

```
14     end_poly()
15
16     handForm = get_poly()
17     register_shape(name, handForm)
18
19     def init():
20         global secHand, minHand, hurHand, show
21         mode("logo")
22         #初始化三个表针元素
23         mkHand("secHand", 135)
24         mkHand("minHand", 110)
25         mkHand("hurHand", 90)
26         secHand = Turtle()
27         secHand.shape("secHand")
28         minHand = Turtle()
29         minHand.shape("minHand")
30         hurHand = Turtle()
31         hurHand.shape("hurHand")
32         for hand in secHand, minHand, hurHand:
33             hand.shapesize(1, 1, 3)
34             hand.speed(0)
35         #实例化Turtle对象
36         show = Turtle()
37         show.hideturtle()
38         show.penup()
39
40     def setup(radius):
41         #设置表框
42         reset()
43         pensize(7)
44         for i in range(60):
45             skip(radius)
46             if i % 5 == 0:
47                 forward(20)
48                 skip(-radius-20)
49             else:
50                 dot(5)
51                 skip(-radius)
52                 right(6)
53
54     def Week(t):
55         week = ["星期一", "星期二", "星期三",
56                 "星期四", "星期五", "星期六", "星期日"]
```

```
56     return week[t.weekday()]
57
58 def Date(t):
59     year = t.year
60     month = t.month
61     day = t.day
62     return "%s %d %d" % (year, month, day)
63
64 def tick():
65     #表针动态显示
66     t = datetime.today()
67     second = t.second + t.microsecond*0.000001
68     minute = t.minute + second/60.0
69     hour = t.hour + minute/60.0
70     secHand.setheading(6*second) #每秒秒针转动6度
71     minHand.setheading(6*minute) #每分钟分针转动6度
72     hurHand.setheading(30*hour) #每小时时针转动30度
73     tracer(False) #不显示绘制过程，直接显示结果
74     show.forward(65)
75     show.write(Week(t), align="center",
76                 font=("Courier", 14, "bold"))
77     show.back(130)
78     show.write(Date(t), align="center",
79                 font=("Courier", 14, "bold"))
80     show.back(50)
81     show.write("时钟", align="center",
82                 font=("Courier", 14, "bold"))
83     show.home()
84     tracer(True)
85     ontimer(tick, 1000) #刷新时间为1s，即为1s后继续调用tick
86
87 def main():
88     tracer(False)
89     init()
90     setup(160)
91     tracer(True)
92     tick()
93     mainloop()
94 main()
```

项目15：整人代码

```
1 import os
2 import time
3
4 print('*****')
5 print('欢迎来到喊猪游戏')
6 print('*****\n')
7 content = input('说我是猪,不然就关你的机, 赶紧的! , "说不说"')
8
9 if content == "我是猪":
10     time.sleep(1)
11     print('明白就好啦, 真乖~\n')
12
13 else:
14     print("既然如此...")
15     time.sleep(3)
16     print("10秒后制裁你的电脑, 啊哈哈~~~~~")
17     time.sleep(1)
18     os.system('shutdown -r -t 10')
```

