关于python的学习报告（9.14）

1. print()打印

print(‘hello world’)

print(‘1+2+3’)

1. input 输入

X = input（‘输入我们想输入’）

1. 变量
2. 变量是代表某个数据（值）或名称。
3. 变量类型：

A.字符串：表示一串字符，需要用’’或””引起来

b整数

C.布尔（bool）：True（1），False

1. 运算符

算术运算符（ +，-， \*， /, %(取模之后返回余数), \*\*(幂), //(取商的整数部分)）

比较运算符（==，!=, >, <, >=,<=）

逻辑运算符（not，and， or）

优先级：算术>比较>逻辑（not>and>or）

1. 条件语句

If条件：当判定条件成立时，执行后面的指令。

其中有个三元操作符：x if 条件 else y（如果条件为真输出x，不是则输出y）

1. 循环语句

While（条件的值为True时，执行后面的语句，当判断条件为Flase时，循环结束。）

8.列表

1. 列表中添加元素

append（）只能在结尾添加，只能放一个元素

将参数作为一个元素增加到了列表末尾。

extend（）同样只能放在结尾，但是对象就可以多个

将参数作为一个列表去扩展列表末尾。

insert（）可以随意排列

1. 从列表删除元素

remove（）在括号内写下你要删除的元素，就可以删除该元素。

del（）写上元素的位置，就可以删除所在位置的元素

pop（）默认是最后一个元素，可以指定位置删除元素

9.列表操作符

reverse（）: 将列表顺序翻转。

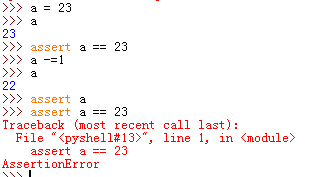
Sort（）：用指定方式对列表进行排序，默认从小到大的排序。

10.元组：

元组和列表很类似，但是元组的元素不能修改。

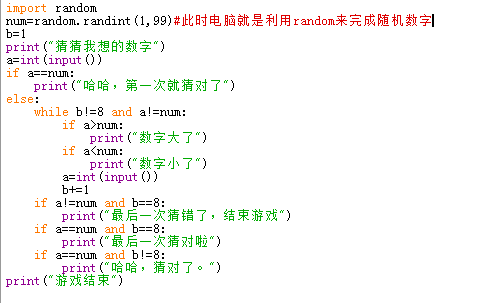
元组只包含一个元素时，需要在元素后面添加逗号。

11.assert用于判断语句的真假，如果条件为假的时候，程序会自动崩溃而且会抛出AssertionError的异常。



assert的异常参数，其实就是在断言表达式后添加字符串信息，用来解释断言并更好的知道哪里去了问题。

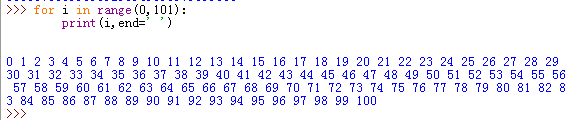
12.random随机



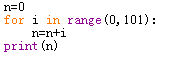
13.for 循环

For ... in ..

同while一样，for循环可以用来重复做一件事，比如说我们需要1到100的各个值。



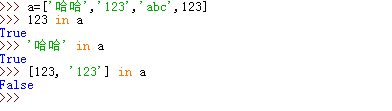
在这里我们可以很快的将各个只给打印出来，在使用时也可以迅速的使用我们想要的数字。因此我们可以试一试求和（1到100的各个值的和）





14.成员资格运算符

Python中有一个成员运算符：in 。用于检查一个值是否在序列中，如果在序列中，返回True否则就返回False。







在上面中我在列表里输入了字符串和整型，可以看到字符串和整型都能在该列表中判断是否存在。而当我这边输入一个带有相同的元素的列表进去，发现输出错误。这里可以发现in只能针对列表的信息进行反馈，可以很好的帮助我们进行更好的查找相关的需要。

15.短路逻辑



在一些进行此题的求解时，本来以为是9但在python中验证时确发现是4，这是我就有些疑问了，于是我在网上去查一下怎么回事。**布尔运算符有个特性：只有在需求值时才进行求值。表达式x and y需要两个变量都为真时才为真，所以如果x为假，表达式就会立刻返回false，而不管y的值（事实上各个语言都有这个特性）。实际上，如果x为假，表达式会返回x得值----否则它就返回y的值。这种行为被称为短路逻辑（short-circuit logic）或惰性求值（lazy evaluaion）：布尔运算符通常被称为逻辑运算符，就像你看到的那样第2个值有时“被短路了”。这种行为对于or来说也同样适用。在表达式x or y中，x为真时，它直接返回x的值，否则返回y值。注意，这意味着在布尔运算符之后的所有代码都不会执行。**

这段程序原来是这样来分解：因为优先级（not>and>or）

代码=（not 1）or （0 and 1）or （3 and 4）or （5 and 6）or （7 and 8 and 9）

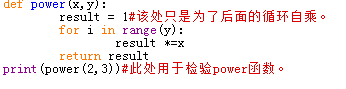
=0 or 0 or 4 or 6 or 9

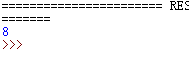
电脑从左到右判断时

(0 or 0=0) (0 or 4=4 ) (4 or.....)（此时电脑已经判断了‘4’为真时，直接跳过后面的‘4’，直接输出‘4’），这就是所谓的短路逻辑。

1. python def定义函数

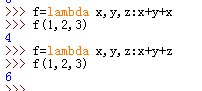
1.我们可以定义一个幂函数





1. Lambda

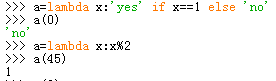
Lambda表达式是起到一个函数速写的作用。允许在代码内嵌入一个函数的定义。



我们用lambda表达式中直接得到我们想要的数：



我们也可以通过lambda表达式来进行很多方式：（求余，判断等）



使用lambda函数应该注意的几点：

* lambda定义的是单行函数，如果需要复杂的函数，应该定义普通函数
* lambda参数列表可以包含多个参数，如 lambda x, y: x + y
* lambda中的表达式不能含有命令，而且只限一条表达式