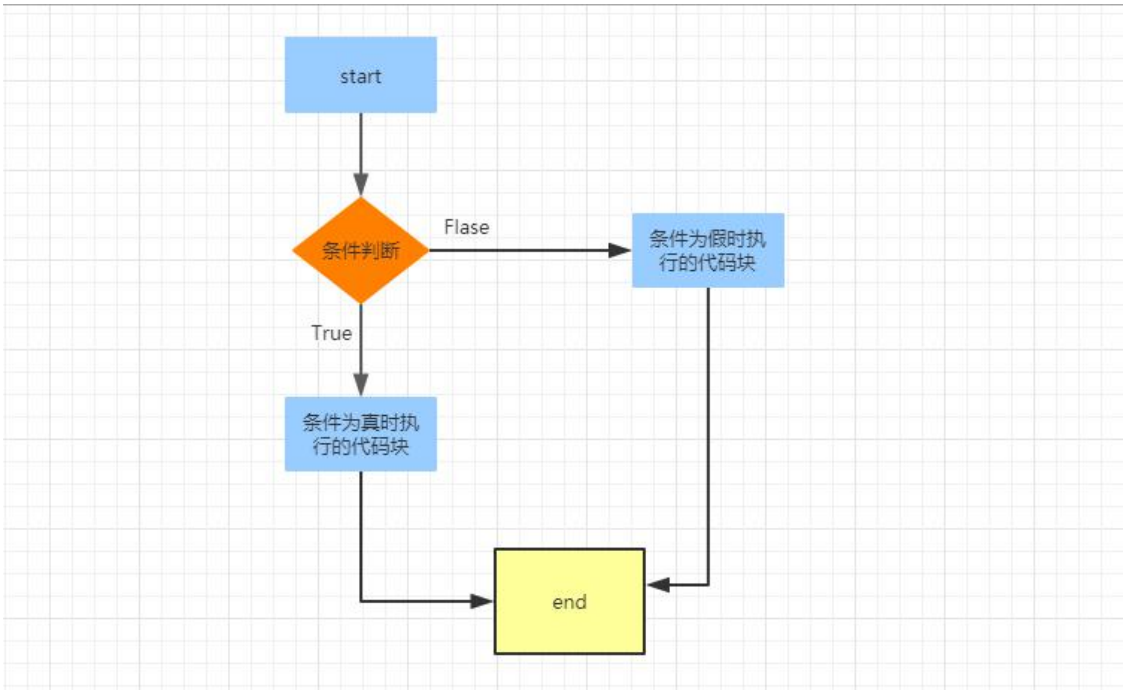
一、if-else 判断语句

1、if-else

1、条件判断语句

Python 条件语句是通过一条或多条语句的执行结果（True 或者 False）来决定执行的代码块。



2、if-else 语句

在生活中我们遇到问题也经常需要一个条件来判断是否去做一件事。

比如： 如果明天不下雨，我们就去爬山，否则打羽毛球。

伪代码实现：

```
if 明天不下雨：
    我们去爬山
else:
    我们去打羽毛球
```

3、猜数字游戏

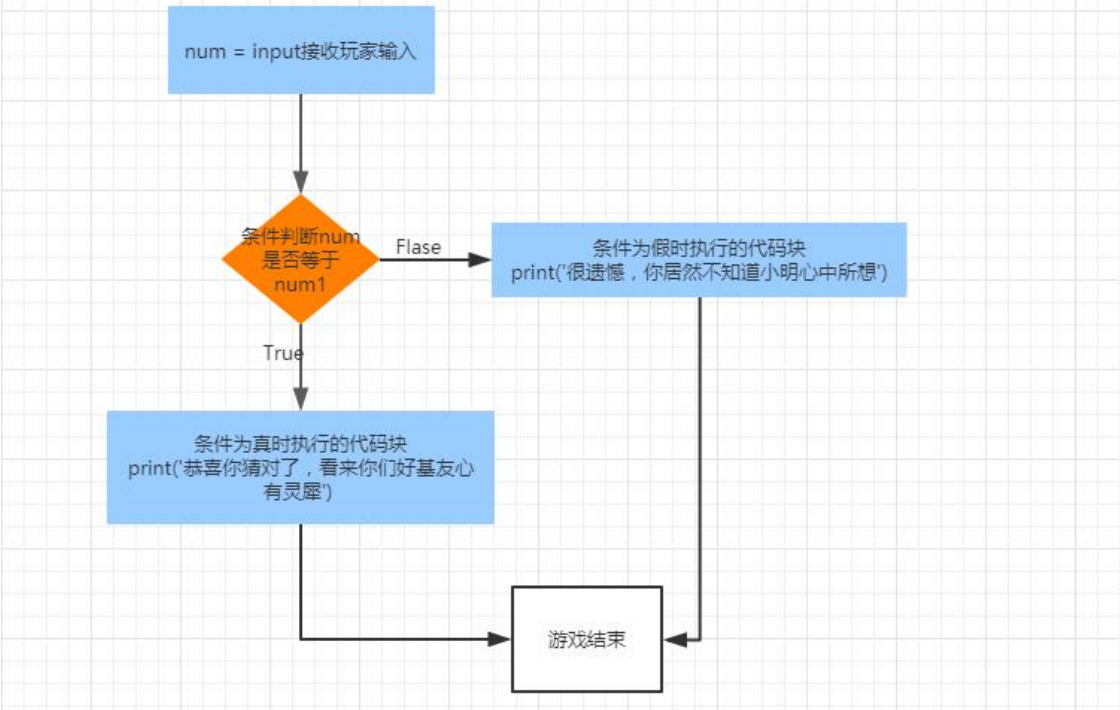
```
"""
猜数字游戏
```

```
"""
# 接收玩家输入数字
num = input('你觉得现在小明想的数字是什么：')

# 定义一个变量，表示小明心中所想
num1 = 5

if num == num1:  # 条件判断
    print('恭喜你猜对了，看来你们是好基友心有灵犀')    # 条件成立打印这句话
else:
    print('很遗憾，你居然不知道你基友心中所想')    #  条件不成立打印这句话
```

4、用流程图分析猜数字游戏



2、if-elif-else

1、if-else 语句只能判断当一个条件。那么多个条件时我们就用到 elif 语句。

```
if 条件 1:
    条件 1 满足时执行
elif 条件 2:
    条件 2 满足时执行
elif 条件 3:
    条件 3 满足时执行
```

2、使用示例，判断考试分数在那个等级。

```
# 小明考试考了75分，编写一个程序判断小明的成绩属于哪个等级
# 100-90 A
# 89-70 B
# 69-50 C
# 49 以下 D
score = 75
