流程控制语句

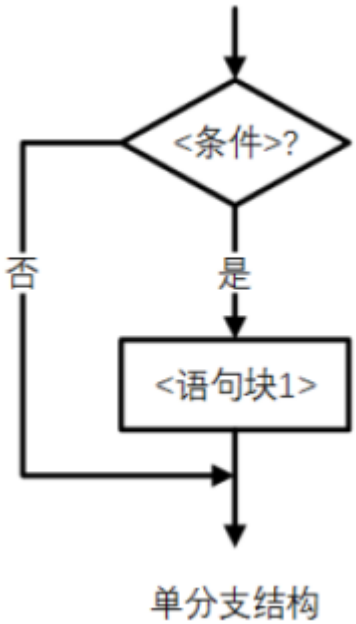
In [17]:
```
1 %pwd
```
executed in 9ms, finished 13:55:53 2019-04-29

Out[17]: 'D:\\讲师'

1 选择语句

1.1 if 语句

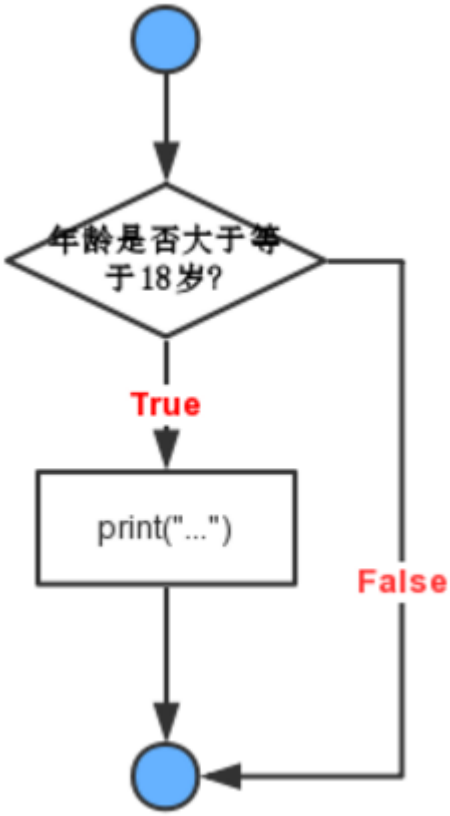
```
if 条件:
    条件为真（True）执行操作
```



单分支结构

In [5]:

```
age = 10

if age >= 18:
    print('your age is', age)
    print('adult')
```



1.2 练习1

先定义一个字符串作为密码。
再让用户输入密码，赋值到password。

- 使用if语句判断输入的是否和密码一致，如果是则打印以下语句："密码正确，恭喜"。
- 如果输入错误则结束if语句。

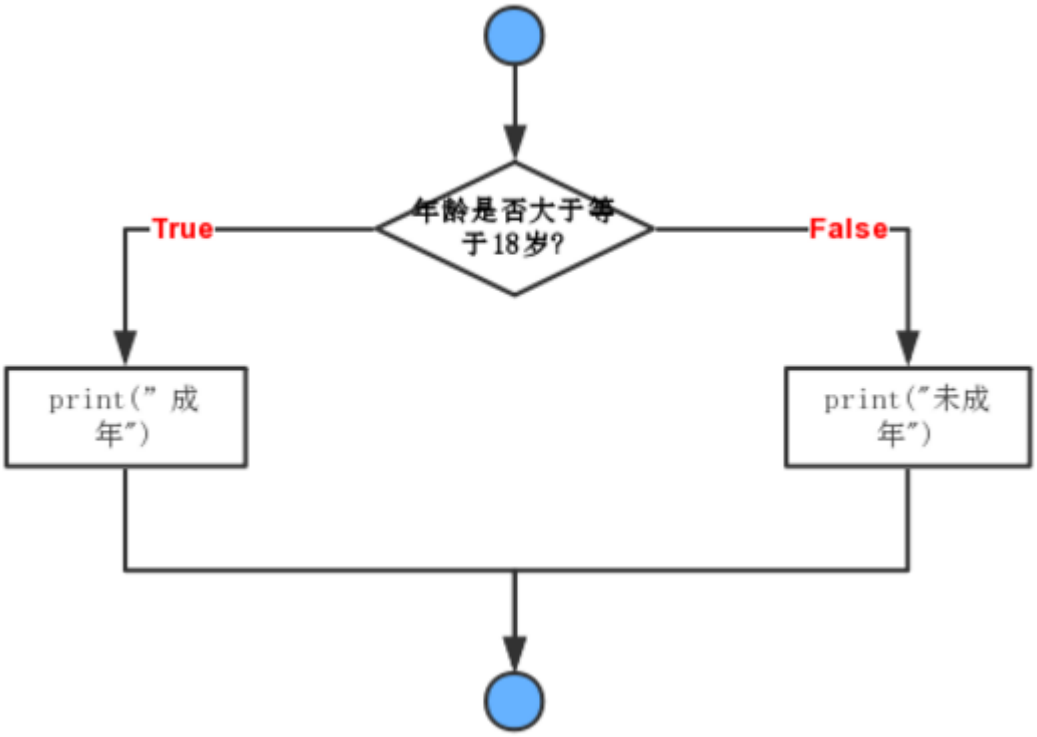
In [1]:

```
1
```

```
请设置你的密码：123
请输入你的密码：123
密码正确，恭喜
```

