大学计算机基础

1

课程内容

- 逻辑运算
- 逻辑电路
- 数据的存储
- 程序设计：分支结构

2

逻辑运算

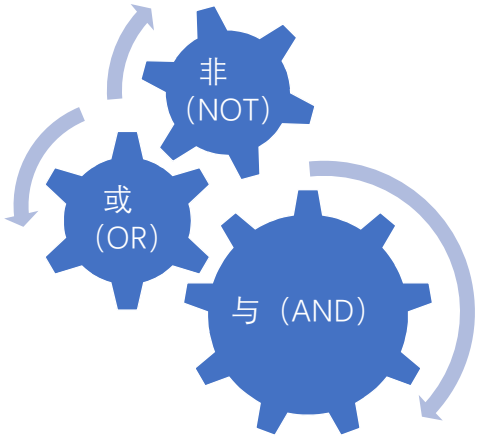
3

逻辑

- 用数学的方法研究关于推理，证明等问题的学科
- 逻辑关系可以被解释成因果关系：
  - 因是条件
  - 条件之间的关系用逻辑连接词组合
  - 根据不同条件得到结果

4

逻辑关系



5

逻辑关系——与

- 运算符：AND
- 特点：两个输入，一个输出；所有条件都真，结果为真
- 表示：A·B，A AND B，AB
- 真值表

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

6

逻辑关系——或

- 运算符：OR
- 特点：两个输入，一个输出；一个条件为真，结果为真
- 表示：A OR B，A+B
- 真值表

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

7

逻辑关系——非

- 运算符：NOT
- 特点：一个输入，一个输出；输入条件为真，结果为假
- 表示：NOT A，$\overline{A}$
- 真值表

A	$\overline{A}$
0	1
1	0

8

逻辑代数

- 定义：用基本逻辑关系与、或、非的运算符连接逻辑变量表示
- 作用：通过代数学的方法研究逻辑关系，通过变换、简化或组合等方法进行逻辑设计。
- 基本公式：见课本表2-6

9

逻辑代数

- 证明分配率(A+B)(A+C)=A+BC
- 证明：

(A+B)(A+C)

=AA+AB+AC+BC

=AA+A(B+C)+BC

=A+A(B+C)+BC

=A(1+B+C)+BC

=A+BC

10

逻辑电路

11

逻辑电路

- 定义：是一种电子线路，制造计算机硬件的基本电路
- 作用：实现逻辑运算
- 单元电路：门电路

12

逻辑电路——门电路

- 定义：是逻辑电路中的单元电路
- 作用：通过高低电平对应逻辑状态的真（1）和假（0）
- 基本门电路

13

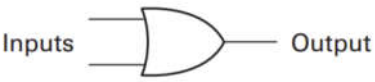
逻辑电路——门电路

AND

Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

与门F=AB

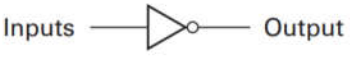
OR



Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

或门F=A+B

NOT



Inputs	Output
0	1
1	0

非门F= $\bar{A}$

14

逻辑电路——组合门电路

- 使用基本门电路可以组成各种复杂的逻辑功能电路。
- 与非门：$\overline{AB}$
- 或非门：$\overline{A+B}$
- 异或门：F=A⊕B= $A\overline{B}+\overline{A}B$

与非门

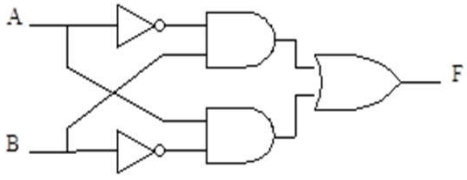
或非门

异或门

15

逻辑电路——异或门

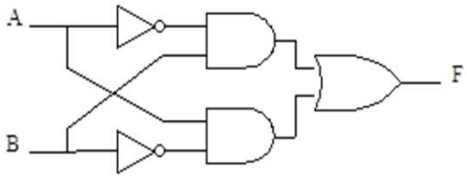
- 表达式
- 基本逻辑电路
- 真值表



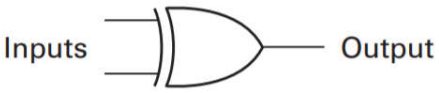
16

逻辑电路——异或门

- 表达式
- 基本逻辑电路
- 真值表



XOR



Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

17

逻辑电路的用途?

