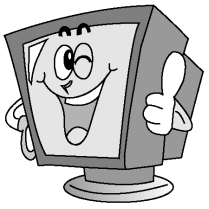

项目 3　条件语句——简易计算器的实现

Python 中的程序流程主要有顺序、分支和循环三种基本流程。分支是根据条件选择执行相应的语句，即只有满足了条件，才能做对应的事情，不满足条件，就不允许做对应的事情。在 Python 中可以使用 if、elif 和 else 关键字来构造单分支（if）、二分支（if…else）和多分支（if…elif…else）的分支结构。

本项目主要实现一个简易计算器，能根据用户输入的数进行加减乘除的计算。

【内容提要】

- 输入/输出语句的使用
- 单分支、二分支、多分支结构的使用
- Python 的数据类型及转换
- Python 运算符的正确使用
- Python 列表内置函数的正确使用
- if 语句的嵌套使用

任务 1　实现 1+1=2

3.1　接收从键盘输入的数字并进行加法计算

在项目 2 的课后作业中已经使用 print()函数输出了相关内容，那么要实现 1+1=2，这里的 2 还要输出在屏幕上，因此也可以使用 print()函数来实现。通过以下代码来测试 1+1 的输出是否等于 2。

示例 1：

```
print('1+1')
print(1+1)
```

运行结果如下：

```
1+1
2
```

从以上输出结果来看，当 1+1 放入引号内时，1+1 被 print()函数认定为字符串；当 1+1 没有用引号引起时，1+1 被 print()函数认定为进行算术运算。至此要想实现 1+1=2，在 python

中可以使用 print(1+1)来实现。

作为一个简易计算器，仅实现 1+1=2 显然是不够的，还有 1+2，2+5，…在等着被计算，那么如果都直接将算术运算写入 print()函数中进行计算，显然是不可能的。让我们动脑筋想想：当加数和被加数发生变化时该怎么办呢？

任务 2　接收从键盘输入的数字并进行简单计算


3.2　接收从键盘输入的数字进行简单计算

从键盘读取用户输入的信息是一种最基本的输入方式。Python 3 提供了输入函数 input()，可以实现接收从键盘输入的信息。其格式如下：

```
变量=input('提示信息')
```

Python 允许定义变量来存储结果或表示值。变量可以是大小写英文字母、数字和“_”的组合，Python 的变量名是区分大小写的，但不能以数字作为变量名的开头。

下面通过 input()函数来修改任务 1 中的 print(1+1)代码，实现加法计算中两个加数可以通过键盘输入来完成加法计算。实现代码如下。

示例 2：

```
x = input("请输入第一个数：")
y = input("请输入第二个数：")
print(x+y)
```

运行结果如下：

```
请输入第一个数：1
请输入第二个数：2
12
```

从输出结果看并没有实现 1+2=3，而是把 1+2 连在了一起变成 12。这是由于 input()函数通常输入的是字符型数据，当执行 print(x+y)代码时变量类型是字符类型，可以通过以下代码进行检验。

示例 3：

```
x=input("请输入第一个数：")
print(type(x))
y=input("请输入第二个数：")
print(type(y))
print(x+y)
```

运行结果如下：

```
请输入第一个数：1
<class 'str'>
请输入第二个数：2
<class 'str'>
12
```

从输出结果可以看出，使用 input()函数从键盘输入的内容为字符类型的数据，为了能进行算术运算，输入的内容必须为数字。让我们回忆一下曾经使用过的计算器，计算器既可以实现 1+1，也能实现 1.5+2.5，这就需要使用 input()函数将从键盘输入的内容转换为整数或浮点数。可以使用 int()或 float()函数进行强制类型转换，将之前的代码改进如下。

示例 4：

```
x=float(input("请输入第一个数："))          #将从键盘输入的数转换为浮点数
print(type(x))                              #显示 x 变量的类型
y=float(input("请输入第二个数："))
print(type(y))
print(x+y)
```

运行结果如下：

```
请输入第一个数：1
<class 'float'>
请输入第二个数：2
<class 'float'>
3.0
```

示例 4 中虽然实现了正确的加法，但如果按照这个代码，则只能做加法运算吗？一般的计算器是可以进行加、减、乘、除操作的。其实，Python 的功能非常强大，它支持的运算符种类也很多，主要包括：算术运算符、关系运算符、逻辑运算符、赋值运算符、位运算符、成员运算符和身份运算符等。常见的算术运算符有：+加、−减、*乘、/除和()括号。让我们继续改进代码，让它可以根据用户输入的运算符来判断需要做哪种算术运算。

