【Python小课堂】第三课－分支循环玩神曲PPAP

Original 2016-10-09 哈哈小 菜猿观世界

ppap
来自菜猿观世界 1:23

1:23 **ppap** 来自菜猿观世界

前两次课讲的内容可以说是编程语言入门必备的基础知识，所以希望大家掌握牢了，因为那些东西对后面的学习可大有帮助哦～

今天又有什么好玩的呢?

没错，就是**分支与循环**～

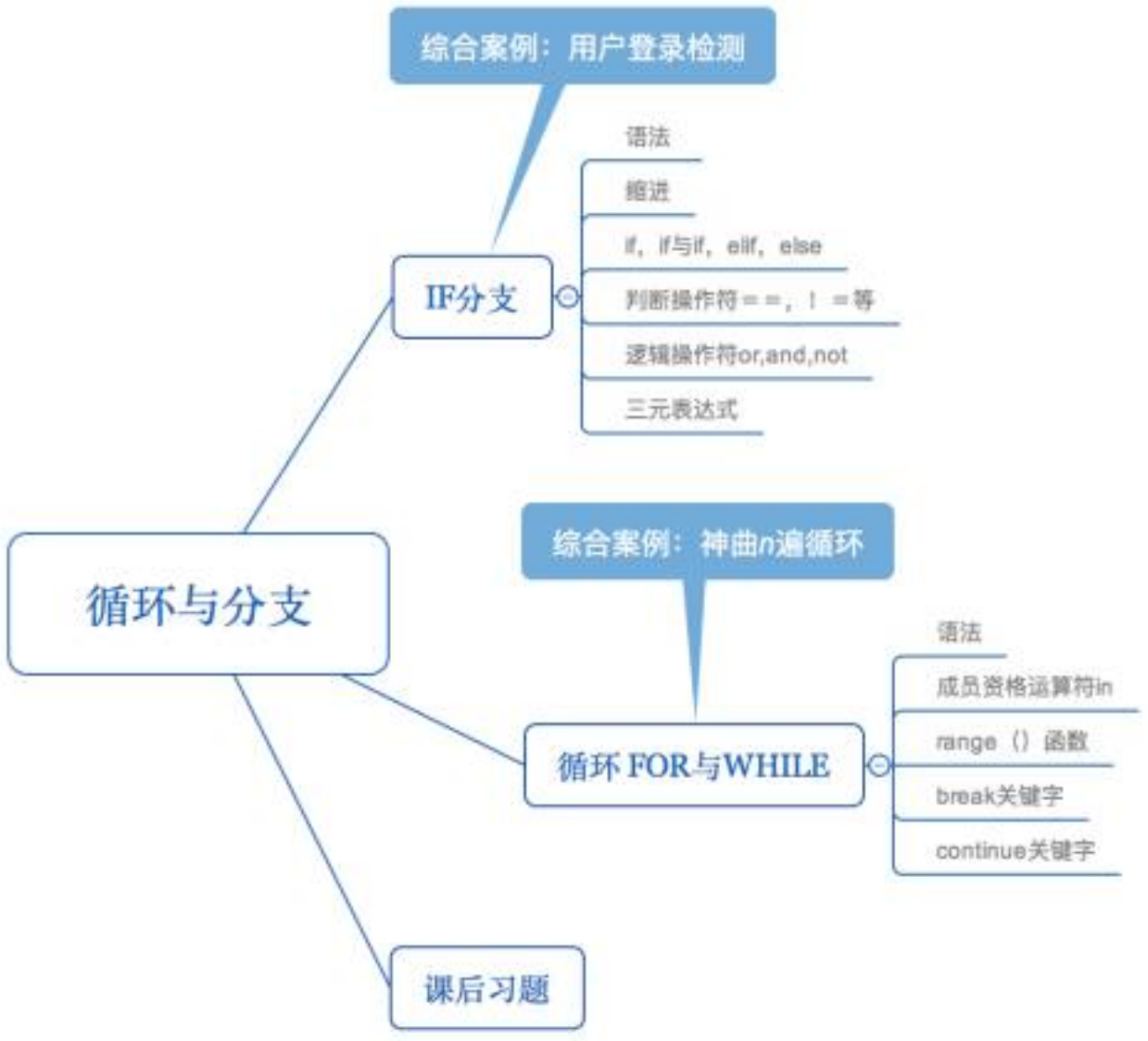
可以说，程序如果没有分支和循环，剩下的将会是一长串的无味和繁琐；