if score >=90:
    print('该学生的成绩为A')
elif 70 <= score < 90:
    print('该学生的成绩为B')
elif 50 <= score < 70:
    print('该学生的成绩为C')
else:
    print('该学生的成绩为D')
Copy
```

提示： if-elif 语句只要有一个条件满足，就会执行条件满足时执行的代码，后面那些条件将不会再判断。

3、猜拳案例

```
import random  # 导入生成随机数模块

inp = int(input('请出拳：'))
# 电脑随机出拳
computer = random.randint(0,2)
print('电脑出拳：%d'%computer)

if inp >2:
    print('输入错误')
elif inp == 0 and computer == 2:
    print('厉害了，居然赢了')
elif inp == 1 and computer == 0:
    print('厉害了，居然赢了')
elif inp == 2 and computer == 1 :
    print('厉害了，居然赢了')
elif inp == computer:
    print('不错，居然平手')
else:
    print('哈哈，输了吧！')
```

3、if 语句的嵌套

1、if 语句嵌套格式：

```
if 条件 1：
    if 条件 2：
        print('满足条件 1'，同时满足条件 2)
    else:
        print('满足条件 1'，但是不满足条件 2)
elif 条件 3:
    if 条件 4:
        print('满足条件 3'，同时满足条件 2)
```

