**python运行方式**

交互式解释器

进入交互式解释器

python3

退出交互式解释器

exit()货ctrl+d

文件形式

明确指定解释器

python3 hello.py

赋予python文件可执行权限

chmod +x hello.py

./hello.py

**python语法结构**

语句块缩进

python代码块通过缩进对齐表达式代码缩进而不是用大括号，缩进表达一个语句属于哪个代码块

缩进风格

1或2：可能不够，很难确定代码语句属于哪个代码块

8至10个：可能太多，如果代码内嵌的层次太多，就会使得代码很难读

4哥空格：非常流行。

缩进相同的一组语句构成一个代码块，称为代码组

首行以关键字开始，以冒号：结束，该行至后的一行或多行代码构成代码组，如果代码组只有一行，可以将其直接写在冒号后面，但是这样的可读性比较差，不推荐

注释和续行

注释以#开头，注释科一在一行的任何地方开始，解释器会忽略掉该行#至后的所有内容

一行过长的语句可以用反斜杠\分解成几行

同行多个语句

分号;允许将多个语句写在同一行上，但是这些语句不能在这行开始一个新的代码块，因为可读性会变差，所以不推荐使用

**python语法**

获取帮助

help(print)

输出语句print

print('hello world!') # 字符串必须放在引导中

print('hello', 'world', '!', 123) # 数字不用放在引号中

print('hello', 'world', '!', 123, sep='\*\*\*') # 各项间用\*\*\*分隔

print('hello' + 'world') # 字符串拼接

print('hello', 'world', '!', 123, end='') # 不打印回车

输入语句input

a = input('number:') # input读入的数据是字符类型

# a+5错误，不能把数字和字符相加

print(int(a) + 5) # 将字符转换成数字再加5

print(a + str(5)) # 将数字5转换成字符，再和字符串拼接

**python变量**

变量定义

变量名称约定

第一个字符只能是大小写字母或下划线

后续字符职能是大小写字母货数字或下划线

区分大小写

python是动态类型语言，即不需要预先声明变量的类型

推荐才用全名的办法

变量名全部采用小写字母

简短、有意义

多个单词间用下划线分割

变量名用名词，函数名用谓语（动词+名词）

类名采用驼峰形式

变量赋值

变量的类型喝值在赋值的那一刻被初始化

变量赋值通过等号来执行

python也支持增量赋值

a = 10 + 5 # 变量赋值，自右向左进行

a = a + 1 # 可以简化位以下形式

a += 10

# b+=10错误，因为等价于b=b+10,b没有赋值

运算符

标准算术运算符：+-\*/ // % \*\*

比较运算符：< <= > >= == !=

逻辑运算符：and not or

print(5 / 2) # 2.5

print(5 // 2) # 2

print(5 % 2) # 之遥余数，模运算，1

print(2 \*\* 3) # 2的3次方，8

20 > 10 > 5 # true相当于20>10and10>5

**python数据类型**

数字

基本数字类型

int：有符号整数

bool：布尔值（True：1，False：0）

float：浮点数

数字表示方法

python默认使用十进制数表示；数字以0o或者0O开头表示为8进制数；数字以0x或0X开头表示为16进制数，数字以0b或0B开头表示为2进制数

a=True+2

print(a)#bool类型也是整型

print(0o11)#8进制数11换算成10进制数为9

print(0x11)#16进制数11换算成10进制数是17

print(0b11)#二进制数11换算成10进制数是3

字符串

定义字符串

python种字符串被定义为引号之间的字符集合，可以使用成对的单引号货双引号，两个意义相同，还支持三引号（三个连续的引号或者双引号），可以用来包含特殊字符。不区分字符和字符串

py=’python’

字符串切片（类似与截取）

使用索引运算符[]和切片运算符[:]可得到字符串，第一个字符串的索引是0,最后一个字符的索引是-1，字符串包含切片中的起始下标，但不包含下标，切片没有下标不会报错

py[0]#取出第一个字符

py[-1]#取出最后一个字符

py[2:4]#切片，取出2到4个字符

py[2:]#结尾不写，表示取到最后部分结束

py[:]#取所有

py[::-1]#步长为负，表示从右往左取

字符串连接

使用+号可以将多个字符串拼接在一起，使用\*号可以将一个字符串重复多次

py+’is good’

