

Python 2 基础知识点讲解

- 仅整理部分主要知识点，供辅导、串讲使用
- 未加入字典、集合、文件的操作，函数仅节选了一小部分
- 多多做题



如何学习?

- 实践出真知，不要只看不做，多动手

模仿→改写→编写

多动手，增加参与感

- 学会理解并通过实操记住代码等，不要死记硬背

多动手、多做题、多思考，理解并记忆

- 多思考，针对问题进行分析，利用计算机的语言解决问题

这是个什么问题？

我要如何解决？我解决的思路是什么，具体化？

如何使用我学过的代码将我的思路加以实现？

- 考试 → 机考，随机，只看结果（按测试用例给分）



书写规则

Python程序的书写规则如下：

- 1) 使用换行符分隔，一般情况下，**一行一条语句**。
- 2) 从第一列开始，**前面不能有任何空格**，否则会产生语法错误。
- 3) Python程序中所有的语法符号，都必须在**英文输入法**下输入，字符串中的符号除外。
- 4) 在Python中代码的**缩进**非常重要。
- 5) 以符号#开始，表示本行#之后的内容为注释。
- 6) 包含在一对**三引号**"..."或"..."之间且不属于任何语句的内容将被解释器认为是**注释**。



数据的输入

输入语句：input、raw_input

将用户输入的数据用一个变量来标识，使用如下的语句形式：

变量 = input ("提示字符串")

变量 = raw_input("提示字符串")

注意：数值等输入为input()；字符串输入为raw_input()

在Python2中，input()只能得到数值；在Python3中得到字符串

变量赋值时，自动确认数据类型

链式赋值x=y=z=input()

解包赋值x,y,z=input()

x,y=y,x

```
>>> a,b,c=1,1.0,"a"
>>> type(a)
<type 'int'>
>>> type(b)
<type 'float'>
>>> type(c)
<type 'str'>
```



输出语句：print

print语句以文本形式显示信息，所有提供的表达式都从左到右求值，结果值以从左到右的方式显示在输出行上。

默认情况下，print输出完所有提供的表达式之后会**自动换行**，如果希望print输出完数据后不换行，需加逗号。

- 直接输出：print a
- 同一行输出：print a,
- 换行：print、print “
- 末尾加空格print ‘ ’
- 特殊%d、%f、%s：print(“a=%d,b=%s,c=%.2f”%(a,b,c))



数据的输出

输出语句print的格式化输出： %d、 %f、 %s

%d: 数字; %f: 小数位数; %s: 字符串, 后面还会有format函数

```
#encoding=utf-8
a=raw_input()
b=input()
c=input()
print("字符串为：%s, 数字为：%d, 保留二位小数为：%.2f"%(a,b,c))
```

```
python
3.1415
3.1415
字符串为：python, 数字为：3, 保留二位小数为：3.14
```

%d输出整数, 无四舍五入

%f可以四舍五入

注意: 代码中有中文字符时, 提交可能报错, 需在代码前添加语句:

-*- coding:UTF-8 -*-或 **#encoding=utf8** 或 **#coding=utf-8** 等



数据类型

对象类型	类型名称	示例	简要说明
数字	int float complex	123456789 3.1415926 1+4j	基本类型，不可变，大小没有限制
字符串	str	'swfu' "I'm a student" """Python """	容器类型，不可变，可以使用单引号、双引号、三引号作为定界符
列表	list	[1, 2, 3] ['a', 'b', ['c', 2]]	容器类型，可变，所有元素放在一对方括号中，元素之间使用逗号分隔，其中的元素可以是任意类型
元组	tuple	(2, -5, 6)(3,)	容器类型，不可变，所有元素放在一对圆括号中，元素之间使用逗号分隔，如果元组中只有一个元素，后面必须多写一个逗号
集合	set	{'a', 'b', 'c'}	容器类型，可变，所有元素放在一对大括号中，元素之间使用逗号分隔，元素必须为不可变类型的数据，并且不允许重复
字典	dict	{1:'food', 2:'taste', 3:'import'}	容器类型，可变，所有元素放在一对大括号中，元素之间使用逗号分隔，元素形式为“键:值”，其中“键”不允许重复且必须不可变



标识符及命名规则

- 在 Python 中定义变量名、函数名或类名时，需要注意以下问题。
 - **必须**以汉字、字母或下画线开头。
 - **不能**包含空格或标点符号。
 - **不能**使用关键字。如 if、else、for、while、return 等不能作为变量名，也不能作为函数和类的名字。
 - 对英文字母的**大小写敏感**，如 score 和 Score 不是同一个变量。
 - **不建议**使用系统内置的模块名、类型名或函数名和已导入的模块名及其成员名，这会改变其类型和含义，甚至会导致其他代码无法正常执行。



标识符及命名规则

有一些标识符是Python本身的一部分，这些特殊的标识符被称为“关键字”或者是“保留字”，它们不能像普通标识符那样使用。Python关键字的完整列表可以用以下方式查看：

```
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
 'return', 'try', 'while', 'with', 'yield']
>>>
```



