辽宁省职业教育精品在线开放课程配套教材
“人工智能+”融媒体一体化教材

Python程序设计工单式教程

主编◎孙　婷　吴　东

北京交通大学出版社
http://www.bjtup.com.cn

内 容 简 介

本书从 Python 语言的基础知识入手，逐步引导学生深入学习 Python 的核心概念，如数据类型、控制结构、函数、类与对象等，进而掌握 Python 编程技术。在内容编排上，本书充分考虑高职学生的认知水平和学习特点，遵循由浅入深、循序渐进的原则。同时，本书还融入了丰富且贴近实际应用的案例，涵盖数据分析、人工智能、Web（万维网）开发等多个热门领域，使学生在学习过程中能够真切感受到 Python 语言的强大功能和广泛应用前景。通过对这些案例的学习和实践，学生能够不断积累经验，提高解决实际问题的能力，为今后从事相关工作奠定坚实的基础。本书适合作为计算机类和通信类专业的教材。

图书在版编目（CIP）数据

Python程序设计工单式教程 / 孙婷，吴东主编. -- 北京 : 北京交通大学出版社，2026. 1.
-- ISBN 978-7-5121-4587-0

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 2026JB3226 号

Python 程序设计工单式教程

Python CHENGXU SHEJI GONGDANSHI JIAOCHENG

责任编辑：宋伟
出版发行：北京交通大学出版社　　电话：010-51686414　　http://www.bjtup.com.cn
地　　址：北京市海淀区高梁桥斜街 44 号　　邮编：100044
印 刷 者：三河市华骏印务包装有限公司
经　　销：全国新华书店
开　　本：210 mm × 285 mm　　印张：16　　字数：472 千字
版 印 次：2026 年 1 月第 1 版　　2026 年 1 月第 1 次印刷
印　　数：1—3 000 册　　定价：52.00 元

本书如有质量问题，请向北京交通大学出版社质监组反映。对您的意见和批评，我们表示欢迎和感谢。
投诉电话：010-51686043，51686008；传真：010-62225406；E-mail：press@bjtu.edu.cn。

前言

随着信息技术的飞速发展，Python 作为一门简洁高效、应用广泛的编程语言，已成为人工智能、数据分析、Web（万维网）开发等领域的核心工具。党的二十大报告中提到“教育、科技、人才是全面建设社会主义现代化国家的基础性、战略性支撑”。高职教育以培养高素质技术技能人才为目标，本书紧密贴合高职教育特点，以“工单式”教学模式为创新点，旨在帮助学生在实践中掌握 Python 编程的核心能力，为其职业发展奠定坚实基础。

本书的编写始终围绕“学中做，做中学”的教育理念，将知识点拆解为具体的工单任务，通过任务驱动引导学生逐步构建编程思维。书中的每个工单均以实际应用场景为背景，从需求分析、代码实现到调试优化，完整呈现项目开发流程。学生在完成工单的过程中，不仅能够系统地掌握 Python 编程的相关知识和技能，更能深刻体会到如何将所学知识应用于实际工作场景中，从而有效提升职业素养和综合能力，能够在毕业后迅速适应岗位需求。本书适合作为计算机类和通信类专业的教材。

本书特色鲜明：①以工单为导向，将理论知识融入实践任务，避免了传统教学中理论与实践脱节的问题；②案例设计贴近职业场景，涵盖数据分析、自动化办公等热门领域，有助于增强学生的职业竞争力；③内容编排由浅入深，从基础语法到高级应用层层递进，兼顾不同层次学生的学习需求；④注重编程规范与代码质量，能够帮助学生培养良好的编码习惯；⑤将技术成果、社会主义核心价值观及政策理念等要素巧妙融入学习导航设计、程序代码编写等教学环节，实现专业知识与思政教育的深度融合，促进价值引领、知识传授与能力培养的有机统一。

此外，本书编者还为广大一线教师提供了服务于本书的教学资源库，有需要者可致电 13810412048 或发邮件至 2393867076@qq.com 获取。

本书编者团队汇聚了众多来自高职教育一线的资深教师和企业的行业专家。孙婷、吴东任主编，张俊宁、马诗朦、王娜任副主编，曹成、王小亮、金丽任参编。具体编写分工如下：学习活动 1、学习活动 2 由马诗朦编写；学习活动 3、学习活动 4、课程标准由孙婷编写；学习活动 5、学习活动 6、学习活动 9、学习活动 10 由吴东编写；学习活动 7、学习活动 8 由张俊宁编写；王娜、曹成、王小亮、金丽参与了本教材部分案例和习题的编写工作，为教材完成付出了辛勤努力，在此一并致谢。团队凭借丰富的教学经验和实践经验，深入研究高职学生的学习需求和企业的用人标准，力求使本书在内容上既紧密贴合教学大纲要求，又充分反映行业发展的最新动态和实际需求。本书的顺利完成，离不开团队每一位成员的辛勤付出与无私奉献，在此谨致以最诚挚的谢意。由于编者水平有限，书中可能存在疏漏之处，恳请广大读者批评指正。