‘\*’ \* 50#打印\*50次

列表

可以将列表当成普通的“数组”，它能保存任意数量任意类型的python对象，像字符串一样，列表也支持下标和切片操作，列表中的项目可以改变

列表操作

使用in或not in判断成员关系，使用append方法向列表中追加元素

alist=[‘bob’,’alice’,10,[1,2,3]]

[‘tom’]+alist

alist.append(100)#向列表中追加元素

元组

可以认为元组是“静态”的列表，元组一旦定义，不能改变

aaa=(‘bob’,10,28)

aaa[2:]

(100,)+aaa

字典

字典是由键值对构成的映射数据类型，通过键取值，不支持下标操作

adict={‘name’:’zhangsan’,’age’:25}

25 in adict#false

adict[‘name’]#’zhangsan’

数据类型比较

按存储模型分类

标量类型：数值、字符串

容器类型：列表、元组、字典

按更新模型分类

可变类型：列表、字典

不可变类型：数字、字符串、元组

按访问模型分类

直接访问：数字

顺序访问：字符串、列表、元组

映射访问：字典

**python判断语句**

if语句

语法：if expression:

if\_suite

else:

else\_suite

如果表达式的值非0或者为布尔值true，则代码组if suite被执行；否则就去执行ekse suite，只要表达式数字为非零值即为true，空字符串、空列表、空元组、空字典的值均为false

if “”:

print(‘yes’)

else:

print(‘no’)

扩展if语句

语法：if expression1:

if\_suite

elif expression2:

elif\_suite

else:

else\_suite

条件表达式

语法：x if c else y中间是条件，两边是结果

a=10

b=5

s=a if a<b else b

print(s)

**python循环**

python中有while和for循环，循环次数未知的情况下，建议采用while循环，循环次数可以预知的情况下，可以采用for循环

while循环语句结构

while expression:

while\_suite

语句while\_suite会被不断的执行循环，直到表达式的值为-或FLASE

sum=0

counter=1

while counter<=100:

sum+=counter

counter+=1

print(sum)

while进阶

break语句

break跳出循环体，执行下面的语句

continue语句

当遇到continue语句时，程序循环终止当前循环，并忽略剩余的语句，然后回到循环的顶端，如果条件满足，循环体内的语句继续执行，如果不满足则退出

else语句

python中的while语句也支持else子句，else子句只在循环正常完成后执行，break语句会跳过else块

for循环详解

python中的for接受可迭代对象（例如迭代器或序列）作为其参数，每次迭代其中一个参数

for iter\_var in iterable:

suite\_to\_repeat

与while循环一样，支持break、continue、else语句

alist=[‘bob’,’alice’]

for name in alist:

print(name)

range函数

for循环常与range函数一起使用

range函数提供循环语法

range函数的完整语法为：

range(start,end,step=1)

list(range(10))#只写10，表示结束数字10，但是不包含10，开始数字是0

list(range(6,10))#[6,7,8,9,10]

list(range(1,10,2))#[1,3,5,7,9]

list(range(10,0,-1))#[10,9,8,7,6,5,4,3,2,1]

for i in range(1,101):

print(‘hello’)#输出三次hello

列表解析

是一个非常有用、简单、而且灵活的工具，科一动态创建列表

语法：

[expr for i in iterable]

[3+2 for i in range(10)]#循环控制3+2运行多少次,[5,5,5,5,5,5,5,5,5,5]

**文件对象**

文件打开方法

open及file内建函数

内建函数open()以及file()提供了初始化输入/输出（I/O）操作的通用接口

成功打开文件时候会返回一个对象，否则引发一个错误

open()方法和file()方法可以完全相互替换

基本语法：

aaa=open(file\_name,’r’,buffering=-1)