算法更无从谈起，如今的科技大厦更是无法筑建。

发展至今，也没听过哪门编程少了分支和循环。

既然na么重要，看来非学不可了！

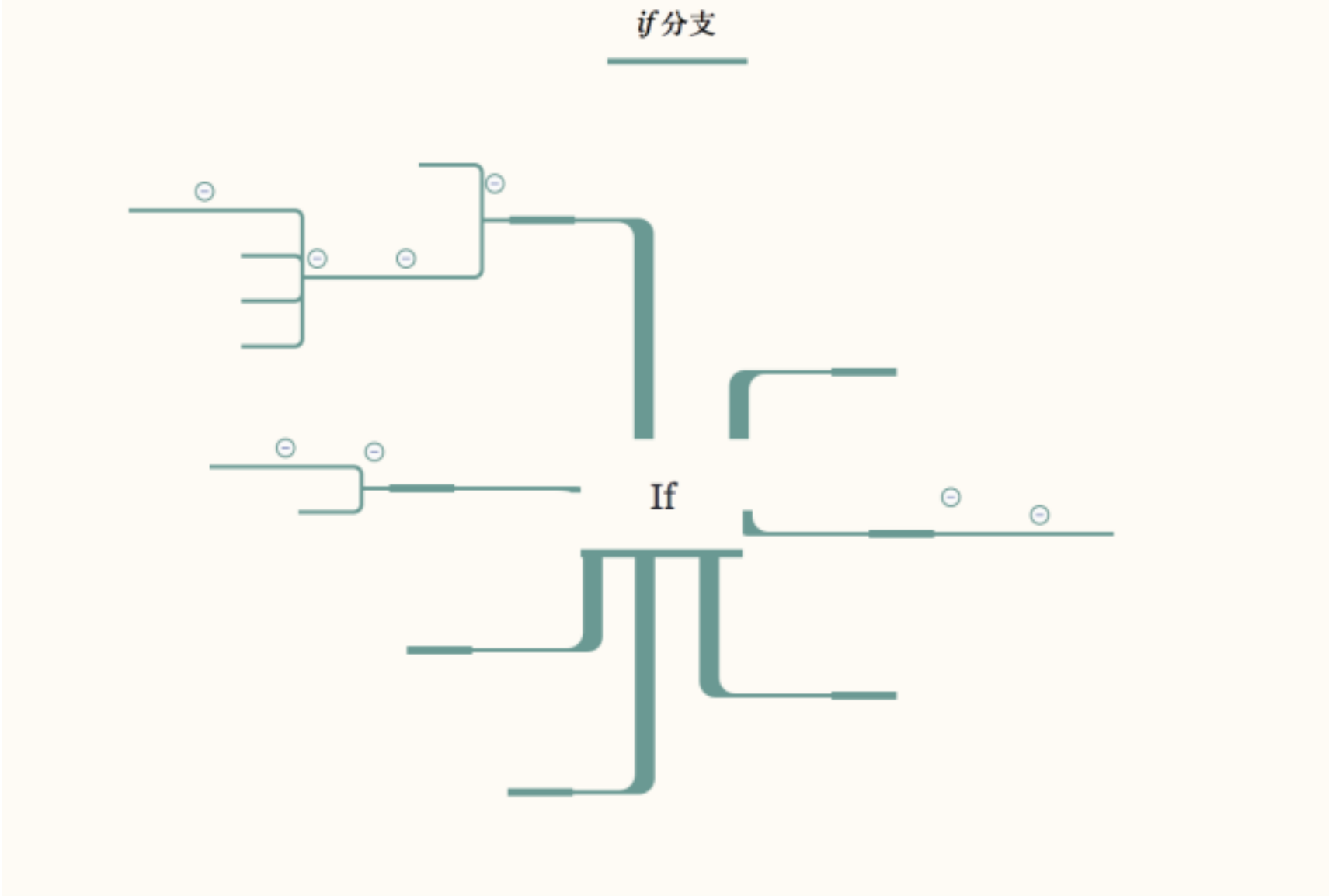
> 通过本课的学习，你将了解以下内容：



1

If分支（if statement）

分支，大脑中的第一印象可能会是这样的：



没错，大概就是这么个情况。

大家可能注意到，今天的题目是分支循环玩神曲PPAP，哈哈小啰里八嗦那么久，也没着题啊?