希望本书能成为读者探索 Python 编程世界的罗盘，助力大家在数字化时代的浪潮中扬帆远航！

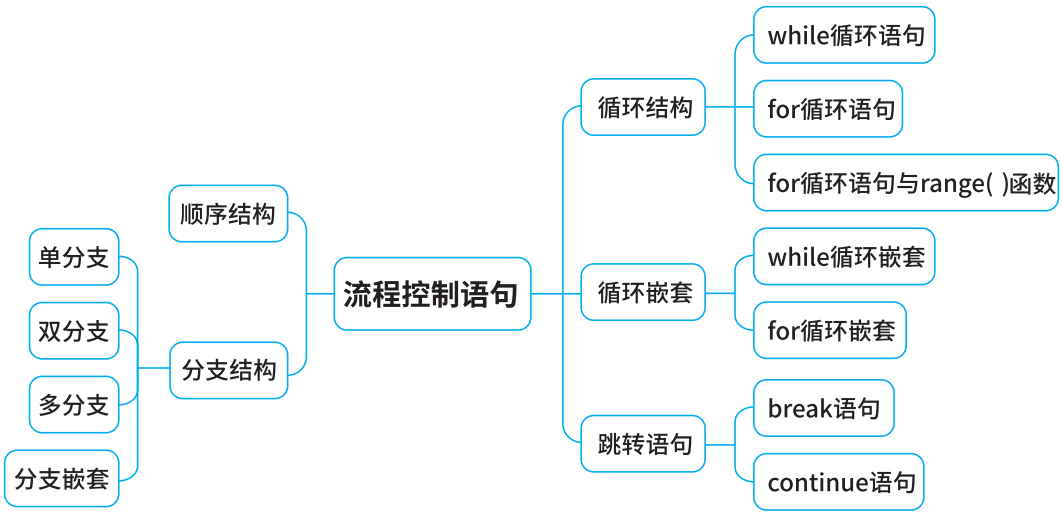
目录

学习活动3　流程控制语句

学习导航

在国产自动驾驶芯片“地平线征程 6”的算法体系构建中，顺序、分支、循环三种流程控制结构与芯片自主研发创新理念深度结合。该系统通过顺序结构的标准化执行确保算法逻辑严谨性，借助分支结构的条件判断实现多场景决策精准性，依托循环结构的持续迭代监测维持系统运行稳定性，从而将程序设计范式转化为保障道路交通安全的核心技术。这一实践生动诠释了“实现高水平科技自立自强”“构建现代化产业体系”的战略部署。

任何程序均由三种基本结构组成：顺序结构、分支结构和循环结构。顺序结构按线性方式依次执行步骤；分支结构（选择结构）根据判断结果选择不同路径执行，可细分为单分支、双分支和多分支；循环结构根据条件满足与否决定是否反复执行程序，满足则反复执行，直至条件不满足时退出。例如：要输出 1~100 之间的所有整数，如果不使用循环语句，就需要 100 行代码才能完成，如果使用循环语句，则只需几行代码即可。这三种结构相结合，能够实现很多复杂的功能。



学习目标

知识目标

（1）理解单分支、双分支、多分支和分支嵌套的语法结构，掌握其使用方法。

（2）掌握 while 循环语句、for 循环语句和循环嵌套在实际编程中的使用方法。

（3）掌握 break 语句和 continue 语句在程序中的使用方法。

能力目标

（1）能够应用流程控制结构编写较复杂的程序。

（2）能够按照自上而下、逐步细化的要求，完成典型模块化程序设计。

素质目标

（1）通过设计和实现流程控制结构，锻炼分析问题、抽象逻辑、规划步骤的系统性思维能力。

（2）在编写包含分支、循环的程序时，培养注重代码可读性、健壮性和效率（避免不必要的嵌套或循环）的意识。

（3）在调试语句的过程中培养科学严谨的态度，树立探索试错与挑战精神。

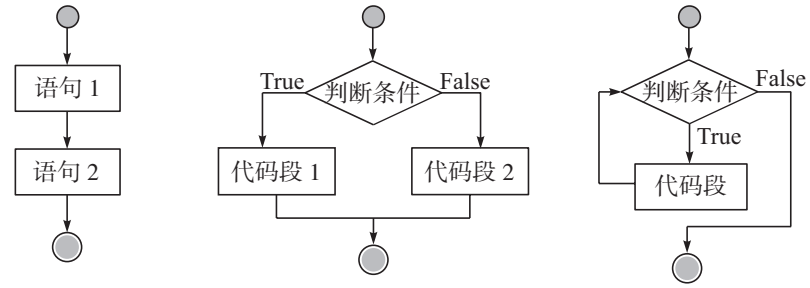
操作工单

课前：预习任务			
背景信息	现实生活中有很多复杂的结构，要根据不同的情况执行不同的操作，在 Python 中，这通常通过 if、elif 和 else 等关键字实现。程序开发中同样可能出现代码的重复执行，这时可以利用循环结构程序设计思路来解决问题		
预习目标	1. 理解顺序结构在编程中的作用 2. 掌握单分支结构、双分支结构、多分支结构和分支嵌套的使用 3. 掌握 while 循环语句、for 循环语句、循环嵌套在实际编程中的使用 4. 掌握 break 语句和 continue 语句在程序中的使用		
预习任务	1. 阅读 Python 官方文档中关于条件语句和循环语句的部分 2. 思考当需要判断多个条件时，应如何使用 elif 3. 思考在不同的应用场景下，应如何选择合适的循环语句		
预习检验	1. 尝试编写一个简单的程序，可根据用户输入的分数判断其成绩等级 2. 一次性输入 10 门课程的分数，判断每门课程的成绩等级		
存在疑问（如有更多问题，可另附纸张）	1. _____ 2. _____		
课中：学习活动实施			
工单编号		工单名称	流程控制语句
工单大类	程序设计	面向专业	计算机类
职业岗位	软件工程师		
实施方式	理实一体	考核方式	操作演示
工单分值	10	完成时限	4 学时
工单来源	教学案例	工单属性	院校工单
考核点	正确使用 Python 流程控制语句		
设备环境	Python3.13.4（64-bit）和 PyCharm 2025		
实施人员信息			
姓名、学号		班级、电话	
隶属组		组长	
岗位分工		小组成员	
学习活动日志			
日期	工作内容	问题及原因	解决方案
课后：过程记录与总结反思			