提示：

外层的 if 判断，也可以是 if-else

内层的 if 判断，也可以是 if-else

根据实际开发的情况进行选择

2、if 嵌套案例：

```
num =int(input("输入一个数字： "))
if num%2==0:
    if num%3==0:
        print('你输入的数字可以整除 2 和 3')
    else:
        print('你输入的数字可以整除 2,但不能整除 3')
else:
    if num%3==0:
        print('你输入的数字可以整除 3，但是不能整除 2')
    else:
        print('你输入的数字不能整除 2 和 3')
```

二、while 循环

1 、while 循环

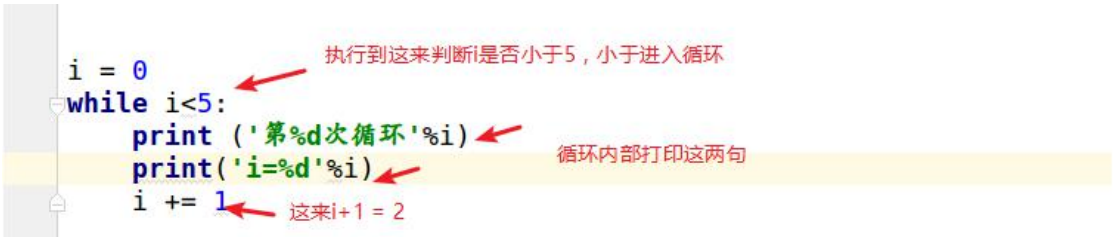
刚才写的猜拳游戏每次运行一次只能猜一次，那么运行一次可以一直玩呢？

那么就要用到循环。

while 循环格式

```
while 条件：
    满足条件时，执行语句 1
    满足条件时，执行语句 2
    满足条件时，执行语句 3
```

while 循环执行,执行完一遍后，再回去判断 i 的值。




```
i = 0
while i<5:
    print ('第%d次循环'%i)
    print('i=%d'%i)
    i += 1
```

猜拳游戏改进：

```
import random  # 导入生成随机数模块

while True:

    inp = int(input('请出拳：'))
    # 电脑随机出拳
    computer = random.randint(0,2)
    print('电脑出拳：%d'%computer)

    if inp >2:
        print('输入错误')
    elif inp == 0 and computer == 2:
        print('厉害了，居然赢了')
    elif inp == 1 and computer == 0:
        print('厉害了，居然赢了')
    elif inp == 2 and computer == 1 :
        print('厉害了，居然赢了')
    elif inp == computer:
        print('不错，居然平手')
    else:
        print('哈哈，输了吧！')
```

计算 1-100 的和

```
i = 1
sum = 0
while i <= 100:
    print(i)
    sum += i
    i += 1
print(sum)
```

2、while 循环嵌套

while 循环也是可以嵌套使用的。

1、while 循环嵌套格式

```
while 条件1:
    条件成立时执行语句
    条件成立时执行语句
    条件成立时执行语句
    while 条件2:
        条件成立时执行语句
        条件成立时执行语句
Copy
```

2、while 循环嵌套使用，打印三角形

```
i = 1
while i <= 5:
    j=1
    while j <=i:
        print('*',end=" ")
        j += 1
    print('\n')
    i += 1
Copy
i = 9
while i >= 1:
    j = 1
    while j <= i:
        print('%d*%d=%2d\t'%(j,i,i*j),end='')
        j +=1
    i -= 1
    print()
```

三、for 循环

for 跟 while 循环一样也可以完成循环。

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 循环格式：

```
for 临时变量 in 字符串，列表等:
    执行代码块
    执行代码块
    执行代码块
Copy
```

