第一部分 实验指导

实验一　安装和熟悉 Python 语言开发环境

一、实验目的

1. 掌握 Python 开发环境的下载和安装方法。
2. 掌握 Python 开发环境的使用技巧。
3. 能够编写简单的 Python 程序，掌握 Python 程序的运行方式。
4. 能够安装和卸载第三方库。
5. 掌握输入函数和输出函数的基本使用方法。

二、知识要点

1. Python 是跨平台的，可以运行在 Windows、Mac 和各种 Linux/UNIX 系统中。在 Windows 系统中编写的 Python 程序，放到 Linux 系统中也是能够运行的。

2. Python 开发环境 IDLE 是 Python 官方提供的交互式运行编程环境。在安装 Python 后，可以单击“开始”按钮，通过选择“Python 3.7”→“IDLE(Python 3.7)”来启动 IDLE。通过 IDLE 可以使用交互式编程模式来执行 Python 命令。IDLE 有一个编辑器，用来编辑 Python 程序文件并运行；还有一个调试器，用来调试 Python 程序文件。

3. PyCharm 是一款功能强大的 Python IDE(集成开发环境)，带有一整套可以帮助用户在使用 Python 语言进行开发时提高效率的工具，如调试、语法高亮显示、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制等。

4. 最常用且最高效的 Python 第三方库的安装方式是使用 pip 安装。pip 是 Python 官方提供并维护的在线第三方库安装工具。对于同时安装 Python 2 和 Python 3 环境的系统，建议使用 pip3 命令专门为 Python 3 安装第三方库。

例如，安装 pygame 库，pip 默认从网络下载 pygame 库安装文件并自动安装到系统中。注意，pip(或者 pip3) 是在命令行(cmd)下运行的工具。

```
D:\>pip install pygame
```

用户也可以卸载 pygame 库，卸载过程可能需要用户确认。

```
D:\>pip uninstall pygame
```

可以通过 list 子命令列出当前系统中已经安装的第三方库，例如：

```
D:\>pip list
```

使用 pip 安装是 Python 第三方库的最主要的安装方式，可以安装 90%以上的第三方库。然而，由于一些历史、技术等原因，还有一些第三方库暂时无法使用 pip 安装，此时需要使用其他的安装方式（如下载库安装文件后手动安装），读者可以参考第三方库提供的步骤和方式进行安装。

三、实验内容

1. 了解待安装计算机的硬件配置（操作系统是 64 位的还是 32 位的）并下载、安装 Python。

从 Windows 7 开始，Windows 系统分为 32 位和 64 位。在 Windows 10 系统中右击桌面上的“此电脑”图标，在弹出的快捷菜单中选择“属性”命令，然后在弹出的对话框中单击“高级系统设置”，就可以看到 32 位或者 64 位操作系统的信息，如图 1-1-1 所示。

图 1-1-1　查看操作系统的信息

根据 Windows 版本（64 位还是 32 位）从 Python 的官方网站（https://www.python.org/）下载 Python 3.7 对应的 64 位安装程序或 32 位安装程序，然后运行下载的 EXE 安装包。

在安装 Python 时要特别注意在图 1-1-2 中勾选“Add Python 3.7 to PATH”复选框，然后单击“Install Now”进行安装。

说明：64 位操作系统只能安装在 64 位计算机上（CPU 必须是 64 位的），对计算机硬件的要求也相对较高，同时需要安装 64 位常用软件，以发挥 64 位（x64）的最佳性能。

2. IDLE 的交互式编程模式。

在安装 Python 后，可以单击“开始”按钮，通过选择“Python 3.7”→“IDLE（Python 3.7）”来启动 IDLE。IDLE 启动后的初始窗口如图 1-1-3 所示。

图 1-1-2　在 Windows 10 上安装 Python 3.7 的界面

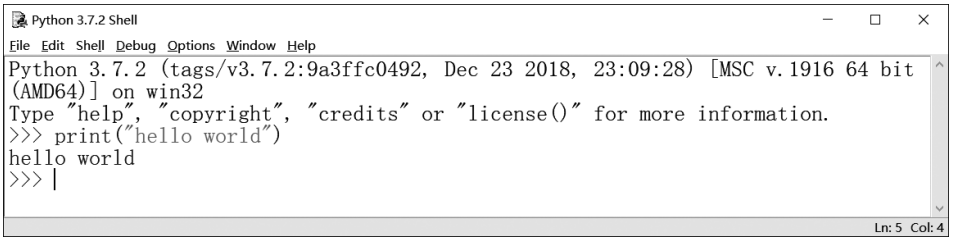

图 1-1-3　IDLE 启动后的初始窗口

在 IDLE 中使用交互式编程模式来执行 Python 命令，直接在 IDLE 提示符“>>> ”的后面输入相应的命令并回车即可。

上机实验如下例：

```
>>> print("hello world")
>>>3+5
>>>3**2
>>>3*2
>>>a=1
>>>b=2
>>>c=3
>>>print(a,b,c,sep=',')        #结果是 1,2,3
```