Python 提供了 if 语句用于控制程序的执行，通过判断条件来决定执行的代码块，其语法格式如下。

1.基本的条件语句（if 语句）

```
if 判断条件:
    执行语句……
```

经常坐公交车的人应该有这样的经历：上车刷卡时，如果卡里的钱大于等于需要支付的车费，就能刷卡成功。用 if 语句来描述这个过程如下：

```
if 卡里的钱>=支付的车费:
    刷卡成功
```

需要注意的是“卡里的钱>=支付的车费”是判断条件，只有当判断条件为真（True）时，才执行接下来的语句块。每个 if 语句后面都要使用冒号（:），执行的语句块要使用缩进，相同缩进数的语句在一起组成一个语句块。

将刷卡坐公交车的例子用 if 语句来实现，其实现代码如下：

```
Cary_money = 50      #初始化卡中的钱有 50 元
if Cary_money >= 2:
    print('刷卡成功！')
```

运行结果如下：

```
刷卡成功!
```

程序中可以有多个 if 语句，让我们使用第一种基本的条件语句来完善示例 4 中的代码，实现代码如下。

示例 5：

```
x=float(input("请输入第一个数："))                    #将从键盘输入的数转换为浮点数
z=input("请输入运算符：")
y=float(input("请输入第二个数："))
if z=='+':                                            #判断变量 z 的值
    print(x+y)
if z=='-':
    print(x-y)
if z=='*':
    print(x*y)
if z=='/':
    print(x/y)
```

运行结果如下：

```
请输入第一个数：6
请输入运算符：/
请输入第二个数：3
2.0
```

从上述代码中，虽然实现了功能，但是使用了多个 if 语句，如果我们选择的是做加法，执行完第一个 if 语句后，程序中的每个 if 语句还是会被执行，来判断条件是否为 True。这样的程序执行效率就较低，那有没有更好的解决方法呢？

2. 有分支的条件语句（if…else 语句）

```
if 判断条件:
    执行语句……
else:
    执行语句……
```

if…else 语句表达的意思是：如果……否则……。如果 if 后面的判断条件为真（True），那么程序就执行 if 下面的语句块；如果 if 后面的判断条件为假（False），那么程序就执行 else 下面的语句块。

我们还是以刷卡坐公交车为例，上车刷卡时，如果卡里的钱大于等于需要支付的车费，就能刷卡成功，否则刷卡失败。用 if 语句来描述这个过程如下。

```
if 卡里的钱>=支付的车费:
    刷卡成功
else:
    刷卡失败
```

需要注意的是，在 Python 的 if 语句中，else 代码块是可选的，可以根据具体情况决定是否需要包含它。Python 是用缩进来标识代码块的，因此在编程时一定要注意代码块的缩进量，多一个空格或少一个空格都会引起程序出错。else 语句后面也需要使用冒号（:），执行的语句块也要使用缩进。

将刷卡坐公交车的例子用 if 语句来实现，其实现代码如下：

```
Cary_money = 1       #初始化卡中的钱有 1 元
if Cary_money >= 2:
    print('刷卡成功！')
else:
     print('刷卡失败！')
```

运行结果如下：

```
刷卡失败！
```

3. 连缀的条件语句（if…elif…else 语句）

```
if 判断条件:
     执行语句……
elif 判断条件:
     执行语句……
elif 判断条件:
     执行语句……
else:
     执行语句……
```

if…elif…else 语句表达的意思是：如果 if 后面的判断条件为真（True），则执行 if 后面的语句块，如果满足 elif 后面的判断条件为真（True），则执行 elif 后面的语句块，如果都不满足则执行 else 后面的语句块。

我们还是以刷卡坐公交车为例，上车刷卡时，如果卡里的钱大于等于需要支付的车费，就能刷卡成功，如果卡里的钱低于 10 元，就提示“卡中余额低于 10 元!”，否则刷卡失败。用 if 语句来描述这个过程如下：

```
if 卡里的钱>=支付的车费 and 卡里的钱>=10:
    刷卡成功
elif 卡里的钱>=支付的车费 and 卡里的钱<10:
    刷卡成功，卡中余额低于 10 元!
else:
     刷卡失败!
```