文件对象访问方法

r以读的方式打开（文件不存在则报错）

w以写方式打开（文件存在则清空，不存在则创建）

a以追加模式打开（必要时创建文件）

r+以读写模式打开（参见r）

w+以读写模式打开（参见w）

a+以读写模式打开（参见a）

b以二进制模式打开

文件输入

read方法

read()方法用来直接读取字节到字符串中，最多读取给定数目个字节

如果没有给定size参数（默认值为-1）或者size值为负，文件将读取至末尾

data=fobj.read()

print(data)

readline方法

读取打开文件的一行（读取下个行结束符之前的所有字节）

然后整行，包括行结束符，作为字符串返回，有一个可选的size参数，默认为-1，代表读至行尾结束符，如果提供了该参数，那么在超过size个字符后返回不完整的行

data=fobj.readline()

print(data)

readlines方法

readlines()方法读取所有（剩余的）行然后把它们作为一个字符串列表返回

data=fobj.readlines()

print(data)

文件迭代

如果需要逐行处理文件，可以结合for循环迭代文件

迭代文件的方法与处理其他序列类型的数据类似

fobj=open(‘str.py’)

for eachline in fobj:

print(eachline,end=’’)

forj=open(‘/tmp/mima’)

fobj.read(4)#读取四个字节，建议一次是1024的倍数

fobj.readline()#适合文本，读取一行

fobj.readlines()#适合文本，把所有行读到列表中

fobj.close()

文件输出

write方法

write()内建方法功能与read()和readline()相反。它包含有文本数据或二进制数据块的字符串写入到文件中，写入文件时，不会自动添加行结束标志，需要程序员手工输入

writelines方法

和readlines()一样，主要是针对列表的操作，它接受一个字符串列表作为参数，将它们写入文件，行结束符不会自动被加入，如果需要的话，必须在调用前每行结尾加上行结束符

fobj=open(‘/tmp/mima’,’w’)

fobj.write(‘hello world!\n’)#写数据时，先放到缓冲区

#当数据量达到一定程度时，自动写入磁盘

fobj.flush()#立即将数据同步到磁盘

fobj.writelines([‘3rd line.\n’,’new line.\n’])

fobj.close()#关闭文件，数据自动写入磁盘

操作文件

with子句

with语句是用来简化代码的，再将打开文件的操作放在with语句中，代码结束后，文件将自动关闭

with open(‘/tmp/mima’) as fobj:#with语句结束后，文件自动关闭

print(fobj.readline())

文件内移动

seek(offset[,whence]):移动文件指针到不同位置

offset是相当于某个位置的偏移量

whence的值，0表示文件开头，1表示当前位置，2表示文件的末尾

tell():返回当前文件指针的位置

fobj=open(‘/tmp/mima’,’rb’)#非文本要加上b，文本文件也可以加上b，表示byte

fobj.tell()#返回当前文件指针的位置

fobj.seek(10,0)#第一个数字是偏移量;第二个数字是相对位置，0表示开头，1是当前

fobj.read(4)#读到第四个字节，因为是bytes方式打开，所以显示b’xxxx'

fobj.seek(-5,2)#移动到文件倒数第5个字节的位置。以’rb’擦爱能写负数，’r’不行

标准文件

程序一执行，就可以访问三个文件

标准输入：一般是键盘，使用sys.stdin

标准输出：一般是显示器缓存输出，使用sys.stdout

标准错误：一般是显示器的非缓存输出，使用sys.stderr

import sys

a=sys.stdin.readline()#读键盘输入，回车也作为一个字符\n输入

print(a)

sys.stdout.write(a)

sys.stderr.write(a)

**函数基础**

创建函数

函数是用def语句来创建的，语法如下

def function\_name(arguments):

function\_body\_suite

标题行由def关键字，函数的名字，以及参数的集合（如果有的话）组成，def子句的剩余部分包括了一个虽然可选但是强烈推荐的文档字符串，和必须的函数体

调用函数

同大多数语言相同，python用一对圆括号调用函数，如果没有加圆括号，只是对函数的引用

def foo():

print(‘hello’)

foo()

函数的返回值

形式参数：函数定义时，紧跟在函数名后（圆括号内）的参数被称为形参

实际参数：在主调函数中调用一个函数时，函数后面的圆括号中的参数（可以是一个表达式）称为实参

调用函数时，实参的个数需要与形参个数一致，实参将以此传递给形参

def foo(a,b):

print(‘a=%d,b=%d’ % (a,b))

foo(3,4)

