

PYTHON程序设计

张树兵

20230228改

基本语法

Python基本语法元素

- 缩进、注释、命名、变量、保留字
- 数据类型、字符串、整数、浮点数、列表
- 赋值语句、分支语句、函数
- input()、print()、eval()、print()格式化

缩进

缩进表达程序的格式框架

- **严格明确**：缩进是语法的一部分，缩进不正确程序运行错误
- **所属关系**：表达代码间包含和层次关系的唯一手段
- **长度一致**：程序内一致即可，一般用4个空格或1个TAB

缩进 if例子

- Python 缩进是一种编程规则，它规定了代码段之间的关系。缩进的意思就是把代码段向右移动，以示它们之间的关系。
- 例如：

if 2 > 1:

```
    print("2大于1")  
    print("这句话会被执行")
```

```
print("这句话不会被执行")
```

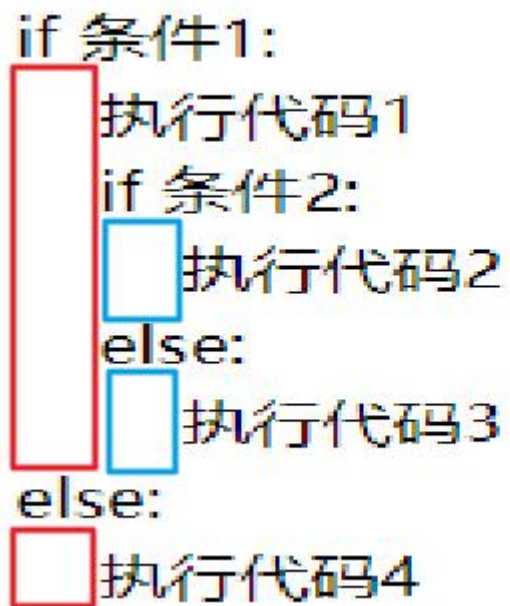
上面的代码中，if语句后面的两句print语句都被缩进了，表示它们是if语句的一部分，只有当条件满足时，它们才会被执行。最后一句print语句没有被缩进，它不属于if语句，所以不管if语句的条件是否满足，它都会被执行。

多层缩进：if语句

- Python 缩进是指给程序添加空格，以此来表示代码块的层次结构。

- 举个例子：

```
if 条件1:  
    执行代码1  
    if 条件2:  
        执行代码2  
    else:  
        执行代码3  
else:  
    执行代码4
```



- Python 缩进就是为了表示上面这个程序中的代码块之间的关系，
- 比如代码1 和代码4 是同一层次的，而代码2 和代码3 则是同一层次
- 次的，Python 会根据空格的多少来区分不同层次的代码块。

缩进复杂例子

200440024王玮楠.py ×

进入循环的条件判断:



```
1 while (1):  
2     province=input("您来自哪个省/区/市? ")  
3     if province=="山东":  
4         city=input("您来自哪个城市? ")  
5         if city=="菏泽":  
6             county=input("您来自哪个区/县? ")  
7             if county=="曹县":  
8                 print("山东菏泽曹县, 牛批, 666, 我滴宝贝!!! ")  
9             else:  
10                print("欢迎您来到中国民航大学! ")  
11        else:  
12            print("欢迎您来到中国民航大学! ")  
13    else:  
14        print("欢迎您来到中国民航大学! ")  
15  
16
```

条件不成立, 退出循环



注释

不被程序执行的辅助性说明信息

- 单行注释：以#开头，其后内容为注释

这里是单行注释

- 多行注释：以'''开头和结尾

'''

这是多行注释第一行

这是多行注释第二行

'''

注释

Python 注释是给程序员用来解释代码的文字。比如，

```
# 这是一个注释
```

```
# 这句代码将变量a设置为5
```

```
a = 5
```

- 这句注释的作用是告诉程序员，下一行代码将变量a设置为5。

注释

- Python 注释就是用来帮助我们理解代码的文字说明。比如，我们可以在程序中添加一行注释：

```
# 这是一个注释，它不会影响程序的执行  
print("Hello World!")
```

