**基础**

python中变量的数据类型会随值的类型的改变而改变

\：转义符

//：整除

global关键字：

作用：声明函数内部的变量为全局型

效果：在函数内部可以对函数外部的变量进行操作（只能是同一文件内）

**Print函数**

格式化输出：print（“格式化字符串”%变量1）

Print（“格式化字符串”%（变量1，变量2，））

格式化字符串：带有%的字符串被称为格式化字符串

格式化字符：%f：浮点数，%.02f表示小数点后只显示两位（%.2f效果相同）

%d：有符号的十进制整数，%06d表示输出的整数显示位数，不足的用0补全

%s：字符串

%%：输出%，百分数

Eg：a=2.5

B=10

Print（“苹果%.02f每公斤，买%d斤”%（a，b））

**字符串**

1.字符串前加‘r’表示原始字符串，解决转义符（\）带来的问题。如：‘e:\ test.txt’中的\t会变成制表符。而r‘e:\ test.txt’则不会。

2.字符串用‘ ’和“ ”效果相同，如果引号内存在引号，则应用不同的引号括起。如：‘她说：“早上好。”’或者“她说：‘早上好。’”这两种方式python语法上都对。

**标识符**

Python的标识符区分大小写，关键字同样区别大小写

**If语句**

判断条件结尾加冒号

If语句块内代码缩进四个空格或者一个Tab键。

Else结尾加冒号，配合if语句使用，用于条件不成立时

Elif结尾同样加冒号

Python将非空字符串解读为True。

判断条件为布尔值时：

1.if a： 当a为True时执行if语句块。即使a=False，也按上述判断执行。

2.if not a：当a为False时执行。

3.if a==True： 当a为True时执行。

**转换函数**

Int（）转换为整型数 float（）转换为浮点数 str（）转换为字符串

**逻辑运算符**

And与 or或 not非

Not用来给条件取反，如if B（一个布尔变量）： 意为当B=TRUE时执行，而if not B： 意为当B=False时执行

**列表**

1. list[ ]=新值 可改变列表内元素的值
2. list[-1]=列表最后一个元素

3.切片：list[起始下标：终止下标：变化量]

起始下标：同其它语言一样，从0开始。序列从左向右方向中，第一个值的索引为0,最后一个为-1

终止下标：切片操作符将取到该索引为止，不包含该索引的值。

变化量：默认是一个接着一个切取，如果为2,则表示进行隔一取一操作。步长值为正时表示从左向右取，如果为负，则表示从右向左取。步长值不能为0

**元组**

name=( )不可改变元组内元素的值

**字典**

dict={ }可随时添加修改键值对。向字典内添加键值对（访问用print）：dict[‘要添加的键’]=值

**for循环**

1. for 变量，变量 in 元组range（）、列表、字典：将元组、列表、字典内的值提取出来赋给变量。
2. 字典.items（）导出键、值两个量，所以需要两个变量接收。字典=字典.key（）导出键。字典.value（）导出值。

**while循环**

1. 直到型循环。while 结束条件：循环直到条件不满足为止。

2.结束条件为数字或者True时会无限循环。break语句用来结束循环。continue语句返回循环开头，并根据条件测试结果决定是否继续执行循环。

**函数**

1.def function\_name（形参）:

