1个#是单行注释，多行注释：**"""内容"""**或前后都是3个单引号。

1. print函数，输出内容一定要加小括号包裹
2. If 1==1: print(“1等于1”)
3. 导入模块的某个函数from math import sqrt

sqrt(9)

结果是3

导入函数，复数计算写法Import cmath

cmath.sqrt(-1)

结果是1j

int(3.9) #结果是3，int()是向下取整。

1. Variable(变量)=raw\_input(“”) #3.x版本都统一用input(“”)，2.x中，input(“”)会把数入的内容带上’’符号，3.x取消了差别，都是input(“”)，结果就是输入的内容，无添加。
2. 直接输入’Let\’s go’，IDLE输出”Let’s go” \是转义符。
3. Str()和repr()都是用来将值转换成字符串。Str易阅读，repr或反引号转成合法的python表达式。

长字符串多行的写法：print (**'''ddd  
ab  
dd  
'''**)

1. 原始字符串前面加个r，示例：原本\n是换行，此处加了r后，就是打印原本的内容，不换行。print (**r"c:\nhere"**)。若最后是“\”结果会报错。解决方法print (**r"c:\nhere"”\\”**)。
2. **"""序列分为列表、字符串和元组。列表可变，字符串和元组不可变。  
   类表和元组和索引示例  
   months=['January','February','March','April','May','June','July','August','Septemper','October','November', 'December']  
   endings=['st','nd','rd']+17\*['th']\  
    +['st','nd','rd']+17\*['th']\  
    +['st']  
   year=input("Year:")  
   month=input("month(1-12):")  
   day=input("Day[1-31]:")  
   month\_number=int(month)  
   day\_number=int(day)  
   month\_name=months[month\_number-1]  
   ordinal=day+endings[day\_number-1]  
   print(month\_name+' '+ordinal+'.'+year)  
   """  
   """分片  
   numbers=[1,2,3,4,5,6]  
   numbers[1:2] #结果是[2]因为从索引0开始编号，冒号为分隔，冒号前面的包含，后面的不包含在分片中。  
   numbers[-2:]#结果是[4,5,6]就是倒数，然后一直到最后。  
   numbers[:3]#结果是[1,2,3],就是正数，索引从第一个0的数字1，到索引为2的3数字3截止。  
   numbers[:]#结果就是整个序列。  
   #第三个参数：分片的步长  
   numbers[0:4:5] #1,2,3,4,5中，每隔5个数取一次值。运行结果就只有1.步长可以是负数，从右到左取值numbers[4:0:-5],结果是5。  
   """  
   """序列成员资格示例  
   database=[['a','1'],['b','2'],['c','3'],['d','4']]  
   username=input("username:")  
   pin=input("pin code：")  
   if[username,pin] in database:print ("yes")  
     
   list('hello')#序列改成列表  
   x=[1,2,3]  
   x[1]=4 #列表元素赋值  
   del x[2] #删除列表元素  
   #分片赋值增加  
   numbers=[1,5]  
   numbers[1:1]=[2,3,4]  
   #分片赋值减少  
   numbers=[1,2,3,4,5]  
   numbers[1:4]=[]  
   numbers.append(6) #返回的还是原来的列表a  
   numbers.count(6)  
   c=[7,8,9]  
   numbers.extend(c) #extend只是修改了原来的列表。列表a+列表b返回一个全新的列表(效率低)  
   numbers[len(numbers):]=c #这样也可以，不会修改原列表，但是代码可读性低。  
   numbers.index('2')  
   numbers[1]  
   numbers.insert(3,"four")#将对象插入到列表中  
   numbers[3:3]=["four"]#insert的操作用分片赋值实现。  
   #pop()方法介绍：会先移除掉列表元素，然后返回被移除的值。相当于出栈（LIFO后进先出拿碗碟）队列就是地铁排队出去，先进到队伍先出去  
   x=[1,2,3]  
   x.pop()#返回值为3，此时x=[1,2]  
   x.pop(0)#返回值为1，此时x=[2]  
   x.remove(2)#remove移除列表的值，没有返回值。  
   list(reversed(x))#反向迭代序列，只返回一个迭代器对象，用list()函数把返回的对象转换成列表。  
   a=[3,1,5]  
   b=a  
   b.sort()#此时a和b都指向[1,3,5]，不好可以用分片复制，y=x[:] y.sort() #列表.sort()返回值为None  
   #或者用sorted()，两者内容一样，但是占用两个空间  
   y=sorted(x)  
     
