1. print 打印
2. syntaxerror 语法错误
3. # -\*- coding: uft-8 -\*- ASCII编码
4. # 注释
5. % 求余符号
6. variable 变量
7. %s 格式化字符串 print "%s, %s" % (a, b)
8. \ 转义字符（\b退格，\n换行符，\r回车符）
9. raw\_input("一些提示 ") 接受人的输入

int(raw\_input()) 将输入的内容进行格式转换

1. from sys import argv 从sys库导入argv

script, first, second = argv 通过“python 脚本 参数”来传参

11. txt = open(filename) 打开文件

Print txt.read() 读取文件

txt.close() 关闭文件

12. from os.path import exists 导入exists

Exists（file） 检查是否存在

txt = open(filename, ‘w’) 以写入方式打开文件

Txt.truncate() 清空文件

txt.seek(0) 回到文件开始

txt.readline() 读取文件的一行

13. def name(parameter1，parameter2)： 定义函数

Return parameter1 + parameter2 函数返回结果

14.word.split(‘ ’) 将word的每个单词分割出来，中间加入空格

Sorted（words） 将列表进行排序

Words.pop(0) 获得第1个元素，pop(-1)是获得最后一个元素

15.for i in range(1,10) i在1~10之间取值

16.if 条件1：

结果1

Elif 条件2：

结果2

Else：

结果3

17.列表。Elements.append() 在列表尾部追加元素。

18. from sys import exit

Exit(0) 退出:exit(0)表示程序是正常退出，exit(1)表示发生了错误。

19. while True： 当条件为真。真、假的开头字母要大写：True、False

20. **If** 语句的规则

1. 每一个“if 语句”必须包含一个 else.

2. 如果这个 else 永远都不应该被执行到，因为它本身没有任何意义，那你必须在 else 语句后面使用一个叫做 die 的函数，让它打印出错误信息并且死给你看，这和上一节的习题类似，这样你可以找到很多的错误。

3. “if 语句”的嵌套不要超过 2 层，最好尽量保持只有 1 层。 这意味着如果你在 if 里边又有了一个 if，那你就需要把第二个 if 移到另一个函数里面。

4. 将“if 语句”当做段落来对待，其中的每一个 if, elif, else 组合就跟一个段落的句子组合一样。在这种组合的最前面和最后面留一个空行以作区分。

5. 你的布尔测试应该很简单，如果它们很复杂的话，你需要将它们的运算事先放到一个变量里，并且为变量取一个好名字。

21.循环的规则

1.只有在循环永不停止时使用“while 循环”，这意味着你可能永远都用不到。这条只有 Python中成立，其他的语言另当别论。

2. 其他类型的循环都使用“for 循环”，尤其是在循环的对象数量固定或者有限的情况下。

22.调试的小技巧

1. 不要使用 “debugger”。 Debugger 所作的相当于对病人的全身扫描。你并不会得到某方面的有用信息，而且你会发现它输出的信息态度，而且大部分没有用，或者只会让你更困惑。

2. 最好的调试程序的方法是使用 print 在各个你想要检查的关键环节将关键变量打印出来，从而检查哪里是否有错。

3. 让程序一部分一部分地运行起来。不要等一个很长的脚本写完后才去运行它。写一点，运行一点，再修改一点。

23.列表

sample\_list = ['a',1,('a','b')]

'#'.join(stuff[3:5]) 用#将第3和第4个字母连接起来

list的方法  
L.append(var) #追加元素  
L.insert(index,var)  
L.pop(var) #返回最后一个元素，并从list中删除之  
L.remove(var) #删除第一次出现的该元素  
L.count(var) #该元素在列表中出现的个数  
L.index(var) #该元素的位置,无则抛异常   
L.extend(list) #追加list，即合并list到L上  
L.sort() #排序  
L.reverse() #倒序  
list操作符:,+,\*，关键字del  
a[1:] #片段操作符，用于子list的提取  
[1,2]+[3,4] #为[1,2,3,4]。同extend()  
[2]\*4 #为[2,2,2,2]  
del L[1] #删除指定下标的元素  
del L[1:3] #删除指定下标范围的元素  
list的复制  
L1 = L #L1为L的别名，用C来说就是指针地址相同，对L1操作即对L操作。函数参数就是这样传递的  
L1 = L[:] #L1为L的克隆，即另一个拷贝。