运算符与表达式

运算符	功能说明
+	算术加法，列表、元组、字符串合并与连接，正号
-	算术减法，集合差集，负号
*	算术乘法，列表、元组或字符串与整数相乘时表示序列重复
/	算术除法
//	求整商
%	求余数，结果的符号与除数相同
**	幂运算
<、<=、>、>=、==、!=	关系运算符
and	逻辑与
or	逻辑或
not	逻辑非
in	成员测试
is	同一性测试，测试两个对象是否引用同一个对象
&、 、^	集合交集、并集、对称差集



运算符

复合赋值运算符

所有二元运算符（+、-、*、/、//、%、**）都可以跟赋值运算符结合在一起，形成复合赋值运算符（+=、-=、*=、/=、//=、%=、**=），复合赋值运算符中间不可有空格。若a和b为操作数，则a += b，等价于a = a + b；a *= b，等价于a = a*b

```
>>> a,b=10,20
>>> a+=b
>>> print(a,b)
30 20
```

例题

问题：需要找钱给用户，现在只有50元，5元和1元的人民币若干张。输入一个整数金额值，给出找钱的方案，假设人民币足够多，且优先使用面额大的钱币

```
#encoding=utf8  
money=input("输入金额： ")  
m50=money/50  
money=money%50  
m5=money/5  
money=money%5  
m1=money  
print("50元面额需要的张数： %d"%m50)  
print("5元面额需要的张数： %d"%m5)  
print("1元面额需要的张数： %d"%m1)
```

```
输入金额： 857  
50元面额需要的张数： 17  
5元面额需要的张数： 1  
1元面额需要的张数： 2
```

程序流程控制



条件表达式

条件表达式主要为关系、逻辑运算，配合选择结构、循环结构等一起使用。

关系运算：>、>=、<、<=、==、<>、!=

逻辑运算：and、or、not

1) 假设有整数x，表示x为一个偶数。→ $x \% 2 == 0$

2) 假设有整数x，表示x是3的倍数且个位上数字为5。

$x \% 3 == 0$ and $x \% 10 == 5$

3) 假设三角形的3条边分别为a, b, c，表示a, b, c能构成一个三角形。

$(a+b>c)$ and $(b+c>a)$ and $(a+c>b)$ and $(a>0)$ and $(b>0)$ and $(c>0)$

4) 假设年份year，则表示year为闰年条件为：

$(year \% 4 == 0$ and $year \% 100 != 0)$ or $(year \% 400 == 0)$



选择结构

单分支结构: if 条件表达式:
 语句块

双分支结构: if 条件表达式:
 语句块1

else:
 语句块2

多分支结构: if 条件表达式1:
 语句块1
elif 条件表达式2:
 语句块2

...

else:
 语句块n

无论是if、if...else、if...elif...else语句，
都要注意英文冒号、缩进等问题。

若符合条件，返回True。

此外，还可以使用if的嵌套。

条件不要重合，避免结果有误。



选择结构-例题

问题：输入三角形三条边的边长，计算三角形的面积。

```
a=float(input("请输入三角形的边长a: "))
```

```
b=float(input("请输入三角形的边长b: "))
```

```
c=float(input("请输入三角形的边长c: "))
```

```
if (a+b>c) and (b+c>a) and (a+c>b) and (a>0) and (b>0) and (c>0):
```

```
    h=(a+b+c)/2
```

```
    area=(h*(h-a)*(h-b)*(h-c))**0.5
```

```
    print("三角形的面积为: %.2f"%(area))
```

```
else:
```

```
    print("不能构成三角形")
```

```
请输入三角形的边长a: 3
请输入三角形的边长b: 5
请输入三角形的边长c: 7
三角形的面积为: 6.50
```

```
请输入三角形的边长a: 2
请输入三角形的边长b: 3
请输入三角形的边长c: 5
不能构成三角形
```




选择结构-例题

问题：BMI=体重（千克）÷身高（米）²，计算BMI，给出评级。

```
height = input("请输入您的身高（米）：")
weight = input("请输入您的体重（千克）：")
BMI = weight/height/height
print("您的BMI指数是：%.1f"%(BMI))
if BMI < 18.5: print("您的体型偏瘦，要多吃多运动哦！")
elif 18.5 <= BMI < 24: print("您的体型正常，继续保持哟！")
elif 24 <= BMI < 28: print("您的体型偏胖，有发福迹象！")
elif 28 <= BMI < 32: print("不要悲伤，您是个迷人的胖子！")
else: print("什么也不说了，您照照镜子就知道了.....")
```

成人的BMI数值	
过轻	低于18.5
正常	18.5-23.9
过重	24-27.9
肥胖	28-32
过于肥胖	32以上



循环结构

当需要做一系列**重复操作**，且这些操作有规律、能说清时，可以使用循环。

for循环一般用于**循环次数可以提前确定**的情况，尤其适用于枚举或遍历序列或迭代对象中元素的场合。

while循环一般用于**循环次数难以提前确定**的情况，当然也可以用于循环次数确定的情况。

for和while很多时候可以替换使用；循环通常也会**配合选择结构**使用，可以**嵌套使用**；使用过程中也需要注意**英文冒号**、**缩进**等问题。

```
for 取值 in 序列:  
    语句块
```

```
while 条件表达式:  
    语句块
```