   x=['a','abxxxxxxxx','abcddd','abcde','abcd']  
   x.sort(key=len)#按长度排序从小到大  
   x=[4,1,5]  
   x.sort(reverse=True)#按大小反向排序，结果是从大到小。  
   #元组，用逗号分隔了一些值，就创建了元组，大部分是用()圆括号括起来的。例如（40，）。区别是不可变序列  
   tuple([1,2,3]) #把一个序列作为参数转换为元组，参数是元组原样返回。  
   x=1,2,3  
   x[0:2]#结果是（1，2）  
   """  
   """格式化字符串操作符%，转换说明符%s(s表示值会被格式化为字符串)，如果整个要转换的字符串（如format）中有百分号，要写成两个%%**

**#输出数字比如20，需要用%d（整型数字），不确定的类型用%r。**

**#最简单的，使用时不用标点符号：**

**Width=“这个”**

**Print（“Test” 这个）**

**#其它的：  
format="hello,%s,%s enough for ya%%?"  
values=('world','Hot')  
print (format % values)  
#格式化实数（浮点数），可以用f说明符类型，一个句点加上希望保留的小数位数加上类型字符  
x=('圆周率：%.12f')  
from math import pi  
print (x % pi)  
#另一种格式化值的方法：模板字符串，传参替换，用$。美元符号就两个$$。替换单词的一部分，参数名要用括号括起来  
from string import Template  
a=Template('$x，这是$x，$$，It\'s sel${x}nium!')  
a.substitute(x="e")  
#用字典变量提供值/名称对  
b=Template('A $a and $b')  
d={}  
d['a']='a'  
d['b']='b'  
b.substitute(d)  
#find方法寻找字符串中的字符串。join方法是split的逆方法  
seq=['1','2','3']  
a='+'  
a.join(seq)#结果是‘1+2+3’  
dirs='','usr','bin','env'  
'/'.join(dird)#结果是'/usr/bin/env'  
print ('c:'+'\\'.join(dirs))#结果是c:**\usr\b**in\env  
"that's all.".title()#结果是That'S All.  
import string  
string.capwords("that's all.")#结果是That's All.  
#replace()查找并替换。split()分隔符。strip方法返回去除左侧和右侧（不包含内部）空格的字符串，如果要指定去除的字符，比如\*号，放到括号中当做参数即可。strip('\*')  
translate()函数。string.maketrans(from，to),创建用于转换的转换表。  
创建字典的三种方法：字典中的键是唯一的，值并不唯一。字典的好处：不用像序列一样一定要在序号范围内插入，可以任意加到字典的某个位置中。  
#（1）键值对  
test={'1':'a','2':'b','3':'c','4':'d'}  
#（2）dict函数（不是真正的函数，是像list,tuple,str一样的类型）  
items=[('name','xiaoxue'),('age',18)]  
d=dict(items) #d的值{'age':18,'name':'xiaoxue'}  
#（3）dict函数通过关键字参数创建字典，不带参数则返回一个新的空字典{}。  
d=dict(name='xiaoxue',age=18)#d的值{'age':18,'name':'xiaoxue'}  
dict.fromkeys(['name','age'])结果是{'age':None,'name':None}  
或dict.fromkeys(['name','age'],'X')结果是{'name':'X','age':'X'}#X是自定义的值。  
copy函数（浅复制）返回的是一个相同键-值对的新字典，副本替换值f['a']='x'，原来不受影响。副本修改比如remove/append一个，原来的也会改变。c.copy()  
deepcopy函数（深复制）用法：deepcopy(d),结果就是原来的怎么变化都不会影响副本。  
from copy import deepcopy  
#get方法，如果访问字典中不存在的项会报错，如d['who'],但是print d.get('who')得到None。还可以替换None，d.get('who','X')，get()如果键存在就是查对应的值  
#dictionary.has\_key(''),返回False or True  
"""**