如果执行顺利，马上就可以看到执行结果。

3. IDLE 的文件编程模式。

在启动 IDLE 后，选择“File”→“New File”命令，可以启动 IDLE 开发环境的编辑器，通过文件的方式组织 Python 程序，文件的扩展名为.py。

在程序编辑完成后，可以选择“File”→“Save/Save as”命令保存文件，可以选择“Run”→“Run Module”命令（或按 F5 键）运行程序。在 IDLE 中可以完成中、小规模 Python 程序的编写与调试，IDLE 为开发人员提供了许多有用的特性，如自动缩进、语法高亮显示、单词自

动完成及显示命令历史等，在这些功能的帮助下，开发人员能够有效地提高自己的开发效率。本书中的所有程序均在 IDLE 环境下编写与调试。

(1) 上机实验在文件编程模式下计算两个数的和。

代码如下：

```
a=int(input("请输入第 1 个数: "))
b=int(input("请输入第 2 个数: "))
print(a,'+',b,'=',a+b)
print(a,'-',b,'=',a-b)
print(a,'*',b,'=',a*b)
```

分别输入 3 和 4，运行结果如下：

```
3 + 4 = 7
3 - 4 = -1
3 *4 = 12
```

(2) 上机实验在文件编程模式下将摄氏温度转换为华氏温度。

$$F=(9/5)C+32$$

代码如下：

```
C=float(input("请输入摄氏温度: "))
F=(9/5) * C+32
print("华氏温度: ", F)     #输出
```

(3) 上机实验在文件编程模式下解决鸡兔同笼问题。鸡兔同笼问题即一个笼子里面装有鸡和兔子(鸡有两只脚，兔子有 4 只脚)，已知鸡和兔子的总数为 h，鸡和兔子的脚的总数为 f，计算鸡和兔子分别有多少只。

代码如下：

```
h=int(input("头总数: "))
f=int(input("脚总数: "))
x=2*h-f/2
y=f/2-h
print("鸡: ", x,"兔子: ",y)
print("鸡: %d 兔子: %d"%(x,y))     #格式控制输出
```

分别输入 30 和 80，运行结果如下：

```
头总数: 30↙
脚总数: 80↙
鸡: 20.0  兔子: 10.0
鸡: 20  兔子: 10
```

4. 请读者自行练习使用 IDLE 的调试器。

5. 请读者尝试安装第三方库。

6. 请读者自行尝试安装其他开发环境，如 PyCharm、Anaconda、VSCode。

实验二　运算符和表达式

一、实验目的

1. 掌握 Python 数值类型数据和字符串的使用方法。
2. 掌握将数学表达式转换成 Python 语言表达式的方法及注意事项。
3. 掌握有关运算符的使用方法。
4. 能够编写简单的 Python 程序，掌握 Python 程序的运行方式。
5. 能够安装和卸载第三方库。

二、知识要点

1. input()函数可以用于数据的输入，默认接收字符串类型。对于输入的数据，可以使用 int()函数转换成整数类型或者使用 eval()函数转换成数值类型，也可以使用 split()等函数放到多个变量中。

例如：

```
>>>a=input("请输入第 1 个数: ")
>>> a                                          #字符串类型
'23'
>>> type(a)
<class 'str'>
>>> b=int(a)
>>> type(b)
<class 'int'>
>>> b                                          #整数类型
23
#Python 3.7 同时输入多个值
#同时输入多个字符串,字符串之间用逗号/空格分隔
>>>a,b,c=input("请输入 3 个数: ").split(",")    #输入 3 个数,用逗号分隔
请输入 3 个数: 3,4,5
>>> a
'3'
>>> b
'4'
>>> c
'5'
#同时输入多个数值,字符串之间用逗号/空格分隔
>>> a,b,c=eval(input('3 个数: '))
3 个数: 3,4,5
>>> a
3
>>> b
4
>>> c
5
>>>
```

2. print()函数可以格式化输出各种数据，支持%d、%f 等格式符，也支持使用 format()函数控制数据的格式。

3. 在 Python 中运算符有算术运算符、关系运算符、逻辑运算符、赋值运算符、位运算符、成员运算符和标识运算符 7 个类型。

(1) 算术运算符：+、-、*、/、//(整除)、%、**(指数)。

(2) 关系运算符：>、<、==、!=、<=、>=。

(3) 逻辑运算符：and、or、not。

(4) 赋值运算符：=、+=、-=、*=、/=、%=、**=、//=。

(5) 位运算符：&、|、^表示二进制的 and、or、xor 运算，>>、<<表示右移和左移，~表示按二进制位求反。

(6) 成员运算符：in、not in。

(7) 标识运算符：is、not is。

4. 在数据类型转换函数中，int(x)将 x 转换成整数类型；float(x)将 x 转换成浮点数类型；complex(real,[,imag])创建一个复数；str(x)将 x 转换为字符串；repr(x)将 x 转换为表达式字符串；eval(str)计算字符串中有效的 Python 表达式；chr(x)将整数 x 转换为一个字符；ord(x)将一个字符 x 转换为它对应的整数值；hex(x)将一个整数 x 转换为一个十六进制的字符串；oct(x)将一个整数 x 转换为一个八进制的字符串。注意，在使用数据类型转换函数时，提供给数据类型转换函数的数据必须是有意义的。