组合设计处理器中的ALU，从而实现：

- 算术运算
- 逻辑运算

18

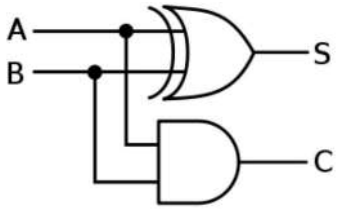
加法器

定义：用逻辑电路实现加法运算的电路叫做加法器（Adder）

半加器：A和B分别是一位二进制数，S为A与B之和，C为A加B产生的进位

和数表达式：$S = A\overline{B} + \overline{A}B$

进位表达式：$C = AB$



19

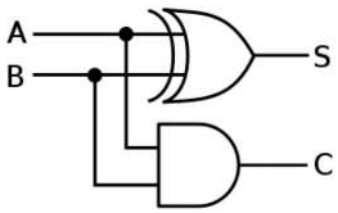
加法器

定义：用逻辑电路实现加法运算的电路叫做加法器（Adder）

半加器：A和B分别是一位二进制数，S为A与B之和，C为A加B产生的进位

和数表达式：$S = A\overline{B} + \overline{A}B$

进位表达式：$C = AB$

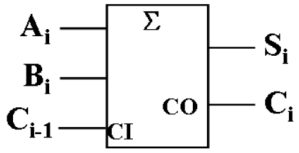


A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

20

全加器

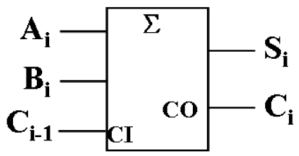
全加器=半加器+来自低位的进位

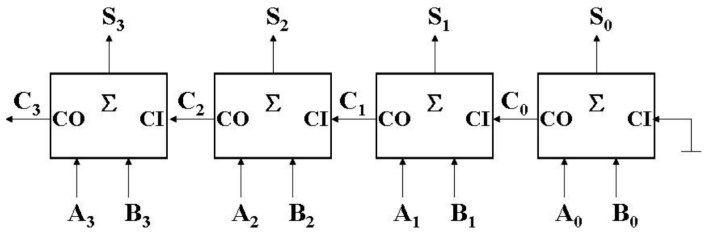


21

全加器

全加器=半加器+来自低位的进位





22

计算机如何进行乘法运算

计算15(x)*13(y) 即1111*1101

a.首先y的最低位为1(2^0)，x左移0位得到1111

b.然后y的最低第二位为0，没有2^1存在，因此本次无运算

c.然后y的最低第三位为1(2^2)，x左移2位得到111100

d.然后y的最低第四位为1(2^3)，x左移3位得到1111000

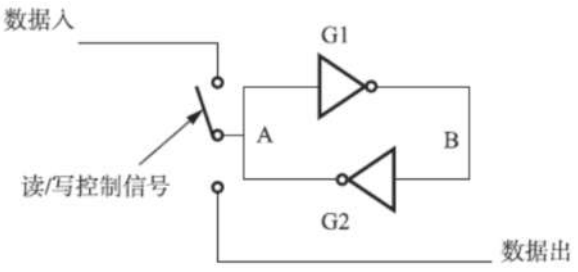
e.把a、b、c、d的结果相加1111+0+111100+1111000=11000011(195),该结果就是乘法的结果