1. **如果是二维数组，和java中的方法是不一样的，要注意。**
2. Python3中只有items()方法返回迭代器对象。把字典项以列表方式返回。
3. Python中是没有办法删除值的。删除的只是名称，不是列表本身。
4. 定义内部类私有方法或属性，前面加上双下滑线\_\_，定义的关键字def。
5. 看一个类是否是另一个类的子类，可以使用内建的issubclass函数，issubclass(子类，父类)，True。查询子类的基类，子类.\_bases\_
6. 查看一个对象是否为一个类的示例，isinstance(对象，类)，查询对象属于那个类，对象.\_class\_
7. 多重继承，多个超类，先继承的类中的方法会重写后继承的类中的方法。继承的写法：class c(A,B)
8. Python3，检查一个特性是否可用，hasattr(x,’\_call\_’)
9. 声明异常：raise Exception()只能抛出它自己定义好的异常类，用try:/except:捕获异常,else:/finally:

捕获系统异常：try：换行内容换行except BaseException as msg:换行print(msg)换行else:换行print(“内容”)换行finally:换行print(“内容”)

1. 构造方法：def \_init\_(self):
2. 子类本身有构造方法，继承超类的构造方法：def \_init\_(self):

Super(子类类名，self).\_init\_()#或者是父类类名.\_init\_(self)

1. 集合Set，set([0,1,2,3,4])和字典一样，元素顺序是随意的。两个集合的并集，可用a.union(b)，或a | b
2. 结合本身只能包含不可变的值，不能包含其他集合。比如a.add(b)会报错。但是可以用a.add(frozenset(b)) frozenset构造函数创建给定集合的副本，代表不可变的集合。
3. Shelve.open函数返回已存储的数值。Import shelve s=shelve.open(‘test.dat’) s[‘x’]=[‘a’,’b’,’c’] s[‘x’].append(‘d’) s[‘x’]值为[‘a’,’b’,’c’] 解决方法：将临时变量绑定到获得的副本上，在它被修改后重新存储这个副本。temp=s[‘x’] temp.append(‘d’) s[‘x’]=temp
4. 迭代文件1：f=open(filename) for char/line in f.read(): process(char/line) f.close() process()方法返回逐一字符或行。Read方法是默认的，可以省略。
5. 迭代文件2：import fileinput for line in fileinput.input(filename): process(line)
6. 函数参数分类
7. 普通参数，即命名参数arg1
8. 带默认值参数，arg2=0
9. 元组参数\*arg3
10. 词典参数\*\*arg4

\*args表示任何多个无名参数，它是一个tuple；\*\*kwargs表示关键字参数，它是一个dict。并且同时使用\*args和\*\*kwargs时，\*args参数列要在\*\*kwargs前，像foo(a=1, b='2', c=3, a', 1, None, )这样调用的话，会提示语法错误“SyntaxError: non-keyword arg after keyword arg”。

还有一个很漂亮的用法，就是创建字典：

def kw\_dict(\*\*kwargs):

return kwargs

print kw\_dict(a=1,b=2,c=3) == {'a':1, 'b':2, 'c':3}

其实python中就带有dict类，使用dict(a=1,b=2,c=3)即可创建一个字典了。

1. GUI图形界面wxPython工具下载地址<https://wxpython.org/pages/downloads/>

联网后cmd输入命令自动下载：pip install -U wxPython

1. *"""  
   #猜数小游戏  
   import random  
   #生成随机数  
   answer=random.randint(1,100)  
   #玩家输入数值  
   number=int(input("请输入数字（1-100）："))  
   #判断输入数字大小  
   while number!=answer:  
    if number>answer:  
    number=int(input("大了，请输入数字（1-100）："))  
    elif number<answer:  
    number=int(input("小了，请输入数字（1-100）："))  
   #输入数字和生成的随机数相同，游戏结束  
   print("恭喜你猜对了，数字是：" +str(number))  
   """***"""  
   #读取txt内容  
   f=open("E:\\Test.txt","r")  
   lines=f.readlines()  
   print(lines)  
   for cids in lines:  
    print(cids.split(":")[0])  
   """  
   """  
   #读取csv文件，和excel比有看不见的逗号  
   import csv  
   csv\_file=csv.reader(open("Test.csv","r"))  
   print(csv\_file)  
   for abc in csv\_file:  
    print(abc)  
   #写入csv  
   stu=["C","c"]  
   #打开文件  
   out=open("Test.csv","a",newline="")#把a（追加）换成w就是覆盖最后一行  
   #设定写入模式  
   csv\_write=csv.writer(out,dialect="excel")  
   #写入内容  
   csv\_write.writerow(stu)  
   print("write over!")  
     
