项目 3

流 程 控 制

在实际生活中，人们经常要面临各种选择或进行周而复始的重复任务，比如根据路程的远近选择出行方式，除了周末没有特殊情况每天都要正常上下班等。程序设计也是如此，代码中的语句默认是按照自上而下的顺序执行，这就是程序设计中最简单的顺序结构，但在解决复杂逻辑问题时，单一的顺序结构有时无法满足要求，此时就需要使用流程控制。流程控制是指在程序执行时，通过一些特定的指令更改程序中语句的执行顺序，使程序产生跳跃、回溯等现象。

本项目将介绍分支结构和循环结构的用法和操作。

学习目标

【掌握】

- 程序的控制结构。
- 选择结构的作用和设计方法。
- 各种选择结构的用法。
- 循环结构的作用和设计方法。
- while 循环的用法。
- for 循环的用法。
- 循环控制语句的用法。

【应用】

- 能够使用 if 语句进行单分支、双分支和多分支结构程序的设计。
- 能够使用 while、for 语句进行循环结构程序的设计。
- 能够使用 break、continue 语句控制循环流程。

项目描述

本项目以中国空间站任务为引导，学习程序流程控制及其实现方法。

首先制作中国空间站组成知识问答，其次制作中国空间站发展历程介绍，最后制作中国空间站轨道参数查询系统。从中国空间站的发展历程，看我国科技的飞速发展和对世界的贡献。在项目案例中贯彻精益求精、开拓创新的工匠精神，激发学生的民族自豪感和

科技自信心，具体项目内容如图 3-1 所示。

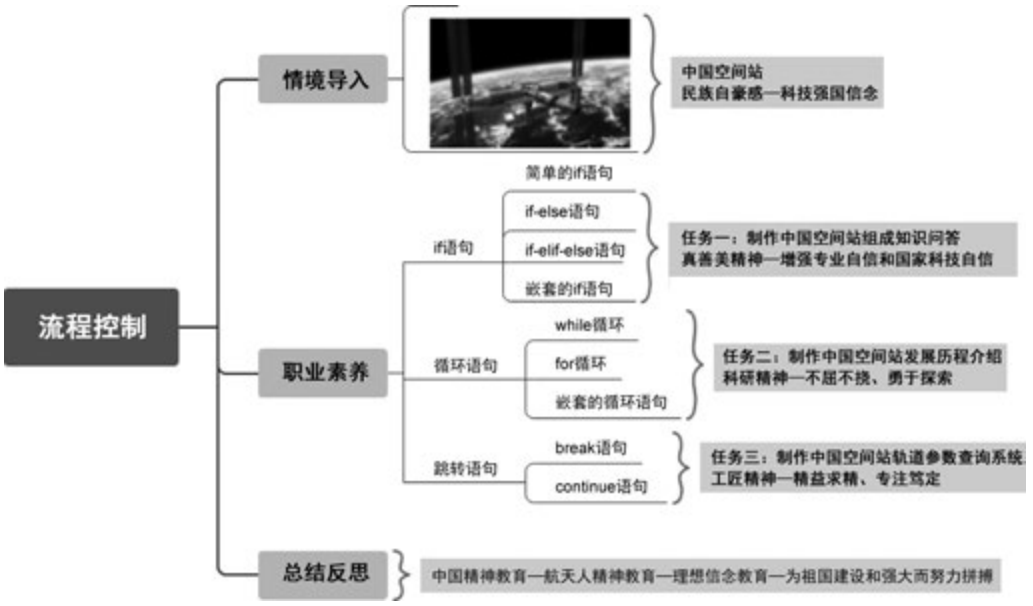

图 3-1 流程控制的项目描述

知识准备

3.1 if 语句

分支结构也称为条件判断结构，是指通过一条或多条语句的执行结果来决定将要执行的代码块，根据分支的数量可分为单分支结构、双分支结构和多分支结构，如图 3-2 所示。

在 Python 中使用 if 语句实现分支结构。

3.1.1 简单的 if 语句

简单的 if 语句是实现单分支结构的方法，该语句由 if 关键字、条件表达式和代码块构成，语法格式如下。

```
if 条件表达式:
  代码块
```

当条件表达式结果为 True 时，执行代码块；反之，不执行代码块。此处的条件表达式可以是一个单纯的布尔值或变量，也可以是比较表达式或逻辑表达式。

单分支结构执行流程图如图 3-3 所示。

单分支 if 语句的使用方法如下：首先使用 Python 基本输出语句设计题干，再使用函数 input()提示用户输入一个答案。使用 if 语句判断答案是否为 B。如果答案正确，则执行语句块输出“回答正确!”；反之，则没有输出。