1.3 if- else语句

```
if 条件:
    条件为真（True）执行操作
else:
    条件为假（False）执行操作
```



In [10]:

```
1  age = 25
2
3  if age >= 18:
4      print('your age is', age)
5      print('adult')
6  else:
7      print('your age is', age)
8      print('teenager')
```

```
your age is 25
adult
```

1.4 练习题

1.4.1 练习2

先定义一个密码，再用input用户输入，使用if语句判断用户输入的是否和密码一致。

- 如果一致则打印以下语句："密码正确，恭喜"。
- 如果输入错误则打印："密码错误"。

In [2]:

```
1
```

```
请设置你的密码：123
请输入你的密码：31
密码错误
```

1.4.2 练习3

看看这个代码错在哪里?
提示一下：关于字符串和整型数字之间的矛盾。大家来解释下为什么会出这样的错误!

In [1]:

```
birth = input('你的出生年份是：')

if birth < 2000:
    print('00前')
else:
    print('00后')
```

```
你的出生年份是：1991
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-fa5b662eebe7> in <module>()
      1 birth = input('你的出生年份是：')
----> 2 if birth < 2000:
      3     print('00前')
      4 else:
      5     print('00后')

TypeError: '<' not supported between instances of 'str' and 'int'
```

如何修改上面的报错，使得if-elif语句能正常运行?

In [3]:

```
birth = int(input('你的出生年份是：'))

if birth < 2000:
    print('00前')
else:
    print('00后')
```

```
你的出生年份是：2019
00后
```

1.4.3 练习4

要求：从键盘获取自己的年龄，判断是否大于或者等于18岁。

- 如果满足条件就输出“成年了，可以去网吧了”
- 如果不满足条件，则输出“未成年，不能入内”