5. 对于 Python 而言，变量存储的只是一个变量的值所在的内存地址，而不是这个变量本身的值。

三、实验内容

1. 表达式的书写。

(1) 将数学表达式 $x=\frac{-b+\sqrt{b^2-4ac}}{2a}$ 书写成 Python 表达式：

```
x=-b+math.sqrt(b**2-4*a*c)/(2*a)
```

(2) 将数学表达式 $\frac{2}{3}[2ab-(|x-y|+b^2)]$ 书写成 Python 表达式：

```
x=2*(2*a*b-(math.abs(x-y)+b*b))/3
```

Python 表达式遵循下列书写规则：

(1) 表达式从左到右在同一个基准上书写，无高低、大小之分。

(2) 表达式中的乘号(*)不能省略。

(3) 括号可以嵌套使用，但是必须成对出现，不能使用中括号和大括号。

2. 输入一个十进制整数，输出其对应的二进制、八进制、十六进制形式。

代码如下：

```
a=int(input('请输入一个不大于 255 的正整数: '))
print('二进制: ',bin(a))
print('八进制: ',oct(a))
print('  十六进制: ',hex(a))
```

运行结果如下：

```
请输入一个不大于 255 的正整数：10
二进制：0b1010
八进制：0o12
十六进制：0xa
```

3. 从键盘输入一个 4 位十进制整数，要求输出各位数字之和。

方法一：由于 input()函数可以用于数据的输入，默认接收字符串类型，所以使用字符串切片方式实现各位数字的获取。

代码如下：

```
number = input("请输入 4 位数：")                          #读取输入内容
bitstr_1 = number[: 1]                                    #读取第 1 位
bitstr_2 = number[1: 2]                                   #读取第 2 位
bitstr_3 = number[2: 3]                                   #读取第 3 位
bitstr_4 = number[3: ]                                    #读取第 4 位
bitint_1 = int(bitstr_1)                                  #把第 1 位转换为 int 型
bitint_2 = int(bitstr_2)                                  #把第 2 位转换为 int 型
bitint_3 = int(bitstr_3)                                  #把第 3 位转换为 int 型
bitint_4 = int(bitstr_4)                                  #把第 4 位转换为 int 型
sum = bitint_1 + bitint_2 + bitint_3 + bitint_4           #执行加法
print(sum)                                                #输出和
```

方法二：通过算术运算获取各位数字。

代码如下：

```
number = int(input("请输入 4 位整数："))
gewei = number %10
shiwei = number // 10 % 10
baiwei = number // 100 % 10
qianwei = number // 1000
result =qianwei + baiwei + shiwei + gewei
print("计算每位相加和："+str(qianwei)+"+"+str(baiwei)
      +"+"+str(shiwei)+"+"+str(gewei)+"="+str(result))
```

方法三：使用 map()函数。map()会根据提供的 function 函数对指定序列做映射，通过把 function 函数依次作用在序列的每个元素上得到一个新序列并返回。

代码如下：

```
number =input("请输入 4 位整数：")
result = map(int,number)          #得到新序列
print(sum(result))                #对序列求和
```

4. 编写程序，输入球的半径，计算球的表面积和体积(结果保留两位小数)。

代码如下：

```
import math
r = float(input("请输入球的半径："))
```

```
area = 4 * math.pi * r**2
volume = 4/3 * math.pi * r**3
print(str.format("球的表面积为: {0: 2.2f},体积为: {1: 2.2f}",area,volume))
```

5. 编写程序,输入本金、年利率和年数,计算复利(结果保留两位小数)。

代码如下:

```
money = int(input("请输入本金: "))
rate = float(input("请输入年利率: "))
years = int(input("请输入年数: "))
amount = money * ((1+rate/100)**years)
print("本金利率和为: %.2f"%amount)
```

6. 编写程序,输入一元二次方程的系数 a、b、c,输出方程的解。

代码如下:

```
import math
a,b,c=eval(input('请输入方程的系数 a、b、c: '))
d=b * b-4 * a * c
if d>0:
    x1=(-b+math.sqrt(d))/(2 * a)
    x2=(-b-math.sqrt(d))/(2 * a)
    print('方程的解 x1=%.2f'%x1)
    print('方程的解 x2=%.2f'%x2)
if d==0:
    print('方程的解 x1=x2=%.2f'%(-b/(2 * a)))
if d<0:
    x1=-b/(2 * a)
    x2=math.sqrt(-d)/(2 * a)
    print('方程的解 x1=%.2f+%.2fj'%(x1,x2))
    print('方程的解 x2=%.2f-%.2fj'%(x1,x2))
```