很不幸，哈哈小本来想立刻就搬出来玩，谁知：

什么gui？看个视频，还要身份验证，好吧，我试试，看还记不记得密码.....

试了几次，还好，顺利进入了。

哎，等等，这时，哈哈小寻根究底的心开始泛滥了，仔细琢磨着这漂亮的界面背后代码实现的原理大概是怎么个样子的。

于是，哈哈小就用python写了如下两个实现方法：

方法一

```
#method 1
#password should be numeric
#user can try many times
correctaccount='cc'
correctpsd='123'

while True:
    youraccount=input("input your name:")

    yourpsd=input("input your password,please:")

    if yourpsd.isnumeric():
        if youraccount!=correctaccount or yourpsd!=correctpsd:
            if youraccount!=correctaccount:
                print("account not correct, try again\n")
            if yourpsd!=correctpsd:
                print("psd not correct, try again\n")
        else:
            break
    elif not yourpsd.isnumeric():#此行也可以写成else:
        print("password should be numeric,try again:\n")

print("succesfully,welcome!")
```

```
input your name:cc
input your password,please:a12
password should be numeric,try again:

psd not correct, try again

input your name:cc
input your password,please:111
psd not correct, try again

input your name:aa
input your password,please:123
account not correct, try again

input your name:cc
input your password,please:123
succesfully,welcome!
>>>
```

方法二

```
#method 2
#password should be numeric
#user can try many times
correctaccount='cc'
correctpsd='123'

while True:
    youraccount=input("input your name:")
    yourpsd=input("input your password,please:")

    if not yourpsd.isnumeric():
        print("password should be numeric,try again:\n")
    if youraccount!=correctaccount:
        print("account not correct, try again\n")
    if yourpsd!=correctpsd:
        print("psd not correct, try again\n")
    if youraccount==correctaccount and yourpsd==correctpsd:
        break
print("succesfully,welcome!")
```

```
input your name:cc
input your password,please:a12
password should be numeric,try again:

input your name:cc
input your password,please:111
psd not correct, try again

input your name:aa
input your password,please:123
account not correct, try again

input your name:cc
input your password,please:123
succesfully,welcome!
>>>
```

哈哈小实现的是在登录时，验证账户和密码（必须为数字）的程序。

代码中出现的while，break大家暂且放一边，等读完此文，再回来理解。

针对分支part，说明一下：

1）python中if的用法很简单：

if 条件：

**　语句**

如果条件（表达式）满足，程序就执行其下面的语句。嗯， 是的，这一点哈哈小就好比今天是今天明天绝不会是昨天这般确信不疑，哈哈～

不信，给你来一例：

```
>>> if "hahaxiao"!="哈哈小":
        print("yes,you are right.")


yes,you are right.
>>>
```

2）从上图可以看出，当写嵌套分支时，python代码中，看不到很多其他编程语言用大括号来格式化代码的现象。所以说，**注重缩进，格式与代码的规整**对于python编写者尤为重要。

3）elif是elseif的缩写。if，if...与if，elif...的**区别**在于，**前者每次**到达该条分支口，都要去检测后面的**条件是否满足，后者是只要有一条分支满足条件**，其他的同层次分支**不再检测**。所以，在符合逻辑的情况下，使用**后者的效率明显要高**。

举个例子，某校面相对象编程这门学科的评分规定是这样的：该门课程采用百分制， 90以上为A,，80分以上为B， 60分以上为C，60分以下为D。据此，我们可以用程序的方式来表现这个评分标准：

方法一

```
score=int(input("your score:"))

if score>=90:
    print("A")
if 90>score>=80:
    print("B")
if 80>score>=60:
    print("C")
if 60>score>=0:
    print("D")
if score<0 or score>100:
    print("error")
```

方法二

```
score=int(input("your score:"))

if score>=90:
    print("A")
elif 90>score>=80:
    print("B")
elif 80>score>=60:
    print("C")
elif 60>score>=0:
    print("D")
else :
    print("error")
```

上图两种方式，方法二的效率要高。再作进一步解释，第一种方法就好像摆在我面前十扇门，开了一条顺利之门之后，后面每条门还都要开一下（这少年好奇心有点强），去看看这扇门所通向的路是不是适合我走的。也是够累的额。