需要注意的是，elif 语句后面也需要使用冒号（:），执行的语句块也要使用缩进。使用连缀的条件语句，只要满足一个判断条件，那么其他的判断条件将不再执行。

将刷卡坐公交车的例子用 if 语句来实现，其实现代码如下：

```
Cary_money = 8       #初始化卡中的钱有 8 元
if Cary_money >= 2 and Cary_money >= 10:
    print('刷卡成功！')
elif Cary_money >= 2 and Cary_money < 10:
```

```
        print('刷卡成功，卡中余额低于 10 元！')
    else:
        print('刷卡失败！')
```

下面验证使用连缀的条件语句，只要满足一个判断条件，那么其他的判断条件将不再执行。

```
Cary_money = 8      #初始化卡中的钱有 8 元
if Cary_money >= 2:
    print('刷卡成功！')
elif Cary_money < 10:
    print('刷卡成功，卡中余额低于 10 元！')
else:
    print('刷卡失败！')
```

这个代码中首先满足了判断条件 Cary_money >= 2，执行了语句 print('刷卡成功！')，虽然 elif 后面的判断条件 Cary_money <= 10 也是满足的，但程序其实是没有这个 elif 语句的。上述代码的运行结果为：

```
刷卡成功！
```

让我们使用连缀的条件语句来继续改进示例 5，实现代码如下。

示例 6：

```
x=float(input("请输入第一个数："))
z=input("请输入运算符：")
y=float(input("请输入第二个数："))
if z=='+':
    print(x+y)
elif z=='-':
    print(x-y)
elif z=='*':
    print(x*y)
elif z=='/':
    print(x/y)
else:
    print("你输入的运算符不符合要求")
```

运行结果如下：

```
请输入第一个数：6
请输入运算符：/
请输入第二个数：3
2.0
```

通过分析示例 5 和示例 6 的代码可以发现，在示例 5 中判断运算符只使用了 if 语句，而在示例 6 中判断运算符使用了 if…elif…else，虽然输出结果一致，但两者还是有所区别的。使用多个 if 语句，程序在执行时，不管前面 if 中的条件是否为 True，后面的 if 语句都将被执行。而使用 if…elif…else，程序在执行时，按顺序判断条件是否为 True，如果有一个满足了，那么之后的 if…elif…else 语句将不再被执行，这样做可以提高程序执行效率。

任务 3　设置简易计算器的计算上下限

在一个简易计算器的屏幕上能显示的数字位数是有限制的，当计算得到的数字超过位数限制时，计算结果就会出错。假设我们制作的简易计算器屏幕显示的最大位数为 10 位，如果计算结果的位数超过 10 位就显示“ERROR”。Python 提供了 len()函数来返回对象（字符、列表、元组等）的长度或项目个数，首先我们需要将计算结果使用 str()函数进行强制类型转换，转换成字符类型，然后使用 len()函数来返回对象长度或项目个数，让我们继续改进示例 6，实现代码如下。

示例 7：

```
x=float(input("请输入第一个数："))
z=input("请输入运算符：")
y=float(input("请输入第二个数："))
if z=='+':
    Calc=x+y
elif z=='-':
    Calc=x-y
elif z=='*':
    Calc=x*y
elif z=='/':
    Calc=x/y
else:
    print("你输入的运算符不符合要求")
 if (len(str(Calc)))>=10:              #str(Calc)将计算结果强制类型转换为字符类型，使用
len(str(Calc))来返回对象长度
    print("ERROR!你的运算超出计算器的范围")
else:
    print(Calc)
```

运行结果如下：

```
请输入第一个数：99999999999999999999
请输入运算符：*
请输入第二个数：99999999999999999999
ERROR!你的运算超出计算器的范围
```

任务 4　项目回顾与知识拓展

本项目要求完成一个简易计算器，让我们回顾一下整个项目所用到的知识和技术。

1. Python 基础输入/输出

（1）用 Python 进行程序设计时，可以通过 input()函数实现输入，其一般格式为：

```
x=input("提示：")
```

本项目示例 2 中就使用了“x=input("请输入第一个数：")”。

input()函数返回输入的对象，可以输入数字、字符串和其他任意类型对象。

（2）用 Python 进行程序设计时，可以通过 print()函数实现输出，本项目示例 2 中就使用了“print(x+y)”。当 x 和 y 两个变量为数字时，x+y 就进行算术运算；当 x 和 y 两个变量为字符串时，x+y 就进行两个字符串的连接。例如：

```
x=3
y=4
print(x+y)
x="您好，"
y="欢迎光临！"
print(x+y)
```

