for循环

range函数

• for循环常与range函数一起使用

• range函数提供循环条件

• range函数的完整语法为:

课堂练习：

斐波那契数列

1. 斐波那契数列就是某一个数,总是前两个数之和,比如0,1,1,2,3,5,8

2. 使用for循环和range函数编写一个程序,计算有10个数字的斐波那契数列

3. 改进程序,要求用户输入一个数字,可以生成用户需要长度的斐波那契数列

# fib = [0,1]

# aa = int(input('请输入一个数:'))

# for i in range(aa):

# fib.append(fib[-1] + fib[-2])

#

# print(fib)

九九乘法表

1. 创建mtable.py程序

2. 乘序运行后,可以在屏幕上打印出九九乘法表

3. 修改程序,由用户输入数字,可打印任意数字的乘法表

# for i in range(1,10):

# for j in range(1,i+1):

# print('%s\*%s=%s' % (j,i,i\*j),end=' ')

# print()

二．文件对象

open及file内建函数

open()方法和file()方法可以完全相互替换

文件对象访问模式



三．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,文件将被读取直至末尾

# fobj = open('/etc/passwd','rb')

# data = fobj.read()

# print(data,end=' ')

[3 + i for i in range(10)] #range(10) 从0开始

[3 +i for i in range(10) if i % 2 == 1]

['192.168.1.%s' % i for i in range(1,255)]

#cp /etc/passwd /tmp/mima

#fobj =open('mima') 报错

fobj = open('/tmp/mima')

data = fobj.read()

print(data) #打印内容

data = fobj.read() #因为文件指针已经移动到尾部，没有数据可读，所以data是空串

print(data)

fobj.close()

fobj = open('/tmp/mima')

fobj.read(4) #指定读取4字节，建议一次读1024的倍数

fobj.readline() #适合文本文件，读一行

fobj.readlines() #适合文本文件，把所有行读到列表中

fobj.close()

fobj = open('/tmp/mima')

for line in fobj:

print(line,end='')

fobj.close()

fobj = open('/tmp/mima', 'w')

fobj.write('hello world!\n') #写数据时，先放到缓存区，

# 当数据达到一定程度时，自动写入磁盘

fobj.flush() #立即将文件同步至磁盘

fobj.writelines(['haha\n','new line\n'])

fobj.close() #关闭文件时，数据自动写入磁盘

with open('/tmp/mima') as fobj:

print(fobj.readline())

################## 不重要

fobj = open('/tmp/mima','rb') #非文本文件要加上b，文本文件也可以加上b，表示bytes

fobj.tell() #返回文件指针的位置

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) #从当前位置读4字节，因为时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’才能写负数，‘r’不行

标准文件

import sys

a = sys.stdin.readline() #读取键盘输入，回车为作为一个字符\n读入

print(a)

sys.stdout.write(a) #标准输出

sys.stderr.write(a) #标准错误

#########cp

a = open('/bin/ls','rb')

b = open('/root/ls','wb')

data = a.read()

b.write(data)

a.close()

b.close()

课堂练习

模拟cp操作

1. 创建cp.py文件

2. 将/bin/ls“拷贝”到/root/目录下

3. 不要修改原始文件

# aa = open('/tmp/mi','rb')

# bb = open('/root/mi','wb')

# data = aa.read()

# bb.writelines(data)

# aa.close()

# bb.close()

1. 函数基础
2. 函数基本操作

函数是用def语句来创建的

标题行由def关键字,函数的名字,以及参数的集合组成、

1. 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()

#

# foo

1. 函数的返回值

• 多数情况下,函数并不直接输出数据,而是向调用者返回值

• 函数的返回值使用return关键字

• 没有return的话,函数默认返回None

# def foo():

# res = 3+4

#

# i = foo()

# print(i)

4.函数参数

定义参数

形式参数

– 函数定义时,紧跟在函数名后(圆括号内)的参数被称为形式参数,简称形参。于它不是实际存在变量,所以又称虚拟变量

• 实际参数

– 在主调函数中调用一个函数时,函数名后面括弧中的参数(可以是一个表达式)称为“实际参数”,简称实参

传递参数

• 调用函数时,实参的个数需要与形参个数一致

• 实参将依次传递给形参

#

# def foo(x,y):

# print('x=%d,y=%d' % (x,y))

#

# 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方式传递给python程序

• 使用sys模块的argv列表接收

默认参数

• 默认参数就是声明了默认值的参数

• 因为给参数赋予了默认值,所以在函数调用时,不向该参数传入值也是允许的

# def pstar(num = 30):

# print('\*' \* num)

#

# pstar()

# pstar(40)

五．模块基础

1.定义模块

导入模块(import)

模块加载(load)

一个模块只被加载一次,无论它被导入多少次

只加载一次可以阻止多重导入时代码被多次执行

如果两个文件相互导入,防止了无限的相互加载

模块加载时,顶层代码会自动执行,所以只将函放入模块的顶层是良好的编程习惯

模块导入的特性

• 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

课堂练习

生成随机密码

• 创建randpass.py脚本,要求如下:

1. 编写一个能生成8位随机密码的程序

2. 使用random的choice函数随机取出字符

3. 改进程序,用户可以自己决定生成多少位的密码

#

# from random import choice

# from string import ascii\_letters,digits,punctuation

#

# all\_chs = ascii\_letters + digits + punctuation

#

#

#

#sj = '1234567890qwertyuiopasdfghjklzxcvbnmmnbvcxzlkjhgfdsapoiuytrewq0987654321'

# #aa = random.choice(sj)

#

# def randpass(n=8):

# result=''

# for i i n range(n):

# ch = choice(all\_chs)

# result += ch

# return result