学习任务 3.1 使用顺序结构计算三角形的面积

程序控制方式是指在程序控制下进行的数据传递方式。程序控制结构是以某种顺序执行的一系列动作，用于解决某个问题。理论和实践证明，无论多复杂的算法，都可以通过顺序、选择、循环 3 种基本控制结构构造出来。每种结构仅有一个入口和一个出口。由这 3 种基本结构组成的多层嵌套程序称为结构化程序。

Python 程序具有三种典型的控制结构（见图 3-1）。



(a) 顺序结构　(b) 分支结构　(c) 循环结构

图 3-1　Python 程序的三种典型控制结构

（1）顺序结构：在程序执行时，按照语句的顺序，从上而下、一条一条地顺序执行。该结构是结构化程序中最简单的结构。

（2）分支结构：又称为“选择结构”，分支语句可根据一定的条件决定执行哪一部分代码段。

（3）循环结构：使同一个代码段根据一定的条件执行若干次。采用循环结构可以实现有规律的重复计算处理。

【任务实现】

编写程序，要求输入三角形的三条边长（假设给定的三条边符合构成三角形的条件：任意两边之和大于第三边），计算三角形的面积并输出。

任务实现程序代码如下：

```
import math
a=int(input(" 请输入三角形的第一条边："))
b=int(input(" 请输入三角形的第二条边："))
c=int(input(" 请输入三角形的第三条边："))
s=1/2*(a+b+c)
area=math.sqrt(s*(s-a)*(s-b)*(s-c))
print(" 此三角形面积为：",area)
```

写出运行结果：

程序说明：解此题的关键是要找到求三角形面积的公式 $area=\sqrt{s(s-a)(s-b)(s-c)}$，其中 $s=\frac{1}{2}\times(a+b+c)$。

程序的第 2~4 行都使用 int() 函数先将从键盘输入的数据转换为整型数据，再赋值给变量。

程序第 6 行中，sqrt() 函数的功能是求平方根，但是该函数无法直接访问，需要导入 math 模块，再通过静态对象调用该函数。

拓展任务

通过键盘输入某商品的单价和数量，求出商品的总价并输出。

基本思路如下。

（1）通过 input() 函数让用户输入商品的价格，并将其转化成 float 型数值。

（2）通过 input() 函数让用户输入商品的数量，并将其转化成 int 型数值。

（3）通过公式计算商品的总价（商品总价 = 商品单价 * 商品数量），最后将结果输出。

借助 AI 完成“计算商品总价”

在以下程序的空白处填写正确的代码，代码如下：

```
a=float(input(" 请输入商品的价格："))
b=int(input(" 请输入商品的数量："))
c=(                    )
print(" 商品总价为：", (                ))
```

读者也可以展开头脑风暴想出更多的方法，并思考如果输入值为负数将如何编写程序。

任务完成效果分析

请对本次学习任务的内容进行梳理与汇总，填写表 3-1。

表 3-1　知识点梳理与汇总

学习任务 3.1　使用顺序结构计算三角形的面积		
学习任务描述：顺序结构会在程序执行时按照语句的顺序，从上而下、一条一条地顺序执行，是结构化程序中最简单的结构		
知识点	**知识内容盘点**	**权重**
顺序结构的流程图	画出顺序结构的流程图	50%
技能点	**技能内容盘点**	**权重**
导入 math 模块	举例说明如何实现	50%

续表

学习评价			
评价内容	自评	互评	师评
知识点掌握情况（40%）			
实践练习完成情况（60%）			
本人签名：	组长签名：	教师签名：	

学习任务 3.2 使用条件语句实现考试评估程序

在 12306 网站购票时需要先验证身份，验证通过后才可进入购票页面，验证失败则需重新验证。在代码编写工作中，大家可以使用条件语句为程序增设条件，使程序产生分支，进而有选择地执行不同的语句。根据分支数量的不同，if 语句分为单分支、双分支和多分支语句。