四、编程并上机调试

1. 输入某年某月某日,编写程序,判断这一天是这一年的第几天。

2. 输入一个三位正整数,求其各位数字之和。例如,输入 123,各位数字之和为 6。

3. 编写程序,实现输入两个两位正整数 a、b,合并形成一个整数放在 c 中。合并的方式为将 a 数的十位和个位数依次放在 c 数的百位和个位上,将 b 数的十位和个位数依次放在 c 数的十位和千位上。

实验三　序列的创建和使用

一、实验目的

1. 了解序列的基本概念。

2. 掌握序列的索引方法。

3. 掌握列表的创建和使用方法。

4. 掌握字典的创建和使用方法。

二、知识要点

1. 可以通过列表的索引下标取出、修改、删除列表中的值，但是不能通过索引下标向列表中增加值。

2. 列表的主要操作。

(1) list()：创建空列表，或者将集合、字典、元组、字符串转换为列表。

(2) append()：向列表添加新元素，新元素可以是列表、字典、元组、集合、字符串。

(3) copy()：复制列表元素。

(4) count(x)：统计列表中指定元素的个数。

(5) index(x)：检索指定元素在列表中的位置。

(6) insert(i,x)：在列表中的指定位置插入元素，i 代表位置，x 代表元素。

(7) pop(i)：取出列表中指定位置的元素并从列表中删除。

(8) remove(x)：删除列表中指定的元素。

(9) del 命令：对列表的元素或列表中的一段数据进行删除。

(10) sort()：对列表的元素进行排序。

(11) sum(list)：内置函数，返回列表元素的和。

(12) len(list)：内置函数，返回列表元素的个数。

(13) max(list)：内置函数，返回列表元素的最大值。

3. Python 字典(dict)是一种可变容器模型，并且可以存储任何类型的对象，如字符串、数字、元组等。字典由键和对应值(key＝＞value)成对组成，字典的每个键-值对中的键和值用冒号分隔，键-值对之间用逗号分隔，整个字典包括在大括号中。

字典的主要操作如下。

(1) 创建字典。

基本语法形式：

```
dict = {key1: value1, key2: value2}
```

另一种形式：

```
dict = dict(key1=value1,key2=value2)
```

(2) 添加或修改字典：通过 dict[key]＝ value 进行操作。

(3) 删除字典或字典中的元素。

删除字典：del dict。

删除字典中的元素：del dict[key]。

(4) 遍历字典的方式。

① 通过 dict.items()遍历获取字典中的所有键-值对。

② 通过 dict.keys()遍历获取字典中的所有键。

③ 通过 dict.values()遍历获取字典中的所有值。

(5) 字典的内置函数。

① cmp(dict1,dict2)：比较两个字典元素。

② len(dict)：计算字典元素的个数,即键的总数。

③ str(dict)：输出字典可打印的字符串表示。

(6) 字典的内置方法。

① dict1.clear()：删除字典中的所有元素。

② dict1.copy()：返回一个字典副本(浅复制)。

③ dict1.fromkeys(seq[,val])：创建一个新字典,以序列 seq 中的元素做字典的键,val 为字典中所有键对应的初始值。

④ dict1.get(key, default = None)：返回指定键的值,如果值不在字典中,返回 default 值。

⑤ dict1.update(dict2)：把字典 dict2 的键-值对更新到 dict1 中。

三、实验内容

1. 将一个整数列表中的数据复制到另一个列表中。

分析：这里要求的是复制数据到一个新的列表中。Python 列表中数据的复制可以使用切片“[:]”实现,也可以使用 copy()实现。

代码如下：

```
>>> ls = [1,2,3,4]
>>> x = ls[:]                    #列表中数据的复制使用切片实现
>>> m=ls.copy()                  #列表中数据的复制使用 copy()实现
```

这里修改 x、m 的第 4 个元素为 100,代码如下：

```
>>> x[3]=100
>>> m[3]=100
>>> x                            #结果为[1, 2, 3, 100]
>>> m                            #结果为[1, 2, 3, 100]
>>>ls                           #结果为[1, 2, 3, 4]
```

通过观察,发现修改 x、m 列表不影响 ls 列表。

2. 假如 ls =[123,456,789,923,458,898],求各整数元素的和。

分析：使用内置函数 sum()实现求和。

代码如下：

```
ls =[123,456,789,923,458,898]
s=sum(ls)
print(s)
```

3. 假如有一个字典 cars={'BMW': 8.5, 'BENS': 8.3, 'AUDI': 7.9},实现遍历字典的键和值的操作。

分析：items()、keys()、values()分别用于获取字典中的所有键-值对、所有键、所有值。

这 3 个方法依次返回 dict_items、dict_keys 和 dict_values 对象，由于 Python 不希望用户直接操作这几个对象，所以使用 list()把它们转换成列表。