1、for 循环示例

```
a = 'python'
for i in a:
    print(i)
p
y
t
h
o
n
Copy
```

2、for 循环遍历列表

```
li = ['a','b','c','d']
for i in li:
    print(i)
a
b
c
d
```

四、break、continue 语句

回想下我们刚才写的猜数字游戏，是不是运行了一直运行到天荒地老，永远不会退出。除非你把程序关掉。那么结束循环应该怎么办？

Python 中使用 break 来结束整个循环。程序遇到 break 将跳出循环，不再执行循环里面的代码。

给猜数字游戏加一个功能，只要玩家胜利三次就退出游戏。

```
import random
win = 0
while True:
    if win >= 3:
        print('你已经胜利三次')
        break
    else:
        # 胜利小于三次，继续完
        inp = int(input('请出拳：'))
        computer = random.randint(0,2)
        print('电脑出拳：%d'%computer)
        if inp >2:
            print('输入错误')
        elif (inp ==0  and computer==2) or (inp == 1 and computer==0) o
r (inp == 2 and computer==1 ):
            print("厉害了，居然赢了")
            win += 1  # 如果胜利一次，胜利次数加1
        elif inp == computer: # 如果两个人出异样就打平
```

```
            print('不错，居然打平了')
        else:
            print('呵呵，输了吧')
Copy
```

continue 跳过本次循环，继续下一次循环。

普通循环和使用 continue 跳过的循环：

```
for i in 'python':
    print(i)
p
y
t
h
o
n

for i in 'python':
    if i == 'h':
        continue
    print(i)
p
y
t
o
n
Copy
```

提示：

break 是直接结束循环，continue 是跳过本次循环，继续下一次循环。
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

五、多条件与短路运算

在条件判断中如果有多个条件，就需要用到昨天将的逻辑运算符。

逻辑运算符：

- and 逻辑与
- or 逻辑或
- not 逻辑非

示例：

```
# 如果条件1 和条件2 都为true,那么整个条件判断才为true,执行if 里面的语句
if 条件1 and 条件2:
    print('haha')

# 如果条件1 或者条件2 有一个为true，就执行if 里面的语句
if 条件1 or 条件2:
    print('haha')

# 取反条件1，如果条件1 为true，则not 条件1 为False
# 如果条件1 为False，那么not 条件1 就为True
if not 条件1:
    print('haha')
Copy
```

1、and

```
a = 1
b = 2
c = 3

# 两个条件都为True,整个判断为True
if (a < b) and (b < c):
    print('哈哈')
else:
    print('呵呵')

# c < b 为False 所以整个条件为False

if (a < b) and (c < b):
    print('哈哈')
else:
    print('呵呵')
Copy
```

2、or

```
a = 1
b = 2
c = 3

# 两个条件都为True,整个判断为True
if (a < b) and (b < c):
    print('哈哈')
else:
    print('呵呵')

# or 只要有一个条件为True 整个条件为True

if (a < b) or (c < b):
```

```
    print('哈哈')
else:
    print('呵呵')
Copy
```

3、not

```
a = 1
b = 2
c = 3

# a<b 是true ,取反后为false,所以执行else 的内容
if not (a < b):
    print('哈哈')
else:
    print('呵呵')

# a = b false ， 取反后为true,所以执行if 里面的内容

if not (a == b):
    print('哈哈')
else:
    print('呵呵')
Copy
```

4、短路运算

短路运算规则：

表达式从左至右运算，若 or 的左侧逻辑值为 True ， 则短路 or 后所有的表达式（不管是 and 还是 or）， 直接输出 or 左侧表达式 。

表达式从左至右运算，若 and 的左侧逻辑值为 False ， 则短路其后所有 and 表达式，直到有 or 出现，输出 and 左侧表达式到 or 的左侧，参与接下来的逻辑运算。

若 or 的左侧为 False ， 或者 and 的左侧为 True 则不能实现短路逻辑。

简单的说要实现短路，表达式用 and 的则将容易为 fasle 的条件放到前面，表达式有 or 的则将容易为 True 的条件放到前面。

```
a = 1
b = 2
c = 3

# and 逻辑运算，代码执行从左到右，先运算 a==2,为 False
# and 逻辑与需要两个条件都为 True,整个条件才为 True
# 所以当条件判断执行第一个条件时为 False,整个条件为 False,不再执行后面的条件判断
# 这就是短路运算
```

```
if a == 2 and b == 2:
    print('haha')
```