23

存储器

存储器由存储单元电路组成，而存储单元电路也由门电路构成。

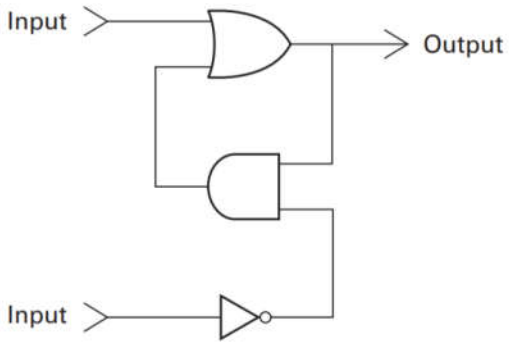
锁存器和触发器



24

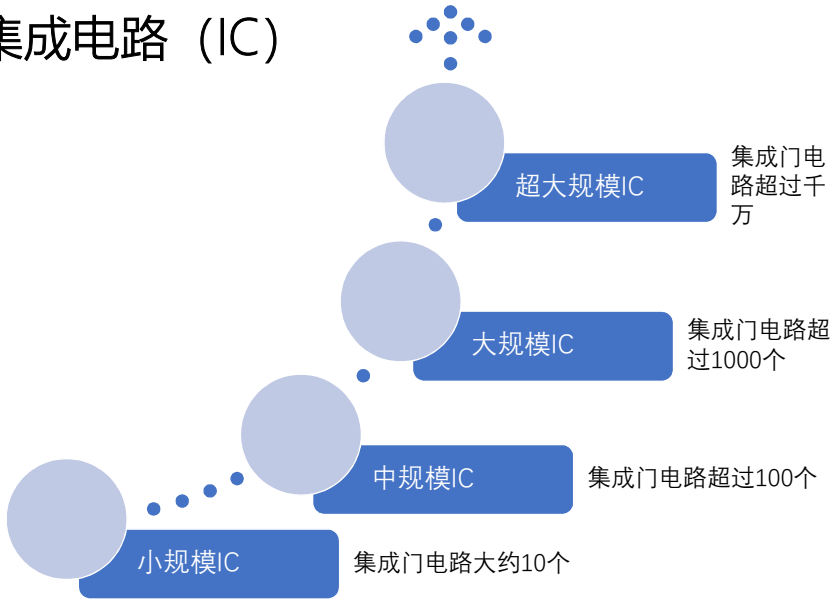
触发器

触发器是计算机存储器的基本部件，它是一个可以产生0或1输出值的电路，它的值会一直保持不变，直到有一个电路过来的临时脉冲，使其变换成其他值。



25

集成电路（IC）



26

数据的存储

27

数据编码

数据按功能可分为两类：数和码

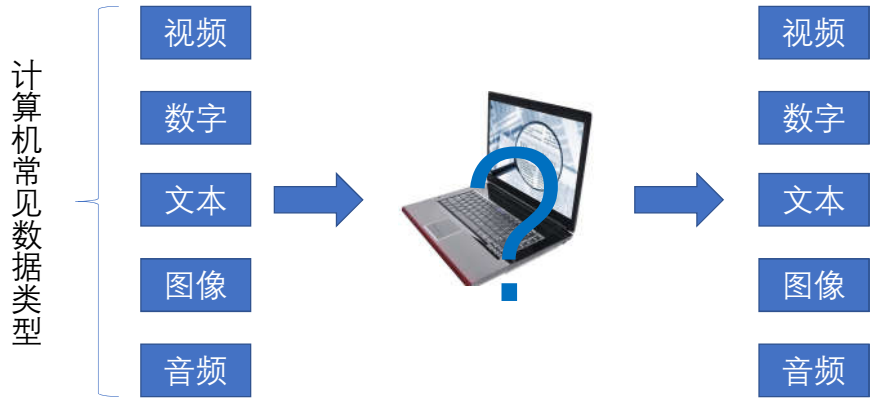
数的作用：表示量

码的作用：对特定对象进行唯一标识，以便检索、交换和处理

码制：编码的规则

28

数据的存储



29

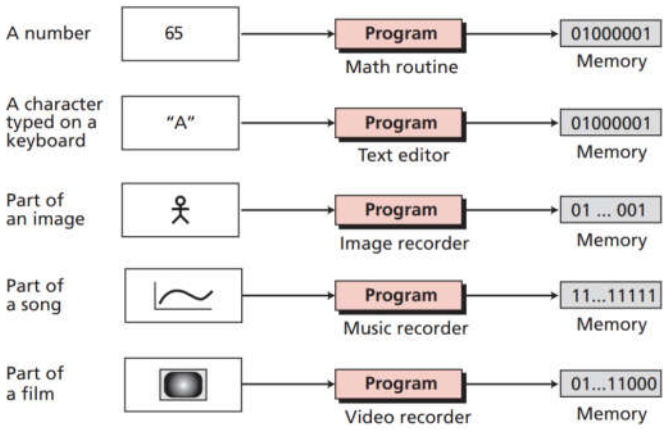
数据的存储——位