提示：

- 使用input从键盘中获取数据，并且赋值到一个变量中
- 使用if语句，来判断 age>=18是否成立

In [5]:

```

```

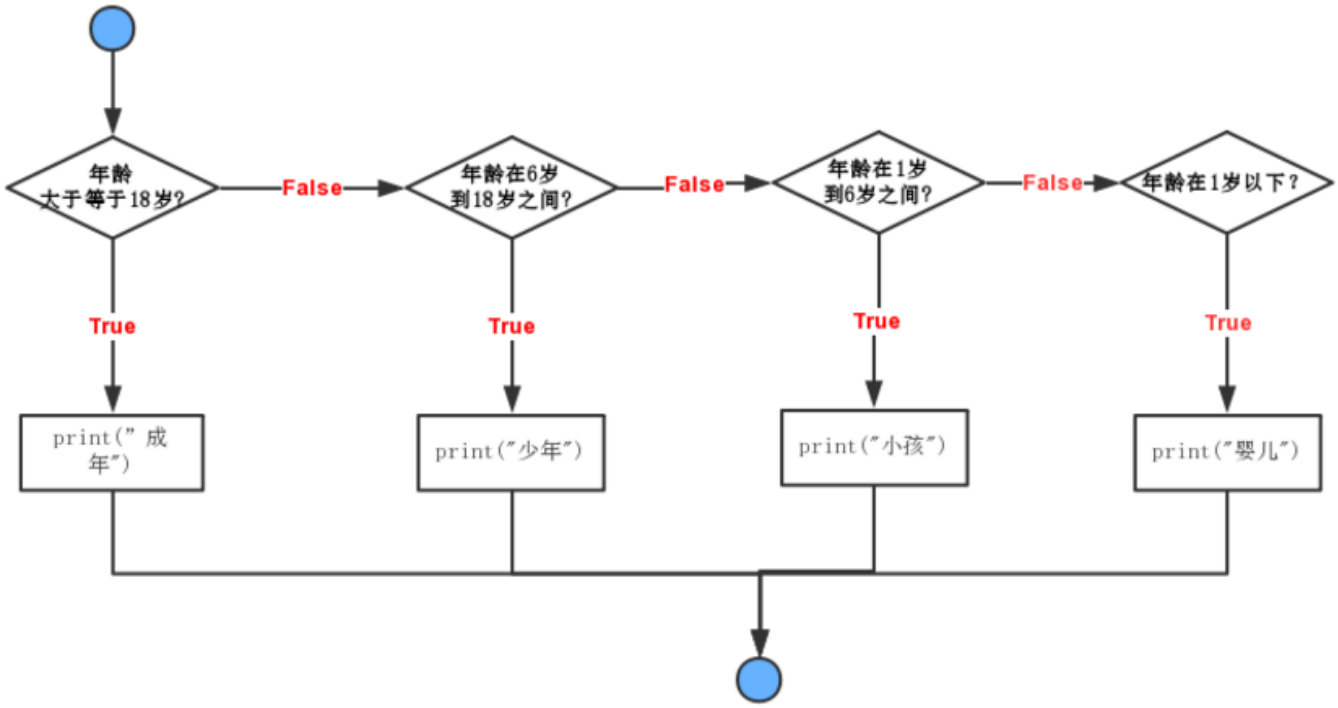
```
你的年龄是：20
成年了，可以去网吧了
```

1.5 if - elif - else语句

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>
```

- if语句执行有个特点，它是从上往下判断
- 如果在某个判断上是True，把该判断对应的语句执行后，就忽略掉剩下的elif和else
- 所以，请测试并解释为什么下面的程序打印的是“少年”：



In [4]:

```
age = 15

if age >= 18:
    print('成人')
elif age >= 6:
    print('少年')
elif age >= 5:
    print('小孩')
else:
    print('婴儿')
```

少年

In [6]:

```
age = 16

if age >= 18:
    print('成人')
elif age >= 13:
    print('少年')
else:
    print('小孩')
```

少年

1.6 练习题

1.6.1 练习5

- 输入自己的身高和体重，根据BMI公式计算自己的BMI指数：体质指数（BMI）=体重（kg）÷身高^2（m）
- 并根据BMI指数判断自己体型：

低于18.5：过轻
18.5-25：正常
25-28：过重
28-32：肥胖
高于32：严重肥胖
用if-elif-else语句判断并打印结果：

In [7]:
```
1
```

```
请输入你的体重（kg）：68
请输入你的身高（m）：1.68
正常
```

In [8]:
```
1
```

```
请输入你的体重（kg）：68
请输入你的身高（m）：1.68
正常
```

1.6.2 练习6

从键盘输入一个整数，判断：

- 该整数是否能被 2 和 3 同时整除?
- 是否能被 2 整除?
- 是否能被 3 整除?
- 是否不能被 2 或 3 整除?
  最终输出相应信息。

In [9]:
```
1
```