项目17：制作一个计算器

```
1 from tkinter import Tk, Button, Text, END
2 import math
3 # 定义一个栈
4 class Calculator():
5     def __init__(self):
6         self.items = []
7     def isEmpty(self):
8         return self.items == []
9     def push(self, item):
10        self.items.append(item)
11    def pop(self):
12        return self.items.pop()
13    def peek(self):
14        return self.items[-1]
15    def size(self):
16        return len(self.items)
17
```

```
18 # 将计算式转化为后序表达式
19 def postfixExpr(formulaList):
20     # 定义等级
21     prior = {}
22     prior["*"] = 3
23     prior["/] = 3
24     prior["+"] = 2
25     prior["-"] = 2
26     prior["("] = 1
27
28     opstack = Calculator()
29     postfixList = []
30     for token in formulaList:
31         if token == '(':
32             opstack.push(token)
33         elif token == ")":
34             topToken = opstack.pop()
35             while topToken != "(":
36                 postfixList.append(topToken)
37                 topToken = opstack.pop()
38             elif token in "*/+":
39                 while (not opstack.isEmpty()) and (prior[opstack.peek()]>= prior[token]) :
40                     postfixList.append(opstack.pop())
41                 opstack.push(token)
42             else :
43                 postfixList.append(token)
44
45         while not opstack.isEmpty():
46             postfixList.append(opstack.pop())
47
48     return " ".join(postfixList)      # 返回后序表达式
49
50 # 后序表达式计算
51 def postfixEval(postfixExpr):
52     operandStack = Calculator()
53     tokenLIst = postfixExpr.split()
54     for token in tokenLIst:
55         if token in "*/+":
56             number1 = operandStack.pop()
57             number2 = operandStack.pop()
58             result = doMath(token, number1, number2)
59             operandStack.push(result)
```

```
60         else:
61             operandStack.push(float(token))
62     return operandStack.pop()  # 返回计算结果
63
64 # 分步计算函数
65 def doMath(op, num1, num2):
66     if op == "*":
67         return num2 * num1
68     elif op == "/":
69         return num2 / num1
70     elif op == "+":
71         return num2 + num1
72     elif op == "-":
73         return num2 - num1
74
75 # 定义Value类，用于把值传到GUI界面的Text框中
76 class Value():
77
78     def __init__(self, value):
79         self.value = value
80     # 将输入按钮的值传到文本框中
81     def insert_value(self):
82         self.result_text=Application.result_text
83         self.result_text.insert(END, self.value)
84
85 # 定义Application类，用于画GUI和计算结果。
86 class Application(object):
87     window = Tk()    # 创建一个GUI界面
88     # 创建一个文本框
89     result_text = Text(window, background='azure')
90     # 更多背景色: http://www.science.smith.edu/dftwiki/index.php/Color_Charts_for_TKinter
91     result_text.place(x=12.5, y=8, width=325, height=60)
92
93     def __init__(self):
94         self.window.title(u'我的计算器') # 定义界面名称
95
96         # 设置窗口大小和位置
97         self.window.geometry('350x310+500+300')
98         self.window.minsize(350, 315)
99         self.window.maxsize(350, 315)
100
101        # 第1行按钮
```

```
102         self.submit_btn0 =
103             Button(self.window, text=u'(' , command=Value('(').insert_value)
104             self.submit_btn0.place(x=12.5, y=76, width=60, height=40)
105             self.submit_btn1 = Button(self.window, text=u'C' , command = self.clean)
106             self.submit_btn1.place(x=78.75, y=76, width=60, height=40)
107             self.submit_btn2 = Button(self.window, text=u'/' ,
108             command=Value('/').insert_value)
109             self.submit_btn2.place(x=145, y=76, width=60, height=40)      #
110             12.5+(60+6.25)*3
110             self.submit_btn4 = Button(self.window, text=u'←' , command=self.backspace)
111             self.submit_btn4.place(x=277.5, y=76, width=60, height=40)
112             # 第2行按钮
113             self.submit_btn5 =
114                 Button(self.window, text=u')' , command=Value(')').insert_value)
114             self.submit_btn5.place(x=12.5, y=76+8+40, width=60, height=40)
115             self.submit_btn6 = Button(self.window, text=u'7' , command =
116                 Value(7).insert_value)
116             self.submit_btn6.place(x=78.75, y=124, width=60, height=40)
117             self.submit_btn7 = Button(self.window, text=u'8' ,
118             command=Value(8).insert_value)
118             self.submit_btn7.place(x=145, y=124, width=60, height=40)
119             self.submit_btn8 = Button(self.window, text=u'9' ,
120             command=Value(9).insert_value)
120             self.submit_btn8.place(x=211.25, y=124, width=60, height=40)
121             self.submit_btn9 = Button(self.window, text=u'-' ,
122             command=Value('-').insert_value)
122             self.submit_btn9.place(x=277.5, y=124, width=60, height=40)
123             # 第3行按钮
124             self.submit_btn10 =
125                 Button(self.window, text=u'π' , command=Value(math.pi).insert_value)
125             self.submit_btn10.place(x=12.5, y=172, width=60, height=40)
126             self.submit_btn11 = Button(self.window, text=u'4' , command =
127                 Value(4).insert_value)
127             self.submit_btn11.place(x=78.75, y=172, width=60, height=40)