24. 字典

dictionary的方法  
D.get(key, 缺省值) #同dict[key]，多了个没有则返回缺省值。[]没有则抛异常  
D.has\_key(key) #有该键返回TRUE，否则FALSE  
D.keys() #返回字典键的列表  
D.values()  
D.items()

D.update(dict2) #增加合并字典  
D.popitem() #得到一个pair，并从字典中删除它。已空则抛异常  
D.clear() #清空字典，同del dict  
D.copy() #拷贝字典  
D.cmp(dict1,dict2) #比较字典，(优先级为元素个数、键大小、键值大小)  
#第一个大返回1，小返回-1，一样返回0  
del stuff[1] 删除第一组元素

dictionary的复制  
dict1 = dict #别名  
dict2=dict.copy() #克隆，即另一个拷贝。

25.元组

tuple = ('a', 'b', 'c', 'd', 'e')

26. 字符串

S.find(substring, [start [,end]]) #可指范围查找子串，返回索引值，否则返回-1  
S.rfind(substring,[start [,end]]) #反向查找  
S.index(substring,[start [,end]]) #同find，只是找不到产生ValueError异常  
S.rindex(substring,[start [,end]])#同上反向查找  
S.count(substring,[start [,end]]) #返回找到子串的个数

S.lowercase()  
S.capitalize() #首字母大写  
S.lower() #转小写  
S.upper() #转大写  
S.swapcase() #大小写互换

S.split(str, ' ') #将string转list，以空格切分  
S.join(list, ' ') #将list转string，以空格连接

处理字符串的内置函数  
len(str) #串长度  
cmp("my friend", str) #字符串比较。第一个大，返回1  
max('abcxyz') #寻找字符串中最大的字符  
min('abcxyz') #寻找字符串中最小的字符

string的转换  
  
oat(str) #变成浮点数，float("1e-1") 结果为0.1  
int(str) #变成整型， int("12") 结果为12  
int(str,base) #变成base进制整型数，int("11",2) 结果为2  
long(str) #变成长整型，  
long(str,base) #变成base进制长整型，

字符串的格式化（注意其转义字符，大多如C语言的，略）  
str\_format % (参数列表) #参数列表是以tuple的形式定义的，即不可运行中改变  
>>>print ""%s's height is %dcm" % ("My brother", 180)  
#结果显示为 My brother's height is 180cm

27. 现在我有三种方法可以从某个东西里获取它的内容：

# dict style

mystuff['apples']

# module style

mystuff.apples()

print mystuff.tangerine

# class style

thing = MyStuff()

thing.apples()

print thing.tangerine

28. 用super实现多继承（避免使用）

class student(person,shool\_member):  
def \_\_init\_\_(self,name,course,age):  
person.\_\_init\_\_(self,name)  
shool\_member.\_\_init\_\_(self,course)  
self.age=age  
#如何使用super实现多继承？？   
#super(student,self).\_\_init\_\_(name,course)   
#self.age=age

1. **class** Child(Parent):

4. **def** altered(self):
5. **print** "child, before parent altered()"
6. super(Child, self).altered()
7. **在\_\_init\_\_中使用super()**
8. 通常super()都是在\_\_init\_\_中使用的，完成父类的初始化，下面是一个例子：
9. class Child(Parent):
10. def \_\_init\_\_(self, stuff):
11. self.stuff = stuff
12. super(Child, self).\_\_init\_\_()

29. 你的 class 应该使用 “camel case（驼峰式大小写）”，你的其它函数应该使用 “underscore format（下划线隔词）”。

永远永远都使用 class Name(object) 的方式定义 class

30. w**.**capitalize**() 将字符串w的首字母大写。**

random.sample的函数原型为：random.sample(sequence, k)，从指定序列中随机获取指定长度的片断。sample函数不会修改原有序列。

1. list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2. slice = random.sample(list, 5)  #从list中随机获取5个元素，作为一个片断返回