运行结果如下：

```
7
您好，欢迎光临！
```

请读者思考：当 x 是数字、y 是字符串时的输出结果是什么？请动手试试。

此外，在 print()函数中还可以嵌套函数，本项目示例 3 中就使用了“print(type(x))”。

（3）可以使用 split()函数配合 input()函数，实现在一行中输入多个值。实现代码如下：

```
x, y = input("请输入 2 个数，并用空格隔开：").split( )
print(x, y)
x,y = input("请输入 2 个数，并用逗号隔开：").split(',')
print(x, y)
```

运行结果如下：

```
请输入 2 个数，并用空格隔开：1 2
1 2
请输入 2 个数，并用逗号隔开：1,2
1 2
```

（4）Python 中的 print()函数默认的是执行一次换一行，如果想实现不换行，可以在 print()函数中使用 end=''。实现代码如下：

```
print(1)
print(2)
print(3)
print(1,end='')
print(2,end='')
print(3,end='')
```

运行结果如下：

```
1
2
3
123
```

如果想在中间用空格分隔，只需要使用 end=' '。实现代码如下：

```
print(1)
print(2)
print(3)
print(1,end=' ')
print(2,end=' ')
print(3,end=' ')
```

运行结果如下：

```
1
2
3
1 2 3
```

同样的方法可以实现用逗号或其他符合分隔的效果，可以动手实践一下。

2. Python 数据类型转换

在 Python 程序设计中，不同类型的数据类型之间可以借助一些函数实现转换，常见的数据类型之间的转换函数，如表 3-1 所示。

表 3-1 Python 中常见的数据类型之间的转换函数

函 数	说 明	举例（x=12.5，y=10）
int(x[,base])	将 x 转换为一个整数	print(int(x)) #打印结果为 12
float(x)	将 x 转换为一个浮点数	print(float(y)) #打印结果为 10.0
complex(real[,imag])	创建一个复数	print(complex(x)) #打印结果为 12.5+0j
str(x)	将 x 转换为字符串	print(str(x)) #打印结果为字符串 12.5
chr(x)	将一个整数 ASCII 转换为一个字符	print(chr(80)) #打印结果为字符 P
ord(x)	将一个字符转换为它的 ASCII 整数值	print(ord('A')) #打印结果为 ASCII 整数值 65
hex(x)	将一个整数转换为一个十六进制字符串	print(hex(x)) #打印结果为 0xa
oct(x)	将一个整数转换为一个八进制字符串	print(oct(x)) #打印结果为 0o12

本项目示例 4 中就使用了“x=float(input("请输入第一个数："))”，将从键盘输入的数转换为浮点数。为了便于读者更好地理解数据类型转换函数，下面通过示例演示 Python 中常见的数据类型转换函数的使用。

示例 8：

```
a = 6.5
b = 8
c = 97
#将浮点数转换为整数
print(int(a))      #执行结果：6
#将整数转换为浮点数
```

```
print(float(b))      #执行结果：8.0
#将浮点数转换为复数
print(complex(a))      #执行结果：(6.5+0j)
#将整数转换为字符串
print('hello'+str(b))     #执行结果：hello 8
#将一个整数 ASCII 转换为一个字符
print(chr(c))      #执行结果：a
#将一个字符转换为它的 ASCII 整数值
print(ord('a'))      #执行结果：97
#将一个整数转换为一个十六进制字符串
print(hex(c))      #执行结果：0x61
#将一个整数转换为一个八进制字符串
print(oct(c))      #执行结果：0o141
```

3. Python 运算符

Python 支持多种类型的运算符：算术运算符、关系运算符、赋值运算符、逻辑运算符、位运算符、成员操作符、标识操作符。本项目示例 7 中使用了“Calc=x+y”，这里的+就是算术运算符，=就是赋值运算符；在“if z=='+':”中，==就是关系运算符。

1）算术运算符

算术运算符是用于实现数学运算的，Python 中常用的算术运算符，如表 3-2 所示。

表 3-2　Python 中常用的算术运算符

运算符	说　明	举例（x=12，y=10）
+	加法	x+y=22
-	减法	x-y=2
*	乘法	x*y=120
/	除法	x/y=1.2
%	模运算符又称求余运算符，返回余数	x%y=2
**	指数，执行对操作数幂的计算	$x^{**}y=12^{10}=61917364224$
//	整除，其结果是将商的小数点后的数舍去	x//y=1