方法二就不同了，一旦找到适合自己的路，走完了事，管其他门呢，想要我开，门儿都没有，真够累的，我还是去躺躺，哈哈～（这少年比较按捺得住内心的躁动）

有学过C 语言或者其他语言的同学可能会说：我有一种更好的办法，用switch。哈哈小要很不好意思地告诉你，python没有预定义的所谓switch语句，至于为什么，可见下节或下某节分晓了哦。

4) 代码中出现的 ＝＝（判断等式两边是否相等），!=（判断等式两边是否不相等）为python中的**判断操作符**。除了这两个，还有值得注意的是 <=(左边是否小于或等于右边), >=(左边是否大于或等于右边), <（左边是否小于右边）, >（左边是否大于右边）。

举几个例子，

3>=3，2e4>20001, True<False返回结果分别为True, False, False。

如果还是感觉有点不明白，那可得回去看上节课的内容了哦。

5）代码中出现的or，and， not等这样的关键字是python的**逻辑操作符**，什么意思呢？比如，x and y，x,y都为真返回后面那一个数y，x or y任何一个满足返回最先满足的那个数（这种特性叫短路逻辑），not x 表示x为false返回true。它们的优先级顺序是：**not>and>or**。

这次干脆就来半打例子吧：

```
>>> 3 and 4
4
>>> 3 or 4
3
>>> not 2
False
>>> 2 or not 3 and 4
2
>>> not False
True
>>> True or False and not False
True
>>>
```

通过demos，理解起来是不是要明白多了呢？如果是的话，那就请用闪电般的速度说出这个表达式的结果吧：not 1 and 3 or 4 and 9 not 7 and 8 or 10

6）最后附加另外一个好玩的东西－**三元操作符**，假如我想输出两个数中较大的那个，我们有两种方法。由图可知，用三元，四行变一行，简洁力简直max啊～

三元表达式语法：x if 条件 else y

```
x,y=3,4

if x>y:
    large=x
else:
    large=y

print("the larger one is %d"%large)
```

```
x,y=3,4

large=x if x>y else y

print("the larger one is %d"%large)
```

好吧，人也不少了，时候也不早了， 总算弄明白了，也该开始玩了。

（请配文头声音观看～）

stop，如果我们想n次循环听神曲，循环那可是必不可少啊，so抱歉，在玩之前还需要我们来认识一下循环。

循环（loop）

循环，大脑中的第一印象可能会是这样的：

没错，具体点，代码执行起来应该会是这样的：

好的，终于可以开玩了～

哈哈小的玩法是，用python写了一小段程序来玩神曲

先看运行结果：

```
Pen-Pineapple-Apple-pen!

-------------End------------------

---ppap----------第8遍------------

I have a pen,I have an apple.

(Eh~)Apple-pen!

I have a pen,I have pineapple.

(Eh~)Pineapple-pen!

Apple-pen~Pineapple-pen(Eh~)

Pen-Pineapple-Apple-pen!

Pen-Pineapple-Apple-pen!

-------------End------------------

---ppap----------第9遍------------

I have a pen,I have an apple.

(Eh~)Apple-pen!

I have a pen,I have pineapple.

(Eh~)Pineapple-pen!

Apple-pen~Pineapple-pen(Eh~)
```

Ln: 50 Col: 0

哈哈小表示玩的有点过了头，不小心按了23遍，所以只好用ctrl＋c结束程序了。

哎，这个有点意思啊，怎么做到的呢?

下图见真章（此时可不必纠结于代码哦，且看完我后文的分析之后，再阅）：

```python
while True:
    time=int(input("你想听几遍啊？"))
    if(time<=0):
        print("既然来都来了，可不能一遍都不听哦～\n")
        continue
    title=input("请输入PPAP:")
    #判断时不区分大小写
    if(title.upper()!='PPAP'):
        print("你输入的不是PPAP，是不是？没关系，再来一遍吧\n")
    else:
        break
print("神曲开始！")

#计遍数
count=1

#为了唱完一句之后可能想放点什么，哈哈小这里使用的是一句一个循环哦（你们也可以改成一篇一个循环哦）
for each in range(0,time*4):
        #每一遍为四句z
    if each%4==0:
        print("———%s——————第%d遍——————————\n"%(title,count))
        print("I have a pen,I have an apple.\n")
        print("(Eh~)Apple-pen!\n")
    elif each%4==1:
        print("I have a pen,I have pineapple.\n")
        print("(Eh~)Pineapple-pen!\n")
    elif each%4==2:
        print("Apple-pen~Pineapple-pen(Eh~)\n")
    else:
        print("Pen-Pineapple-Apple-pen!\n")
        print("Pen-Pineapple-Apple-pen!\n")
        print("——————————End——————————————\n")
        #最后一句结束，遍数加一
        count+=1
```