循环结构

for语句:

for 取值 in 序列:
语句块

for语句中的迭代: **range()**函数生成迭代序列

$\text{range}(n) \rightarrow \mathbf{0}, 1, 2, \dots, \mathbf{n-1}$ $\rightarrow n$ 小于或等于0时序列为空

$\text{range}(m,n) \rightarrow \mathbf{m}, m+1, \dots, \mathbf{n-1}$ \rightarrow 左闭右开

$\text{range}(m,n,d) \rightarrow \mathbf{m}, m+d, m+2d, \dots, \mathbf{\text{最接近但小于}n\text{的值}}$



循环结构

while语句:

```
while 条件表达式:  
    语句块
```

执行while语句的时候，先求条件表达式的值，如果值为**True**就**执行循环体语句块**一次，然后重复上述动作；当条件表达式的值为**False**的时候，while语句执行**结束**。



循环结构

break和continue语句:

当满足特定条件需要结束循环或者立刻转去做下一次循环时, 可以使用循环控制语句break和continue。

```
for i in range(1, 10+1):  
    if i%3==0:  
        break  
    print(i),
```

输出: 1 2

```
for i in range(1, 10+1):  
    if i%3==0:  
        continue  
    print(i),
```

输出: 1 2 4 5 7 8 10

此外, 当for、break、else组合使用时, else可以判断循环是否自然终止。



循环结构-例题

问题：求1至100中奇数和偶数的和分别是多少。

```
sum1=0
sum2=0
for i in range(1,101):
    if i%2==0:
        sum1+=i
    else:
        sum2+=i
print("偶数和=%d" %sum1)
print("奇数和=%d" %sum2)
```

设初始值，作为“记录”，例如需要加和时可以设一个变量为0，不断累加；需要判断大小可以利用初始变量作比较；需要计数可以设初始变量为0，每次符合条件加1；需要乘积可以设初始变量为1，不断相乘。



循环结构-例题

问题： 求非负数字序列中的最小值、最大值和平均值。用户输入-1表示序列终止。

```
count=0
sum=0
num=input()
max=min=num
while(num!=-1):
    count+=1
    sum+=num
    if(num<min):min=num
    if num>max: max=num
    num=input()
print("最大值: %d" %max)
print("最小值: %d" %min)
print("平均值: %.2f" %(1.0*sum/count))
```



循环结构-例题

问题：输入一个数，判断是否为素数。

```
n=input()
for i in range(2,a):
    if n%i==0:
        print 'no'
        break
    else:print 'yes'
```

```
n=input()
count=0
for i in range(2, n):
    if n%i==0:
        count+=1
if count==0:
    print("yes")
else:
    print("no")
```

一个大于1的自然数，除了1和它自身外，不能被其它整数整除的数叫做素数；否则称为合数。



循环结构-例题

问题：输出区间内所有素数（一行内输入和输出）；输出区间第n个素数，没有则输出NO。

```
m,n=input()
for i in range(m,n+1):
    for j in range(2,i):
        if i%j==0:
            break
    else:print i,
```

输出第n个、最大最小等，均可设初始值计数。

```
a,b,n=input()
c=0
for i in range(a,b+1):
    for j in range(2,i):
        if i%j==0:
            break
    else:
        c+=1
        if c==n:
            print i
            break
    else:print 'NO'
```



循环结构-例题

问题：n人搬n砖，每人：男人搬4块，女人搬3块，2小孩搬1块。

```
n=input()
for a in range(0,n+1):
    for b in range(0,n+1):
        for c in range(0,n+1):
            if a*4+b*3+c/2.0==n and a+b+c==n:
                print a,b,c
```



循环结构-例题

问题：聚餐有k人参加，吃饭共花了50先令，每人：男人花3先令，女人花2先令，小孩花1先令；各多少人？共有多少种可能？

```
k=input()
count=0
for man in range(0,k+1):
    for woman in range(0,k+1):
        for children in range(0,k+1):
            if man+woman+children==k and 3*man+2*woman+1*children==50:
                print("%d %d %d" %(man,woman,children))
                count+=1
if count==0:
    print("no result")
```

列表与元组



列表(list)

列表用来**有序**存放一组相关数据，以便进行统一的处理。通常将一组数据放在一对方括号“**[]**”中，定义列表。

列表可以为空列表，列表中可以放入各类数据，如数字、字符串等，列表中也可以包含列表。

列表中数据的个数为列表的**长度**，列表中的每个数据为列表的**元素**。列表元素有序排放，元素之间以逗号隔开，列表可变，可直接使用添加、删除元素、排序等操作。

列表中每个数据都对应**索引值**，可通过索引获取或修改数据。

例如：ls=[12,53,123,413,542]，正向索引为0、1、2、3、4

反向索引为-1、-2、-3、-4、-5



列表(list)

添加元素: `ls.append(元素)` → 在列表末尾添加一个元素, 括号内直接写**元素**

`ls.extend(列表)` → 用一个列表扩充另一个列表, 括号内是**列表**

`ls.insert(索引,元素)` → 插入元素, 括号内为**索引值**、**元素**

为方便查看, 我们定义一个列表为`ls=[0,1,2,3,4]` → 元素与索引对应

```
>>> ls=[0,1,2,3,4]
>>> ls.append(5)
>>> ls
[0, 1, 2, 3, 4, 5]
```

```
>>> ls=[0,1,2,3,4]
>>> ls.extend([3,4,5,6])
>>> ls
[0, 1, 2, 3, 4, 3, 4, 5, 6]
```

```
>>> ls=[0,1,2,3,4]
>>> ls.insert(3,500)
>>> ls
[0, 1, 2, 500, 3, 4]
>>> ls.insert(-2,'python')
>>> ls
[0, 1, 2, 500, 'python', 3, 4]
```