位置参数

与shell脚本类似，程序名以及参数都以位置参数的形式传递给python程序

使用sys模块的argv列表接收

import sys

print sys.argv

默认参数

默认参数就是为了声明默认值的参数，因为给参数赋予了默认值，所以在函数调用时，不向该参数传值也是允许的

def pstar(num=10):

print(‘\*’ \* num)

pstar()

**模块基础**

创建模块

模块在物理层面上组织模块的方法是文件，每一个以.py结尾的python文件都是一个模块；模块名称不要与系统中已存在的模块重名；文件名字去掉后面的扩展名（.py）就是模块名

导入模块（import）

使用import导入，模块被导入后，程序会自动生成pyc的字节码文件以提高性能，模块通过“模块名.属性”的方法调用，如果仅需模块中的某些属性，也可以单独导入

import sys

import os,string

from random import randint

模块加载（load）

一个模块只允许被加载一次，无论被导入多少次；只加载一次可以阻止多重导入时代码被多次执行；模块夹杂爱时，顶层代码会自动执行，所以只将函数放入模块的顶层是良好的编程习惯

模块的导入特性

模块具有一个\_\_name\_\_特殊属性，当模块文件直接执行时，\_\_name\_\_的值为’\_\_main\_\_’，当模块被另一个文件导入时，\_\_name\_\_的值就是该模块的名字

vim foo.py

hi=’hello world’

if \_\_name\_\_ == ‘\_\_main\_\_’:

print(hi)

vim bar.py

import foo

**shell相关模块**

shutil模块

复制和移动

shutil.copyfileobj(fsrc,fdst[,length])

将类似文件的对象fsrc的内容复制到类似文件的对象fdst

shutil.copyfile(fsrc,fdst,\*,follow\_symlinks=True)

将文件src的文件内容（无元数据）复制到名为dst的文件，然后的返回dst

shutil.copy(src,dst,\*,follow\_symlinks=True)

将文件src复制到文件或目录dst。src和dst应为字符串

shutil.copy2(src,dst,\*,follow\_symlinks=True)

与copy相同，但copy2也尝试保留所有文件元数据

shutil.move(src,dst,copy\_function=copy2)

递归的将文件或目录移动到另一个位置，并返回目标

import shutil

# https://yiyibooks.cn/

f1 = open('/etc/passwd', 'rb')

f2 = open('/tmp/abcd', 'wb')

shutil.copyfileobj(f1, f2)

f1.close()

f2.close() # 复制出来/tmp/abcd

shutil.copyfile('/etc/passwd', '/tmp/xyz') # 复制出来xyz

shutil.copy('/etc/passwd', '/tmp') # 复制到tmp目录底下

shutil.copy('/etc/passwd', '/tmp/aabb') # 复制出来aabb

shutil.copy2('/etc/passwd', '/tmp/aabbcc') # 复制出来aabbcc，并且不改变权限

shutil.move('/tmp/abcd', '/home') # 移动abcd到另外一个地方

目录操作

shutil.copytree(src,dst,symlins=False,ignore=None,copy\_function=copy2,ignore\_dangling\_symlinks=False)

递归地复制以src为根的整个目录树，返回目标目录。目标目录不能已经存在

shutil.rmtree(path,ignore\_errors=Flase,onerror=None)

删除整个目录树；路径必须指向目录，而不是指向目录的符号链接

shutil.copytree('/home', '/asd') # 复制目录，目标目录必须不存在

shutil.rmtree('/asd') # 删除目录

权限管理

shutil.copymode(src,dst,\*,follow\_symlinks=True)

将权限位从src复制到dst。文件内容，所有者和组不受影响。src和dst是以字符串的形式给出路径名称

shutil.copystat(src,dst,\*,follow)symlinks=True)

将权限位，最后访问时间，上次修改时间和标志从src移动到dst

shutil.chown(path,user=None,group=None)

