换为列表

 radiansdict.setdefault(key, default=None) 和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default

 radiansdict.update(dict2) 把字典dict2的键/值对更新到dict里

 radiansdict.values() 返回一个迭代器，可以使用 list() 来转换为列表

 pop(key[,default]) 删除字典给定键 key 所对应的值，返回值为被删除的值。key值必须给出。 否则，返回default值。

 popitem() 随机返回并删除字典中的一对键和值(一般删除末尾对)。

 扩展：

 1. 浅拷贝：

 只拷贝引用地址，未拷贝内容：

 a = [1,2,3,4,5]

 b = a

 2. 深拷贝：

 拷贝引用地址和内容：

 a = [1,2,3,4,5]

 import copy

 b = copy.deepcopy(a)

 可以递归拷贝；一拷到底

 注意：

 1、对于不可变类型 Number String Tuple,浅复制仅仅是地址指向，不会开辟新空间。

 2、对于可变类型 List、Dictionary、Set，浅复制会开辟新的空间地址(仅仅是最顶层开辟了新的空间，里层的元素地址还是一样的)，进行浅拷贝

 3、浅拷贝后，改变原始对象中为可变类型的元素的值，会同时影响拷贝对象的；改变原始对象中为不可变类型的元素的值，只有原始类型受影响。

# 11. Set集合

集合不支持切片操作。

## 11.1 Set集合创建

 s = {'name','aa','bb'}

 s = set(序列) # dict序列，值添加key

 s = {x for x in range(10) if x not in range(5,10)}

## 11.2 Set集合添加元素

 s.add(x) # 添加单个元素

 s.update(x) # 添加序列元素

## 11.3 移除元素

 s.remove(x) # 移除单个元素

 s.discard(x) # 移除集合(不存在不报错)

 s.pop() # 随机删除集合中的一个元素

## 11.4 集合操作方法

 方法 描述

 len(s) 查看集合的长度

 s.clear() 清空集合

 x in s 判断元素是否在集合中

 add() 为集合添加元素

 clear() 移除集合中的所有元素

 copy() 拷贝一个集合

 difference() 返回多个集合的差集

 difference\_update() 移除集合中的元素，该元素在指定的集合也存在。

 discard() 删除集合中指定的元素

 intersection() 返回集合的交集

 intersection\_update() 删除集合中的元素，该元素在指定的集合中不存在。

 isdisjoint() 判断两个集合是否包含相同的元素，如果没有返回 True，否则返回 False。

 issubset() 判断指定集合是否为该方法参数集合的子集。

 issuperset() 判断该方法的参数集合是否为指定集合的子集

 pop() 随机移除元素

 remove() 移除指定元素

 symmetric\_difference() 返回两个集合中不重复的元素集合。

 symmetric\_difference\_update() 移除当前集合中在另外一个指定集合相同的元素，并将另外一个指定集合中不同的元素插入到当前集合中。

 union() 返回两个集合的并集

 update() 给集合添加元素

# 12. 条件判断

## 12.1 基本语法

 if condition\_1:

 statement\_block\_1

 elif condition\_2:

 statement\_block\_2

 else:

 statement\_block\_3

## 12.2 注意要点

 1、每个条件后面要使用冒号 :，表示接下来是满足条件后要执行的语句块。

 2、使用缩进来划分语句块，相同缩进数的语句在一起组成一个语句块。

 3、在Python中没有switch – case语句。

# 13. 循环语句

## 13.1 while循环

 while bool：

 pass

 else:

 pass

 练习：1-100求和；输出9\*9乘法表

## 13.2 for循环

 for <variable> in <sequence>:

 <statements>

## 13.3 range()函数

 如果你需要遍历数字序列，可以使用内置range()函数。它会生成数列：

 range(start,end,step)

## 13.4 break和continue语句及循环中的else子句

(1) break 语句可以跳出 for 和 while 的循环体。

如果你从 for 或 while 循环中终止，任何对应的循环 else 块将不执行。

(2) continue语句被用来告诉Python跳过当前循环块中的剩余语句，然后继续进行下一轮循环。

## 13.5 pass 语句

 Python pass是空语句，是为了保持程序结构的完整性。pass 不做任何事情，一般用做占位语句。

# 14. 迭代器和生成器

 迭代是Python最强大的功能之一，是访问集合元素的一种方式。迭代器是一个可以记住遍历的位置的对象。迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退。