列表(list)

删除元素: `ls.pop(索引值)` → 将列表中指定元素“弹”出，括号内为索引值

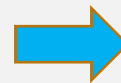
`ls.remove(元素)` → 删除列表中某一元素

`del ls[索引]`、`del ls` → 删除列表中某个/某些元素、删除列表

pop方法: 将列表中指定元素“弹”出来，括号内为索引值，可用其他变量“接住”弹出元素。当括号内不带参数时，默认弹出最后一个元素。

```
>>> ls=[11,22,33,44,55,66]
>>> ls.pop(3)
44
>>> ls
[11, 22, 33, 55, 66]
>>> ls.pop()
66
>>> ls
[11, 22, 33, 55]
>>> a=ls.pop()
>>> a
55
```

```
ls=input()
for i in range(len(ls)):
    a=ls.pop()
    print a,
```



```
[1,2,3,4]
4 3 2 1
```



列表(list)

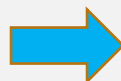
删除元素: `ls.pop(索引值)` → 将列表中指定元素“弹”出，括号内为索引值

`ls.remove(元素)` → 删除列表中某一元素

`del ls[索引]`、`del ls` → 删除列表中某个/某些元素、删除列表

remove方法: 直接删除列表中某一元素，无需知道位置，括号内为元素。如果指定的元素不存在于列表中，程序会报错。当列表中有多个待删除元素，默认先删除索引值最小的。

```
>>> ls=[1,2,4,6,5,4,8,9,4,0]
>>> ls.remove(2)
>>> ls
[1, 4, 6, 5, 4, 8, 9, 4, 0]
>>> ls.remove(4)
>>> ls
[1, 6, 5, 4, 8, 9, 4, 0]
```



```
>>> ls.remove(100)

Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
    ls.remove(100)
ValueError: list.remove(x): x not in list
```




列表(list)

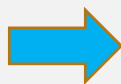
删除元素: `ls.pop(索引值)` → 将列表中指定元素“弹”出, 括号内为索引值

`ls.remove(元素)` → 删除列表中某一元素

`del ls[索引]`、`del ls` → 删除列表中某个/某些元素、删除列表

del语句: Python语句, 不是列表对象的方法, 也不是内置函数; 不仅可以用来删除列表中的某个(些)元素, 还可以直接删除整个列表。

```
>>> ls=[1,2,3,4,5,6,7,8,9,10]
>>> del ls[2]
>>> ls
[1, 2, 4, 5, 6, 7, 8, 9, 10]
>>> del ls[-1]
>>> ls
[1, 2, 4, 5, 6, 7, 8, 9]
>>> del ls[2:4]
>>> ls
[1, 2, 6, 7, 8, 9]
```



```
>>> del ls
>>> ls

Traceback (most recent call last):
  File "<pysHELL#45>", line 1, in <module>
    ls
NameError: name 'ls' is not defined
```



列表(list)

列表切片：应用索引值，操作列表中特定位置的元素

用一个**冒号**隔开两个索引值，**左边**是**开始位置**，**右边**是**结束位置**。这里要注意的一点是：**结束位置**上的元素是**不包含**的。→ **左闭右开**

如果**省略**了**开始位置**，Python会**从0**这个位置开始。

如果要得到从**指定索引值**到列表**末尾**的所有元素，把**结束位置**也**省去**即可。

如果什么都没有，**只有一个冒号**，Python将返回**整个列表**的拷贝。

列表切片**不改变列表**自身的组成结构和数据，它其实是为列表创建一个新的拷贝（副本）并返回。

列表切片操作实际上还可以接受**第三个参数**，其代表的是**步长**，默认值为1。



列表(list)

```
>>> ls=[1,2,3,4,5,6,7,8,9,10]
>>> ls[1:4]
[2, 3, 4]
>>> ls
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> a=ls[1:4]
>>> a
[2, 3, 4]
```

不改变列表、可用其他变量赋值

```
>>> ls=[1,2,3,4,5,6,7,8,9,10]
>>> ls2=[ls[0],ls[2],ls[5]]
>>> ls2
[1, 3, 6]
>>> ls3=ls[3:6]
>>> ls3
[4, 5, 6]
```

利用切片为其他列表赋值

```
>>> ls=[1,2,3,4,5,6,7,8,9,10]
>>> ls[:3]
[1, 2, 3]
>>> ls[5:]
[6, 7, 8, 9, 10]
>>> ls[:]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> ls[-5:]
[6, 7, 8, 9, 10]
>>> ls[::-1]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

可用正向/反向索引；不同省略方式的含义、第三个参数应用

```
>>> ls=[1,2,3,4,5,6,7,8,9,10]
>>> ls[2:5]
[3, 4, 5]
>>> ls[2:5]=["python",[12,25],999]
>>> ls
[1, 2, 'python', [12, 25], 999, 6, 7, 8, 9, 10]
```

利用切片改变列表自身元素



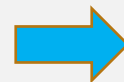
列表(list)

遍历列表：利用for循环、range函数，“从头到尾”访问列表元素。