3.2.1 单分支 if 语句

if 语句由关键字 if、判断条件和冒号组成，if 语句和从属于该语句的代码段可组成分支结构。其语法格式为：

```
if 判断条件：
    代码段
```

if 语句的执行流程如图 3-2 所示，当判断条件的返回值为 True 时，表示条件满足，代码段将被执行，否则该代码段将不被执行。

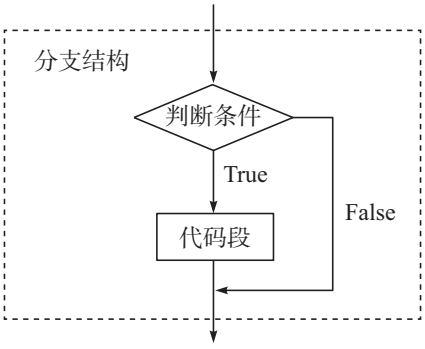


图 3-2 if 语句的执行流程

提示

判断条件后的冒号“:”是不可缺少的，它表示一个语句块的开始。

【任务实现 3.2.1】

用 if 语句实现一个考试成绩评估程序：某学校上周进行了英语模拟考试，考试成绩不低于 60 分的学生考试及格，假设小明的考试成绩为 88 分，输出小明的成绩评估结果。

任务实现程序代码如下：

```
score = 88
if score >=60
    print(" 考试及格 !")
```

写出运行结果：

```

```

程序说明：由例题的输出结果可知，88 大于等于 60，满足条件，执行代码段，输出“考试及格!”

3.2.2 双分支 if 语句

在实际生活中，我们有时不仅需要处理满足条件的情况，也需要对不满足条件的情况做特殊处理。Python 提供了可以同时处理满足和不满足条件情况的 if-else 语句。其语法格式为：

```
if 判断条件 :
    代码段 1
else:
    代码段 2
```

if-else 语句的执行流程如图 3-3 所示，若判断条件成立，执行 if 语句之后的代码段 1；若判断条件不成立，执行 else 语句之后的代码段 2。

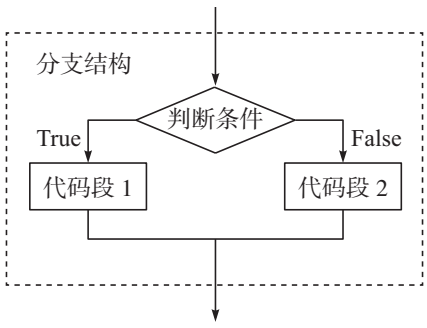


图 3-3　if-else 语句的执行流程

【任务实现 3.2.2】

优化【任务实现 3.2.1】中的考试成绩评估程序，使该程序可以同时兼顾考试及格和不及格这两种评估结果。

（1）程序优化完成后，先设置变量 score 的值为 88，写出运行结果。

（2）将变量 score 的值修改为 55，再次运行代码并写出运行结果。

任务实现程序代码如下：

```
score=88
if score>=60:
    print(" 考试及格！ ")
else:
    print(" 考试不及格！ ")
```

写出运行结果：

（1）变量 score 的值为 88 时，写出运行结果。

```

```

（2）变量 score 的值修改为 55 后，写出运行结果。

```

```

程序说明：通过比较两次的输出结果可知，程序在 score 的值为 88 时执行了 if 语句的代码段，输出了“考试及格！”，修改 score 的值为 55 后，不满足 if 语句的判断条件，因此执行了 else 语句的代码段，输出了“考试不及格！”。

3.2.3 多分支 if 语句

Python 除了提供单分支和双分支条件语句外，还提供多分支条件语句 if-elif-else。多分支条件语句用于处理单分支和双分支无法处理的情况。其语法格式为：

```
if 判断条件 1:
    代码段 1
elif 判断条件 2:
    代码段 2
elif 判断条件 3:
    代码段 3
...
else:
    代码段 n
```

以上格式中的 if 关键字与判断条件 1 构成一个分支，elif 关键字与其他判断条件构成 n−2 个分支，else 语句构成最后一个分支，每个条件语句及 else 语句与代码段之间均采用缩进的形式进行关联，if−elif−else 语句的执行流程如图 3−4 所示。

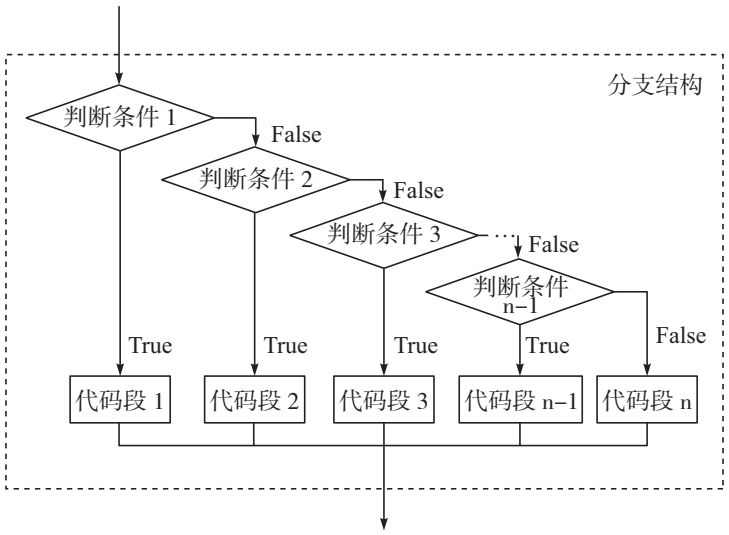


图 3−4　if−elif−else 语句的执行流程

【任务实现 3.2.3】

下面用多分支语句优化考试成绩评估程序，新程序将学生的等级分为五个部分：90~100 分为“优秀”，80~89 分为“良好”，70~79 分为“中等”，60~69 分为“及格”，低于 60 分为“不及格”。