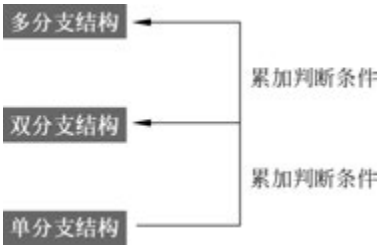

图 3-2　分支结构示意图

图 3-3　单分支结构执行流程图

```
#code_3_1
print("选择题：1992 年,中国政府就制定了载人航天工程()发展战略,建成空间站是发展战略的
重要目标。")
print("A.两步走")
print("B.三步走")
print("C.四步走")
print("D.五步走")
num = input("请输入一个答案\n")
if num == "B":
    print("回答正确!")
```

运行结果如下。

```
选择题：1992 年,中国政府就制定了载人航天工程()发展战略,建成空间站是发展战略的重要
目标。
A.两步走
B.三步走
C.四步走
D.五步走
请输入一个答案
B
回答正确!
```

上述代码中,若用户输入错误答案系统需要输出“回答错误!”的结果,可将程序修改为如下所示：用户在控制台输入了 D 选项,选项 D 与正确答案 B 不符,第 1 个 if 条件不满足,不执行输出“回答正确!”语句。程序按顺序向下执行,满足第 2 个 if 语句的条件表达式答案不是 B,执行输出“回答错误!”语句。由此,本程序的运行结果为输出“回答错误!”。

```
#code_3_2
print("选择题：1992 年,中国政府就制定了载人航天工程()发展战略,建成空间站是发展战略的
重要目标。")
print("A.两步走")
```

```
print("B.三步走")
print("C.四步走")
print("D.五步走")
num = input("请输入一个答案\n")
if num == "B":
    print("回答正确!")
if num!= "B":
    print("回答错误!")
```

运行结果如下。

```
选择题:1992 年,中国政府就制定了载人航天工程()发展战略,建成空间站是发展战略的重要目标。
A.两步走
B.三步走
C.四步走
D.五步走
请输入一个答案
D
回答错误!
```

3.1.2 if-else 语句

Python 的双分支结构使用 if-else 语句实现,语法格式如下。

```
if 条件表达式:
    代码块 1
else:
    代码块 2
```

与单分支结构对比,双分支结构是当条件表达式结果为 True 时,执行代码块 1;反之,执行 else 中的代码块 2。

注意:if 和 else 必须对齐,代码块 1 和代码块 2 必须有相同的缩进。

双分支结构执行流程图如图 3-4 所示。

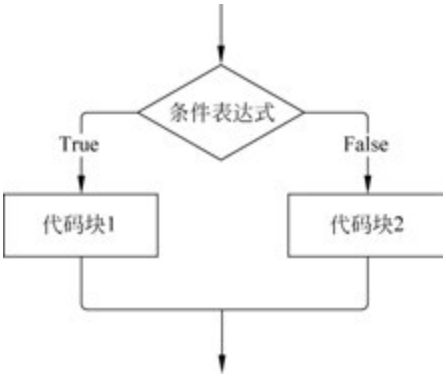

图 3-4 双分支结构执行流程图

双分支 if-else 语句的使用方法如下:用户在控制台输入了选项 A,由于 A 与正确答案 B 不符,if 语句的条件表达式不满足,执行 else 中的语句块输出“回答错误!”。

```
#code_3_3
print("选择：1992 年,中国政府就制定了载人航天工程()发展战略,建成空间站是发展战略的重要目标。")
print("A.两步走")
print("B.三步走")
print("C.四步走")
print("D.五步走")
num = input("请输入一个答案\n")
if num == "B":
    print("回答正确!")
else:
    print("回答错误!")
```

