**第一部分 基础篇**

##### 输入输出

###### 输出

用print()在括号中加上字符串，就可以向屏幕上输出指定的文字。比如输出'hello, world'，用代码实现如下：

print('hello, world')

print()函数也可以接受多个字符串，用逗号“,”隔开，就可以连成一串输出：

print('The quick brown fox', 'jumps over', 'the lazy dog')

print()也可以打印整数，或者计算结果：

print(300)

print(100 + 200)

因此，可以把计算100 + 200的结果打印得更漂亮一点：

print('100 + 200 =', 100 + 200)

注意，对于100 + 200，Python解释器自动计算出结果300，但是，'100 + 200 ='是字符串而非数学公式，Python把它视为字符串。

建议：设置文件编码

# -\*- coding: utf-8 -\*-

###### 输入

现在，已经可以用print()输出想要的结果了。但是，如果要让用户从电脑输入一些字符怎么办？Python提供了一个input()，可以让用户输入字符串，并存放到一个变量里。比如输入用户的名字：

name = input()

当输入name = input()并按下回车后，Python交互式命令行就在等待输入。

输入完成后，不会有任何提示，Python交互式命令行又回到>>>状态了。

要打印出name变量的内容，除了直接写name然后按回车外，还可以用print()函数：

print(name)

有了输入和输出，就可以把上次打印'hello, world'的程序改成有点意义的程序了：

name = input()

print('hello,', name)

代码执行：

python hello.py

##### 数据类型和变量

###### 数据类型

计算机顾名思义就是可以做数学计算的机器，因此，计算机程序理所当然地可以处理各种数值。但是，计算机能处理的远不止数值，还可以处理文本、图形、音频、视频、网页等各种各样的数据，不同的数据，需要定义不同的数据类型。在Python中，能够直接处理的数据类型有以下几种：

###### 整数

Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样，例如：1，100，-8080，0，等等。

计算机由于使用二进制，所以，有时候用十六进制表示整数比较方便，十六进制用0x前缀和0-9，a-f表示，例如：0xff00，0xa5b4c3d2，等等。

###### 浮点数

浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可变的，比如，1.23x109和12.3x108是完全相等的。浮点数可以用数学写法，如1.23，3.14，-9.01，等等。但是对于很大或很小的浮点数，就必须用科学计数法表示，把10用e替代，1.23x109就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。

整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确的，而浮点数运算则可能会有四舍五入的误差。

###### 字符串

字符串是以单引号'或双引号"括起来的任意文本，比如'abc'，"xyz"等等。请注意，''或""本身只是一种表示方式，不是字符串的一部分，因此，字符串'abc'只有a，b，c这3个字符。如果'本身也是一个字符，那就可以用""括起来，比如"I'm OK"包含的字符是I，'，m，空格，O，K这6个字符。

如果字符串内部既包含'又包含"怎么办？可以用转义字符\来标识，比如：

'I\'m \"OK\"!'

表示的字符串内容是：

I'm "OK"!

转义字符\可以转义很多字符，比如\n表示换行，\t表示制表符，字符\本身也要转义，所以\\表示的字符就是\，可以在Python的交互式命令行用print()打印字符串看看：

如果字符串里面有很多字符都需要转义，就需要加很多\，为了简化，Python还允许用r''表示''内部的字符串默认不转义。

###### 布尔值

布尔值和布尔代数的表示完全一致，一个布尔值只有True、False两种值，要么是True，要么是False，在Python中，可以直接用True、False表示布尔值（请注意大小写）：

False、True

布尔值可以用and、or和not运算。

###### 空值

空值是Python里一个特殊的值，用None表示。None不能理解为0，因为0是有意义的，而None是一个特殊的空值。

###### 变量

变量的概念基本上和初中代数的方程变量是一致的，只是在计算机程序中，变量不仅可以是数字，还可以是任意数据类型。

变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和\_的组合，且不能用数字开头。

###### 常量

所谓常量就是不能变的变量，比如常用的数学常数π就是一个常量。在Python中，通常用全部大写的变量名表示常量：

PI = 3.14159265359

但事实上PI仍然是一个变量，Python根本没有任何机制保证PI不会被改变，所以，用全部大写的变量名表示常量只是一个习惯上的用法。

##### list和tuple

###### list

Python内置的一种数据类型是列表：list。list是一种有序的集合，可以随时添加和删除其中的元素。

比如，列出班里所有同学的名字，就可以用一个list表示：

classmates=['Michael','Bob','Tracy']

变量classmates就是一个list。用len()函数可以获得list元素的个数：

len(classmates)

