第五章 程序控制结构

非顺序式的程序控制，往往需要根据一定的条件，决定程序运行的路线。因此，我们首先来认识一下什么叫条件测试。

第一部分 条件测试

1、比较运算

In [1]:

```
a = 10
b = 8
print(a > b)      # 大于
print(a < b)      # 小于
print(a >= b)     # 大于等于
print(a <= b)     # 小于等于
print(a == b)     # 等于
print(a != b)     # 不等于
```

```
True
False
True
False
False
True
```

- 非空

In [3]:

```
ls = [1]
if ls:            # 数据结构不为空、变量不为0、None、False 则条件成立
    print("非空")
else:
    print("空的")
```

```
非空
```

2、逻辑运算

- 与、或、非

In [4]:

```
a = 10
b = 8
c = 12
print((a > b) and (b > c))     # 与
print((a > b) or (b > c))      # 或
print(not(a > b))              # 非
```

```
False
True
False
```

- 复合逻辑运算的优先级
  非 > 与 > 或

In [5]:

```
print(True or True and False)
```

```
True
```

In [6]:

```
print((True or True) and False)
```

```
False
```

3、存在运算

元素 in 列表/字符串

In [8]:

```
cars = ["BYD", "BMW", "AUDI", "TOYOTA"]
```

In [9]:

```
print("BMW" in cars)
print("BENZ" in cars)
```

```
True
False
```

元素 not in 列表/字符串

In [10]:

```
print("BMW" not in cars)
print("BENZ" not in cars)
```

```
False
True
```

第二部分 分支结构——if语句

1、单分支

模板

if 条件:
    缩进的代码块

In [11]:

```
age = 8
if age > 7:
    print("孩子，你该上学啦！")
```

```
孩子，你该上学啦！
```

2、二分支

模板

if 条件:
    缩进的代码块
else:
    缩进的代码块

In [12]:

```
age = 6
if age > 7:
    print("孩子，你该上学啦！")
else:
    print("再玩两年泥巴！")
```

```
再玩两年泥巴！
```

3、 多分支

模板

if 条件:
    缩进的代码块
elif 条件:
    缩进的代码块
elif 条件:
    缩进的代码块
...
else:
    缩进的代码块

In [13]:

```
age = 38
if age < 7:
    print("再玩两年泥巴")
elif age < 13:
    print("孩子，你该上小学啦")
elif age < 16:
    print("孩子，你该上初中了")
elif age < 19:
    print("孩子，你该上高中了")
elif age < 23:
    print("大学生活快乐")
elif age < 60:
    print("辛苦了，各行各业的工作者们")
else:     # 有时为了清楚，也可以写成elif age >= 60:
    print("享受退休生活吧")
```

辛苦了，各行各业的工作者们

不管多少分支，最后只执行一个分支

4、嵌套语句

题目：年满18周岁，在非公共场合方可抽烟，判断某种情形下是否可以抽烟

In [17]:

```
age = eval(input("请输入年龄"))
if age > 18:
    is_public_place = bool(eval(input("公共场合请输入1，非公共场合请输入0")))
    print(is_public_place)
    if not is_public_place:
        print("可以抽烟")
    else:
        print("禁止抽烟")
else:
    print("禁止抽烟")
```

请输入年龄16
禁止抽烟

第三部分 遍历循环——for 循环

主要形式：

- **for** 元素 **in** 可迭代对象：
  执行语句

执行过程：

- 从可迭代对象中，依次取出每一个元素，并进行相应的操作

1、直接迭代——列表[ ]、元组( )、集合{ }、字符串" "

In [18]:

```
graduates = ("李雷", "韩梅梅", "Jim")
for graduate in graduates:
    print("Congratulations, "+graduate)
```

```
Congratulations, 李雷
Congratulations, 韩梅梅
Congratulations, Jim
```

2、变换迭代——字典

In [20]:

```
students = {201901: '小明', 201902: '小红', 201903: '小强'}
for k, v in students.items():
    print(k, v)
for student in students.keys():
    print(student)
```

```
201901 小明
201902 小红
201903 小强
201901
201902
201903
```

3、range()对象

In [22]:

```
res=[]
for i in range(10000):
    res.append(i**2)
print(res[:5])
print(res[-1])
```

```
[0, 1, 4, 9, 16]
99980001
```

In [23]:

```
res=[]
for i in range(1,10,2):
    res.append(i**2)
print(res)
```

[1, 9, 25, 49, 81]

循环控制：break 和 continue

- break 结束整个循环

In [24]:

```
product_scores = [89, 90, 99, 70, 67, 78, 85, 92, 77, 82] # 1组10个产品的性能评分
 # 如果低于75分的超过1个，则该组产品不合格
i = 0
for score in product_scores:
    if score < 75:
        i += 1
    if i == 2:
        print("产品抽检不合格")
        break
```

产品抽检不合格

- continue 结束本次循环

In [25]:

```
product_scores = [89, 90, 99, 70, 67, 78, 85, 92, 77, 82] # 1组10个产品的性能评分
 # 如果低于75分,输出警示
print(len(product_scores))
for i in range(len(product_scores)):
    if product_scores[i] >= 75:
        continue
    print("第{0}个产品，分数为{1}，不合格".format(i, product_scores[i]))
```