运行结果如下。

```
选择：1992 年,中国政府就制定了载人航天工程()发展战略,建成空间站是发展战略的重要目标。
A.两步走
B.三步走
C.四步走
D.五步走
请输入一个答案
A
回答错误!
```

3.1.3　if-elif-else 语句

多分支结构能够对多个分支进行判断,根据不同条件的执行结果执行不同的分支语句。Python 中使用 if-elif-else 语句实现多分支结构,语法格式如下。

```
if 条件表达式 1:
  代码块 1
elif 条件表达式 2:
    代码块 2
elif 条件表达式 3:
    代码块 3
…
elif 条件表达式 n:
    代码块 n
else:
    代码块 n + 1
```

多分支结构首先判断条件表达式 1 是否为 True,如是,则执行代码块 1 后结束整个 if 语句；否则判断条件表达式 2 是否为 True,如是,则执行代码块 2 后结束整个 if 语句；以此类推,如果条件表达式 n 结果为 True,则执行代码块 n 后结束整个 if 语句；否则执行代码块 $n+1$ 后结束整个 if 语句。

多分支结构执行流程图如图 3-5 所示。

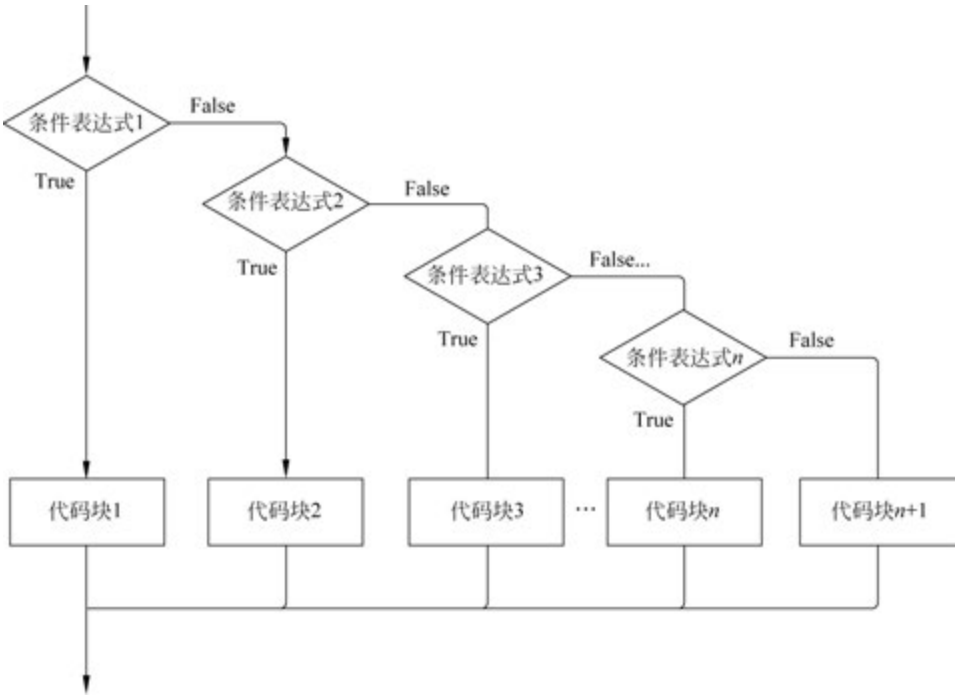

图 3-5　多分支结构执行流程图

多分支 if-elif-else 语句的使用方法如下：用户输入 2024，由于 2024 不等于 2021，if 语句的条件表达式结果为 False，输出恭喜答对的语句不执行。程序继续执行 elif 语句，继续判断是否条件表达式 2024＜2021，结果为 False，输出猜得早了一些的语句不执行。程序继续执行 else 语句，执行输出猜得晚了一些的语句。

```
#code_3_4
print("请猜一猜长征五号 B 遥二运载火箭成功将空间站首个舱段——天和核心舱送入太空是在
哪一年?")
num = int(input("请输入年份: \n"))
if num == 2021:
    print("恭喜你,答对啦!")
elif num < 2021:
    print("你猜得早了一些,记住应该是 2021 年。")
else:
    print("你猜得晚了一些,记住应该是 2021 年。")
```