上面的代码会输出 “Hello World!”，但注释里的文字**不会被执行**。我们把它放在代码上，是因为它能帮助我们了解代码的作用，**提高程序的可读性**。

保留字（关键字）

被编程语言内部定义并保留使用的标识符

- Python语言有33个保留字(也叫关键字)

if, elif, else, in

- 保留字是编程语言的基本单词，大小写敏感

if 是保留字，If 是变量

保留字 (关键字)

保留字

| | | | | |
|----------|---------|--------|--------|----------|
| and | elif | import | raise | global |
| as | else | in | return | nonlocal |
| assert | except | is | try | True |
| break | finally | lambda | while | False |
| class | for | not | with | None |
| continue | from | or | yield | |
| def | if | pass | del | |

(26/33)

保留字（关键字）

- Python 保留字是预先定义的特殊字符或关键字
- 它们表示特定的程序指令，不能用作变量、函数和其他标识符的名称。
- 例如1，你不能用`and`作为一个变量的名字，因为它是Python的保留字，它只能用来表示逻辑与的意思。
- 例如2：假设你想创建一个叫`print`的变量，但是`print`是python的一个保留字，所以你不能用它来作为变量的名字，你必须用别的名字。
- 比如你可以把它改成`printer`。

保留字（关键字）

- Python 保留字是特殊的关键字，它们用于定义Python语言的语法和结构。
- 比如，“if” 是一个Python保留字，它告诉程序，如果某个条件
- 为真，那么就执行某些操作。

if 3 > 2:

print("3 is bigger than 2!")

赋值语句

由赋值符号构成的一行代码

- 赋值语句用来给变量赋予新的数据值

`C=(eval(F)-32)/1.8` #右侧运算结果赋给变量C

- 赋值语句右侧的数据类型同时作用于变量

`TempStr=input("")` #input()返回一个字符串，TempStr也是字符串

赋值语句

- Python赋值语句就是把一个值存储到一个变量中。
- 比如，我们可以把数字5赋值给变量“a”：就像我们拿书包一样，
- 变量就像书包，里面装的东西就是赋给它的值。

a = 5

- 这样，变量a就保存了数字5，以后可以用a来表示数字5。

b = (4+5)*6

- 你可以把 (4+5) *6的结果赋给变量b：这样变量b的值就是54了。

分支语句

由判断条件决定程序运行方向的语句

- 使用保留字 *if elif else* 构成条件判断的分支结构

```
if int(height)>170:
```

```
    print("你好高呀，太帅了")
```

#如果条件为True则执行冒号后语句

- 每个保留字所在行最后存在一个冒号(:)，语法的一部分

冒号及后续缩进用来表示后续语句与条件的所属关系

分支语句： if

- Python分支语句就像是在让电脑做一个决定，可以根据不同的情况做出不同的反应。
举个例子：

- 假设你想要让电脑根据你的答案来判断你喜欢什么颜色： **(要表达的意思)**

如果你回答"蓝色"，电脑就会显示："你喜欢蓝色！"

如果你回答"红色"，电脑就会显示："你喜欢红色！"

- 这就是Python分支语句的一个简单例子。 **(逐行翻译成 python代码)**

```
color=input("输入你喜欢的颜色： 一个汉字 ")
```

```
if color=="蓝":
```

```
    print("你喜欢的颜色是蓝色")
```

```
if color=="红":
```

```
    print("你喜欢的颜色是红色")
```

分支语句

```
1 color=input("输入你喜欢的颜色：一个汉字 ")
2
3 if color=="蓝":
4     print("你喜欢的颜色是蓝色")
5 if color=="红":
6     print("你喜欢的颜色是红色")
```

Shell ×

```
>>> %Run 'if入门1.py'
```

```
输入你喜欢的颜色：一个汉字 蓝
你喜欢的颜色是蓝色
```

```
>>> %Run 'if入门1.py'
```

```
输入你喜欢的颜色：一个汉字 红
你喜欢的颜色是红色
```

```
>>> |
```