128             self.submit_btn12 = Button(self.window, text=u'5' ,
129             command=Value(5).insert_value)
129             self.submit_btn12.place(x=145, y=172, width=60, height=40)
130             self.submit_btn13 = Button(self.window, text=u'6' ,
130             command=Value(6).insert_value)
```

```
131         self.submit_btn13.place(x=211.25, y=172, width=60, height=40)
132         self.submit_btn14 = Button(self.window, text=u'+' ,
133             command=Value('+').insert_value)
134         self.submit_btn14.place(x=277.5, y=172, width=60, height=40)
135         # 第4行按钮
136         self.submit_btn15 = Button(self.window, text=u'1/x', command=
137             Value('1/').insert_value)
138         self.submit_btn15.place(x=12.5,y=220,width=60,height=40)
139         self.submit_btn16 = Button(self.window, text=u'1', command =
140             Value(1).insert_value)
141         self.submit_btn16.place(x=78.75,y=220,width=60,height=40)
142         self.submit_btn17 = Button(self.window, text=u'2',
143             command=Value(2).insert_value)
144         self.submit_btn17.place(x=145, y=220, width=60, height=40)
145         self.submit_btn18 = Button(self.window, text=u'3',
146             command=Value(3).insert_value)
147         self.submit_btn18.place(x=211.25, y=220, width=60, height=40)
148         self.submit_btn19 = Button(self.window, text=u'=', command=self.equal)
149         self.submit_btn19.place(x=277.5, y=220, width=60, height=88)
150         # 第5行按钮
151         self.submit_btn20 =
152             Button(self.window, text=u'x2', command=Value('^2').insert_value)
153         self.submit_btn20.place(x=12.5,y=268,width=60,height=40)
154         self.submit_btn21 = Button(self.window, text=u'%',
155             command=Value('%').insert_value)
156         self.submit_btn21.place(x=78.75,y=268,width=60,height=40)
157         self.submit_btn22 = Button(self.window, text=u'0',
158             command=Value(0).insert_value)
159         self.submit_btn22.place(x=145, y=268, width=60, height=40)
160         self.submit_btn23 = Button(self.window, text=u'.',
161             command=Value('.').insert_value)
162         self.submit_btn23.place(x=211.25, y=268, width=60, height=40)

155     def clean(self):          # 清空键
156         self.result_text.delete(0.0, END)
157         return
158     def backspace(self):      # 退格键
159         content = self.result_text.get(0.0, END).strip().split("\n")
160         self.result_text.delete(0.0, END)
161         for i in content[:-1]:
162             self.result_text.insert(END, i)
```

```
163         self.result_text.insert(END, '\n')
164
165         self.result_text.insert(END, content[-1][-1])
166         self.result_text.see(END) # 定位光标到最后
167
168     def equal(self):          # 等于键
169         formula = self.result_text.get(0.0, END).strip().split("\n")[-1] # 获取界面的计
算式子
170
171         print(formula, end=" ")
172         formula = formula.replace("^2", "") # 将平方保留一个符号^方便后面计算
173         try:
174             # 将数字和运算符号分离
175             formulaList1 = []
176             i = 0
177             while i < len(formula):
178                 if formula[i] in "0123456789.":
179                     cc = ''
180                     while i < len(formula):
181                         if formula[i] in "0123456789.":
182                             cc += formula[i]
183                         i += 1
184                     else:
185                         break
186                     i -= 1
187                     formulaList1.append(cc)
188
189             i += 1
190             # 将平方和百分号做处理
191             formulaList2 = []
192             for i in formulaList1:
193                 formulaList2.append(i)
194                 if i == "^":
195                     index = formulaList1.index("^")
196                     formulaList1[index] = "-" # 将^符号替换掉，方便查找第2个^
197                     number = formulaList1[index - 1]
198                     formulaList2.pop()
199                     formulaList2.append('*')
200                     formulaList2.append(number) # 转化为两个数相乘
201
202                 elif i == "%":
203                     index = formulaList1.index("%")
204                     formulaList1[index] = "-" # 将%符号替换掉，方便查找第2个%
```

```
204         formulaList2.pop()
205
206
207     global result      # 方便在except中调用
208     b = postfixExpr(formulaList2)          # 获取后序表达式
209     result = postfixEval(b)    # 获取计算结果值
210     print(result)
211
212     self.result_text.insert(END, '=')
213     self.result_text.insert(END, result)
214     self.result_text.insert(END, '\n')  # 另起一行
215     self.result_text.see(END)    # 定位光标到最后
216
217 except ValueError:
218
219     self.result_text.insert(END, result)
220     self.result_text.insert(END, '\n')
221     self.result_text.see(END)    # 定位光标到最后
222     print(result)
223
224 except:
225     self.result_text.insert(END, 'Error!')
226     self.result_text.insert(END, '\n')
227     self.result_text.see(END)    # 定位光标到最后
228     print("Error!")
229
230 def run(self):  # 运行tkinter, 展示界面
231     self.window.mainloop()
232
233 if __name__=="__main__":
234     app = Application()
235     app.run()
```