任务实现程序代码如下：

```
score=88
if score>=90:
    print(" 优秀 ")
elif  score>=80:
    print(" 良好 ")
elif  score>=70:
    print(" 中等 ")
elif  score>=60:
    print(" 及格 ")
else:
    print(" 不及格 ")
```

写出运行结果：

3.2.4 分支嵌套

大家在火车站乘坐高铁出行时一般需要历经安检和检票两道程序：安检符合条件后方可进入检票程序，符合检票条件后方可进站乘坐列车。这个场景中虽然涉及两个判断条件，但这两个条件并非选择关系，而是嵌套关系：先判断外层条件，条件满足后才去判断内层条件，两层条件都满足时才执行内层的操作。这就是分支嵌套。Python 中通过 if 嵌套可以实现程序中条件语句的嵌套逻辑。其语法格式为：

```
if 判断条件 1:          # 外层条件
    代码段 1
    if 判断条件 2:      # 内层条件
        代码段 2
```

if 嵌套的执行流程如图 3-5 所示，若外层判断条件（即判断条件 1）的值为 True，执行代码段 1，并对内层判断条件（即判断条件 2）进行判断：若判断条件 2 的值为 True，则执行代码段 2，否则将跳出内层分支结构，顺序执行外层分支结构中内层分支结构之后的代码。若外层判断条件的值为 False，则会直接跳过条件语句，既不执行代码段 1，也不执行内层分支结构。

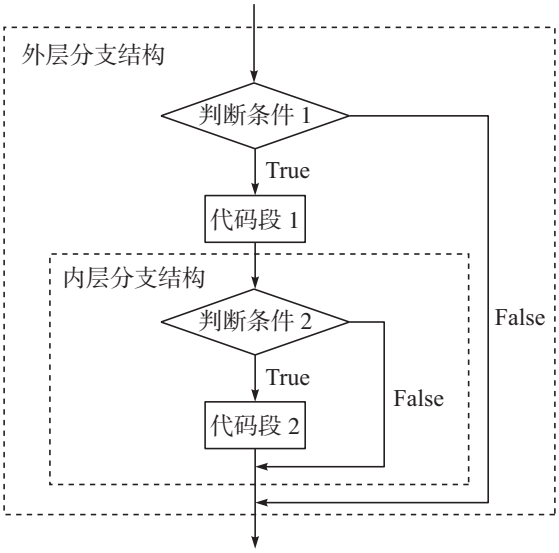


图 3-5　if 嵌套的执行流程

【任务实现 3.2.4】

可以利用 if 嵌套优化【任务实现 3.2.3】。先利用 if 嵌套判断输入的数值合不合法，当输入的数值不合法，即不在 0~100 之间时，输出提示语句“请输入 0~100 之间的分数”。

任务实现程序代码如下：

```
score=int(input(" 请输入您的分数 "))
if score>100 or score<0:
    print(" 请输入 0~100 的分数。")
else:
```

```
if score>=90:
    print(" 优秀 ")
elif  score>=80:
    print(" 良好 ")
elif  score>=70:
    print(" 中等 ")
elif  score>=60:
    print(" 及格 ")
else:
    print(" 不及格 ")
```

运行代码，当输入的数值为 120 时，写出运行结果：

当输入的数值为 88 时，写出运行结果：

拓展任务

通过键盘输入三个数字，判断三个数字中的最大值。

本任务的解法较多，读者可以开拓思路，使用多种方法实现，不断提高自己的编程能力和思考能力。

方法一基本思路：通过 input() 函数让用户输入三个数字，并将其转化成 float 型数值，接着穷举各种大小关系，从而找到最大值。此案例判断条件为多个，可以使用 if-elif-else 多分支结构来进行编程。首先用 n1 与 n2、n3 比较，若 n1 既大于 n2 又大于 n3，则 n1 为最大值；若 n2 既大于 n1 又大于 n3，则 n2 为最大值；若都不成立，则 n3 为最大值，最后将结果输出。

在以下程序的空白处填写正确的代码，代码如下：

```
n1=float(input('Please enter the first number:'))
n2=float(input('Please enter the second number:'))
n3=float(input('Please enter the third number:'))
if n1(          )n3 and n2(          )n3:
    max_num=n3
elif n1(          )n2 and n3<n2:
    max_num=(          )
else:
    max_num=(          )
print('the max is:{max_num}')
```

借助 AI 完成“判断三个数字中的最大值”

方法二基本思路：首先假定三个数字中 n1 最大，将其赋值给 max，然后将 max 逐一与剩下的两个数字进行比较，如果假定的最大值 max 比 n2 小，则将 n2 重新赋值给 max，此时找出的是 n1 和 n2 中的较大值，再将其与 n3 进行比较，若 n3 较大，则将 n3 赋值给 max，最后将结果输出。

在以下程序的空白处填写正确的代码，代码如下：

```
n1=float(input('Please enter the first number:'))
n2=float(input('Please enter the second number:'))
n3=float(input('Please enter the third number:'))
max=n1
if max<n2:
    max = (          )
if max<n3:
    max = (          )
print('the max is:%f'%max)
```