运行结果如下。

```
请猜一猜长征五号 B 遥二运载火箭成功将空间站首个舱段——天和核心舱送入太空是在哪一年?
请输入年份:
2024
你猜得晚了一些,记住应该是 2021 年。
```

3.1.4　嵌套的 if 语句

if 语句的语句块中除了上述基本语句外，还可以包含 if 语句，这种 if 语句中又包含 if 语句的结构称为 if 语句的嵌套。嵌套 if 语句的语法格式如下。

```
if 条件表达式 1:
  if 条件表达式 1.1:
        代码块 1.1
    elif 条件表达式 1.2:
        代码块 1.2
    elif 条件表达式 1.3:
        代码块 1.3
    ...
    elif 条件表达式 1.n:
        代码块 1.n
    else:
        代码块 1.n + 1
elif 条件表达式 2:
    代码块 2
elif 条件表达式 3:
    代码块 3
...
elif 条件表达式 n:
    代码块 n
else:
    代码块 n + 1
```

嵌套 if 语句可以是单分支、双分支和多分支之间的相互嵌套，使用方法如下：用户在控制台输入了成绩 105。成绩 105 满足大于或等于 0 的条件表达式，执行其对应的 if-elif-else 多分支结果语句。首先判断 105 是否满足＜60 的条件表达式，显然不满足；继续判断是否满足＜70 的条件表达式，依旧不满足；继续判断是否满足＜85 的条件表达式，依旧不满足；继续判断是否满足≤100 的条件表达式，依旧不满足，只能执行 else 语句，输出 105 成绩大于最高分 100，无法判定。

```
#code_3_5
num = int(input("请输入一个 0～100 的成绩\n"))
if num >= 0:
    if num < 60:
        print("%d 为不及格." %num)
    elif num < 70:
        print("%d 为及格." %num)
    elif num < 85:
        print("%d 为良好." %num)
    elif num <= 100:
        print("%d 为优秀." %num)
    else:
        print("%d 成绩大于最高分 100,无法判定。" %num)
```

```
else:
    print("%d 成绩小于最低分 0,无法判定。" %num)
```

运行结果如下。

```
请输入一个 0～100 的成绩
105
105 成绩大于最高分 100,无法判定。
```

3.2 循环语句

循环结构是指在程序中重复执行一条或某几条语句,一般会有一个退出循环的条件,满足这个条件,循环不再继续。循环语句可以减少重复书写相同代码的工作量,需要重复的代码(循环体)只书写一次即可。

Python 中的循环结构主要通过受条件控制的 while 循环语句和受次数控制的 for 循环语句实现。

3.2.1 while 循环

while 循环语句由 while 关键字、循环条件和循环体三部分构成,只要条件表达式结果为 True,就不断重复执行代码块,直至条件表达式不满足时才退出该循环。while 语句的语法格式如下。

```
while 条件表达式:
  循环体
```

此处的条件表达式可以是一个单纯的布尔值或变量,也可以是比较表达式或逻辑表达式。在循环体代码中应包含修改表达式值的语句,否则该循环将被无限执行下去,称为死循环。当循环体语句为多条语句时,必须将这部分语句缩进对齐。