更改给定路径的所有用户和/或组

shutil.copymode('/home', '/asd') # 复制权限，目标必粗存在

shutil.copystat('/home', '/asd') # 复制访问时间等

shutil.chown('/asd', user='www', group='www') # 更改目录所属组所属主

**语法风格**

变量赋值（支持链式多重赋值）：x=y=10，a,b=10,20

合法标识符

第一个字符必须是字母或下划线，剩下的字符可以是数字、字母或下划线，区分大小写

关键字

关键字列表和iskeyword()函数都放入了keyword模块以便查询

内建

除了关键字外，python还可以在任何一级代码使用“内建”的名字集合，但是应该把它当做“系统保留字”，常见的系统保留字由：True、False、None

模块结构及布局

编写程序时，应该建立一种同一切容易阅读的结构，并将它应用到每一个文件中趣

起始行（使用什么翻译器#!/usr/bin/env python）

模块文档字符串

导入模块

全局变量声明

类定义

函数定义

程序主体（if \_\_name\_\_==’\_\_main\_\_’）

**字符串详解**

序列

序列操作符

seq[ind]获取下标为ind的元素

seq[ind1:ind2]获取下标从ind1到ind2的元素集合

seq \* expr序列重复expr次

seq1+seq2连接序列seq1和seq2

obj in seq判断obj元素是否包含在seq中

obj not in seq判断obj元素是否不包含在seq中

内建函数

list(iter)把可迭代对象转换为列表

str(obj)把obj对象转换成字符串

tuple(iter)把一个可迭代对象转换成一个元组

len(seq)返回seq的长度

max(iter,key=none)返回iter中的最大值

enumerate接收一个可迭代对象座位参数，返回一个enumerate对象

reversed(seq)：接收一个序列作为参数，返回一个有序的序列

sorted(iter)接受一个可迭代对象作为参数，返回一个有序的列表

str(100)

str([10, 20, 30])

max((10, 200, 30))

min((10, 200, 30))

users = ['bob', 'alice', 'join']

for i in range(len(users)):

print('#%s:%s' % (i, users[i]))

list(enumerate(users)) # 应用时不必转换

for item in enumerate(users):

print('#%s:%s' % (item[0], item[1]))

for ind, user in enumerate(users): # (0,bob),(1,alice)

print('#%s:%s' % (ind, users))

for u in reversed(users): # 翻转

print(u)

print(sorted(users)) # 排序

字符串

字符串操作符

比较操作符：字符串大笑按ASCII码值进行大小比较

切片字符串：[]、[:]、[::]

成员关系操作符：in、not in

格式化操作符

%c转换成字符

%s优先用str()函数进行字符串转换

%d%i转换成有符号十进制数

%o转换成无符号八进制数

%e/%E转换成科学计数法

%f/%F转换成浮点数

\*定义宽度或者小数点精度

-左对齐

+在正数前面显示+号

<sp>在正数前面显示空格

#在八进制数前面显示零0，在十六进制前面显示0x或者0X

0显示的数字前面填充0而不是默认的空格

'%s is %s years old' % ('bob', 25)

'%s is %d years old' % ('bpb', 25)

'%12s%8s' % ('name', 'age') # 总宽度是12个字符，name占了4个，还剩8个空格，默认右对齐

'%-12s%-8s' % ('name', 'age') # 总宽度是12个字符，name占了4个，还剩8个空格，默认左对齐

'%c' % 98 # ascii为8的字符

'%#o' % 30 # 8进制数形式

'%#x' % 30 # 16进制数形式

'%f' % (5 / 3) # 浮点数

'%5.2f' % (5 / 3) # 总宽度5位，小数占2位

'%+d' % 10 # 正数前面加+号

'% d' % 10 # 正数前面加空格

'%010d' % 55 # 总宽度为10，不够宽度不够，前面补0

format函数

使用位置参数

‘my name is {},age {}’.format(‘hoho’,18)

使用关键字参数

‘my name is {name},age is {age}’.format({‘name’:’bob’,’age’:23})

填充与格式化

{:[填充字符][对齐方式 <^>][宽度]}

使用索引

'name is {0[0]} age is {0[1]}'.format(['bob', 23])

'{} is {} years old'.format('bob',25)#'bob is 25 years old'

'{1} is {0} years old'.format(25,'bob')#'bob is 25 years old'

'{:<20}'.format('hello') # 'hello '

'{:>20}'.format('hello') # ' hello'