分支语句 if else

- Python 分支语句可以控制程序的流程，来决定程序应该做什么。例如，当我们想要根据某个条件来决定一个人是否可以进入一个游乐场时，可以使用分支语句：

如果一个人的年龄大于18岁，那么他可以进入游乐场；

如果一个人的年龄小于18岁，那么他不能进入游乐场。

python代码：

```
age=int(input("输入你的年龄："))
```

```
if age>=18:
```

```
    print("你可以进入游乐场")
```

```
else:
```

```
    print("你不可以进入游乐场")
```


分支语句

```
1 age=int(input("输入你的年龄："))
2
3 if age>=18:
4     print("你可以进入游乐场")
5 else:
6     print("你不可以进入游乐场")
```

Shell ×

```
>>> %Run 'if入门2.py'
```

```
输入你的年龄： 22
你可以进入游乐场
```

```
>>> %Run 'if入门2.py'
```

```
输入你的年龄： 6
你不可以进入游乐场
```

```
>>> |
```


分支语句 if else

- Python 分支语句可以让程序按照不同的条件来执行不同的操作，举个简单的例子：
- 假设你要写一个程序，当用户输入的数字大于5时，程序就会打印出"大于5"；如果用户输入的数字小于等于5，程序就会打印出"小于等于5"。
- 这时就可以用Python的分支语句来实现：

```
num = int(input('请输入一个数字： '))
```

```
if num > 5:
```

```
    print(num, '大于5')
```

```
else:
```

```
    print(num, '小于等于5')
```

分支语句

```
1 num = int(input('请输入一个数字: '))
2 if num > 5:
3      print(num, '大于5')
4 else:
5      print(num, '小于等于5')
```

Shell ×

```
>>> %Run 'if入门4.py'
```

请输入一个数字: 9

9 大于5

```
>>> %Run 'if入门4.py'
```

请输入一个数字: 3

3 小于等于5

```
>>> |
```

分支语句 if elif

- Python 分支语句可以让程序按照不同的条件来执行不同的操作，举个简单的例子：
- 假设你要写一个程序，当用户输入的数字大于5时，程序就会打印出"大于5"；如果用户输入的数字小于等于5，程序就会打印出"小于等于5"。
- 这时就可以用Python的分支语句来实现：

```
num = int(input('请输入一个数字： '))
```

```
if num > 5:
```

```
    print(num, '大于5')
```

```
elif num <= 5:
```

```
    print(num, '小于等于5')
```

分支语句

```
1 num = int(input('请输入一个数字: '))
2 if num > 5:
3      print(num, '大于5')
4 elif num <= 5:
5      print(num, '小于等于5')
```

Shell ×

```
>>> %Run 'if入门3.py'
```

```
请输入一个数字: 9
9 大于5
```

```
>>> %Run 'if入门3.py'
```

```
请输入一个数字: 3
3 小于等于5
```

```
>>>
```

分支语句 if elif elif ...else

- Python 多分支语句是一种条件判断语句，它可以根据不同的条件，选择不同的操作。例如：
- 如果你的年龄是7岁，那么你应该去上一年级；
- 如果你的年龄是8岁，那么你应该去上二年级；
- 如果你的年龄是9岁，那么你应该去上三年级；
- 如果你的年龄不是7、8、9岁，那么你是一个没进笼子的神兽
- 以此类推...
- Python 多分支语句的语法如下：

if 条件1:

 **执行语句1**

elif 条件2:

 **执行语句2**

else:

 **执行语句3**

分支语句 if elif elif ...else

```
1 age=int(input("输入你的年龄: "))
2
3 if age==7:
4     print("你应该去上一年级")
5 elif age==8:
6     print("你应该去上二年级")
7 elif age==9:
8     print("你应该去上三年级")
9 else:
10    print("你是一个没进笼子的神兽")
```

```
>>> %Run 'if入门5.py'
```

输入你的年龄: 7
你应该去上一年级

```
>>> %Run 'if入门5.py'
```

输入你的年龄: 8
你应该去上二年级

```
>>> %Run 'if入门5.py'
```

输入你的年龄: 9
你应该去上三年级

```
>>> %Run 'if入门5.py'
```

输入你的年龄: 18
你是一个没进笼子的神兽

多分支语句:

- Python多分支语句是一种判断语句，可以根据不同的条件执行不同的语句。
- 例如，假设你想要根据某个数字的大小来打印不同的信息：

```
num = 7
```

```
if num > 5:
```

```
    print("数字大于5")
```

```
elif num == 5:
```

```
    print("数字等于5")
```

```
else:
```

```
    print("数字小于5")
```

- 上面的代码中，使用了Python的if-elif-else语句，根据num的值来打印不同的信息。其中，
- if语句判断num是否大于5， elif语句判断num是否等于5， else语句判断num是否小于5。

分支语句 概念:

分支型程序更有价值。最简单的分支语句是条件语句。如图2-3大矩形部分所示，条件语句包括3部分。

- ❑ 一个测试：即一个表达式，取值或者为True，或者为False。
- ❑ 一个代码块：测试条件取值为True时执行。
- ❑ 一个可选代码块：测试条件取值为False时执行。

条件语句执行完毕后，程序会接着执行后面的语句。

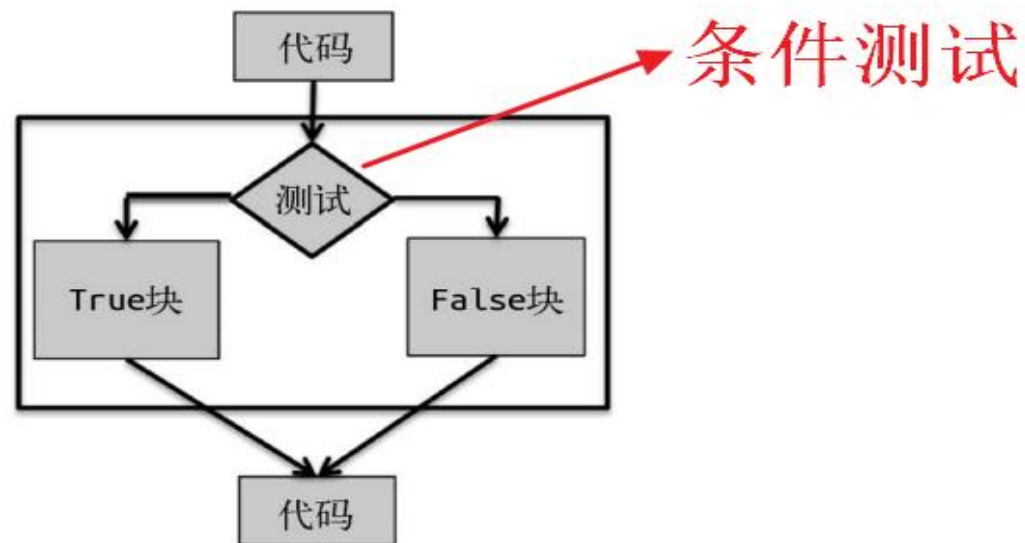


图2-3 条件语句流程图

分支语句语法框架：

在Python中，条件语句具有以下形式：

if 条件:

 **执行语句块1**

else:

 **执行语句块2**

或者：

if 条件:

 **执行语句**

双分支结构

单分支结构

描述Python语句形式的时候，我们用斜体描述可能出现在程序该处的代码。例如，*Boolean expression*表示一个表达式，它的取值为True或False，可以用在if保留字后面。*block of code*表示可以跟在else:后面的Python语句序列。

分支语句语法框架：

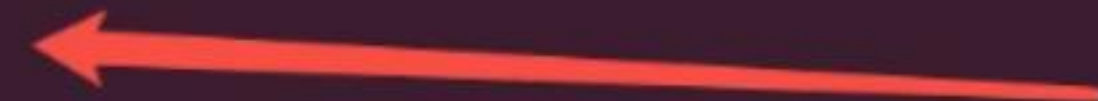
```
if 条件1:  
    执行语句1  
elif 条件2:  
    执行语句2  
elif 条件3:  
    执行语句3  
elif 条件4:  
    执行语句4  
...  
else:  
    执行语句n+1
```