三、原Python小课课后练习及解析

基础语法课程

第0关 print()与变量

课后练习1：打印皮卡丘

- 题目解析:

- 直接复制题目要求打印的“皮卡丘”(在【书写代码】步骤里直接提供)，用print()打印。
- 皮卡丘涉及到换行，所以这里要使用三引号来换行。
- 注意：标点符号都要用英文。

- 参考代码:

```
1 print('''
2     ^    /|
3     /\_   <_/
4     / |   / /
5     | z _ , < /  /`\
6     |     \ /  \
7     Y     ^ / /
8     仁 、 •  c o < / 
9     () ^   + \ <
10    >- ._  イ | // 
11    / ^   / /<+ \ \
12    \_ /  (_/  | // 
13    7     +/
14    >-r____`--_
15    ''')
```

课后练习2：听见无脸男的声音

- 题目解析:

方法一：使用单引号+\n打印

- 使用 \' 表示打印一个单引号：使用单引号进行打印的时候，文本中有单引号的地方，使用转义符号把单引号直接打印出来。
- 需要换行，使用 \n。
- 打印函数print()。

- 参考代码:

```
1 print('千寻你好，人们叫我'无脸男'\n这个世界的人都选择无视我\n只有你看到了我并和我打招呼\n我感到很孤单，很孤单\n你愿意和我成为朋友吗？')
```

方法二：使用三引号打印

- 直接使用三引号，打印出内容里的单引号并且实现换行。

- 参考代码:

```
1 print(''千寻你好，人们叫我'无脸男'  
2 这个世界的人都选择无视我  
3 只有你看到了我并和我打招呼  
4 我感到很孤单，很孤单  
5 你愿意和我成为朋友吗？'')
```

方法三：使用双引号+\n

- 使用双引号的时候，不会和单引号产生歧义；
- 需要换行的地方用 \n。

```
1 print("千寻你好，人们叫我'无脸男'\n这个世界的人都选择无视我\n只有你看到了我并和我打招呼\n我感到很孤  
单，很孤单\n你愿意和我成为朋友吗？")
```

第1关 数据类型与转换

课后练习1：程序员的一人饮酒醉

- 题目解析：
 - 非字符串要与字符串进行拼接的时候，需要使用str()先转换为字符串；
 - 使用 + 将字符串进行拼接；
 - 使用print()函数打印拼接结果。
- 参考代码：

```
1 number1 = 1  
2 number2 = 2  
3 unit1 = '人'  
4 unit2 = '眼'  
5 line1 = '我编程累'  
6 line2 = '是bug相随'  
7 sentence1 = '碎掉的节操满地堆'  
8 sentence2 = '我只求今日能早归'  
9  
10 print(str(number1) + unit1 + line1 + sentence1)  
11 print(str(number2) + unit2 + line2 + sentence2)
```