代码如下：

```
cars = {'BMW': 8.5, 'BENS': 8.3, 'AUDI': 7.9}
#获取字典中的所有键-值对，返回一个 dict_items 对象
ims = cars.items()
#将 dict_items 对象转换成列表
print(list(ims))                #[('BMW', 8.5), ('BENS', 8.3), ('AUDI', 7.9)]
#访问第 2 个键-值对
print(list(ims)[1])             #('BENS', 8.3)
#获取字典中的所有键，返回一个 dict_keys 对象
kys = cars.keys()
#将 dict_keys 对象转换成列表
print(list(kys))                #['BMW', 'BENS', 'AUDI']
#访问第 2 个键
print(list(kys)[1])             #'BENS'
#获取字典中的所有值，返回一个 dict_values 对象
vals = cars.values()
#将 dict_values 对象转换成列表
print(type(vals))               #[8.5, 8.3, 7.9]
#访问第 2 个值
print(list(vals)[1])            #8.3
```

4. 编写程序，生成一个包含 20 个 1～100 的随机整数的列表，然后对偶数下标的元素进行降序排列，奇数下标的元素保持不变(提示：使用切片)。

分析：对于列表 x，通过[::2]形式可以获取偶数下标元素的切片。

代码如下：

```
import random
x=[random.randint(1,100) for i in range(20)]    #生成 20 个 1~100 的随机整数的列表
print(x)
y=x[:: 2]
y.sort()                                        #默认为升序排列
y.reverse()                                     #反转列表元素，即可实现降序排列
x[:: 2]=y                                       #赋值
print(x)
```

运行结果如下：

```
[87, 34, 61, 36, 98, 68, 10, 43, 74, 47, 46, 54, 25, 27, 53, 38, 5, 99, 79, 22]
[98, 34, 87, 36, 79, 68, 74, 43, 61, 47, 53, 54, 46, 27, 25, 38, 10, 99, 5, 22]
```

5. 假如有列表 li=["alex", "eric", "rain"]，编写程序实现下列功能。

(1) 计算列表的长度并输出。

(2) 在列表中追加元素"seven"，并输出添加元素后的列表。

(3) 在列表中的第 1 个位置插入元素"Tony"，并输出添加元素后的列表。

(4) 修改列表中第 2 个位置的元素为"Kelly"，并输出修改元素后的列表。

(5) 删除列表中的元素"eric",并输出删除元素后的列表。

(6) 删除列表中的第 2 个元素,并输出所删除元素的值和删除元素后的列表。

(7) 删除列表中的第 2 个和第 3 个元素,并输出删除元素后的列表。

(8) 将列表中的所有元素反转,并输出反转元素后的列表。

代码如下:

```
li = ["alex", "eric", "rain"]        #(1)
print(len(li))
li.append("seven")                   #(2)
print(li)
li.insert(0,"Tony")                  #(3)
print(li)
li[1]="Kelly"                        #(4)
print(li)
li.remove("eric")                    #(5)
print(li)
new_li=li.pop(1)                     #(6)
print(li,new_li)
del li[1: 3]                         #(7)
print(li)
li = ["alex", "eric", "rain"]
li.reverse()                         #(8)
print(li)
```

运行结果如下:

```
3
['alex', 'eric', 'rain', 'seven']
['Tony', 'alex', 'eric', 'rain', 'seven']
['Tony', 'Kelly', 'eric', 'rain', 'seven']
['Tony', 'Kelly', 'rain', 'seven']
['Tony', 'rain', 'seven'] Kelly
['Tony']
['rain', 'eric', 'alex']
```

四、编程并上机调试

1. 编写程序,生成一个包含 10 个随机整数的列表,并求出最大值和最小值。

2. 编写程序,生成一个包含 20 个随机整数的列表,并删除第一个元素,并将列表中的所有元素之和追加到列表中。

3. 编写程序,输入一个整数(1～12)表示月份,输出该月份对应的花名(用字典实现)。月份对应的花名为:

```
"1月":"梅花","2月":"杏花","3月":"桃花","4月":"牡丹花","5月":"石榴花","6月":"莲花","7月":"玉簪花","8月":"桂花","9月":"菊花","10月":"芙蓉花","11月":"山茶花","12月":"水仙花"
```