'{:^20}'.format('hello') # ' hello '

原始字符串操作符

原始字符串操作符是为了对付那些在字符串中出现的特殊字符，在原始字符串里，所有的字符都是直接按照字面的意思来使用，没有转义或不能打印字符

内建函数

string.capitalize():把字符串的第一个字符大写

string.center(width):返回一个源字符串居中，并使用空格填充至长度width的新字符串

string.count(str,beg=0,end=len(string)):返回str在string里面出现的次数，如果beg活着end指定则返回指定范围内str出现的次数

string.endswith(obj, beg=0,end=len(string)):检查字符串是否以obj结束，如果beg活着end指定则检查指定范围内是否以obj结束，如果是，则返回true，否则返回false

string.islower():如果string中包含至少一个区分大小写的字符，并且所有这些字符都是小写，则返回true，否则返回false

string.strip():删除string字符串两端的空白

string.upper():转换string中的小写字母为大写

string.split(str=””,num=string.count(str)):以str为分隔符切片string，如果num有指定，则仅分隔num个子字串

astr='\thello '

astr.split()#去除两端空白字符

astr.lstrip()#去除左端空白字符

astr.rsplit()#去除右端空白字符

'hello.tar.gz'.split('.')#['hello', 'tar', 'gz']，用点来切割，默认是空格，把切出来的放到列表里面

hi='hello world'

hi.title()#'Hello World'，每个单词字母大写，本身不会变，只会返回一个改完的

hi.upper()#'HELLO WORLD'，转成大写字母

hi.lower()#'hello world'，转成小写字母

hi.islower()#true，所有字母是不是小写，不关数字

'hao123'.isdigit()#false，所以都是数字，有不是数字的返回false

hi.isidentifier()#false，是不是标识符

hi.center(50)#' hello world '，居中，宽度是50

hi.center(50,'#')#'###################hello world####################'，用井号填充

hi.ljust(50)#'hello world '，实现左对齐

hi.rjust(50)#' hello world'，实现右对齐

hi.startswith('he')#true，是不是以he开头

hi.endswith('d')#true，是不是以d结尾，可以是多个字符

**列表和元组**

列表

创建及访问列表

列表是有序、可变的数据类型，列表中可以包含不同类型的对象，列表可以由[]或工厂函数创建，支持下标及切片操作

更新列表

通过下标只能更新值，不能使用标添加新值

列表内建函数

list.append(obj)向列表中添加一个对象obj

list.count(obj)返回一个对象obj在列表中出现的次数

list.extend(seq)把序列seq的内容添加到列表中

list.index(obj)返回obj对象的下标

list.insert(index.obj)在索引为index的位置插入对象obj

list.reverse()原地反转列表

list.sort()排序

alist=[1,50,60,100]

alist[0]=10#把第一项修改为10

alist[1:3]=[30,40]#修改第二项和第三项的值

alist[3:3]=[70,80,90]#在第三项后添加三个

alist.append(30)#在列表最后追加30

alist.sort()#排序，默认升序

alist.reverse()#排序，以降序排列

alist.count(30)#统计30出现了几回

alist.remove(30)#将第一个出现的30删除

alist.pop(2)#弹出下标为2的，默认弹出最后一个

alist.index(50)#返回50的下标

alist.extend('abc')#以列表的形式扩充进去

alist.extend(['abc','xyz'])

alist.insert(3,1000)#在下标为3的地方插入1000

元组

创建元组

通过()或工厂函数tuple()创建元组、元组是有序的、不可变类型、与列表类似，作用于列表的操作，绝大多数也可以作用域元组

单元素元组

如果一个元组只有一个元素，那么创建该元组的时候，需要加上一个逗号atuple=(‘hello’,)

更新元组

虽然元组本身是不可变的，但是因为它属于容器类型，也就意味着元组的某一个元素是可变的容器类型，那么这个元素中的项目仍然是可变的

atuple = ('bob', ['bob', 23]) # ('bob', ['bob', 23])

atuple[-1] = ['boy', 22] # 'tuple' object does not support item assignment

atuple[-1][-1] = 22 # ('bob', ['bob', 22])

atuple[-1].append(175) # ('bob', ['bob', 22, 175])