用索引来访问list中每一个位置的元素，记得索引是从0开始的：

classmates[0]

classmates[1]

当索引超出了范围时，Python会报一个IndexError错误，所以，要确保索引不要越界，记得最后一个元素的索引是len(classmates)-1。

如果要取最后一个元素，除了计算索引位置外，还可以用-1做索引，直接获取最后一个元素：

classmates[-1]

以此类推，可以获取倒数第2个、倒数第3个：

classmates[-2]

classmates[-3]

classmates[-4]

list是一个可变的有序表，所以，可以往list中追加元素到末尾：

classmates.append('Adam')

也可以把元素插入到指定的位置，比如索引号为1的位置：

classmates.insert(1,'Jack')

要删除list末尾的元素，用pop()方法：

classmates.pop()

要删除指定位置的元素，用pop(i)方法，其中i是索引位置：

classmates.pop(1)

要把某个元素替换成别的元素，可以直接赋值给对应的索引位置：

classmates[1]='Sarah'

list里面的元素的数据类型也可以不同，比如：

L=['Apple',123,True]

list元素也可以是另一个list，比如：

s=['python','java',['asp','php'],'scheme']

要注意s只有4个元素，其中s[2]又是一个list，如果拆开写就更容易理解了：

如果一个list中一个元素也没有，就是一个空的list，它的长度为0：

L=[]

len(L)

###### tuple

另一种有序列表叫元组：tuple。tuple和list非常类似，但是tuple一旦初始化就不能修改，比如同样是列出同学的名字：

classmates=('Michael','Bob','Tracy')

现在，classmates这个tuple不能变了，它也没有append()，insert()这样的方法。其他获取元素的方法和list是一样的，可以正常地使用classmates[0]，classmates[-1]，但不能赋值成另外的元素。

不可变的tuple有什么意义？因为tuple不可变，所以代码更安全。如果可能，能用tuple代替list就尽量用tuple。

tuple的陷阱：当定义一个tuple时，在定义的时候，tuple的元素就必须被确定下来，比如：

t=(1,2)

t

如果要定义一个空的tuple，可以写成()：

t=()

t

但是，要定义一个只有1个元素的tuple，如果这么定义：

t=(1)

t

定义的不是tuple，是1这个数！这是因为括号()既可以表示tuple，又可以表示数学公式中的小括号，这就产生了歧义，因此，Python规定，这种情况下，按小括号进行计算，计算结果自然是1。

所以，只有1个元素的tuple定义时必须加一个逗号,，来消除歧义：

t=(1,)

Python在显示只有1个元素的tuple时，也会加一个逗号,，以免误解成数学计算意义上的括号。

最后来看一个“可变的”tuple：

t=('a','b',['A','B'])

t[2][0]='X'

t[2][1]='Y'

t

这个tuple定义的时候有3个元素，分别是'a'，'b'和一个list。不是说tuple一旦定义后就不可变了吗？怎么后来又变了？

表面上看，tuple的元素确实变了，但其实变的不是tuple的元素，而是list的元素。tuple一开始指向的list并没有改成别的list，所以，tuple所谓的“不变”是说，tuple的每个元素，指向永远不变。即指向'a'，就不能改成指向'b'，指向一个list，就不能改成指向其他对象，但指向的这个list本身是可变的！

##### dict和set

###### dict

Python内置了字典：dict的支持，dict全称dictionary，在其他语言中也称为map，使用键-值（key-value）存储，具有极快的查找速度。

举个例子，假设要根据同学的名字查找对应的成绩，如果用list实现，需要两个list：

names = ['Michael', 'Bob', 'Tracy']

scores = [95, 75, 85]

给定一个名字，要查找对应的成绩，就先要在names中找到对应的位置，再从scores取出对应的成绩，list越长，耗时越长。

如果用dict实现，只需要一个“名字”-“成绩”的对照表，直接根据名字查找成绩，无论这个表有多大，查找速度都不会变慢。用Python写一个dict如下：

d = {'Michael': 95, 'Bob': 75, 'Tracy': 85}

d['Michael']

为什么dict查找速度这么快？因为dict的实现原理和查字典是一样的。假设字典包含了1万个汉字，要查某一个字，一个办法是把字典从第一页往后翻，直到找到想要的字为止，这种方法就是在list中查找元素的方法，list越大，查找越慢。

第二种方法是先在字典的索引表里（比如部首表）查这个字对应的页码，然后直接翻到该页，找到这个字。无论找哪个字，这种查找速度都非常快，不会随着字典大小的增加而变慢。

dict就是第二种实现方式，给定一个名字，比如'Michael'，dict在内部就可以直接计算出Michael对应的存放成绩的“页码”，也就是95这个数字存放的内存地址，直接取出来，所以速度非常快。