例如,输入 6,则输出 6 月是莲花。

4. 已知有列表 lst=[54,36,75,28,50],请完成以下操作。

(1) 在列表的尾部插入元素 42。

(2) 在元素 28 的前面插入 66。

(3) 删除并输出 28。

(4) 将列表按降序排列。

(5) 清空整个列表。

实验四　选择结构的使用

一、实验目的

1. 了解程序控制的基本概念。

2. 掌握选择结构及选择嵌套结构的程序设计。

3. 掌握 if 语句、if…else 语句和 if…elif…else 语句的使用方法。

二、知识要点

在程序中选择结构可以用 if 语句、if…else 语句和 if…elif…else 语句实现。

1. if 语句。if 语句用来判断给出的条件是否满足,然后根据判断的结果(即真或假)决定是否执行给出的操作。

2. if…else 语句。if…else 语句是一种双选结构,用于在两组备选动作中选择其一。if…else语句的语法格式如下:

```
if 表达式:
    语句 1
else:
    语句 2
```

3. if…elif…else 语句。有时候需要在多组动作中选择一组执行,这时就会用到多选结构,对于 Python 语言来说就是 if…elif…else 语句。该语句可以利用一系列条件表达式进行检查,并在某个表达式为真的情况下执行相应的代码。需要注意的是,虽然 if…elif…else 语句的备选动作较多,但是有且只有一组动作被执行。该语句的语法格式如下:

```
if 表达式 1:
    语句 1
elif 表达式 2:
    语句 2
    …
elif 表达式 n:
    语句 n
else:
    语句 n+1
```

注意:最后一个 elif 子句之后的 else 子句没有进行条件判断,它实际上处理和前面所有条件都不匹配的情况,所以 else 子句必须放在最后。

三、实验内容

1. 编写程序，输入一个人的体重和身高，计算出 BMI 指数并判断该人的体形。

BMI 指数是用体重除以身高的平方得到的数字，它是目前国际上常用的衡量人体胖瘦程度及是否健康的一个标准。成人的 BMI 指数低于 18.5 为偏瘦，18.5～23.9 属于正常，24～27 属于偏胖。

代码如下：

```
weight = eval(input('请输入您的体重(kg): '))
height = eval(input('请输入您的身高(m): '))
BMI = weight/height**2
if BMI<18.5:
    print('您的体形偏瘦')
elif BMI >=18.5 and BMI<25:
    print('您的体形正常')
else:
    print('您的体形偏胖')
```

2. 编写程序，输入多个整数(用逗号分隔)，完成以下操作。

(1) 用这些整数创建一个元组。从键盘输入一个整数，判断元组中是否存在该整数，如果存在则提示存在，否则提示不存在。

代码如下：

```
a=eval(input("输入多个整数,并用逗号分隔: "))     #eval()将字符串当成有效的 Python
                                                  #表达式来求值
b=int(input("输入要查询的整数: "))
if  b in a:
    print("存在")
else:
    print("不存在")
```

说明：使用 eval()可以实现列表、字典、元组和字符串之间的转换。

(2) 用这些整数创建一个列表。从键盘输入一个整数，判断列表中是否存在该整数，如果存在则将其删除，否则将其添加到列表中。

代码如下：

```
m=input("输入多个整数,并用逗号分隔: ")
a=eval("["+m+"]")
b=int(input("输入要查询的整数: "))
if  b in a:
    a.remove(b)
else:
    a.append(b)
print(a)
```

运行结果如下：

```
输入多个整数,并用逗号分隔: 2,3,4,5↙
输入要查询的整数: 4↙
[2, 3, 5]
```

注意:↙代表按回车键。

3. 编写程序,输入一个三位正整数,找出其中最大的一位数字并输出。

分析:先将三位正整数的各位上的数字分离出来,再进行比较。

代码如下:

```
a=int(input("请输入一个三位正整数: "))
a1=a %10                #个位
a2=a/10 %10             #十位
a3=a/100                #百位
if a1>a2:
   max_a =a1
else:
   max_a =a2
if max_a>a3:
   max_a = max_a
else:
   max_a =a3
print(max_a)
```

实际上,分离一个三位正整数的数字也可以使用 divmod(),它可以同时进行整除和求余。

```
a1, a2 = divmod(a, 100)
a2, a3 = divmod(a2, 10)
print(a1,a2,a3)          #百位、十位、个位
```

思考:5 位正整数如何找出其中最大的一位数字并输出?

四、编程并上机调试

1. 编写程序计算购买地铁票时的应付款。购买地铁票的规定为乘 1～4 站,3 元/位;乘 5～9 站,4 元/位;乘 9 站以上,5 元/位。输入站数和人数,输出应付款。例如:

```
请输入站数和人数: 5,3↙
应付款: 12
```