读者也可以展开头脑风暴想出更多的方法，并思考如果是求 n 个数字中的最大值将如何编写程序。

任务完成效果分析

请对本次学习任务的内容进行梳理与汇总，填写表 3-2。

表 3-2　知识点梳理与汇总

学习任务 3.2　使用条件语句实现考试评估程序		
学习任务描述：条件语句在程序设计中使用频率非常高，该语句可以对判断条件值为 True 和 False 的情况分别执行不同的处理过程		
知识点	**知识内容盘点**	**权重**
条件语句的语法	写出条件语句的语法	20%
条件语句的流程图	画出条件语句的流程图	20%
技能点	**技能内容盘点**	**权重**
优化考试评估程序（将成绩分成优、良、中、及格、不及格五个等级）	举例说明如何实现	30%
当输入的数值不合法，即不在 0~100 之间时，输出提示语句“请输入 0~100 之间的分数”	举例说明如何实现	30%

学习评价			
评价内容	**自评**	**互评**	**师评**
知识点掌握情况（40%）			
实践练习完成情况（60%）			
本人签名：		组长签名：	教师签名：

学习任务 3.3　使用循环语句计算 1~1000 的整数和

现实生活中存在着很多重复的事情，如地球一直围绕着太阳旋转，月球始终围绕地球旋转，每年都会经历四季的更替，每天都会经历从白天到黑夜的过程……程序开发中同样可能出现代码的重复执行，Python 提供了循环语句，使用该语句就能以简洁的代码实现重复操作。

采用循环结构可以实现有规律的重复计算处理。当程序执行到循环语句时，会根据循环判定条件对一组语句重复执行多次。循环结构可以看成是一个条件判断语句和一个向回转向语句的组合。

在 Python 语言中我们主要学习 while、for 两种循环。while 循环和 for 循环都是先判断表达式，后执行循环体。

提示

while、for 两种循环都需要特别注意：在循环体内应包含趋于结束的语句（即循环变量值的改变），否则就可能成为一个死循环，这是初学者的一个常见错误。

3.3.1 while 循环语句

Python 语言中，while 语句是用一个表达式来控制循环的语句，它的一般形式为：

```
while 判断条件：
    代码段
```

当判断条件的返回值为 True 时，执行代码段（或称为“循环体”），然后重新判断该条件的返回值，直到判断条件的返回值为 False 时，退出循环。while 语句的执行流程如图 3-6 所示。

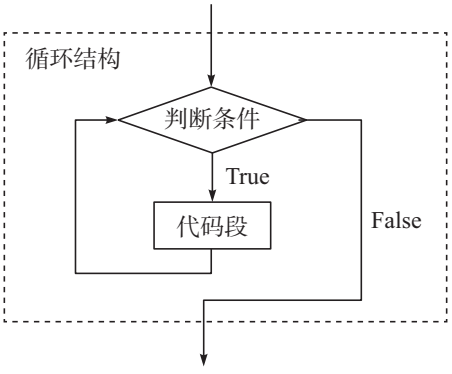


图 3-6　while 语句的执行流程

【任务实现 3.3.1】

编写程序，求 s = 1 + 2 + 3 + … + 1000 的值。

任务实现程序代码如下：

```
i=1
s=0
while i<=1000:
    s+=i
    i+=1
print("s=1+2+3+…+1000=",s)
```

写出运行结果：

3.3.2 for 循环语句

while 循环语句非常灵活，基本能够满足循环结构程序设计的需要，但有时还需要“量体裁衣”。例如，要让一个集合中的每个元素都执行一个代码段，此时 for 循环语句更为合适。for 循环语句是最常用的循环语句，一般用在循环次数已知的情况下。Python 语言中，for 循环的语法格式为：

```
for 循环变量 in 序列：
    代码段
```

for 循环中循环变量用于保存读取出的值。如果序列中包含表达式，则应先进行计算求值，然后将序列中的第一个元素赋给循环变量，执行代码段，接着将序列中的第二个元素赋给变量，执行代码段，依此类推，直到序列中的最后一个元素赋给循环变量，执行代码段后，for 循环结束，开始执行 for 语句后的语句。

【任务实现 3.3.2】

使用 for 循环输出字符串“实现中华民族伟大复兴的中国梦”中的每一个字符。

任务实现程序代码如下：

```
for x in " 实现中华民族伟大复兴的中国梦 ":
    print(x,end = " ")
```

写出运行结果：

程序说明：Python 中的 for 循环常用于遍历列表、元组、字符串及字典等序列中的元素，具体使用

方法将陆续在后续内容中介绍。

3.3.3 for 循环语句与 range () 函数

for 循环语句经常与 range() 函数一起使用，range() 函数是 Python 的内置函数，可创建一个整数列表。range() 函数的语法格式为：

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

（1）start：可迭代对象的开始值为 start，默认是从 0 开始。例如，range(3) 等价于 range(0,3)。

（2）end：可迭代对象到 end 结束，但不包括 end。例如，range(0,3) 是指 0，1，2 数字序列，不包括 3。

（3）step：步长，返回的可迭代对象元素之间的间隔，默认值为 1。例如，range(0,3) 等价于 range(0,3,1)。当 step 为负数时，开始值大于结束值，range() 函数将生成一个从大到小的数字序列。例如，range(3,0,−1) 是指数字序列 3，2，1。

【任务实现 3.3.3】

使用 for 循环输出 s = 1 + 2 + 3 + … + 1000 的值。

任务实现程序代码如下：

```
s=0
for i in range(1,1001):
    s+=i
print("s=1+2+3+…+1000=",s)
```