```
请输入一个整数：5
不能被 2 或 3 整除
```

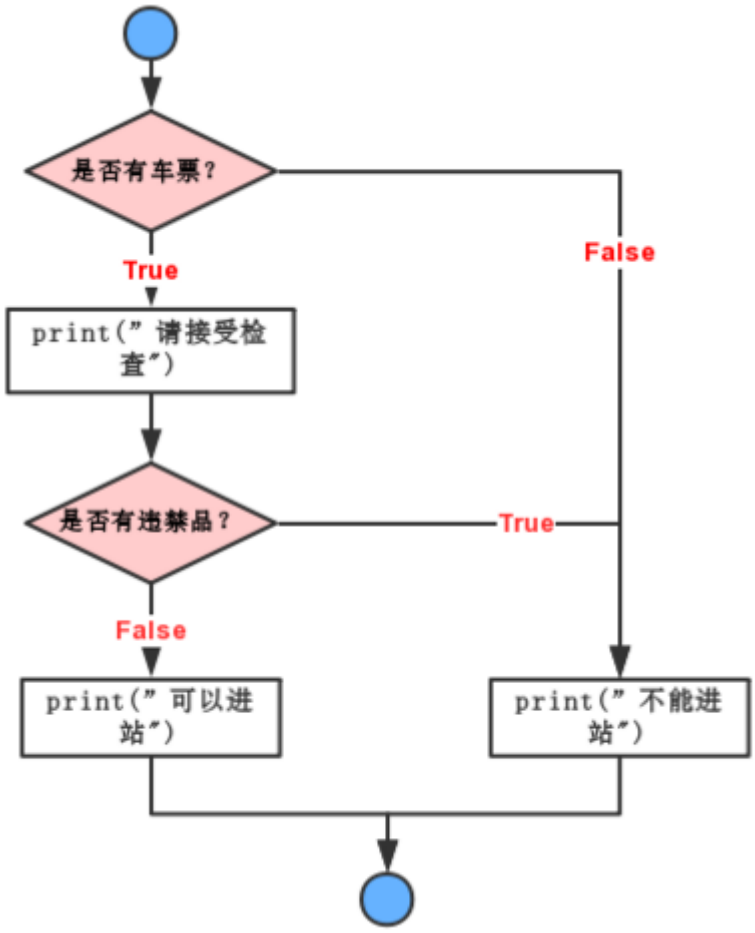
1.7 if 嵌套

- 通过学习if的基本用法，已经知道了
  - 当需要满足条件去做事情的这种情况需要使用if

    - 当满足条件时做事情A，不满足条件做事情B的这种情况使用if-else
- 想一想：坐火车或者地铁的实际情况是：
    - 先检查是否有车票之后，才会进行安检，
    - 即实际的情况某个判断是再另外一个判断成立的基础上进行的，这样的情况该怎样解决呢？

if嵌套的格式

```
if 条件1:

        满足条件1 做的事情1
        满足条件1 做的事情2
        ...(省略)...

        if 条件2:
            满足条件2 做的事情1
            满足条件2 做的事情2
            ...(省略)...
```



In [32]:

```
Ticket= int(input("是否有车票？如果有则填写“1”，没有则填写“0”："))

if Ticket == 1:
    print("有车票，请接受检查")
    contraband = int(input("是否有带违禁品？如果有则填写“1”，没有则填写“0”："))

    if contraband == 1:
        print("不能进站")
    elif contraband == 0:
        print("可以进站")

elif Ticket == 0:
    print("不能进站")
```

```
是否有车票？如果有则填写“1”，没有则填写“0”：0
不能进站
```

上面的小测试并不完美，仅作为if嵌套的简单例子。

1.8 练习7

- 情节描述：上公交车，并且可以有座位坐下。
- 要求：输入公交卡当前的余额，只要超过2元，就可以上公交车；如果空座位的数量大于0，就可以坐下。

In [10]:

```
```

```
请输入公交卡余额：3
余额充足，请上车
请输入空座位数量：0
没空座位，站着吧！
```

1.9 练习8

用if嵌套的方法实现以下功能：

- 输入一个整数，判断是否为8，如果是，则打印：“对啦~”
- 如果不是，则继续判断：
  - 输入的整数如果大于8，则打印：“比8大了”
  - 输入的整数如果小于8，则打印：“比8小了”

In [11]:

```
```

```
请输入一个整数：9
比8大了
```

1.10 条件表达式

In [2]:
```
a=10
b=5

if a>b:
    r="a更大"
else:
    r="b更大"

r
```

Out[2]: 'a更大'

上面的代码可以用条件表达式简化如下：

In [ ]:
```
a=10
b=5

r="a更大" if a>b else "b更大"