2. 建立一个姓名和成绩对应的字典,从键盘输入一个姓名,判断该姓名在字典中是否存在,如果存在,将其成绩改成 90 分,否则提示不存在。

实验五　循环结构的使用

一、实验目的

1. 了解程序控制的基本概念。

2. 掌握循环结构及循环嵌套结构的程序设计。

3. 掌握 while 语句和 for 语句的使用方法。

二、知识要点

在程序中循环结构可以用 while 语句和 for 语句实现。

1. while 语句。while 语句用于循环执行程序，即在某条件下循环执行某段程序，以处理需要重复处理的相同任务。其语法格式如下：

```
while 判断条件:
    语句 1
else:
    语句 2      #当循环条件不满足时执行语句 2
```

while 循环通常不用 else 关键字。

2. for 语句。使用 for 语句可以遍历任何序列，如一个列表、一个元组或者一个字符串。通俗而言，遍历就是从第一个元素到最后一个元素依次访问一次。for 语句的语法格式如下：

```
for 元素 in 序列
    循环体
else:
    语句 2      #当循环结束时执行语句 2
```

for 语句每循环一次便从待遍历序列中依次取出一个元素供循环使用，直到所有元素取完为止，当循环结束时执行语句 2。for 循环通常不用 else 关键字。

注意： *当循环结构中有 else 关键字时，else 后面的语句 2 在 while 循环和 for 循环正常结束时执行。如果循环被 break 语句结束，则不会执行 else 后面的语句 2。*

三、实验内容

1. 编写程序，求级数 1＊2＋2＊3＋3＊4＋4＊5＋…＋n＊(n＋1)＋…的前 n 项的和。

分析：这是一个计数循环，每一个加数可以由数列的通项公式求得。

代码如下：

```
n=int(input("请输入一个正整数: "))
sum=0
for i in range(1,n+1):
   a = i * (1+i)
   sum = sum + a
print(sum)
```

思考：求和问题的算法设计实际上应用了迭代算法的思想，请读者自行理解。

2. 一个小球从 n 米的高度落下，每次落地后反弹为原来高度的一半再落下，球在 10 次落地后未再弹起。编写程序，求小球共经过了多少米？（要求 n 为偶数）

分析：这是一个 9 次的计数循环求和，其中的加数（小球弹起的高度）又应用了迭代算

法。(提示：需要将和初始化为 n)

代码如下：

```
n=int(input("请输入一个正整数: "))
s=n
for i in range(1,10):
   n=n/2
   s+=2*n
print(s)
```

3. 一个整数，加上 100 后是一个完全平方数(如果一个正整数 a 是某个整数 b 的平方，那么这个正整数 a 是完全平方数)，再加上 168 又是一个完全平方数。编写程序，输出 n 以内符合这种特征的整数的个数。

分析：这是一个双重循环嵌套应用穷举法的题目。外层循环遍历 n 以内的所有整数，两个内层循环又分别应用穷举法判断此数加上 100 及此数加上 268 后得到的两个新数是否为完全平方数。

代码如下：

```
n=int(input("请输入一个正整数: "))
k=1
for i in range(1,n):
   a=i+100
   flag1=False                #标志位
   for j in range(1,a+1):     #j<=a
       if a==j*j :
           flag1=True
           break
   a=i+268
   flag2=False                #标志位
   for j in range(1,a+1):
       if a==j*j:
          flag2=True
          break
   if flag1 and flag2:
      print("符合这种特征的整数: ",i,"加 100",i+100,"加 268",i+268)
      k=k+1
print("个数: ",k)
```

运行结果如下：

```
请输入一个正整数: 1000↙
符合这种特征的整数: 21 加 100 121 加 268 289
符合这种特征的整数: 261 加 100 361 加 268 529
个数: 3
```

思考：找出一个范围内的所有指定数，这是一个常见的应用，通常使用穷举法进行遍历。这道题又借鉴了判断素数的方法，请对这种应用进行总结。

4. 斐波那契数列由著名的意大利数学家 Fibonacci 以兔子繁殖为例而引入，故又称“兔

子数列”,其数值为1,1,2,3,5,8,13,21,34,…。在数学上,这一数列以如下递推方法定义:F(0)=1,F(1)=1,F(n)=F(n−1)+F(n−2)(n≥2)。

编写程序,输出斐波那契数列。

分析:首先调用input()输入要输出的斐波那契数列的长度,然后把斐波那契数列存储在一个序列中,并逐个输出序列的元素。

代码如下:

```
#输入斐波那契数列的长度
FibonacciUptoNumber = int(input('Please input a Number : '))
n = FibonacciUptoNumber
fibs = [1, 1]
for number in range(n-2):
    fibs.append(fibs[-2] + fibs[-1])
print(fibs)
```

5. 设计删除一个列表中重复元素的程序。

分析:首先调用List.sort()对序列进行排序,然后调用last = List[−1]语句从后向前找出重复的元素,最后逐个输出不重复的元素。

代码如下:

```
#List=eval(input("请输入列表: "))          #从键盘输入列表
List=[34,67,67,45,67,33,44,34,68,23]
List.sort()
last = List[-1]                            #最后一个元素
print(List)
print(range(len(List)-2, -1, -1))
for i in range(len(List)-2, -1, -1):       #从倒数第二个元素开始
    if last==List[i]:
        del List[i]
    else:
        last=List[i]
print(List)
```