```
ls=input()
for i in range(len(ls)):
    print(ls[i]),
```

此处i刚好对应索引，
通过索引值输出列表中的
每个元素。

```
ls=input()
for i in ls:
    print i,
```



```
[1, 2, 3]
1 2 3
```

此处i对应列表中的
的每一个元素，直接
输出。



列表(list)

常用操作:

len函数: `len(ls)` → 统计和返回指定列表的长度, 即列表中元素的个数

in运算 和 not in运算: 判断指定的元素是否在列表中

ls.count(): 统计某个元素在列表中出现的次数

ls.index(): 返回某个元素在列表中**第一次出现**的索引值

限定查找的范围: `ls.index(n,start,stop)`

```
>>> ls=[1,2,3,4,5,6,4,7,8,4,10]
>>> a=len(ls)
>>> a
11
>>> 2 in ls
True
>>> 9 in ls
False
```



```
>>> ls.count(4)
3
>>> ls.index(4)
3
>>> ls.index(4,4,7)
6
>>> ls.index(4,7,10)
9
```



列表(list)

常用操作:

`ls.reverse()`: 翻转整个列表, 改变列表本身

`ls.sort()`: 对列表元素从小到大进行排序 (元素均为数字), 改变列表本身

`ls.sort(reverse=True)`: 从大到小排序

`sorted(ls)`: 从小到大排序, 生成新列表, 不改变列表本身

`sorted(ls,reverse=True)`: 从大到小排序

```
>>> ls=[2,8,3,1,5,4,7,9]
>>> ls.reverse()
>>> ls
[9, 7, 4, 5, 1, 3, 8, 2]
>>> ls.sort()
>>> ls
[1, 2, 3, 4, 5, 7, 8, 9]
>>> ls.sort(reverse=True)
>>> ls
[9, 8, 7, 5, 4, 3, 2, 1]
```

```
>>> ls=[2,8,3,1,5,4,7,9]
>>> sorted(ls)
[1, 2, 3, 4, 5, 7, 8, 9]
>>> ls
[2, 8, 3, 1, 5, 4, 7, 9]
>>> a=sorted(ls)
>>> a
[1, 2, 3, 4, 5, 7, 8, 9]
>>> b=sorted(ls,reverse=True)
>>> b
[9, 8, 7, 5, 4, 3, 2, 1]
```



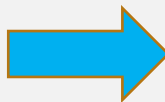
列表(list)

常用操作:

“+” 运算: 将多个列表对象合并在一起生成新列表, 类似extend()方法的效果, 但**不改变**参与运算的列表本身

“*” 运算, 通过重复指定遍数扩充列表长度

```
>>> ls1=[0,1,2,3]
>>> ls2=[1,2,3]
>>> ls3=[9,7]
>>> ls1+ls2+ls3
[0, 1, 2, 3, 1, 2, 3, 9, 7]
>>> ls1
[0, 1, 2, 3]
>>> ls4=ls1+ls2+ls3
>>> ls4
[0, 1, 2, 3, 1, 2, 3, 9, 7]
```



```
>>> ls3*3
[9, 7, 9, 7, 9, 7]
>>> ls3
[9, 7]
>>> ls5=ls3*3
>>> ls5
[9, 7, 9, 7, 9, 7]
```



列表(list)

常用操作:

输入列表: `input()`

转换列表: `list()` → 需要赋值给变量

`min`函数: 求数值列表中的最小值元素

`max`函数: 求数值列表中的最大值元素

`sum`函数: 求数值列表中元素之和

```
>>> t=1,2,3,4,5
>>> t
(1, 2, 3, 4, 5)
>>> type(t)
<type 'tuple'>
>>> list(t)
[1, 2, 3, 4, 5]
>>> t
(1, 2, 3, 4, 5)
>>> ls=list(t)
>>> ls
[1, 2, 3, 4, 5]
```



```
>>> min(ls)
1
>>> max(ls)
5
>>> sum(ls)
15
>>> ls
[1, 2, 3, 4, 5]
```




元组(tuple)

元组：戴上了“枷锁”的列表，与列表类似，也是用来存放一组相关的数据。

不同之处主要有两点：

元组使用**圆括号**（ ），列表使用方括号[]；

元组的元素**不能修改**

元组与列表之间的转换：tuple()、list()

元组是“不能修改”的列表，因此列表中**不涉及元素修改**的操作都适用于元组。



列表与元组-例题

问题： 逆序输出列表。

```
ls=input()
for i in range(len(ls)):
    n=ls.pop()
    print n,
```

```
ls=input()
ls.reverse()
for i in ls:
    print i,
```

```
ls=input()
for i in ls[::-1]:
    print i,
```

方法不唯一，重要的是理解和掌握各类函数和操作、多加思考



列表与元组-例题

问题：输入n，并输入n个数字，输出其中最大的数字。

```
n=input()
list=[]
for i in range(n):
    a=input()
    list.append(a)
    b=max(list)
print("max=%d"%b)
```