在现今的计算机中，信息是以0和1的模式编码的，这些数字称为**位（bit）**，尽管你可能倾向把他们与数值联系起来，但它们的确只是一些符号，其意义取决于正在处理的应用，他们有时表示数字，有时表示数字里的字符和标点符号，有时表示图像，还有时表示声音。

30

数据的存储——位模式

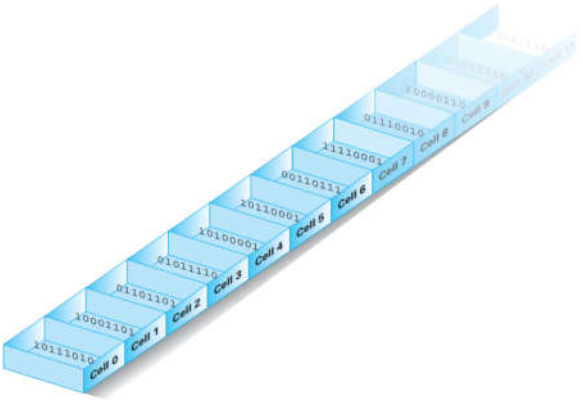
一个由0和1二进制位序列，用来表示数据的不同类型，长度为8的位模式被称为一个字节。



31

数据的存储——主存储器

- 计算机包含大量的电路，每一个电路能够存储一个位，被称为计算机的**主存储器**。
- 计算机的主存储器是由**存储单元（Cell）**组成，典型的存储单元容量为8位。
- 每一个存储单元都被赋予了唯一的地址。



32

数据的存储——存储器容量

- 千字节（KB）= 1024 Byte
- 兆字节（MB）=1024 KB
- 吉字节（GB）= 1024 MB
- 太字节（TB）= 1024 GB

33

数据的存储——存储数字

存储整数:

通常使用**定点表示法**

原码，反码，补码

存储实数

通常使用**浮点表示法**

包括：数的符号、阶码符号、阶码值和尾数

34

程序设计基础：分支结构

35

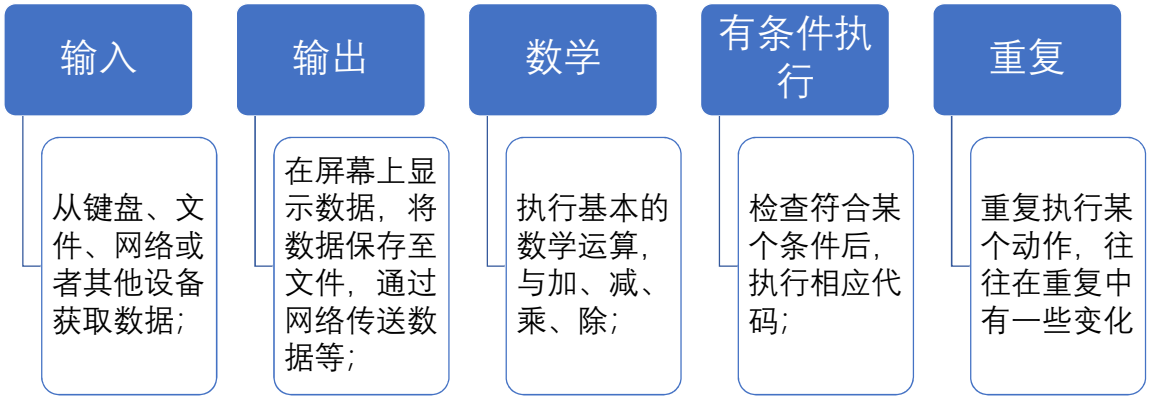
随堂练习讲解

1. 新建一个hw1_3.py文件，并编写一个程序。要求在程序中读入一个表示星期几的数字（1~7），输出对应的星期字符串名称。例如，输入3，返回“星期三”。（提示：使用一个字符串变量存储字符串“星期一星期二星期三星期四星期五星期六星期日”并使用字符串切片）

```
c=int(input('请输入1-7其中一个数字'))
d='星期一星期二星期三星期四星期五星期六星期天'
print(d[3*(c-1):c*3])
```

36

程序的基本指令



37

布尔值

“布尔”数据类型只有两个值：True和False。在解释器中输入以下代码并得到如下结果：

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> Var=True
>>> Var
True
>>>
```

这里的True不是字符串，两边没有引号，它是变量Var的值。请使用Type()函数，查看变量Var的数据类型

38

运算符

算术运算符

关系运算符

逻辑预算符

39

关系操作符

关系运算符的作用是用来比较两个值，其比较结果是一个布尔值。

下表列出来常见的关系运算符:

操作符	数学符号	操作符含义
<	<	小于
<=	≤	小于等于
>=	≥	大于等于
>	>	大于
==	=	等于
!=	≠	不等于

40

关系操作符

如果两边的值一样， ==求值为 True。如果两边的值不同，!=（求值为 True。 ==和!=操作符实际上可以用于所有数据类型的值。

```
>>> 42 == 42
True
>>> 42 == 99
False
>>> 2 != 3
True
>>> 2 != 2
False
```

```
>>> 'hello' == 'hello'
True
>>> 'hello' == 'Hello'
False
>>> 'dog' != 'cat'
True
>>> True == True
True
>>> True != False
True
>>> 42 == 42.0
True
❶ >>> 42 == '42'
False
```

41

关系操作符

<、 >、 <=和>=操作符主要用于整型和浮点型值。