说明一下：

1）两个循环，第一个为while循环，第二个为for循环，语法分别为：

语法	举例

while 条件：
循环体

```
>>> i=0
>>> while i<4:
        print("hahaxiao\n")
        i+=1


hahaxiao

hahaxiao

hahaxiao

hahaxiao

>>>
```

for 目标 in 表达式：
循环体

```
>>>
>>> string="hahaxiao"
>>> for each in string:
        print(each,end=' ')

h a h a x i a o
```

再作进一步的解释，一但条件满足，就去执行循环体一次，一直执行下去，直到条件不满足为止。

比如第一个例子，i<4 这个条件等执行了4次循环体之后，i 递增到了4，不再符合 i<4 这个条件了，于是就跳出循环了，say goodbye了。

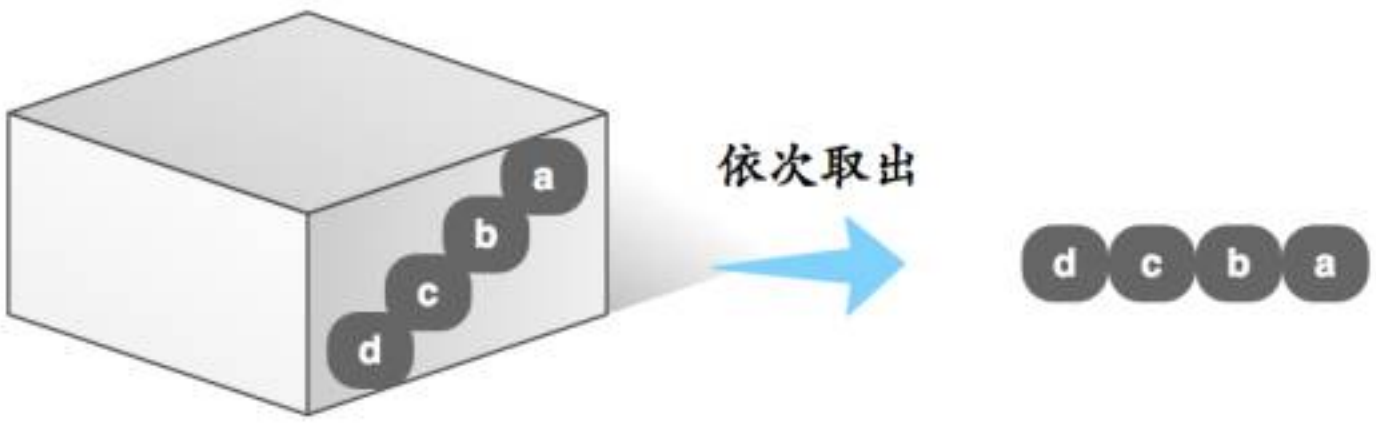
对于第二例子， 这里提一下print函数中的end。这个的意思是，每打印一次字符串each，并在其尾巴加上你赋给end 的字符串（为了方便看出结果，这里哈哈小给的是空格）。

第二个例子进一步解释请从这往下看。

2）大家可能有注意到for循环用到的in这个关键字，in在这叫做**成员资格运算符**，使用来检查一个值是否位

于序列中，如果在则返回True，不在则返回False。

就好比上面的例子，for each in string，每执行一次循环，就从string这个字符串中取出一个值，并将这个值赋给变量each。而且这种索取按从左至右的顺序索取的，直到取完为止，有种迭代的感觉。至于什么是迭代，大家目前阶段有下面这张图的概念就行了。

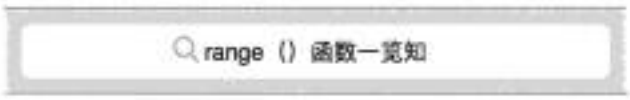


讲到这里，python两种循环for，与while是不是已经变得so easy了呢?