```
4
1
6
3
2
max=6
```

利用循环，实现指定次数输出
善用列表，直接输出最大值



列表与元组-例题

问题：输入一组数据，计算均值，保留一位小数，并输出大于均值的数。

```
t=input()
ls=list(t)
ave=sum(ls)/(len(ls)*1.0)
print("ave=%.1f"%ave)
for i in ls:
    if i>ave:print i,
```

```
8,3,1,6,2,5,1,3,2
ave=3.4
8 6 5
```

稍微综合一点，输入数字用空格间隔时，需通过字符串输入操作。



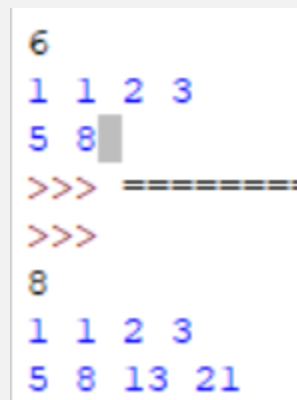
列表与元组-例题

问题：输入数字n，以4个数字为一行，输出斐波那契数列的前n项。

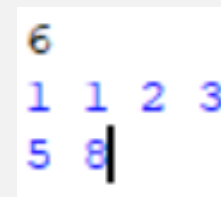
斐波那契数列：1，1，2，3，5，8，13，.....

```
n=input()
count=0
list=[1,1]
for i in range(2,n):
    list.append(list[i-2]+list[i-1])
for j in list:
    print j,
    count+=1
    if count%4==0:print"
    if count%4!=0:print"
```

根据规律，通过初始列表和循环自行制造斐波那契数列；4个数字为一行输出，可以设置变量计数，人为换行。若期望输出数字后有空格，则最终需额外输入。



```
6
1 1 2 3
5 8
>>> -----
>>>
8
1 1 2 3
5 8 13 21
```



```
6
1 1 2 3
5 8
```

无最后一行代码时，
最后一个数字无空格。

字符串



字符串(str)

使用单引号或双引号括起来的内容，称为字符串类型数据。定义方式如下：

- 1) 单引号（' '），其中可以包含双引号。
- 2) 双引号（" "），其中可以包含单引号。
- 3) 三单引号（''' '''），其中可以包含单引号和双引号，可以跨行。
- 4) 三双引号（""" """），其中可以包含单引号和双引号，可以跨行。

字符串输入：**raw_input()**

字符串**不可变**。

```
>>> s1='a'
>>> s2="b"
>>> s3='''c'''
>>> s4="""d"""
>>> s1,s2,s3,s4
('a', 'b', 'c', 'd')
```



字符串(str)

字符串的索引与切片：正向递增序号和反向递减序号。

正向递增序号

0	1	2	3	4	5	6	7	8	9	10	11	12
人	生	苦	短	,	我	学	P	y	t	h	o	n
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

反向递减序号

字符串的索引与切片与列表类似，掌握列表操作，字符串基本可以直接操作。



字符串(str)

字符串的索引与切片与列表类似，掌握列表操作，字符串基本可以直接操作。

```
>>> s='Hello Mike'
>>> print(s[0:5:1])
Hello
>>> print(s[0:6:2])
Hlo
>>> print(s[0:6:-1])

>>> print(s[4:0:-1])
olle
>>> print(s[::-1])
ekiM olleH
>>> print(s[::-3])
eMlH
```

1. s[0:5:1] 正向取
2. s[0:6:2] 正向取，间隔一个字符取
3. s[0:6:-1] 反向取，但是头下标小于尾下标无法反向取，因此输出为空
4. s[4:0:-1] 反向取，索引值为0的字符无法取到
5. s[4::-1] 反向取，从索引值为4的字符依次取到开头字符
6. s[::-1] 反向取整串
7. s[::-3] 反向取，间隔两个字符取



字符串(str)

字符串常用操作：与列表类似

“+” 运算：字符串拼接，例如 “python”+“100”结果为 “python100”

“*” 运算：字符串复制，例如 “python”*3，结果为 “pythonpythonpython”

in运算：判断是否为子串，例如 “H” in “Hello”结果为True；

“h” in “Hello”结果为False。

len函数：len(s) → 返回字符串长度，即字符个数

注意：字符串对象是**不可变**的，字符串对象提供的涉及字符串“修改”的方法都是**返回修改之后的新字符串**，并不对原字符串做任何修改。



字符串(str)

字符串常用操作：查找：find()、rfind()、index()、rindex()、count()

- find()和rfind(): 分别用来查找一个字符串在另一个字符串指定范围（默认是整个字符串）中**首次**和**最后一次**出现的位置，如果**不存在则返回-1**。
- index()和rindex(): 分别用来查找一个字符串在另一个字符串指定范围（默认是整个字符串）中**首次**和**最后一次**出现的位置，如果**不存在则抛出异常**。
- count(): 返回一个字符串在另一个字符串中出现的**次数**，如果**不存在则返回0**。

```
>>> s='hello world'
>>> s.find("l")
2
>>> s.rfind("l")
9
>>> s.index("l")
2
>>> s.rindex("l")
9
```