 迭代器有两个基本的方法：iter() 和 next()。

## 14.1 迭代器生成

 字符串，列表或元组对象都可用于创建迭代器

 list=[1,2,3,4]

 it = iter(list) # 创建迭代器对象

 print(next(it)) # 输出迭代器的下一个元素

 print(next(it))

## 14.2 迭代器遍历

 list=[1,2,3,4]

 it = iter(list)

1. for循环

 for i in it:

 print(i)

1. while循环

 import sys

 while True:

 try:

 print(next(its))

 except StopIteration:

 sys.exit()

## 14.3 创建一个迭代器

 把一个类作为一个迭代器使用需要在类中实现两个方法 \_\_iter\_\_() 与 \_\_next\_\_() 。

 例如：

 class MyNumbers:

 def \_\_iter\_\_(self):

 self.a = 1

 return self

 def \_\_next\_\_(self):

 if self.a < 20:

 x = self.a

 self.a += 1

 return x

 else:

 raise StopIteration

 myclass = MyNumbers()

 myiter = iter(myclass)

 print(next(myiter))

 for x in myiter:

 print(x)

## 14.4 生成器

 在Python中，使用了yield的函数被称为生成器(generator)。跟普通函数不同的是，生成器是一个返回迭代器的函数，只能用于迭代操作，更简单点理解生成器就是一个迭代器。

 在调用生成器运行的过程中，每次遇到 yield 时函数会暂停并保存当前所有的运行信息，返回yield的值, 并在下一次执行next()方法时从当前位置继续运行。

 调用一个生成器函数，返回的是一个迭代器对象。

 import sys

 def fibonacci(n): # 生成器函数 - 斐波那契

 a, b, counter = 0, 1, 0

 while True:

 if (counter > n):

 return

 yield a

 a, b = b, a + b

 counter += 1

 f = fibonacci(10) # f是一个迭代器，由生成器返回生成

 while True:

 try:

 print (next(f), end=" ")

 except StopIteration:

 sys.exit()

# 15. 函数

## 15.1 基本语法

 def 函数名(参数列表):

 函数体

## 15.2 函数分类

 1. 有参数

 (1) 有几个参数，就得传入几个参数

 (2) 在函数调用中输入参数时，参数名称必须对应

 def aa(x):

 print(x)

 aa(x=5)

 (3) 当调用函数时，必须全部定义名称,且名称对应，顺序可以不同

 def aa(x,y):

 print(x)

 aa(y=2,x=5)

 (4) 函数中可以定义默认值

 def aa(x=30,y):

 print(x)

 aa(y=2,x=5)

 (5) 不定长度参数

 def aa(x,y,\*args,\*\*kwargs):

 print(x)

 print(args) # 元组

 print(kwargs) # 字典

 aa(1,2,3,4,5,6,7,a = 8,b=9)

 2. 有返回值

 (1) 单个返回值

 def aa(x):

 return x

 a = aa(10)

 (2) 多个返回值

 def aa(x):

 return x,10

 a = aa(10) # a是一个元组

 a,b = aa(10) # 多个参数接收

## 15.3 匿名函数

 基本语法：

 lambda [arg1 [,arg2,.....argn]]:expression

 sum = lambda arg1, arg2: arg1 + arg2

 # 调用sum函数

 print ("相加后的值为 : ", sum( 10, 20 ))

 print ("相加后的值为 : ", sum( 20, 20 ))

## 15.4 变量作用域

 L （Local） 局部作用域

 E （Enclosing） 闭包函数外的函数中

 G （Global） 全局作用域

 B （Built-in） 内建作用域

 L –> E –> G –>B(在局部找不到，便会去局部外的局部找（例如闭包），再找不到就会去全局找，再者去内建中找)

 x = int(2.9) # 内建作用域

 g\_count = 0 # 全局作用域

 def outer():

 o\_count = 1 # 闭包函数外的函数中

 def inner():

 i\_count = 2 # 局部作用域

 o\_count += 1

 inner()

 outer()

## 15.5 全局变量和局部变量

 定义在函数内部的变量拥有一个局部作用域，定义在函数外的拥有全局作用域。

 局部变量只能在其被声明的函数内部访问，而全局变量可以在整个程序范围内访问。调用函数时，所有在函数内声明的变量名称都将被加入到作用域中。

 total = 0 # 这是一个全局变量

 # 可写函数说明

 def sum( arg1, arg2 ):

 #返回2个参数的和."

 total = arg1 + arg2 # total在这里是局部变量.

 print("函数内是局部变量 : ", total)

 return total

 #调用sum函数

 sum( 10, 20 )

 print ("函数外是全局变量 : ", total)