写出运行结果：

```

```

拓展任务

求 1000 以内所有奇数的和。

本例的解法较多，读者可以开拓思路，使用多种方法实现，不断提高自己的编程能力和思考能力。

方法一基本思路：通过 while 循环语句实现。首先设置变量 s=0（表示和），i=1（表示当前数字，从 1 开始即可），步长设为 2，再通过 while 循环累加，最后输出结果。

借助 AI 完成“求 1000 以内所有奇数的和”

在以下程序的空白处填写正确的代码，代码如下：

```
i=1
s=0
while(          ):
```

```
        s+=i
        i+=2
print(s)
```

方法二基本思路：本案例也可以通过 for 循环语句实现。只要善于思考，使用 for 循环语句编写也有很多种方法，例如，可以在 for 循环中加入 if 条件判断，也可以直接利用 range() 函数。

若利用 range() 函数，应首先设置变量 s=0（表示和），range(1,1001,2) 函数可生产从 1 开始直到 1001（左闭右开，不包含 1001）的从小到大的奇数序列；该函数也可以从大到小生产奇数序列，即将 range() 函数写成 range(999,0,−2)，此时要注意结束位 end 参数为 0，步长 step 为 −2。也可在 for 循环中加入 if 条件判断，将条件设置为“除以 2 后的余数为 1”，即可将 1~1000 以内的所有奇数筛选出来，再进行相加。

在以下程序的空白处填写正确的代码，代码如下：

```
s=0
for i in range(          ):
        s+=2
print(s)
```

读者也可以展开头脑风暴想出更多的方法，并思考如果是求 1000 以内所有偶数的和将如何编写程序。

任务完成效果分析

请对本次学习任务的内容进行梳理与汇总，填写表 3−3。

表 3−3　知识点梳理与汇总

学习任务 3.3　使用循环语句计算 1 ～ 1000 的整数和		
学习任务描述：循环结构在程序设计中使用频率非常高，可以用 while 循环语句或者 for 循环语句来实现		
知识点	**知识内容盘点**	**权重**
循环语句的语法	写出循环语句的语法	20%
循环语句的流程图	画出循环语句的流程图	20%
技能点	**技能内容盘点**	**权重**
求 1 ～ 1000 的奇数和	举例说明如何实现	30%
求 1 ～ 1000 的偶数和	举例说明如何实现	30%

学习评价			
评价内容	**自评**	**互评**	**师评**
知识点掌握情况（40%）			
实践练习完成情况（60%）			
本人签名：	组长签名：	教师签名：	

学习任务 3.4　使用循环嵌套打印九九乘法口诀表

所谓循环嵌套是指在一个循环体中嵌入另一个循环体。循环嵌套既可以是 for 循环嵌套 while 循环，也可以是 while 循环嵌套 for 循环，即各种类型的循环都可以作为外层循环，也都可以作为内层循环。

当程序遇到循环嵌套时，如果满足外层循环的判断条件，则会开始执行外层循环的循环体，而内层循环将被当成外层循环的循环体来执行，程序通过判断内层循环的判断条件进入内层循环。只有当内层循环执行结束后，才会再次判断外层循环的判断条件，以决定是否再次执行外层循环的循环体。

3.4.1 while 循环嵌套

Python 语言中 while 循环嵌套的一般形式为：

```
while 判断条件 1:
    while 判断条件 2:
        内层循环体
    外层循环体
```

【任务实现 3.4.1】

编写程序，打印出九九乘法表。

任务实现程序代码如下：

```
i=1
while i<=9:
    j=1
    while j<=i:
        mut=j*i
        print("%d*%d=%d"%(j,i,mut),end='\t')
        j+=1
    print()
    i+=1
```

写出运行结果：

3.4.2 for 循环嵌套

Python 语言中 for 循环嵌套的一般形式为：

```
for i in 序列：
    for j in 序列：
        内层循环体
    外层循环体
```

提示

循环嵌套可以有多层，但建议不要超过两层。

【任务实现 3.4.2】

编写程序，用 for 语句打印出九九乘法表。

任务实现程序代码如下：

```
for i in range(1, 10):
    for j in range(1,i+1):
        print('{}x{}={}\t'.format(j, i, i*j), end="")
    print()
```

写出运行结果：

拓展任务

分别输入两个学生的 3 门成绩，并分别计算他们的平均成绩。

在以下程序的空白处填写正确的代码，代码如下：

```
j=1                                    # 定义外部循环计数器初始值
while j<=2:                            # 定义外部循环为执行两次
    sum = 0                            # 定义成绩初始值
    i = 1                              # 定义内部循环计数器初始值
    name = input(' 请输入学生姓名 :')
# 接收用户输入的学生姓名，赋值给 name 变量
    while (      ): # 定义内部函数循环 3 次，即接收 3 门课程的成绩
        print(' 请输入第 %d 门的考试成绩 : '%i)        # 提示用户输入成绩
        sum=sum+int(input())                  # 接收用户输入的成绩，赋值给 sum
        (      )
# 变量 i 自增 1，变为 2，继续执行循环，直到 i 等于 4 时，跳出循环
    avg = sum/(i-1)                        # 计算学生的平均成绩，赋值给 avg
    print(name,' 的平均成绩是 %d\n'%avg)    # 输出学生成绩平均值
    j = j + 1
# 内部循环执行完毕后，外部循环计数器 j 自增 1，变为 2，再进行外部循环
print(' 学生成绩输入完成！ ')
```