课后练习2：非酋的吐槽

- 题目解析:

- 小数形式的字符串'7.8'，不是整数形式的字符串，不能直接转化为整数，需要先使用float()转化为浮点数，再使用int()转化为整数，转化成整数之后，因为要用于字符串拼接，所以还要使用str()转化为字符串；
- 使用 + 将字符串进行拼接；
- 使用print()函数打印拼接结果。

- 参考代码:

```
1 slogan = '脸黑怪我咯'
2 number = '7.8'
3 unit = '张'
4 sentence = '蓝票一个SSR都没有'
5 word = slogan + str(int(float(number))) + sentence
6 print(word)
```

第2关 条件判断和条件嵌套

课后练习1：寻找宝石

- 题目解析:

- 我们需要设定一个变量来表示宝石的数量，可以把这个变量命名为stonenumber，并赋值为0。
- 按照题目需求我们需要选择多项选择的结构，条件是拥有宝石的数量，三种情况分别是大于三颗宝石，一到三颗宝石和零颗宝石。

- 参考代码:

```
1 stonenumber = 0
2 if stonenumber >= 4:
3     print('获得了打败灭霸的力量，反杀稳了')
4 elif 1<=stonenumber <= 3:
5     print('可以全员出动，殊死一搏')
6 else:
7     print('没办法了，只能尝试呼叫惊奇队长')
```

课后练习2：美国队长的工资

- 题目解析:

- 根据题干的要求，我们需要使用if...elif...else结构来处理小于500，500-1000和大于1000这三种情况。

- 在小于500的情况下，有小于100和100到500之间的两种情况，可以在这个if语句里面嵌套if..else语句来实现判断。
- 在大于1000的情况下条件也可以被细分为1000到20000和大于20000这两种情况，在这个条件里也去嵌套if...else结构实现判断。

- 参考代码：

```

1 money=80
2 if money<=500:
3     print('欢迎进入史塔克穷人榜前三名！')
4     if money<=100:
5         print("恭喜您荣获'美元队长'称号！")
6     else:
7         print('请您找弗瑞局长加薪。')
8 elif 500<money<=1000:
9     print('恭喜您至少可以温饱了。')
10 else:
11     print('经济危机都难不倒您！')
12     if money<=20000:
13         print('您快比托尼·史塔克有钱了！')
14     else:
15         print('您是不是来自于瓦坎达国？')
16
17 print('程序结束')

```

第3关 input()函数

课后练习1：哈利波特的宠物

- 题目解析：

- 运行代码之后，在终端输入一个名字，传给input()函数，并赋值给name变量；
- 然后和字符串【'哈利·波特的猫头鹰叫做'】拼接，并使用print()函数进行打印。

- 参考代码：

```

1 name = input('为哈利·波特的猫头鹰起个名字：')
2 print('哈利·波特的猫头鹰叫做' + name)

```

课后练习2：罗恩的减肥计划

- 题目解析：

- 运行代码之后，在终端输入一个数字，传给input()函数，并使用int()函数转化为整数，再赋值给number变量。
- 使用双向判断语句if...else...根据游戏规则对number的值进行判断。

- 参考代码：

```
1 number = int(input('请输入罗恩吃的巧克力数量：'))
2 #int将其转换为整数，之后好进行大小的比较
3 if number > 10:
4     print('罗恩要给哈利100块')
5 else:
6     print('哈利就给罗恩100块')
```