可以猜到，这种key-value存储方式，在放进去的时候，必须根据key算出value的存放位置，这样，取的时候才能根据key直接拿到value。

把数据放入dict的方法，除了初始化时指定外，还可以通过key放入：

d['Adam'] = 67

d['Adam']

由于一个key只能对应一个value，所以，多次对一个key放入value，后面的值会把前面的值冲掉：

d['Jack'] = 90

d['Jack']

d['Jack'] = 88

d['Jack']

如果key不存在，dict就会报错：

d['Thomas']

要避免key不存在的错误，有两种办法，一是通过in判断key是否存在：

'Thomas' in d

二是通过dict提供的get()方法，如果key不存在，可以返回None，或者自己指定的value：

d.get('Thomas')

d.get('Thomas', -1)

要删除一个key，用pop(key)方法，对应的value也会从dict中删除：

d.pop('Bob')

请务必注意，dict内部存放的顺序和key放入的顺序是没有关系的。

和list比较，dict有以下几个特点：

查找和插入的速度极快，不会随着key的增加而变慢；

需要占用大量的内存，内存浪费多。

而list相反：

查找和插入的时间随着元素的增加而增加；

占用空间小，浪费内存很少。

所以，dict是用空间来换取时间的一种方法。

dict可以用在需要高速查找的很多地方，在Python代码中几乎无处不在，正确使用dict非常重要，需要牢记的第一条就是dict的key必须是不可变对象。

这是因为dict根据key来计算value的存储位置，如果每次计算相同的key得出的结果不同，那dict内部就完全混乱了。这个通过key计算位置的算法称为哈希算法（Hash）。

要保证hash的正确性，作为key的对象就不能变。在Python中，字符串、整数等都是不可变的，因此，可以放心地作为key。而list是可变的，就不能作为key：

key = [1, 2, 3]

d[key] = 'a list'

###### set

set和dict类似，也是一组key的集合，但不存储value。由于key不能重复，所以，在set中，没有重复的key。

要创建一个set，需要提供一个list作为输入集合：

s = set([1, 2, 3])

s

注意，传入的参数[1, 2, 3]是一个list，而显示的{1, 2, 3}只是告诉这个set内部有1，2，3这3个元素，显示的顺序也不表示set是有序的。。

重复元素在set中自动被过滤：

s = set([1, 1, 2, 2, 3, 3])

s

通过add(key)方法可以添加元素到set中，可以重复添加，但不会有效果：

s.add(4)

s

通过remove(key)方法可以删除元素：

s.remove(4)

s

set可以看成数学意义上的无序和无重复元素的集合，因此，两个set可以做数学意义上的交集、并集等操作：

s1 = set([1, 2, 3])

s2 = set([2, 3, 4])

s1 & s2

s1 | s2

set和dict的唯一区别仅在于没有存储对应的value，但是，set的原理和dict一样，所以，同样不可以放入可变对象，因为无法判断两个可变对象是否相等，也就无法保证set内部“不会有重复元素”。

##### 条件判断

计算机之所以能做很多自动化的任务，因为它可以自己做条件判断。

比如，输入用户年龄，根据年龄打印不同的内容，在Python程序中，用if语句实现：

age = 20

if age >= 18:

print('your age is', age)

print('adult')

根据Python的缩进规则，如果if语句判断是True，就把缩进的两行print语句执行了，否则，什么也不做。

也可以给if添加一个else语句，意思是，如果if判断是False，不要执行if的内容，去把else执行了：

age = 3

if age >= 18:

print('your age is', age)

print('adult')

else:

print('your age is', age)

print('teenager')

注意不要少写了冒号:。

当然上面的判断是很粗略的，完全可以用elif做更细致的判断：

age = 3

if age >= 18:

print('adult')

elif age >= 6:

print('teenager')

else:

print('kid')

elif是else if的缩写，完全可以有多个elif，所以if语句的完整形式就是：

if <条件判断1>:

<执行1>

elif <条件判断2>:

<执行2>

elif <条件判断3>:

<执行3>

else:

<执行4>

##### 循环

###### for...in

Python的循环有两种，一种是for...in循环，依次把list或tuple中的每个元素迭代出来，看例子：

names = ['Michael', 'Bob', 'Tracy']

for name in names:

print(name)

###### while

第二种循环是while循环，只要条件满足，就不断循环，条件不满足时退出循环。比如要计算100以内所有奇数之和，可以用while循环实现：

sum = 0

n = 99

while n > 0:

sum = sum + n

n = n - 2

print(sum)