多分支结构

缩进:

```
1 age = 19
2
3 if age>18:
4     print("-----0-----")
5     print("-----1-----")
6     print("-----1-----")
7     print("-----1-----")
8     print("-----1-----")
9     print("-----1-----")
10    print("-----1-----")
11    print("-----1-----")
12    print("-----9-----")
```

缩进: **Tab**键, 实现



相同的缩进:
表示**age>18**, 条件满足时,
执行**9**个**print**语句

缩进:

```
1 age = 10
2
3 if age>18:
4     print("----0-----")
5     print("----1-----")
6     print("----1-----")
7     print("----1-----")
8 else:
9     print("----1-----")
10    print("----1-----")
11    print("----1-----")
12    print("----1-----")
13 print("----9-----")
```

该语句是if语句里的吗?

If 嵌套:

```
if 条件1:  
    if 条件2:  
        代码块 1  
    else:  
        代码块 2  
else:  
    if 条件3:  
        代码块 3  
    else:  
        代码块 4
```

If 嵌套:

```
if 条件1:  
    if 条件2:  
        代码块 1  
    else:  
        代码块 2
```

趣味编程：

- 你可以使用Python编写一个简单的游戏：**猜数字游戏**。
- 游戏的规则是，
- 电脑会想出一个1到100之间的数字，你要尽量猜出它。
- 如果你猜的数字比电脑想出的数字大，则提示你猜的数字太大了；
如果你猜的数字比电脑想出的数字小，则提示你猜的数字太小了；
如果你猜的数字和电脑想出的数字相等，则游戏结束，你赢了。

趣味编程：

- 你可以使用Python编写一个简单的游戏：
 - **猜数字游戏，并统计次数。**
- 游戏的规则是，
- 电脑会想出一个1到100之间的数字，你要尽量用最少数次数猜出它。
- 如果你猜的数字比电脑想出的数字大，则提示你猜的数字太大了；
如果你猜的数字比电脑想出的数字小，则提示你猜的数字太小了；
- 每猜一次，计入猜数一次
- 如果你猜的数字和电脑想出的数字相等，则游戏结束，你赢了。
- 最后输出你的总次数

趣味编程：角色移动躲避障碍物的流程图

- 四个箭头按钮控制角色移动方向，躲避障碍物：
- 我们需要用 Python 创建一个小角色，
- 它在画面中移动，可以用键盘上的箭头按钮来控制小角色的移动方向
- 当小角色碰到障碍物时，它就不能穿越障碍物。
- 最后，我们可以用 Python 让小角色通过所有的障碍物，然后完成这个穿越画面的任务。

趣味编程：角色移动躲避障碍物的具体步骤

- #导入必要的模块pygame
- #正在初始Pygame
- #设置窗口大小
- #定义颜色
- #创建游戏窗口
- #设置窗口标题
- #设置时钟
- #设置角色的坐标
- #设置角色速度
- #定义障碍物
- #设置窗口背景
- #绘图角色
- #绘制障碍物
- #处理用户输入控制方向按钮消息
- #检测与障碍物的碰撞
- #更新显示窗口
- #退出游戏

趣味编程：摘星星流程图

- Python 星星收集游戏的主要开发步骤如下：
 - 1. 创建一个游戏窗口，设置游戏的大小、背景、标题等。
 - 2. 在游戏中添加星星，设置每个星星的位置，大小，颜色等。
 - 3. 加入鼠标移动事件，当鼠标移动到某个星星上时，表示鼠标已经移动到了星星上。
 - 4. 加入随机出现的功能，设置一定的时间间隔，每隔一定时间星星会在不同的位置上随机出现，玩家需要尽快收集到它们。
 - 5. 添加计分功能，每当玩家收集一个星星，分数就会增加。
 - 6. 添加游戏结束的条件，当玩家达到一定的分数时，游戏就会结束

趣味编程：摘星星的具体步骤

- #导入必要的模块
- #设置窗口大小
- #正在初始化Pygame
- #设置窗口
- #定义星星初始位置
- #绘制星空底图
- #游戏循环进行
- # 跟踪鼠标位置
- # 检查鼠标指针是否靠近星星
- # 绘制星星
- # 更新显示
- #退出游戏
- # 输出得分