课后练习3：古灵阁金币兑换

- 题目解析：

- 根据题目的要求，我们判断可以通过 if条件嵌套的形式来解决问题：
 - 最外层的 if条件根据用户的需求判断是否需要帮助，只有两种情况可以使用双向判断：
 - 需要：嵌套第二层 if条件根据用户的需求判断需要什么类型的帮助，有三种情况使用多向判断：
 - 1 存取款->去存取款窗口；
 - 2 货币兑换：获取用户需要兑换的金加隆数量，并通过计算，打印出来需要支付的人民币；
 - 3 咨询->去存取款窗口。
 - 不需要->再见
 - 需要获取用户的意見的时候，都使用input()进行交互，获取用户的需求。
 - 为了使得用户输错的时候提示输入错误，需要再加一级else进行判断。

- 参考代码：

```
1 choice = input('您好，欢迎古灵阁，请问需要帮助吗？需要or不需要？')
2 if choice == '需要':
3     number = input('请问您需要什么帮助呢？1 存取款；2 货币兑换；3 咨询')
4     if number == '2':
5         print('金加隆和人民币的兑换率为1:51.3，即一金加隆=51.3人民币')
6         print('请问您需要兑换多少金加隆呢？')
7         money = input('请输入你需要兑换的金加隆')
8         print('好的，我知道了，您需要兑换' + money + '金加隆。')
9         print('那么，您需要付给我' + str(int(float(money) * 51.3)) + '人民币。')
10    elif number == '1':
```

```

11     print('请到存取款窗口办理')
12 elif number == '3':
13     print('请到咨询窗口咨询')
14 # 加上else语句当输入不是1、2、3时，提示输入错误
15 else:
16     print('输入错误，没有你需要的服务')
17
18 elif choice == '不需要':
19     print('好的，再见')
20 # 加上else语句当输入不是1、2、3时，提示输入错误
21 else:
22     print('输入错误')

```

第4关 列表和字典

课后练习1：君子爱‘数’取之有道

- 题目解析：
 - 第一题：把列表list1中的'love'取出来，并打印出来。
 - 观察list1 可以发现，list1 是一个嵌套字典的列表，外层是列表，共有3个元素，这3个元素都是字典，而且每个字典都只有一个键值对；
 - 要取出“love”，观察list1，可以找到它在列表最后一个元素中的字典里，作为键“爱”的值。可以使用偏移量-1或2，即list[-1] 或list[2]，取到列表，然后使用键“爱”取得。
 - 第二题：把字典dict1中的'love'取出来，并打印出来。
 - 观察dict1 可以发现，dict1 是一个嵌套列表的字典，外层是字典，共有3个键值对，这3个键值对中的值都是列表；
 - 要取出“love”，观察list1，可以找到它在dict1 中键为 3 对应的值（列表）中，是列表的第一个元素。首先要使用键 3 取出“love”所在的列表，然后再使用偏移量0 或-3 取出。
 - 第三题：将tuple1中的A和list2中的D打印出来。
 - 观察tuple1 可以发现，tuple1 是一个元组，“A”就是元组中的第一个元素，可以使用偏移量0 或-2 取得；
 - 观察list2 可以发现，是一个嵌套元组的列表，列表中共有3个元素，这3个元素都是元组。练习要求提取的字母“D”所在的元组在列表的第二个元素中，可以使用偏移量1 或-2 取得；取到元组之后，可以使用偏移量1 或-1 取到元素“D”。（1 或-2 和1 或-1 两两组合共有4种取法）
- 参考代码：

```

1 # 第一题代码
2 list1 = [{"嫉妒":'envy'}, {"恨":'hatred'}, {"爱":'love'}]
3 print(list1[2]['爱'])
4 print(list1[-1]['爱'])

5

6 # 第二题代码
7 dict1 = {1:['cake', 'scone', 'puff'], 2:['London', 'Bristol', 'Bath'], 3:
8     ['love', 'hatred', 'envy']}
9 print(dict1[3][0])
10 print(dict1[3][-3])

11 # 第三题代码
12 tuple1 = ('A', 'B')
13 list2 = [('A', 'B'), ('C', 'D'), ('E', 'F')]
14 print(tuple1[0])
15 print(list2[1][1])