r
```

从“r="a更大" if a>b else "b更大""可以看到，先计算中间的条件（a>b），如果结果为True，返回if语句左边的值，否则返回else右边的值。

2 循环语句

循环语句的使用场景

这一天老师让你抄写一百遍"我错了"

In [ ]:
```
print("我错了")
print("我错了")
print("我错了")
print("我错了")
print("我错了")
print("......")
```

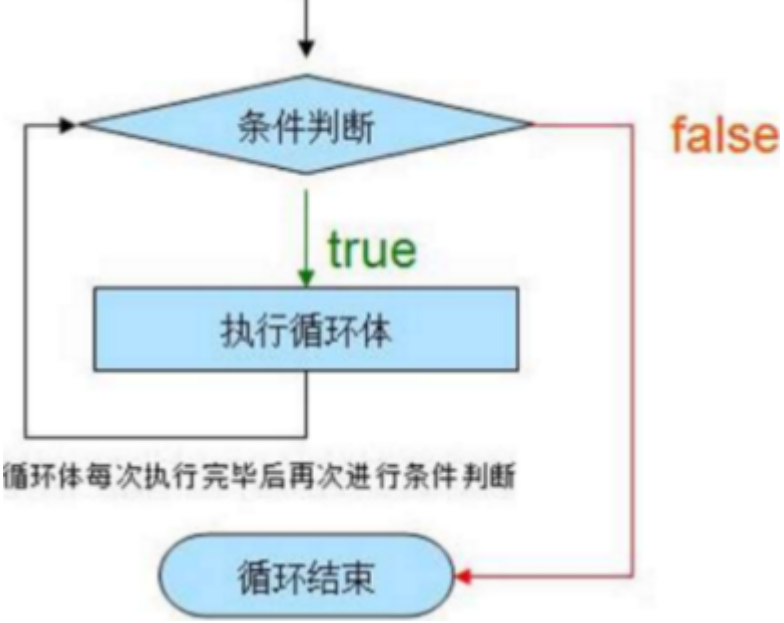
In [9]:
```
i = 1

while i <= 10:
    print("第", i, "次,", "我错了")
    i += 1
```

```
第 1 次, 我错了
第 2 次, 我错了
第 3 次, 我错了
第 4 次, 我错了
第 5 次, 我错了
第 6 次, 我错了
第 7 次, 我错了
第 8 次, 我错了
第 9 次, 我错了
第 10 次, 我错了
```

2.1 while循环

只要while后面的条件判断为True，那么就会一直执行循环体内的语句。

```
while 条件:
        条件满足时，做的事情1
        条件满足时，做的事情2
        条件满足时，做的事情3
        ...(省略)...
```



例1：将0到5赋值到i，打印出来并表明该次是第几次循环。

In [25]:

```
i = 1

while i<5:
    print("当前是第%d次执行循环"%(i))   #字符串的格式化
    print("i=%d"%i)
    i+=1   #加法赋值
```

```
当前是第1次执行循环
i=1
```

while循环和if语句有什么不同？将上面的while替换成if，观察结果：

In [16]:

```
i = 0

if i<5:
    print("当前是第%d次执行循环"%(i+1))
    print("i=%d"%i)
    i+=1
```

```
当前是第1次执行循环
i=0
```

例2：计算1~100的累积和（包含1和100）

In [21]:

```
i = 1
sum = 0

while i<=100:
    sum += i
    i += 1

print("1~100的累积和为:{}".format(sum))
```

```
1~100的累积和为:5050
```

2.1.1 练习9

改进上述代码，求出1~100之间偶数的累积和：

In [12]:
```
1
```

```
1~100的累积和为:2550
```

2.1.2 练习10

新建一个空列表，用while循环的方法，将整数1到20作为元素，插入到空列表中。

In [14]:
```
1
```

Out[14]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

2.1.3 while循环嵌套

使用while循环嵌套打印出九九乘法表：

In [20]:
```
1  i=1          #i是行数
2  while i<=9:
3          j=1  #j是列数
4          while j<=i:  #同一行，列数不超过行数
5                  print("{}*{}={} ".format(j,i,j*i),end=" ")
6                  j+=1
7          i+=1
8          print("")
```

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12  4*4=16
1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

In [21]:

```
i=1          #i是行数
while i<=9:
        j=1  #j是列数
        while j<i:  #同一行，列数不超过行数
                print("{}*{}={} ".format(j,i,j*i),end=" ")
                j+=1
        if j==i:
                print("{}*{}={} ".format(j,i,j*i))
        i+=1
```

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12  4*4=16
1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

2.2 for循环

像while循环一样，for可以完成循环的功能。

在Python中 for循环可以遍历任何序列的项目，如一个列表或者一个字符串等。

for循环的格式

```
for 临时变量 in 列表或者字符串等:
    循环满足条件时执行的代码
```

In [14]:

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

In [15]:

```
[i for i in range(5)]
```

Out[15]: [0, 1, 2, 3, 4]

In [24]:

```
for i,j in {1:"a",2:"b",3:"c"}.items():
    print(i)
    print(j)
    print("---")
```

```
1
a
---
2
b
---
3
c
---
```

2.2.1 for循环嵌套

有一个列表a=[1,2,3,[55,66,77,88]],用for循环嵌套的方式遍历该列表，如果列表a的元素为列表，则将元素为列表中的元素逐一输出。

In [50]:

```
a=[1,2,3,[55,66,77,88]]

for i in a:
    if type(i)==list :
        for j in i:
            print(j)
```

```
55
66
77
88
```

2.2.2 练习11

如何将列表["a","a+","c","a+","d","c","d","a+","e","f","d","f","a+","f","d"]里面所有的"a+"替换成"a"。

In [30]:

```
1
```

executed in 326ms, finished 16:18:37 2019-04-27

Out[30]: ['a', 'a', 'c', 'a', 'd', 'c', 'd', 'a', 'e', 'f', 'd', 'f', 'a', 'f', 'd']

In [32]:

```
1
```

Out[32]: ['a', 'a', 'c', 'a', 'd', 'c', 'd', 'a', 'e', 'f', 'd', 'f', 'a', 'f', 'd']

2.2.3 练习12

有以下两个列表：
a=[1,4,5,6,7]

b=[4,6,2,8,9]
如何计算出两个列表之间的元素相乘的结果，并生成新的一个列表？

In [33]:
```
1
```
executed in 20ms, finished 13:42:49 2019-04-26

Out[33]: [4, 24, 10, 48, 63]

2.2.4 练习13

a=[[3, 5, 2, 4],[7, 6, 8, 8],[1, 6, 7, 7]]
如何取出列表a中每个元素列表的前三个元素,最终生成以下新列表:
[[3, 5, 2], [7, 6, 8], [1, 6, 7]]

In [1]:
```
1
```
executed in 21ms, finished 13:24:49 2019-04-29

Out[1]: [[3, 5, 2], [7, 6, 8], [1, 6, 7]]

2.3 for...else与while...else

遍历循环还有一种扩展模式， 使用方法如下:

```
for 〈循环变量〉 in 〈遍历结构〉:
    〈语句块1〉
else:
    〈语句块2〉
```

同样地，while语句也有相同的扩展

- 当for循环正常执行之后， 程序会继续执行else语句中内容。
- else语句只在循环正常执行之后才执行并结束， 可以在<语句块2>中放置判断循环执行情况的语句

In [26]:
```
1 for i in [1,2,0,3]:
2     print(i)
3 else:
4     print("没问题")
```
0.5
0.5
0.5
0.5
没问题

In [30]:
```
1 for i in [1,2,0,3]:
2     print(i)
3     break
4 else:
5     print("没问题")
```
1

In [31]:

```
i=1

while i<=5:
    print(i)
    i+=1
    break
else:
    print("没问题")
```

```
1
```

3 跳转语句

3.1 break语句

break语句可以终止当前的循环，包括while和for在内的所有控制语句。
这就好比你绕圈跑步，一开始打算跑5圈。
跑了2圈半，感觉要拉肚子了，你就停下来，不跑了，找厕所去了。
——结束当前循环
此时，算作你跑了2圈。

- **break用来跳出最内层for或while循环，脱离该循环后程序从循环后代码继续续执行。**
- **break语句跳出了最内层while循环，但仍然继续执行外层循环。每个break语句只有能力跳出当前层次循环。**

3.1.1 在for循环中

In [3]:

```
name = '世界杯在召唤我'

for x in name:
    print('----')
    if x == '在':
        break
    print(x)

print("这句话在for循环后，但是和for循环无关")    #脱离该循环后程序从循环后代码继续续执行
```

```
----
世
----
界
----
杯
----
这句话在for循环后，但是和for循环无关
```

In [13]:

```
name = '世界杯在召唤我'

for i in range(3):
    for x in name:
        print('----')
        if x == '在':
            break        #跳出了最内层while循环，但仍然继续执行外层循环
        print(x)
    print("\n————————我是{}次大循环结束之后的优美的分割线————————\n".format(i+

print("\n这句话在for循环后，for循环无关\n")
```

```
----
世
----
界
----
杯
----

————————我是1次大循环结束之后的优美的分割线————————

----
世
----
界
----
杯
----

————————我是2次大循环结束之后的优美的分割线————————

----
世
----
界
----
杯
----

————————我是3次大循环结束之后的优美的分割线————————

这句话在for循环后，for循环无关
```

3.1.2 在while循环中

In [36]:
```
i = 0
while i<10:
    i+=1
    print('----')
    if i==5:
        break
    print(i)
```