运行结果如下:

```
[23, 33, 34, 34, 44, 45, 67, 67, 67, 68]
range(8, -1, -1)
[23, 33, 34, 44, 45, 67, 68]
```

6. 编写程序,生成一个包含50个随机整数的列表,然后删除其中的所有奇数。(提示:从后向前删。)

分析:采用反向索引,从列表中的最后一个元素逐一判断是否为奇数,如果是,则使用List.remove(元素)方法删除元素。

代码如下:

```
import random
list_1 = []
```

```
for i in range(10):                          #生成包含 50 个随机整数的列表
    list_1.append(random.randint(0,100))
print('生成的随机整数列表为: \n',list_1)
n = len(list_1)                              #计算列表的长度
s1 = 0                                       #计算被删除的奇数元素的个数
for i in range(n-1,-1,-1):                   #反向索引
    #判断索引元素是否为奇数
    if list_1[i] %2 ! = 0:
        print('\n 被删除的奇数元素为: ',list_1[i])
        list_1.remove(list_1[i])             #删除列表中的奇数元素
        s1 += 1
print('删除的奇数元素的个数为: %d\n     #最终随机整数列表为: \n'%(s1),list_1)
```

运行以上程序，如果循环采用正向索引，是否会出现问题？分析一下原因。

```
for i in range(0,len(list_1)):               #正向索引
```

7. 有一个已升序排列的列表，现输入一个数，要求按原来的规律将其插入列表中。

分析：首先判断边界情况，然后处理在中间插入的情况。

代码如下：

```
a =[1,4,6,9,13,16,19,28,40,100]
insert_num = int(input("请输入插入的数字: "))
if insert_num <= a[0]:
    a.insert(0,insert_num)
elif insert_num >= a[len(a)-1]:
    a.insert(len(a) ,insert_num)
else:
    for i in range(len(a)):
        if insert_num > a[i] and insert_num <= a[i+1]:
            a.insert(i+1,insert_num)
            break
print(a)
```

四、编程并上机调试

1. 输入一个数 n，输出 1～n 中的所有完数。完数是指一个数恰好等于它的因子之和，如 6＝1＋2＋3，6 就是完数。

2. 从键盘上输入一个两位的整数，求该数以内所有能被 3 整除的奇数的个数。

3. 输入一个成绩序列，输出各个成绩等级的人数。

数据：score＝[45，98，65，87，43，83，68，74，20，75，85，67，79，99]。

等级：A(100～90)，B(89～80)，C(79～70)，D(69～60)，E(60 以下)。

4. 统计重复数字。

(1) 随机生成 1000 个整数，数字的范围为[20，100]；

(2) 升序输出所有不同的数字及每个数字重复的次数。

5. 现有一个字典存放学生的学号和成绩，成绩列表中的 3 个数据分别是学生的语文、

数学和英语成绩：

```
dictScore={"01":[67,88,45],"02":[97,68,85],"03":[98,97,95],"04":[67,48,45],
"05":[82,58,75],"06":[96,49,65]}
```

编写程序，返回每个学生的学号及其最高分。

6. 求每个学生的平均成绩。

学生的成绩：

```
s={"Teddy":[100,90,90],"Sandy":[100,90,80],"Elmo":[90,90,80]}
```

输出结果：

```
{'Teddy': 93, 'Sandy': 90, 'Elmo': 86}
```

实验六　函数的定义和使用

一、实验目的

1. 掌握定义函数和调用函数的方法。

2. 掌握函数实参与形参的对应关系，以及传递的方法。

3. 理解函数的嵌套调用，以及局部变量和全局变量的区别。

4. 学习在程序设计中运用函数解决实际问题，体会使用函数在提高代码的可读性及程序开发效率方面的重要性。

二、知识要点

1. 函数的定义。函数是组织好的、可重复使用的、用来实现一定功能的代码段。函数能提高应用的模块性和代码的重复利用率。在 Python 中有内置函数，如 print()，用户也可以自定义函数。用户自定义函数的基本形式如下：

```
def 函数名(函数参数):
    函数体
    return 表达式或者值
```

在 Python 中使用 def 关键字进行函数的定义，不用指定返回值的类型。函数的参数可以是零个、一个或者多个，函数的参数也不用指定类型。函数的返回值是通过函数中的 return 语句获得的。如果没有 return 语句，则自动返回 None(空值)。

2. 函数的调用。在定义函数之前，不允许调用该函数。如果要调用一个函数，需要知道函数的名称和参数。在调用函数的时候，如果传入的参数的数量和类型不对，会报 TypeError 的错误。

3. 函数的形参和实参。形参的全称是形式参数，在使用 def 关键字定义函数时函数名后面括号中的变量称为形式参数。实参的全称为实际参数，在调用函数时提供的值或者变