解答课：

复习知识点：格式化输出、分支结构、循环结构、函数

练习一：

1、计算1000以内所有能被5整除的整数之和
提示：
#步骤1：以1000为最大值，定义循环，步长为1
#步骤2：每循环一次判断能否被5整除，并进行累加

```
# range(初始值,终点值，步长)是一个函数，用于生成一个等差数列，这个数列可以被for循环迭代

# 第一种解答：
c = 0      # 用于累加5的倍数
for i in range(1,1001,1):
    if i%5==0:       # i%5==0 其实是求 变量i中数字除以5的余数是不是 0
        c = c+i     # 每循环一次，将能够被5整除的整数，累加到变量 c 中
print(c)


# 第二种解答：更简单的写法
c = 0
for j in range(5,1001,5):
    c = c+j
print(c)
```

1、计算1000以内所有不能被7整除的整数之和
提示：
#步骤1：以1000为最大值，定义循环，步长为1
#步骤2：每循环一次判断能否被7整除，并进行累加

```
# 解答代码：
c = 0
for a in range(1,1001,1):
    if a%7!=0:        # 判断 a 中数据是否不能被7整除
       c = c+a
print(c)
```

练习二：

```
"""
1、从控制台接收一个不大于1000的正整数n，
统计1-n中不能被5整除的数的个数
累加1-n中能被5整除，且能被2整除的数的和
"""
# input 输入语句，用于从控制台接受一个键盘输入的数据，这个数据会自动保存为字符串数据进行返回
n = int(input("请输入一个不大于1000的正整数："))
c1,c2 = 0,0         # 同时申明两个变量，c1用于统计个数，c2用于累加
for i in range(1,n+1,1):
```

```
    if i%5!=0:     # 判断当前变量是否不能被5整除
        c1 = c1+1
    elif i%5==0 and i%2==0:      # 判断 是否 能被5整除,且能被2整除
        c2 = c2+i
# 当我们在打印的时候，可以用f修饰字符串，这样该字符串就可以是一个格式化字符串，字符串中可以通过{}直接拼接变量
print(f"不能被5整除的数的个数是：{c1}\n能被5整除,且能被2整除的数的和:{c2}")
```

```
"""
2、从控制台接收一个不大于1000的正整数n,并输出以下三类数字：
A1=能被5整除的数字中所有偶数的和；
A2=被7整除后余1的数字的个数；
A3=被9整除,且能够被2整除的数字的个数。
"""

# 解答代码1：
n = int(input("请输入一个不大于1000的正整数："))
A1,A2,A3 = 0,0,0       # 一次性声明三个变量，分别用于累加，统计，统计
for k in range(1,n+1,1):
    if k%5==0 and k%2==0 :
        A1 = A1 + k
    elif k%7==1 and k!=1:       # 1被7整除余后余1
        A2 = A2 + 1
    elif k%9==0 and k%2==0 :
        A3 = A3 + 1
print(f"能被5整除的数字中所有偶数的和:{A1}\n被7整除后余1的数字的个数:{A2}\n被9整除,且能够被2整除的数字的个数:{A3}")

# 问题：有些数据既满足A1又满足A2也满足A3 【比如90】，这个该怎么解决？
# 解答代码2：
n = int(input("请输入一个不大于1000的正整数："))
A1,A2,A3 = 0,0,0       # 一次性声明三个变量，分别用于累加，统计，统计
for k in range(1,n+1,1):
    if k%5==0 and k%2==0 :
        A1 = A1 + k
    if k%7==1 and k!=1:       # 1被7整除余后余1
        A2 = A2 + 1
    if k%9==0 and k%2==0 :
        A3 = A3 + 1
print(f"能被5整除的数字中所有偶数的和:{A1}\n被7整除后余1的数字的个数:{A2}\n被9整除,且能够被2整除的数字的个数:{A3}")


# 总结： if + elif 这样的写法下，这个判断是一个整体，只要满足 if 或 elif 其中的一个条件，其他的条件就不会再判断了，
#       某些情况下我们需要判断多次，可以使用多个if,多个if都是互相独立的。
```