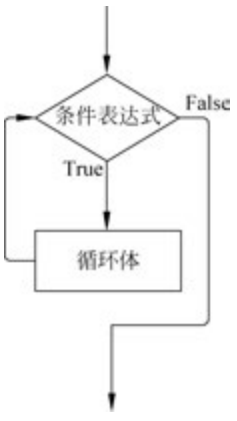

图 3-6 while 语句执行流程图

while 语句执行流程图如图 3-6 所示。

while 语句的使用方法如下:使用函数 input()提示用户输入一个整数,用户通过控制台输入了 100,随后使用函数 int()将该数转换为整数并赋给变量 num。其后又定义了两个变量,sum 用于存放累加和,初始值为 0,i 作为循环变量。开始执行 while 语句,判断是否小于或等于用户输入的数值,由于 0≤100 结果为 True,执行循环体内的语句 sum=sum+i 和 i+=1。执行后,sum 结果为 0,i 变为了 1;再次判断条件表达式 2≤100,结果为 True,执行循环体语句后,sum 值变为 1,i 的值变为 2;继续判断条件表达式 3≤100……直至 i 的值变为 101 时,条件表达式 101≤100 的值变为 False,循环结束,输

出 0＋1＋2＋…＋100 的值为 5050。

```
#code_3_6
num = int(input("请输入一个 0～100 的数\n"))
sum = 0
i = 0
while i <= num:
    sum = sum + i
    i += 1
print("从 0 到 %d 的累加和是 %d。" %(num,sum))
```

运行结果如下。

```
请输入一个 0～100 的数
100
从 0 到 100 的累加和是 5050。
```

3.2.2　for 循环

for 循环语句用于循环次数确定或遍历对象时使用，语法格式如下。

```
for 变量 in 可迭代对象:
  循环体
```

for 语句执行时，依次将对象中的数据赋值给变量（称为迭代）。变量每赋值一次，就执行一次循环体。当 for 语句把可迭代对象中所有元素全部迭代完毕，循环终止。

1. 遍历对象

使用 for 语句遍历对象时使用方法如下：对象为字符串 str，赋值为中国空间站。使用 for 语句对 str 对象进行遍历，依次将中、国、空、间、站赋值给变量 i。每次赋值时执行循环体，输出变量 i 的当前值，即依次输出中、国、空、间、站。

```
#code_3_7
str = "中国空间站"
for i in str:
    print(i)
```

运行结果如下。

```
中
国
空
间
站
```

2. 循环次数确定的循环

for 语句用于实现循环次数确定的程序时，通常可以配合函数 range()实现。

range()函数用于创建一个整数列表，关于列表的相关知识将在后续章节中详细讲解。

函数 range()函数的语法格式如下。

```
range(start,end,step)
```

其中,start 用于定义列表的起始值,若省略则表示从 0 开始; end 用于定义列表的结束值但不包括该值,不可省略; step 用于定义步长,即相邻两个数之间的差值,若省略则步长为 1。

注意:若 range()函数仅有一个参数,即为结束值 end;如有两个参数,则表示起始值 start 和结束值 end。

例如,range(2,6)表示创建含有元素 2、3、4、5 的整数列表。

for 语句和函数 range()使用方法如下:使用函数 input()提示用户输入一个整数,用户通过控制台输入了 100 后使用函数 int()将该数转换为整数并赋给变量 num。其后又定义了变量 sum 用于存放累加和,初始值为 0。开始执行 for 语句,函数 range()创建了一个起始值为 1,结束值为 num+1,即 101,步长为 1 的整数列表,即列表中的元素为 1、2、3、4、5、…、100,根据 for 语句的定义,依次将列表中的整数赋值给 i,每次赋值(迭代)执行一次循环体,将 i 的值累加到 sum 中,直至 100 为止,循环终止。最后,输出 sum 值为从 1 累加到 100 的所有整数之和为 5050。

```
#code_3_8
num = int(input("请输入一个 1～100 的数\n"))
sum = 0
for i in range(1,num + 1):
    sum += i
print("从 1 到 %d 的累加和是 %d。" %(num,sum))
```

运行结果如下。

```
请输入一个 1～100 的数
100
从 1 到 100 的累加和是 5050。
```

3.2.3 嵌套的循环语句

与分支结构相同,循环语句也支持嵌套,且 while 语句和 for 语句既可以自身互相嵌套包含外,两种语句之间也可以相互嵌套。

1. while 循环嵌套

while 循环嵌套是指 while 循环中嵌套 while 循环,语法格式如下。

```
while 条件表达式 1:
    代码块 1
    while 条件表达式 2:
        代码块 2
        …
    …
…
```