10
第3个产品，分数为70，不合格
第4个产品，分数为67，不合格

for 与 else的配合

如果for 循环全部执行完毕，没有被break中止，则运行else块

In [27]:

```
product_scores = [89, 90, 99, 70, 67, 78, 85, 92, 77, 82] # 1组10个产品的性能评分
 # 如果低于75分的超过1个，则该组产品不合格
i = 0
for score in product_scores:
    if score < 75:
        i+=1
    if i == 2:
        print("产品抽检不合格")
        break
else:
    print("产品抽检合格")
```

产品抽检不合格

第四部分 无限循环——while 循环

4.1 为什么要用while 循环

- 经典题目：猜数字

In [ ]:

```
albert_age = 18
#第1次
guess = int(input(">>:"))
if guess > albert_age :
    print("猜的太大了，往小里试试...")
elif guess < albert_age :
    print("猜的太小了，往大里试试...")
else:
    print("恭喜你，猜对了...")
#第2次
guess = int(input(">>:"))
if guess > albert_age :
    print("猜的太大了，往小里试试...")
elif guess < albert_age :
    print("猜的太小了，往大里试试...")
else:
    print("恭喜你，猜对了...")
```

代码可能需要重复执行，可是又不知道具体要执行多少次

4.2 while循环的一般形式

主要形式：

- **while** 判断条件：
  执行语句

条件为真，执行语句
条件为假，while 循环结束

In [ ]:

```
albert_age = 18
guess = int(input(">>:"))
while guess != albert_age:
    if guess > albert_age :
        print("猜的太大了，往小里试试...")
    elif guess < albert_age :
        print("猜的太小了，往大里试试...")
    guess = int(input(">>:"))
print("恭喜你，猜对了...")
```

4.3 while与风向标

In [ ]:

```
albert_age = 18
flag = True     # 布尔类型
while flag:
    guess = int(input(">>:"))
    if guess > albert_age :
        print("猜的太大了，往小里试试...")
    elif guess < albert_age :
        print("猜的太小了，往大里试试...")
    else:
        print("恭喜你，猜对了...")
        flag = False    # 当诉求得到满足，就让风向变一下
```

In [33]:

```
flag=True
while flag:
    pass
    while flag:
        pass
        while flag:
            flag=False     # 循环逐层判断，当flag为false时，循环会逐层退出
```

4.4 while 与循环控制 break、continue

In [ ]:

```
albert_age = 18
while True:
    guess = int(input(">>:"))
    if guess > albert_age :
        print("猜的太大了，往小里试试...")
    elif guess < albert_age :
        print("猜的太小了，往大里试试...")
    else:
        print("恭喜你，猜对了...")
        break    # 当诉求得到满足，就跳出循环
```

输出10以内的奇数

In [1]:

```
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue       # 跳出本次循环，进入下一次循环
    print(i)
```

```
1
3
5
7
9
```

4.5 while与else

如果while 循环全部执行完毕，没有被break中止，则运行else块

In [2]:

```
count = 0
while count <= 5 :
    count += 1
    print("Loop",count)
else:
    print("循环正常执行完啦")
```

```
Loop 1
Loop 2
Loop 3
Loop 4
Loop 5
Loop 6
循环正常执行完啦
```

4.6 再看两个例子

【小例子】删除列表中的特定值

In [4]:

```
pets = ["dog", "cat", "dog", "pig", "goldfish", "rabbit", "cat"]
```

In [5]:

```
while "cat" in pets:
    pets.remove("cat")
pets
```

Out[5]:

['dog', 'dog', 'pig', 'goldfish', 'rabbit']

【小例子】将未读书籍列表中书名分别输出后，存入已读书籍列表

In [6]:

```
not_read = ["红楼梦", "水浒传", "三国演义", "西游记"]
have_read = []
while not_read:      # not_read非空，循环继续，否则中止
    book = not_read.pop()
    have_read.append(book)
    print("我已经读过《{}》了".format(book))
print(not_read)
print(have_read)
```

```
我已经读过《西游记》了
我已经读过《三国演义》了
我已经读过《水浒传》了
我已经读过《红楼梦》了
[]
['西游记', '三国演义', '水浒传', '红楼梦']
```

第五部分 控制语句注意问题

5.1 尽可能减少多层嵌套

- 可读性差，容易把人搞疯掉

In [ ]:

```
if 条件:
    执行语句
    if 条件:
        执行语句
        if...
```

5.2 避免死循环

条件一直成立，循环永无止境

In [ ]:

```
# while True:
    # print("天地之渺渺，时间之无限")
```

5.3 封装过于复杂的判断条件

如果条件判断里的表达式过于复杂

出现了太多的 not/and/or等

导致可读性大打折扣

考虑将条件封装为函数

In [ ]:

```
a, b, c, d, e = 10, 8, 6, 2, 0
if (a > b) and (c >d) and (not e):
    print("我已经晕鸟")
```

In [7]:

```
numbers = (10, 8, 6, 2, 0)


def judge(num):
    a, b, c, d, e = num
    x = a > b
    y = c > d
    z = not e
    return x and y and z


if judge(numbers):
    print("就是这个feel，biu倍儿爽")
```

```
就是这个feel，biu倍儿爽
```

In [ ]:

```

```