写出运行结果：

程序说明：以上程序的基本思路如下。

（1）定义一个外部循环计数器 j 并使其初始值为 1，让外部循环执行 2 次。

（2）定义一个存放总成绩的变量 sum，初始值为 0。

（3）定义一个内部循环计数器 i，初始值为 1。

（4）通过键盘接收学生姓名，赋值给变量 name。由于是 3 门课程的成绩，因此需内部循环 3 次，依次从键盘中接收学生不同课程的成绩并加到 sum 中求和，再除以 3，将结果赋值给 avg，即可得出平均值。

提示

循环变量要自增，否则就会形成死循环。

本案例也可以通过 for 循环语句实现，读者也可以展开头脑风暴想出更多的方法。

任务完成效果分析

请对本次学习任务的内容进行梳理与汇总，填写表 3-4。

表 3-4　知识点梳理与汇总

学习任务 3.4　使用循环嵌套打印九九乘法口诀表		
学习任务描述：循环嵌套在程序设计中使用频率非常高，可以用 while 循环嵌套或者 for 循环嵌套实现		
知识点	**知识内容盘点**	**权重**
循环嵌套的语法	写出循环嵌套的语法	20%
循环嵌套的流程图	画出循环嵌套的流程图	20%
技能点	**技能内容盘点**	**权重**
用 while 循环嵌套打印输出九九乘法口诀表	举例说明如何实现	30%
用 for 循环嵌套打印输出九九乘法口诀表	举例说明如何实现	30%

学习评价			
评价内容	**自评**	**互评**	**师评**
知识点掌握情况（40%）			
实践练习完成情况（60%）			
本人签名：	组长签名：		教师签名：

学习任务 3.5 使用跳转语句控制循环流程

前面介绍的循环都是当循环条件为假时退出循环，然而在某些场合，只要满足一定的条件就应当提前结束正在执行的循环操作。为此，Python 提供了两个辅助控制程序循环的语句：break 语句和 continue 语句。

3.5.1 break 语句

break 语句用来终止整个循环，即使循环条件没有满足 False，或者序列还没递归完，只要执行到 break，就会停止执行循环语句。在循环结构中，break 语句通常与 if 语句一起使用，以便在满足条件时跳出循环。

【任务实现 3.5.1】

计算满足条件的最大整数 n，使得 1+2+3+……+n ≤ 10000。

任务实现程序代码如下：

```
n=1
s=0
while True:
    s+=n
    if s > 10000:
        break
    n+=1
print(" 最大整数 n 为 ",n-1, ", 使得 1+2+3+……+n<=10000。")
```

写出运行结果：

```

```

程序说明：任务实现中用了“ while True:”语句，也就是循环条件恒为 True，因此，如果没有 break 语句，该程序将无限循环下去，成为死循环。但当 n 为 141 时，s 的值为 10011，if 语句中的关系表达式 s>10000 为“True”，于是此时会执行 break 语句，跳出 while 循环，输出 n−1 的值。

3.5.2 continue 语句

有时我们并不希望终止整个循环的操作，而只希望提前结束本次循环，接着执行下次循环，这时，可以用 continue 语句。与 break 语句不同，continue 语句的作用是结束本次循环，即跳过循环体中 continue 语句后面的语句，直接开始下一次循环。

【任务实现 3.5.2】

输出 1~20 之间所有的奇数。

任务实现程序代码如下：

```
for n in range(1,21):
    if n%2==0:
        continue
    else:
        print(n)
```

写出运行结果：

程序说明：若把程序中的 continue 语句换成 break 语句，则只能输出 1。因为 n=2 时，满足条件 n%2==0，所以此时会执行 break 语句，终止循环。

拓展任务

打印出 1~100 中的素数。

写出正确的代码，代码如下：

借助 AI 完成“打印 1~100 的素数”

写出运行结果：

程序说明：定义一个 for 语句，用来遍历 2~100；定义一个变量 i，初始值为 2，用来存储待判断的整数；定义一个变量 j，初始值为 2，用来存储除数；再定义一个 for 语句，用来遍历 2~i 的除数，并依次用 j 保存。计算 i 与 j 取余的结果，结果若为 0，则表示 2~i 中存在 j 能够整除 i，i 不为素数，此时应跳出循环；若一直没有被它整除的因子，则 i 为素数，应把它输出出来。

任务完成效果分析

请对本次学习任务的内容进行梳理与汇总，填写表 3-5。

表 3-5　知识点梳理与汇总

学习任务 3.5　使用跳转语句控制循环流程		
学习任务描述：Python 中有两个辅助控制程序循环的语句，即 break 语句和 continue 语句		
知识点	**知识内容盘点**	**权重**
break 语句	写出 break 语句的用法	20%
continue 语句	写出 continue 语句的用法	20%
技能点	**技能内容盘点**	**权重**
应用 break 语句终止循环	举例说明如何实现	30%
应用 continue 语句结束本次循环	举例说明如何实现	30%
学习评价		

评价内容	自评	互评	师评
知识点掌握情况（40%）			
实践练习完成情况（60%）			
本人签名：	组长签名：	教师签名：	

学习活动 3 任务实现参考结果

学习活动 3
任务实现参考结果

学习活动 3 拓展任务参考结果

学习活动 3
拓展任务参考结果