为了便于读者更好地理解算术运算符，下面通过示例演示 Python 中常用的算术运算符的使用。

示例 9：

```
a = 6.5
b = 8
c = 97
#加法运算
print(a+b)      #执行结果：14.5
#减法运算
print(c-a)      #执行结果：90.5
#乘法运算
print(a*b)      #执行结果：52.0
#除法运算
```

```
print(c/b)      #执行结果：12.125
#模运算
print(c%b)      #执行结果：1
#指数运算
print(b**2)     #执行结果：64
#整除运算
print(c//b)     #执行结果：12
```

2）赋值运算符

赋值运算符是将右侧的表达式求出结果，赋给其左侧的变量，Python 中常用的赋值运算符，如表 3-3 所示。

表 3-3　Python 中常用的赋值运算符

运算符	说　明	举例（x=12，y=10）
=	直接赋值	x=12　将 12 赋值给变量 x
+=	加法赋值	x+=12　相当于 x=x+12，结果为 24
-=	减法赋值	x-=12　相当于 x=x-12，结果为 0
*=	乘法赋值	x*=12　相当于 x=x*12，结果为 144
/=	除法赋值	x/=12　相当于 x=x/12，结果为 1.0
%=	取模赋值	x%=12　相当于 x=x%12，结果为 0
**=	指数幂赋值	x**=12　相当于 x=x**12，结果为 8916100448256
//=	整除赋值	x//=12　相当于 x=x//12，结果为 1

3）关系运算符

关系运算符用于将两个值进行比较，如果满足结果为 True（真），不满足结果为 False（假）。Python 中常用的关系运算符，如表 3-4 所示。

表 3-4　Python 中常用的关系运算符

运算符	说　明	举例（x=12，y=10）
==	检查两个操作数的值是否相等，如果是，则条件成立，结果为 True	(x==y)为 False
!=	检查两个操作数的值是否不相等，如果是，则条件成立，结果为 True	(x!=y)为 True
>	检查左操作数是否大于右操作数，如果是，则条件成立，结果为 True	(x>y)为 True
<	检查左操作数是否小于右操作数，如果是，则条件成立，结果为 True	(x<y)为 False
>=	检查左操作数是否大于等于右操作数，如果是，则条件成立，结果为 True	(x>=y)为 True
<=	检查左操作数是否小于等于右操作数，如果是，则条件成立，结果为 True	(x<=y)为 False

4）逻辑运算符

Python 中的逻辑运算一般可以放在条件语句中作为布尔判断语句。Python 中常用的逻辑运算符，如表 3-5 所示。

表 3-5　Python 中常用的逻辑运算符

运算符	逻辑表达式	说　明	举例（a=15，b=30）
and	x and y	布尔“与”。如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值	(a and b) 返回 30
or	x or y	布尔“或”。如果 x 为非 0，它返回 x 的值，否则它返回 y 的计算值	(a or b) 返回 1
not	not x	布尔“非”。如果 x 为 True，返回 False。如果 x 为 False，它返回 True	not(a and b) 返回 False

5）赋值运算符

Python 中常用的赋值运算符，如表 3-6 所示。

表 3-6　Python 中常用的赋值运算符

运算符	说　明	举　例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

6）成员运算符

除了以上所介绍的运算符，Python 还支持成员运算符，如表 3-7 所示。

表 3-7　成员运算符

运算符	说　明	举　例
in	如果在指定的序列中找到值返回 True，否则返回 False	a = 10 b = [3,5,8,10] if a in b: 　　print('True') 结果为： True
not in	如果在指定的序列中没有找到值返回 True，否则返回 False	a = 20 b = [3,5,8,10] if a not in b: 　　print('True') 结果为： True

4. Python 列表内置函数

Python 支持多个列表内置函数：len()、max()、min()、list()。本项目中示例 7 中使用了“len(str(Calc))”，这里就用到了 len()内置函数来统计字符串变量 Calc 中的字符个数。Python 中常用的列表内置函数，如表 3-8 所示。

表 3-8 Python 中常用的列表内置函数

运算符	说 明	举例（x='abcde'，y=[10,8,5]，z=10,8,5）
len(list)	返回列表元素个数	len(x)，结果为 5
max(list)	返回列表元素最大值	max(y)，结果为 10
min(list)	返回列表元素最小值	min(y)，结果为 5
list(list)	将元组转换为列表	list(y)，结果为[10,8,5]