```
>>> 42 < 100
True
>>> 42 > 100
False
```

```
>>> 42 < 42
False
>>> eggCount = 42
❷ >>> eggCount <= 42
True
>>> myAge = 29
❸ >>> myAge >= 10
True
```

42

逻辑操作符

如何比较两个布尔值？需要用到逻辑运算符。Python中有三个逻辑运算符：and，or和not。

在解释器中输入(4<5) and (5<6)，(4<5) and (9<6)和(1==2) or (2==2)三个代码，显示结果为？

43

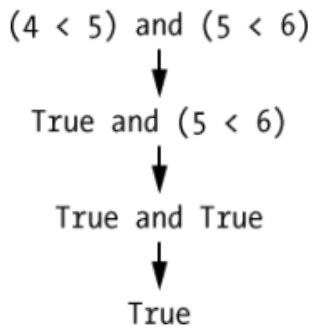
逻辑操作符

思考一下计算机的求值过程，以(4<5) and (5<6)为例，求值过程为？

44

逻辑操作符

思考一下计算机的求值过程，以(4<5) and (5<6)为例，求值过程为？



45

程序结构

程序有三种基本结构组成：顺序结构、分支结构和循环结构

顺序结构：按照代码出现的先后顺序依次执行；

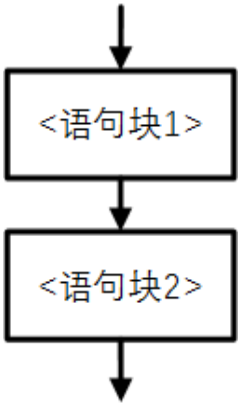
分支结构：根据条件判断是否执行相关语句，又包括单分支结构，二分支结构和多分支结构。

循环结构：当条件成立时，重复执行某些语句。

46

顺序结构

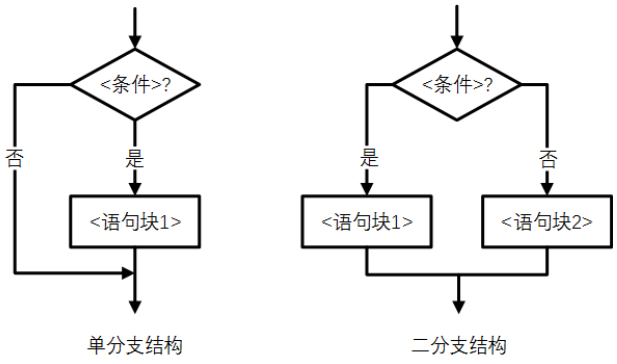
顺序结构是程序按照线性顺序依次执行的一种运行方式，其中语句块S1和语句块S2表示一个或一组顺序执行的语句



47

分支结构

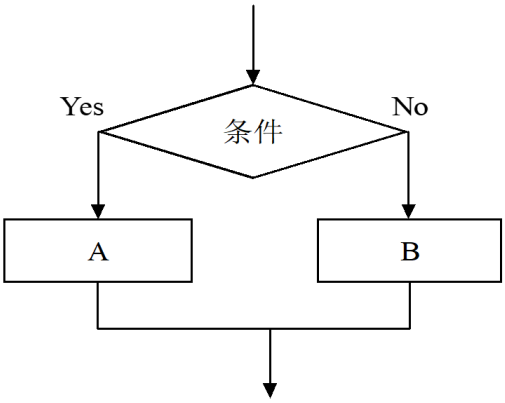
分支结构是程序根据条件判断结果而选择不同向前执行路径的一种运行方式，包括单分支结构和二分支结构。由二分支结构会组合形成多分支结构



48

分支结构

- 条件判断
- 分支代码块



49

单分支结构if语句

Python中if语句的语法格式如下：

if <条件>:

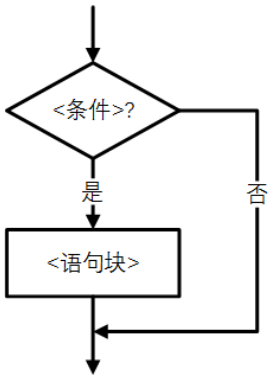
语句块

- 语句块是if条件满足后执行的一个或多个语句序列
- 语句块中语句通过与if所在行形成缩进表达包含关系
- if语句首先评估<条件>的结果值，如果结果为True，则执行语句块里的语句序列，然后控制转向程序的下一条语句。如果结果为False，语句块里的语句会被跳过。

50

单分支结构if语句

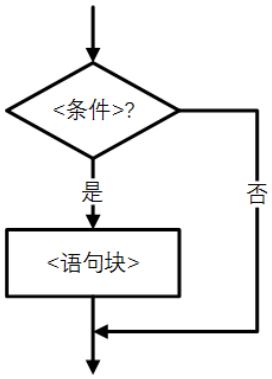
if语句中语句块执行与否依赖于条件判断。但无论什么情况，控制都会转到if语句后与该语句同级别的下一条语句。



51

单分支结构if语句

如何判断条件是否成立?



52

单分支结构if语句

if语中<条件>部分可以使用任何能够产生True或False的语句

形成判断条件最常见的方式是采用关系操作符

Python语言共有6个关系操作符

操作符	数学符号	操作符含义
<	<	小于
<=	≤	小于等于
>=	≥	大于等于
>	>	大于
==	=	等于
!=	≠	不等于

53

单分支结构if语句实例

PM 2.5空气质量提醒

输入：接收外部输入PM2.5值

处理：

if PM2.5值 >= 75，打印空气污染警告

if 35 <= PM2.5值 < 75，打印空气污染警告

if PM2.5值 < 35，打印空气质量优，建议户外运动

输出：打印空气质量提醒

54

单分支结构if语句实例

条件判断?

分支代码块?

55

单分支结构if语句实例