     
   from xml.dom import minidom  
   #加载xml文件  
   dom=minidom.parse("Testxml.xml")  
   root=dom.\_get\_documentElement()  
   # print(root.nodeName)  
   # print(root.nodeValue)  
   # print(root.nodeType)  
   names=root.getElementByTagsName("name")  
   ages=root.getElementByTagsName("age")  
   print(names[0].firstChild.data)  
   print(ages[0].firstChild.data)  
   #读取属性节点  
   logins=root.getElementByTagsName("login")  
   for i in range(2):  
    username=logins[0].getAttribute("username")  
    password=logins[0].getAttribute("password")  
   """  
     
   """  
   #单线程与多线程  
   from time import sleep,ctime  
   import threading  
   def talk(content,loop):  
    for i in range(loop):  
    print("start talk: %r" %ctime())  
    sleep(2)  
   def write(content,loop):  
    for i in range(loop):  
    print("start write: %r" %ctime())  
    sleep(3)  
     
   threads=[]  
   t1=threading.Thread(target=talk,args=("Hello,51zxw",2))  
   threads.append(t1)  
   t2=threading.Thread(target=write,args=("Life is short,you need python!",2))  
   threads.append(t2)  
   if \_\_name\_\_=="\_\_main\_\_":  
    for t in threads:  
    t.start()  
    for t in threads:  
    t.join()  
   print("All the time: %r" %ctime())  
   #如果是进程，就把threading.Thread替换成mutiprocessing.Process  
   """**
2. abspath是当前脚本的绝对路径，路径带文件本身，dirname返回的是不带文件本身的路径，即文件所在路径  
   print(os.path.abspath(\_\_file\_\_)) *#E:\Program\pythonProject\Test\_Framework\src\test\test\_baidu.py，去别就是后面有没有\test\_baidu.py*base\_path = os.path.dirname(os.path.abspath(\_\_file\_\_))+**'\..\..'***#加的'\..\..'起到的效果是向上返回两层*print(base\_path)  
   driver\_path = os.path.abspath(base\_path+**'\drivers\chromedriver.exe'**)  
   print(driver\_path)  
     
   os.path.split(path)将path分割成目录和文件名二元组返回  
   获取当前运行脚本的绝对路径（去掉最后2个路径）  
   path3 = os.path.dirname(os.path.dirname(os.path.dirname(\_\_file\_\_)))  
   print(path3)
3. 如果把方法名的第一个字母大写，程序运行竟然不把它当方法，直接跳过了
4. @property负责把一个方法变成属性，是属性就有getter和setter方法，比如把data方法变成属性(加了@property)后，可以新建方法 @data.setter **def** data(self, value):self.\_data = value
5. 关于yml文件，需要在python的安装路径Scripts目录下，左手按住shift键，右键点击在此处打开命令窗口，输入pip install pyyaml  
   直接在名为config的python package下新建一个文件，命名为config.yml，yml文件，写法URL: http:www.baidu.com。注意冒号后面一定有个空格。
6. 把数据放到excel中，并读取。以便实现数据分离，参数化。在python.exe目录，左手按住shilf，右键在此处打开命令窗口，输入pip install xlrd  
   xlrd（读excel）表、xlwt（写excel）表、openpyxl（可读写excel表）
7. Python3 import IO；Python2 import StringIO。
8. *#超简单九九乘法表***for** i **in** range(1,10):  
    **for** j **in** range(1,10):  
    print(**'{}\*{} = {}'**.format(i,j,i\*j))
9. for ab in c:

with …. as #相当于try catch