5. Python 控制语句——if 语句

在 Python 程序设计中，有时需要根据特定的情况有选择地执行某些语句，这就是 Python 控制语句中的选择结构。if 语句的语法形式如下所示：

```
if 表达式:
    语句 1
```

以上的这种 if 语句是一种单选结构，而 if…else 语句是一种双选结构，其语法形式如下所示：

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

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，其语法形式如下所示：

```
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
```

本项目中示例 7 中使用了：

```
if z=='+':
    Calc=x+y
elif z=='-':
    Calc=x-y
```

```
elif z=='*':
    Calc=x*y
elif z=='/':
    Calc=x/y
else:
    print("你输入的运算符不符合要求")
```

6. if 语句的嵌套

在 if…else 或 if…elif…else 语句中使用另一个 if 结构的语句即为 if 语句的嵌套。Python 支持多层嵌套，其嵌套结构形式如下：

```
if 判断条件 1:
    执行语句块 1
    if 判断条件 2:
        执行语句块 2
    else:
        执行语句块 3
elif 判断条件 3:
    执行语句块 4
else:
    执行语句块 5
```

从上述的嵌套结构形式中可以看出，if 语句的嵌套可以根据多个判断条件，选择执行相应的语句块。

我们将刷卡坐公交的例子继续进行完善，上车刷卡时，如果卡里的钱大于等于需要支付的车费，那就能刷卡成功，如果卡里的钱低于 10 元，那就提示“刷卡成功，卡中余额低于 10 元!”，否则提示“刷卡失败!”。如果乘客年龄大于 60 岁，乘车费用打 5 折。用 if 语句来实现，其实现代码如下：

```
Cary_money = 20      #初始化卡中的钱有 20 元
age = 65      #初始化乘客年龄为 65 岁
if Cary_money >= 2 and Cary_money > 10:
    if age > 60:
        print('刷卡 1 元成功！')
    else:
        print('刷卡成功！')
elif Cary_money >= 2 and Cary_money <= 10:
    if age > 60:
        print('刷卡 1 元成功！')
    else:
        print('刷卡成功，卡中余额低于 10 元！')
else:
    print('刷卡失败！')
```

运行结果如下：

```
刷卡 1 元成功!
```

同步练习：猜猜我的幸运数字

请您在程序中设置 1 个 1 位数的幸运数字，让游戏者通过键盘输入猜猜您的幸运数字，如果猜中了，显示“你好厉害！被你猜中了!”，如果没猜中，显示“继续努力，你的数字猜大了!”或“继续努力，你的数字猜小了!”。

参考代码：

```
lucknum=6
guess=int(input("请猜猜我的 1 位数幸运数字："))
if guess==lucknum:
    print("你好厉害！被你猜中了！")
elif guess<lucknum:
    print("继续努力，你的数字猜小了！")
else:
    print("继续努力，你的数字猜大了！")
```

课后作业

一、选择题

1.（　　）语句是 else 语句和 if 语句的组合。

A. elif　　B. end　　C. else　　D. if

2.（　　）可以单独使用。

A. if　　B. elif　　C. else　　D. end

3. 如果想测试变量的类型，可以使用（　　）来实现。

A. input( )　　B. type( )　　C. print( )　　D. int( )

4. 在 Python 程序设计中 int 表示的是（　　）数据类型。

A. 浮点型　　B. 字符型　　C. 整型　　D. 复数型

5. 已知 a=1，b=2，c=3，以下语句执行后 a，b，c 的值是（　　）。

```
a=1
b=2
c=3
if c>b:
    c=a
    a=b
    b=c
```

A. 3,2,1　　B. 1,2,3　　C. 3,2,2　　D. 2,1,1

二、判断题

（　　）1. Python 使用符号#表示单行注释。

（　　）2. Python 中的代码块使用缩进来表示。

（　　）3. elif 可以单独使用。

（　　）4. else 条件后面需要使用冒号。

三、操作题

1. 编写一个 Python 程序，实现 2 个数交换。

2. 编写一个 Python 程序，实现从键盘输入 3 个整数，按照从大到小的顺序输出。

3. 设计一个汇率换算器，实现输入美元金额后，能输出对应的人民币金额。

4. 编写一个 Python 程序，实现从键盘输入一个字符，判断该字符是数字、字母、空格还是其他。