```
PM = eval(input("请输入PM2.5数值: "))
if 0<= PM < 35:
    print("空气优质，快去户外运动!")
if 35 <= PM <75:
    print("空气良好，适度户外活动! ")
if 75 <= PM:
    print("空气污染，请小心! ")
```

56

二分支结构: if-else语句

Python中if-else语句用来形成二分支结构，语法格式如下：

```
if <条件>:
    <语句块1>
else:
    <语句块2>
```

- <语句块1>是在if条件满足后执行的一个或多个语句序列
- <语句块2>是if条件不满足后执行的语句序列
- 二分支语句用于区分<条件>的两种可能True或者False，分别形成执行路径

57

二分支结构: if-else语句

微实例4.5：PM 2.5空气质量提醒（2）

```
PM = eval(input("请输入PM2.5数值： "))
if PM >= 75:
    print("空气存在污染，请小心！ ")
else:
    print("空气没有污染，可以开展户外运动！")
```

58

二分支结构: if-else语句

二分支结构还有一种更简洁的表达方式，适合通过判断返回特定值，语法格式如下：

<表达式1> if <条件> else <表达式2>

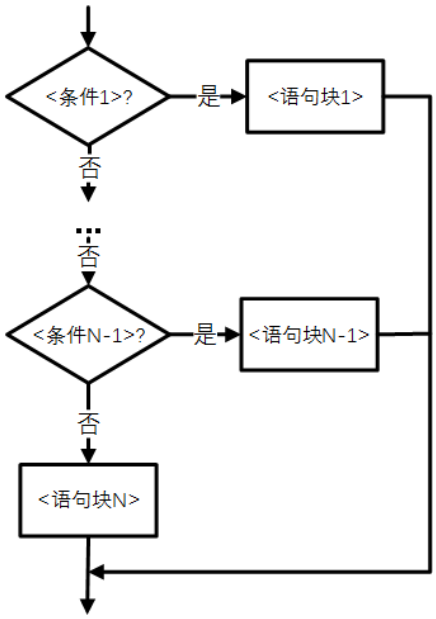
```
PM = eval(input("请输入PM2.5数值: "))
print("空气{}污染!".format("存在" if PM >= 75 else "没有"))
```

59

多分支结构: if-elif-else语句

Python的if-elif-else语句格式如下：

if <条件1>:
    <语句块1>
elif <条件2>:
    <语句块2>
...
else:
    <语句块N>



60

多分支结构: if-elif-else语句

- 多分支结构是二分支结构的扩展，这种形式通常用于设置同一个判断条件的多条执行路径。
- Python依次评估寻找第一个结果为True的条件，执行该条件下的语句块，同时结束后跳过整个if-elif-else结构，执行后面的语句。如果没有任何条件成立，else下面的语句块被执行。else子句是可选的

61

多分支结构: if-elif-else语句

微实例4.4通过多条独立的if语句对同一个变量PM进行判断，这种情况更适合多分支结构，代码如下：

```
PM = eval(input("请输入PM2.5数值: "))
if 0<= PM < 35:
    print("空气优质，快去户外运动!")
elif 35 <= PM <75:
    print("空气良好，适度户外活动! ")
else:
    print("空气污染，请小心! ")
```

62