练习三：

代码一（模拟生成验证码）：

```
# 导入随机工具包
import random
# = 号是赋值符号，用于将右边的值保存到左边变量
# = 右边代码意思是，通过随机工具包中一个名为randint的工具，获取100000 到 999999 中的一个随机数
yanZhenMa = random.randint(100000,999999)
# 打印得到的随机数
print(yanZhenMa)
```

代码二（模拟用户输入验证码）：

```
# 接受用户输入的内容，并保存到 user_input 变量中
user_input = input("请填入正确的验证码：")
# 注意，input函数能够接受用户输入的内容，但是它会将这个内容默认转为字符串类型。
```

题目要求：

结合以上两段代码，实现验证用户输入的验证码是否正确的功能。

需求1：

如果验证码校验成功，则在控制台输出"验证码正确，即将登录！"，

如果验证码校验不对，则在控制台输出"验证码错误，请重新输入！"

需求2：

在需求1实现的基础上，叠加循环，实现 给用户三次输入错误的机会，如果三次机会用完都没有输入正确，则提示"机器已锁死，请明日再试！"

```
# 参考代码：
# 由于需求一的实现代码比较多，为了我们在实现需求二的时候，不影响到需求一代码，我们可以将其封装成一个函数，然后调用其实现需求二
def yzm():
    # 导入随机工具包（Python自带）
    import random
    # = 号是赋值符号，用于将右边的值保存到左边变量
    # = 右边代码意思是，通过随机工具包中一个名为randint的工具，获取100000 到 999999 中的一个随机数
    yanZhenMa = random.randint(100000,999999)
    # 打印得到的随机数【在实际应用中，肯定是不能打印验证码的，这个打印语句得注释或删除】
    print(yanZhenMa)

    # 接受用户输入的内容，并保存到 user_input 变量中
    user_input = int(input("请填入正确的验证码："))
    # 注意，input函数能够接受用户输入的内容，但是它会将这个内容默认转为字符串类型。
    # 【下面的打印语句也是测试代码，在实际使用中得注释或删除】
    print(f"测试参数：\n系统生成的验证码是：{yanZhenMa}\n用户输入的验证码是：{user_input}")
    return  yanZhenMa,user_input       # 返回系统生成的验证码，以及用户输入的验证码
```

```
x = 0       # 声明一个变量，用于记录失败次数
# 需求2：在需求1实现的基础上，叠加循环，实现 给用户三次输入错误的机会，如果三次机会用完都没有输入正确，则提示"机器已锁死，请明日再试！"
while True:
    if x >= 3:     # 当x中记录的值是 >或者=3的情况下，就退出循环
        print("--------------------------机器已锁死，请明日再试！")
        break
    yanZhenMa, user_input = yzm()       # 调用yzm函数可以得到返回的 系统和用户输入的验证码
    # 需求一：结合以上两段代码，实现验证用户输入的验证码是否正确的功能
    if user_input == yanZhenMa:
        print("验证码正确，即将登录！")
        break       # 在验证码正确之后，跳出循环
    else:
        print("验证码错误，请重新输入！")
    x = x + 1       # 在这里实现每循环一次记录一次

"""
总结：
    在将一段代码封装成一个函数的时候，我们这个函数到底需不需要返回值需不需要参数呢？
虽然说，我们之前在学习函数的时候说道，程序员是函数的创造者，函数有没有返回值是程序员完全能够决定的，
但是，在实际使用的时候，作为程序员还是需要结合情况思考，对程序进行设计之后，再实际操作，这样的话，可以尽可能减少代码的改动。
这体现了开发分析的重要性。
开发分析能力不是一朝一夕就能够培养好的，咱们加油，积累代码量，多多思考！
"""
```