```
----
1
----
2
----
3
----
4
----
```

如果是嵌套循环，break语句结束的是哪个循环？

In [10]:
```
for i in range(3):
    for j in range(20,31):
        if j ==25:
            break
        print(j)
    print("----")
```

```
20
21
22
23
24
----
20
21
22
23
24
----
20
21
22
23
24
----
```

3.2 continue

- continue的作用：用来结束本次循环，紧接着执行下一次的循环。

- 这就好比你绕圈跑步，一开始打算跑5圈。
- 跑了2圈半，发现好像落下钱包在起跑点了，终止跑第二圈。
- 然后你回到起跑点找钱包，找到钱包，然后继续重新跑，算作从第3圈开始跑。
- 这里你要注意的是：
  - break用来结束当前代码块的最内层循环。
  - continue用来结束当前代码块的本次循环。

3.2.1 在for循环中

In [17]:

```
name = '世界杯在召唤我'

for x in name:
    print('----')
    if x == '在':
        continue
    print(x)
```

```
----
世
----
界
----
杯
----
----
召
----
唤
----
我
```

3.2.2 在while循环中

In [31]:

```
i = 0
while i<10:
    i = i+1
    print('----')
    if i==5:
        continue
    print(i)
```

```
----
1
----
2
----
3
----
4
----
----
6
----
7
----
8
----
9
----
10
```

注意:

break/continue只能用在循环中，除此以外不能单独使用

break/continue在嵌套循环中，只对最近的一层循环起作用

如果是嵌套循环，continue语句结束的是哪个循环？

In [51]:

```
for i in range(3):
    for j in range(1,8):
        if j ==5:
            continue
        print(j)
    print("----")
```

```
1
2
3
4
6
7
----
1
2
3
4
6
7
----
1
2
3
4
6
7
----
```

3.2.3 练习14

打印1到10之间的整数,但是2、5、7、8这四个数除外。

In [2]:

```
1
```

executed in 9ms, finished 13:26:02 2019-04-29

```
1
3
4
6
9
10
```

In [7]:

```
1
```

executed in 7ms, finished 13:42:16 2019-04-29

```
1
3
4
6
9
10
```

3.2.4 练习15

使用while，完成以下图形的输出

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

In [5]:
```
1
```
executed in 8ms, finished 13:29:15 2019-04-29

```
*
**
***
****
*****
****
***
**
*
```

3.3 pass空语句

In [20]:
```
1 for letter in 'Python':
2     if letter == 'h':
3         passs
4     else:
5         print("当前字母 :",letter)
```

```
当前字母 : P
当前字母 : y
当前字母 : t
当前字母 : o
当前字母 : n
```

如果我们上面的pass不写（或者用注释代替），会报错，因为if语句是不完整的：

从上面对比可以知道，pass是空语句，是为了保持程序结构的完整性。
pass 不做任何事情，一般用做占位语句。

4 课后作业：

4.1 作业一

有以下两个列表：
a=[[1,4,5],[6,7,6]]
b=[[4,6,2],[8,9,7]]
如何计算出两个列表里面的元素相乘的结果，并生成新的一个列表[4, 24, 10, 48, 63, 42]？

In [42]:
```
1
```
executed in 14ms, finished 13:57:29 2019-04-26

Out[42]: [4, 24, 10, 48, 63, 42]

4.2 作业二

如何生成新列表[[4, 24, 10], [48, 63, 42]]?

In [44]:
```
1
```
executed in 15ms, finished 14:06:33 2019-04-26

Out[44]: [[4, 24, 10], [48, 63]]

4.3 作业三

参考2.1.3 while循环嵌套例题，如何使用for循环打印九九乘法表?

In [12]:
```
1
```
executed in 11ms, finished 13:50:14 2019-04-29

```
1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
1*4=4 2*4=8 3*4=12 4*4=16
1*5=5 2*5=10 3*5=15 4*5=20 5*5=25
1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

In [13]:
```
1
```
executed in 12ms, finished 13:52:11 2019-04-29

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12 4*4=16
1*5=5  2*5=10 3*5=15 4*5=20 5*5=25
1*6=6  2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7  2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8  2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9  2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

In [16]:
```
1
```
executed in 11ms, finished 13:54:27 2019-04-29

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12 4*4=16
1*5=5  2*5=10 3*5=15 4*5=20 5*5=25
1*6=6  2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7  2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8  2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9  2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```