```

课后练习2：找到那只狼

- 题目解析：
 - 观察townee 可以发现，它的最外层是一个列表，列表中有6个元素（看逗号，每个元素使用功能逗号分隔），分别是：
 - i. 第一个是字典 {'海底王国':['小美人鱼"海之王"小美人鱼的祖母"五位姐姐'],'上层世界':['王子','邻国公主']}
 - ii. 第二个是字符串 '丑小鸭'；
 - iii. 第三个是字符串 '坚定的锡兵'；
 - iv. 第四个是字符串 '睡美人'；
 - v. 第五个是字符串 '青蛙王子'；
 - vi. 第六个是列表 [{'主角':'小红帽','配角1':'外婆','配角2':'猎人'},{'反面角色':'狼'}]， “狼”就藏该元素中。
 - 分析第六个元素，该元素也是一个列表，拥有2个元素，都是字典，分别是{'主角':'小红帽','配角1':'外婆','配角2':'猎人'}和{'反面角色':'狼'}。“狼”就藏在第二个元素中，以值的形式存在。
 - 综上得知，“狼”藏在townee 列表最后一个元素的列表中的字典里，作为键“反面角色”的值，即print(townee[5][1]['反面角色']), 也可以使用负向取值。
- 参考代码：

```

1 townee = [
2     {'海底王国': ['小美人鱼', '海之王', '小美人鱼的祖母', '五位姐姐'], '上层世界': ['王子', '邻国公主']},
3     '丑小鸭', '坚定的锡兵', '睡美人', '青蛙王子',
4     [{'主角': '小红帽', '配角1': '外婆', '配角2': '猎人'}, {'反面角色': '狼'}]]

```

```
5      ]
6  print(townee[5][1]['反面角色'])
```

第5关 for循环和while循环

课后练习1：数数字

- 题目解析：
 - 使用for循环打印数字时可使用range()函数，循环内部可以加一个条件判断语句，当结果不等于4的时候打印数字。
 - 使用while循环打印数字时需要提前设置一个计数变量，每循环一次这个变量就加一，表示循环一次。因此循环条件可以根据这个计数变量的大小来设置。
- 参考代码：

```
1  # for循环的方法
2  for i in range(1,8):
3      if i !=4:
4          print(i)
5
6  # while循环的方法
7  a=0
8  while a<7:
9      a=a+1
10     if a !=4:
11         print(a)
```

课后练习2：轮流坐前排

- 题目解析：
 - 列表有append()和pop()两个方法，表示给列表增添或者删除元素。
 - append()方法接收一个参数，并且会把这个参数加到列表的末尾处。
 - pop()方法接收一个整数参数，参数表示删除列表中的第几个元素。
 - 可以使用pop()方法删除列表中的第一个元素，并把它赋值给一个变量以记录，然后再使用append()方法把这个变量加到列表的末尾位置，这样就实现了“换座位”的需求。
- 参考代码：

```
1 students = ['小明', '小红', '小刚']
2 for i in range(3):
3     student1 = students.pop(0) # 运用pop()函数，同时完成提取和删除。
4     students.append(student1) # 将移除的student1安排到最后一个座位。
5 print(students)
```

第6关 布尔值和四种语句

课后练习1：囚徒困境

- 题目解析：
 - 可以使用无限循环让双方选择，当满足一定条件时候跳出循环，可以使用input()语句接收双方的选择。
 - 针对双方都不认罪、一方认罪另一方不认罪、双方都认罪这三种情况，可以使用if...elif...else语句来进行条件判断。
- 参考代码：

```
1 while True:
2     a = input('A, 你认罪吗？请回答认罪或者不认：')
3     b = input('B, 你认罪吗？请回答认罪或者不认：')
4     if a == '认罪' and b == '认罪':
5         print('两人都得判10年，唉')
6     elif a == '不认' and b == '认罪':
7         print('A判20年, B判1年, 唉')
8     elif a == '认罪' and b == '不认':
9         print('A判1年, B判20年')
10    else:
11        print('都判3年, 太棒了')
12    break # 当满足开头提到的条件时，跳出循环。
```