2.传递实参方式：1.位置传递 与形参位置对应赋值。2.关键字传递 关键字=实参（关键字传递要放在位置传递的后面，二者不可轻易混用）。

3.给形参赋予默认值时要放在参数列表的最后。

4.return语句用来返回值

5.传递列表切片：function\_name（list\_name[ : ]）。这样就不会修改原列表，修改的是原列表的副本。

6.\*形参：创建一个元组，将传递的任意数量的实参收入元组中。若与其他实参混合使用，则必须将此类型的形参放在最后。

7.\*\*形参：创建一个字典，将传递的任意数量的关键字实参收入字典中。若与其他实参混合使用，则必须将此类型的形参放在最后。

8.从模块导入函数

1）from 模块名 import 函数名，函数名，函数名。导入模块内的函数，使用语法为：直接使用函数。

2）import 模块名 。导入模块，使用其中的函数的语法为：模块名.函数名。

3）from 模块名 import \* 。导入模块内全部的函数，使用语法为：直接使用函数

**类**

1. 从别的模块导入类（类的导入与函数类似）：

1）from 模块名 import 类名，类名，类名。导入模块内的类，使用语法为：实例=类。

2）import 模块名 。导入模块，使用其中的类的语法为：实例=模块.类。

3）from 模块名 import \* 。导入模块内全部的类，使用语法为：实例=类

2.def \_\_init\_\_（self，形参，形参，）:初始化类的属性。

**文件**

**打开**：open（文件路径，打开模式）windows系统中文件路径中使用反斜杠（\）。**打开模式**：（‘w’）写入模式，会覆盖源文件；（‘r’）只读模式；（‘a’）附加模式，不会对源文件进行覆盖操作；（‘r+’）读写模式，优先读取，会覆盖源文件；（‘w+’）读写模式，优先写入，会覆盖源文件。**所有读取模式如果目标文件没有创建则报错，写入模式则会创建一个目标文件。**

**with打开**：with open...as file打开文件并命名为file，关闭时间交给python自己处理（即在with代码块内使用，with代码块执行完毕时关闭）。

**关闭**：close（文件名）手动关闭文件。

**读取**：对象.read（）python读取文本文件中的内容时都解读为字符串。

**逐行读取**：for line in 文件：、对象.readlines（）从文件中读取每一行并存储在一个列表里。

**写入**：文件.write（字符串）写入数据。使用换行符（\n）可写入多行。

**json模块：**

json文件能够存储python的数据结构（如列表、字典、元组等）

1.json.dump（要存储的数据，文件对象）存储数据

2.json.load（文件名）读取文件

**异常及处理**

**python使用被称为异常（trackback）的特殊对象来管理程序执行期间发生的错误。**

处理：try-except-else语句：处理异常，告诉python发生异常时该怎么做。提高程序抵御错误的能力

1.try：python尝试执行try中的代码块，若出现问题则执行except代码块。2.except：若try代码块发生了指定的异常，except代码块会告诉python怎么做。

3.else：try代码块成功运行后需要执行的代码放在else代码块中。

4.pass语句：告诉python什么都不做。

**测试**

测试源代码的输出结果与预期是否一致，过程如下：

1. 导入测试模块unittest
2. 导入待测试对象
3. 创建测试子类（父类：unittest.TestCase）
4. 测试类中创建测试函数：调用待测试函数并将返回值存储在一个变量中；调用self的一个方法assertEqual（返回值，期望的结果）来测试返回值与期望值是否一致。
5. 调用方法unittest.main（）运行测试。输出的分析结果为红色字体；第一行句点表示成功运行了一个测试，F表示因被测试对象的问题失败了一个测试，E表示因测试程序本身有问题失败了一个测试；第二行表示测试运行的耗时；第三行OK表示测试全部通过，FAILED（errors=x）表示测试未通过， 测试程序本身出现X个错误。FAILED（failures=x）表示测试未通过，被测试对象出现X个错误。

测试类：

测试过程与测试函数类似，测试类时需创建类的实例。

1. 方法setUp（）创建供测试用的实例，可用于每个测试方法。若在TestCase的子类中包含了该方法，python将先运行它，再运行各个以test打头的方法。

**pygame模块：开发游戏工具**

pygame刷新一次为一帧，默认1ms一帧。

基本框架：

import pygame

import sys

def start（）:

pygame.init（）  **#初始化pygame**

screen=pygame.display.set\_mode（width，height） **#创建显示窗**

\*pygame.display.set\_caption（字符串） **#设置窗口名，非运行必要条件**

while True:

for event in pygame.event.get（）: **#事件判断，否则进入死循环**

if even.type==pygame.QUIT:

sys.exit（）

pygame.display.flip（） **#刷新屏幕**

start（）  **#运行开始游戏函数**

具备上述**必要**语句则可运行最初始游戏窗口。

**错误的产生**

1.AttributeError：属性没找到错误。可能是位置传参不匹配导致属性找不到。

2.IndentationError：缩进错误

3.SystaxError：语法错误

4.IndexError：索引错误