```
>>> s.index("hello")
0
>>> s.count("l")
3
>>> s1="l"
>>> s.index(s1)
2
```

括号里需要填字符串，若变量为字符串，也可以填入变量。



字符串(str)

字符串常用操作：分隔：split()、rsplit()、partition()、rpartition()

split()：以指定字符为分隔符，从原字符串**左端开始**将其分隔成多个字符串，并**返回**包含分隔结果的**列表**。**默认**按**空白符号**分隔字符串，如空格，换行符，制表符等。

```
>>> s='0 1 2,3 4,5,8 9'
>>> s.split()
['0', '1', '2,3', '4,5,8', '9']
>>> s.split(",")
['0 1 2', '3 4', '5', '8 9']
>>> s.split(",",1)
['0 1 2', '3 4,5,8 9']
>>> s.rsplit(",",1)
['0 1 2,3 4,5', '8 9']
```

默认将整个字符串按分隔符分隔，可填入第二个参数，分隔成特定几部分。

rsplit()为从**右端开始**分隔。



字符串(str)

字符串常用操作：分隔：split()、rsplit()、partition()、rpartition()

partition(): 基于分隔符的字符串拆分为具有**三个字符串**的**元组**。第一个字符串是分隔符之前的部分，第二个字符串是分隔符，第三个字符串是分隔符之后的部分。

```
>>> s='0 1 2,3 4,5,8 9'
>>> s.partition(",")
('0 1 2', ',', '3 4,5,8 9')
>>> s.rpartition(',')
('0 1 2,3 4,5', ',', '8 9')
```

rpartition()为从**右端开始**分隔。

用法：字符串变量.split()

字符串变量.rsplit()

字符串变量.partition()

字符串变量.rpartition()



字符串(str)

字符串常用操作：连接：join()

join(): 字符串的join()方法用来将列表中多个字符串进行连接，并在相邻两个字符串之间插入指定字符，返回新字符串。

```
>>> s='0 1 2 3 4'
>>> ls=s.split()
>>> ls
['0', '1', '2', '3', '4']
>>> ss=', '.join(ls)
>>> ss
'0,1,2,3,4'
```

注意：用法为：“分隔符”.join(列表)

与split()对应



字符串(str)

字符串常用操作：大小写：lower()、upper()、capitalize()、title()、swapcase()

```
>>> s = "I have two big eyes."
>>> s.lower()          # 返回小写字符串
'i have two big eyes.'
>>> s.upper()          # 返回大写字符串
'I HAVE TWO BIG EYES.'
>>> s.capitalize()    # 首字母大写
'I have two big eyes.'
>>> s.title()          # 每个单词首字母大写
'I Have Two Big Eyes.'
>>> s.swapcase()       # 大小写互换
'i HAVE TWO BIG EYES.'
```

注意：不改变原字符串
返回新的字符串



字符串(str)

字符串常用操作：替换：replace()

replace(): 替换字符串中指定字符或子字符串，**每次**只能**替换一个**字符或子串，类似于Word，记事本等文本编辑器的查找和替换功能。该方法**不修改原字符串**，而是**返回**一个**新字符串**。

```
>>> s="你是我的小呀小苹果"
>>> s1=s.replace("小","small")
>>> print s1
你是我的small呀small苹果
```




字符串(str)

字符串常用操作：删除两端字符：strip()、rstrip()、lstrip()

```
>>> s = "   abc   "
>>> s.strip()           # 删除两端空白字符
'abc'
>>> s.rstrip()          # 删除右端空白字符
'   abc'
>>> s.lstrip()           # 删除左端空白字符
'abc   '
>>> s = "====Mike===="
>>> s.strip('=')        # 删除两端指定字符"="
'Mike'
```

注意：不改变原字符串
返回新的字符串



字符串(str)

字符串常用操作：

判断字符串是否以指定字符开始或结束：startswith()、endswith()

```
>>> s = "Python程序设计.py"
>>> s.startswith("Python") # 检测字符串是否以“Python”开始
True
>>> s.endswith("py")      # 检测字符串是否以“py”结束
True
```

可配合if使用



字符串(str)

字符串常用操作:

判断字符串类型: isupper()、islower()、isdigit()、isalnum()、isalpha()

```
>>> s='years'
>>> s.islower()    # 判断字符串是否为全小写
True
>>> s="YEARS"
>>> s.isupper()    # 判断字符串是否为全大写
True
>>> s="20100405"
>>> s.isdigit()    # 判断字符串是否为全数字
True
>>> s="He is 10 years old"    # 字符串s中包含数字字母和空格
>>> s=s.replace(" ", "")    # 去除字符串中的空格
>>> s.isalnum()    # 判断字符串是否为数字或字母
True
>>> s.isalpha()    # 判断字符串是否为全字母
False
```

可配合if使用



字符串(str)

字符串常用操作：**format**格式化 → 以往：**%s**

从Python2.6 开始，新增了一种格式化字符串的方法str.format()，它增强了字符串格式化的功能，基本语法是通过“{}”和“:”来代替以前的“%”。

format方法可以有多个输出项，位置可以按指定顺序。

```
>>> "I am {0}, from {1}".format("xiaowang", "AI")
'I am xiaowang, from AI'
>>> "I am {1}, from {0}".format("xiaowang", "AI")
'I am AI, from xiaowang'
```



字符串-例题

问题：输入两个数字，计算多项式的和，例如：输入1 4，计算 $1+11+111+1111$ 。