课后练习2：记录困境中的选择

- 题目解析：
 - 我们可以使用一个列表来记录每一组实验者的选择，这个列表中的每一个元素就是一组实验者的选择。
 - 当使用input()函数获取到一组实验者的选择后，我们就可以用两个实验者的选择添加到列表中。使用列表的append()方法可以实现给列表中添加新元素。
 - 当满足条件跳出循环后，记录的最后一组就是选择了最优解的一组，也就是列表中的最后一个元素。

- 最后可以使用遍历列表的方式来打印实验者的组数和他们的选择。

- 参考代码:

```

1 n = 0
2 list_answer = []
3 while True:
4     n += 1
5     a = input('A, 你认罪吗? 请回答认罪或者不认: ')
6     b = input('B, 你认罪吗? 请回答认罪或者不认: ')
7     list_answer.append([a,b])
8     if a == '认罪' and b == '认罪':
9         print('两人都得判10年, 唉')
10    elif a == '不认' and b == '认罪':
11        print('A判20年, B判1年, 哎')
12    elif a == '认罪' and b == '不认':
13        print('A判1年, B判20年')
14    else:
15        print('都判3年, 太棒了')
16    break
17 print('第' + str(n) + '对实验者选了最优解。')
18 for i in range(n):
19     print('第' + str(i+1) + '对实验者的选择是: ' + str(list_answer[i]))

```

课后练习3：演员的作品

- 题目解析:

- 我们可以用一个字典来存放电影信息，字典的键是电影名，字典的值是对应电影的演员所组成的列表。
- 使用input()函数可以获取我们要查的演员名。
- 使用for循环遍历字典可以获取到字典的键，使用键值对对应的方式可取到字典的值，即是演员组成的列表，遍历演员列表就可找到具体的演员。

- 参考代码:

```

1 movies = {
2     '妖猫传': ['黄轩', '染谷将太'],
3     '无问西东': ['章子怡', '王力宏', '祖峰'],
4     '超时空同居': ['雷佳音', '佟丽娅'],
5 }
6
7 actor = input('你想查询哪个演员? ')
8 for movie in movies: # 用 for 遍历字典

```

```
9     actors = movies[movie] # 读取各个字典的主演表
10    if actor in actors:
11        print(actor + '出演了电影' + movie)
```

第7关 小游戏大学问

课后练习1：再来一盘

- 题目解析：

- 方案一：开启一个无限循环，设定一个跳出循环的条件。当得到肯定回复时，继续进行循环过程；当得到否定回复时，运行break，停止循环，退出游戏。
- 方案二：设置一个变量并赋值为布尔值True，然后进入while循环，循环的条件为变量的布尔值。接着通过input()函数接收输入的内容，接着使用if...else语句来根据输入的内容改变变量的布尔值决定是否跳出循环。

- 参考代码：

```
1 #方法一
2 import time
3 import random
4 while True:
5     player_victory = 0
6     enemy_victory = 0
7     for i in range(1,4):
8         time.sleep(1.5)
9         print(' \n——现在是第 %s 局——' % i)
10        player_life = random.randint(100,150)
11        player_attack = random.randint(30,50)
12        enemy_life = random.randint(100,150)
13        enemy_attack = random.randint(30,50)
14        print('【玩家】\n血量: %s\n攻击: %s' % (player_life,player_attack))
15        print('-----')
16        time.sleep(1)
17        print('【敌人】\n血量: %s\n攻击: %s' % (enemy_life,enemy_attack))
18        print('-----')
19        time.sleep(1)
20        while player_life > 0 and enemy_life > 0:
21            player_life = player_life - enemy_attack
```

```
22     enemy_life = enemy_life - player_attack
23     print('你发起了攻击, 【玩家】剩余血量%s' % player_life)
24     print('敌人向你发起了攻击, 【敌人】的血量剩余%s' % enemy_life)
25     print('-----')
26     time.sleep(1.2)
27 if player_life > 0 and enemy_life <= 0:
28     player_victory += 1
29     print('敌人死翘翘了, 你赢了! ')
--
```