1. sys.platform只是一个字符串，显示的是python解释器的系统，而不是操作系统。
2. 流重定向，用于文本的输入与输出：

#python script.py >saveit.txt

1. #!/usr/local/bin/python 文件顶端的特定一行告诉系统python解释器保存在哪里。
2. reload函数的返回值是一个python模块对象。reload是不可传递的，重载一个模块只会重载该模块，而不能重载该模块导入的任何模块。
3. dir函数将会返回一个列表，其中包含了对象的所有属性。
4. 真除法“/”与floor除法“//”

* 在python3.0中，/现在总执行真除法，不管操作数的类型，都返回包含任何余数的一个浮点结果。在python2.x中，/表示传统除法，如果两个操作数都是整数的话，执行截断的整数除法。
* //执行floor除法（floor(x),即取不大于x的最大整数），它截除掉余数并且针对整数操作数返回一个整数，如果有任何一个操作数是浮点类型，则返回一个浮点数类型的数（整数）。
* python2.x中使用python3.0的除法，X=Y / float(Z)，或者使用from \_\_futrue\_\_ import division导入python3.0的/.

1. 集合只能包含不可变数据类型，因此，列表和字典不能嵌入到集合中。

* 列表语法（[...]）用于表示列表常量或列表解析表达式。**后者是执行隐形循环**，把表达式的结果收集到新的列表中。
* （...）语法用于表示元组和表达式，以及生成器表达式，后者是产生所需结果的列表解析的一种形式，而不是构建一个最终的列表。
* {...}表示字典常量，在python3.0中可以表示集合常量以及字典和集合的解析。

1. python类型转换

 函数                      描述

int(x [,base ])         将x转换为一个整数  
long(x [,base ])        将x转换为一个长整数  
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  
chr(x)                 将一个整数转换为一个字符  
unichr(x )              将一个整数转换为Unicode字符  
ord(x )                 将一个字符转换为它的整数值  
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

1. python中唯一使用分号的地方是作为语句界定符。
2. “+=”对列表是原处修改的意思。“+”合并总是生成新的对象。
3. 要省略换行字符，print语句后可以多个逗号
4. and和or运算符总会返回对象，不是运算符左侧的对象，就是右侧的对象。
5. if/else三元表达式：

if X:

A = Y

else:

A = Z

可以写为：

A = Y if X else Z

A = [Y, Z][bool(X)]

A = ((X and Y) or Z)

1. while和for循环中else语句，如果循环离开时没有碰到break语句，就会执行一次。
2. True和False只是整数1和0的特殊版本而已。所以用“==”与1和0测试是相等的，“is”测试是不相等的。

17.内置作用域是通过一个名为\_\_builtin\_\_的标准库模块来实现的。

18.Python中的对象包含三要素：id、type、value其中id用来唯一标识一个对象，type标识对象的类型，value是对象的值。

19.命名空间的访问

局部命名空间可以 locals()  BIF来访问，全局 (模块级别)命名空间可以通过 globals() BIF来访问。 locals 与 globals 之间的一个重要的区别，locals 是只读的，globals 不是。

def func1(i, info):

x = 12345

print(locals())

locals()["x"]= 6789

print("x=",x)

y=54321

func1(1 , "first")

globals()["y"]= 9876

print( "y=",y)

执行结果：

{'i': 1, 'x': 12345, 'info': 'first'}

x= 12345

y= 9876

解释：

locals 实际上没有返回局部名字空间，它返回的是一个拷贝。所以对它进行改变对局部名字空间中的变量值并无影响。

globals 返回实际的全局名字空间，而不是一个拷贝。所以对 globals 所返回的 dictionary 的任何的改动都会直接影响到全局变量。