别急，哈哈小再给你加点好玩的东西～

3）主代码的第十八行，有出现 range（）这个方法。哈哈小在这要告诉你，这可是一个好东西，跟for循环最搭，就好比国庆搭假期，白天搭夜晚，星星搭月亮一般，哈哈～

不信你看：



语法：range（[start,] stop[, step=1]）

可传三个参数，中括号括起来的表示可选，step如果不传，默认为1

传入一个参数stop，表示0至x的数字序列（包括0，不包括x），例：
```
>>> list(range(4))
[0, 1, 2, 3]
```

如果三个参数都传，表示生成从start到stop步数为step的数字序列

输出10-20的所有奇数

实例

```
>>> for i in range(11,20,2):
        print(i)


11
13
15
17
19
>>>
```

要输出一些特定数字序列，for搭range（），绝配了。

4）对第五行和第十行出现的break和continue做一番解释，break表示跳出循环，不再执行该循环体（如果循环嵌套的话，只能跳出一级哦）。

大家可以设想设想一下，再进入某个游戏时，就好比进入了一个死循环，点击退出游戏就等同执行了break代码，所以说，break还是很有用的。

至于continue，它的意思是，立马结束本次循环，即这次循环它之后代码不在执行，而是去验证执行下一次循环。

就比如本例中的第五行，如果用户输入的播放次数不合法，我们就直接到下一次循环要用户再次输入，而不是接着去执行接受神曲名的代码。

看了一遍一脸懵c.gif

思考一会.gif

懵c继续时.gif

没关系，再用两个简单易懂的例子说明一下：

Example 1

```
for i in range(10):
    if i%2==0:
        continue
    print(i)
```

一遇到能被2整除的数，就continue循环，不再执行后面的print函数了。以这种方式，就成功避免了偶数的输出，输出0到9的所有质数，用这种方法是不是也不错呢?

Example 2

```
answer=6

while True:
    num=int(input("guess the number:"))
    if num>answer:
        print("too large, try again:\n")
    elif num<answer:
        print("too small,try again:\n")
    else:
        print("correct,congratulations!")
        break;
```

猜数字的游戏，游戏一开始，我们就进入到一个死循环里。没有猜中，就没有break，没有break，就别想结束游戏（除非你偷偷使用了crtl＋c，或着不听话去输入一些乱七八糟的字符），哈哈～

5）本例中用到的while循环，判断的条件是True， 永真，所以是个死循环。值得注意的是，在 python中，只有以下情况会被看作假: False None () {} [] '' ""

来试试：

```
>>> while None:
        print("来呀，来呀，来咬我啊。")


>>> while ():
        print("没咬到吧，瞧你那损色！")


>>> while '':
        print("还是没咬到吧，呵呵。")


>>>
```

3

课后作业

上次哈哈小说过，从今开始，要留课后作业，说到做到，哈哈～

课后作业

1.对于用户身份验证程序，如果只给用户三次验证机会，并且密码升级为：不低于8位数，至少包含字母和数字；如何写呢?

2.请编写一个输出1-1000所有偶数的程序。

3.请编写一个输出1-1000所有素数的程序（只能被自身和1整除的自然数）。

4.如果一个3位数，等于其各位数字的立方和，则称这个数为水仙花数，如：370=3^3+7^3+0^3。请输出100-999之间的所有水仙花数。

5.如何用三元表达式找出三个数的最大值?

赶紧开动脑筋吧，要答案和课程相关代码请扫**下方二维码**加入答疑群找哈哈小哦～

下面是我们的技术交流群二维码，微信群长按二维码可以加入，QQ群可能需要大家手动输入群号去查找。大家有任何技术方面的问题，可以在群里提问，我们会详细解答的噢~以后我们还会在群里举办各种小竞赛、小活动。

（如果二维码过期，可以在公众号下方
菜单栏“菜猿业务”中点击“交流社群”
获取最新二维码）

QQ群（592945062）

群名称：菜猿-编程入门
群　号：592945062

结束语：我们不想给你带来多少多少的知识，我们只想尽绵薄之力给你带来可能的智慧的启迪。

END

菜猿编程宝典 | 重新定义编程入门教育

长按，识别二维码，加关注