```
s=raw_input()
ls=s.split()
a,n=ls[0],int(ls[1])
c=0
for i in range(n):
    c+=int(a*(i+1))
print c
```

```
1 4
1234
>>> ==
>>>
2 5
24690
```

验算：

$$1+11+111+1111=1234$$

$$2+22+222+2222+22222=24690$$



字符串-例题

问题：字符判断，输入一行字符，判断是否含元音字母或数字。

```
a=raw_input()
b=a.lower()
for i in b:
    if i=='a' or i=='e' or i=='i' or i=='o' or i=='u' or i.isdigit():
        print'yes'
        break
else:print'no'
```

```
Axzvnoef214vda
yes
>>> =====
>>>
csfk1sdBNC
no
```

函数

函数用来**将复杂的问题分解为若干子问题**，并逐个解决。函数允许我们按照这样的方式编写或阅读程序：首先关注所有的任务，然后关注如何完成每项任务。一旦我们为一个特定任务编写了函数，就可以在任意时刻调用这个函数来完成这个特定的任务。

使用函数有很多优点：

- 1) 减少程序中的代码重复量。
- 2) 把大而复杂的问题分解成小问题。
- 3) 有助于提升代码的整洁度，使代码更易于理解。
- 4) 使一段代码可以重复使用多次。

Python的乐高积木，函数就是把代码打包成不同形状的乐高积木，以便可以发挥想象力进行随意拼装和反复使用。用的时候很方便，根本不需要去想实现的原理
例如：round()函数可以用来四舍五入。

自建函数：def

具体地：def 函数名称(形式参数):

语句块

(return)

用来完成某些操作时，直接写入操作即可；

用来返回某一个值时，return即可，返回多个值时，生成元组。

自建函数构建形参与函数外的实参不冲突，与不同函数间的形参不冲突。

函数

默认值参数：如果希望函数的一些参数是可选的，可以在声明函数时为这些参数指定默认值。调用该函数时，如果没有传入对应的实参值，则函数使用声明时指定的默认参数值。

```
>>> def babble(words, times=1):  
    print((words+" ")*times)
```

```
>>> babble('hello',3)  
hello hello hello  
>>> babble('Tiger')  
Tiger
```

默认值参数必须写在形参列表的右边

函数

作用域：

局部变量作用域在函数内部，全局变量作用域在函数外，全局可见。

全局变量和局部变量重名时，局部变量优先。

global声明将使用全局变量。

```
def f():  
    x=5  
    print("f内部: x=",x)  
    return x*x  
x=10  
print("f()=",f())  
print("f外部: x=",x)
```

f内部: x= 5
f()= 25
f外部: x= 10

```
def f():  
    global x  
    x=5  
    print("f内部: x=",x)  
    return x*x  
x=10  
print("f()=",f())  
print("f外部: x=",x)
```

访问全局，将全局变量x重新赋值

f内部: x= 5
f()= 25
f外部: x= 5



函数-例题

问题：输出区间内所有素数。

```
def su(n):  
    for i in range(2,n):  
        if n%i==0:  
            break  
    else:  
        print n,  
m,n=input()  
for i in range(m,n+1):  
    su(i)
```

面对不同的要求，要学会思考和分析需要做什么，构建一个什么样的函数，构建的函数可以帮助我们做什么。

例如本例函数直接判断素数并输出，因此输入区间后直接循环使用函数即可。若需要对素数进行一定操作后再输出，则函数需要做出调整。

2,100

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97



函数-例题

问题：输出区间内所有素数，4个为一行。

```
def su(n):  
    for i in range(2,n):  
        if n%i==0:  
            return False  
    else:  
        return True
```



```
c=0  
m,n=input()  
for i in range(m,n+1):  
    if su(i)==True:  
        c+=1  
        print i,  
        if c%4==0:  
            print  
if c%4!=0:print"
```



```
2,100  
2 3 5 7  
11 13 17 19  
23 29 31 37  
41 43 47 53  
59 61 67 71  
73 79 83 89  
97
```

```
2,100  
2 3 5 7  
11 13 17 19  
23 29 31 37  
41 43 47 53  
59 61 67 71  
73 79 83 89  
97
```

当最后无空格时，可删除最后一行



函数-例题

问题：输入平时成绩、期末成绩、权重，计算总分，默认权重3:7，保留两位小数。

```
def fun(a,b,rate=0.3):  
    score=a*rate+b*(1-rate)  
    return score  
a,b,c=input()  
print("总评成绩: %.2f" %fun(a,b,c))
```

```
95,98,0.3  
总评成绩: 97.10  
>>> =====  
>>>  
95,98,0.4  
总评成绩: 96.80  
>>> =====  
>>>  
95,98,0.5  
总评成绩: 96.50
```



函数-例题

问题：编写函数，提取短语的首字母缩略词。缩略词是一个单词，是从短语中每一个单词取首字母组合而成的，且要求大写。例如：“very important person”的缩略词是“VIP”。

```
def sx(s1):  
    ls=s1.split()  
    s2=""  
    for i in ls:  
        s2+=i[0].upper()  
    return s2  
s1=raw_input()  
print sx(s1)
```

```
very important person  
VIP  
>>> =====  
>>>  
capital asset pricing model  
CAPM
```