《大学计算机基础》课程总结与备考方法

北京航空航天大学

计算机学院

艾明晶

2018.6.14

内容提要

一、课程体系与教学案例

二、基于Python的实验体系

三、重点复习内容

四、期末备考方法

一、课程体系与教学案例

■ 课程体系（理科）：48学时。26理论学时+22实验学时

第1章：计算思维与计算机模型	4	计算思维的核心思想，计算的基础；计算机的理论模型与物理实现；信息在计算机中的表示；计算机系统基本结构（自学）；计算思维方法的案例
第2章：问题抽象与建模	2	科学抽象过程与方法；模型的定义和分类；数学建模的基本过程和方法；建模典型案例
第3章：程序设计基础与数据结构	8	程序与程序设计语言；Python的基本语法；程序控制结构；数据与数据结构；Python实现自定义数据结构
第4章：算法设计及优化	8	计算机求解问题与算法；问题求解的经典算法（枚举，递归，动态规划，贪心算法）；算法的分析与评估（求解效率的评估，查找算法，排序算法）
第5章：科学计算与数据处理	4	科学计算基本概念；Python科学计算及图表绘制；常用数据处理（插值，拟合）；GUI设计
第6章：工程思维与领域前沿		综合计算思维的各方法，讲述计算机科学家的思维和工程设计思想

内部教材：基于计算思维的大学计算机基础

目 录

教材是复习备考的重要资料！

教学案例（1）

- **教学案例是教学内容的重要载体**
- 计算思维包含若干核心概念，但是比较抽象，学生一般难以理解
- 体现计算思维主要方法的若干教学案例，使抽象的概念具体化
- **配套的教学案例（74个）**
- 在教学过程中，采用案例驱动的启发式教学方法

第2章 问题抽象与建模（8个）

1	归纳的案例：斐波那契数列
2	数学建模一般步骤的案例：车辆的跟随距离
3	实验建模案例：人口预测问题
4	综合建模案例：小行星绕太阳运行的轨道问题
5	典型案例：地铁自动购票机找零
6	典型案例：学生选课
7	典型案例：百度地图中的最短路径
8	典型案例：食堂排队问题

教学案例（2）

第3章 程序设计基础与数据结构（20个）

1	程序IPO模式：温度转换器
2	程序IPO模式：古巴比伦计算数字x的平方根
3	赋值语句示例：公交车客运能力
4	面向对象的程序设计方法示例：定义直线的类
5	列表POP方法的应用：将一个列表中的各个元素倒序排列
6	元组的应用示例：计算任意两个正整数的所有公约数
7	字符串的应用示例：计算任意一个单词对应的总分数
8	字符串lower方法示例：用户名检查
9	字符串split方法示例：切分单词
10	字典示例：创建一个简单数据库

11	函数示例：计算圆面积
12	递归函数示例：阶乘的计算
13	递归函数示例：兔子问题
14	while语句示例：计算一个整数的平方
15	while True语句示例：成绩连续录入与计算平均分
16	线性表的应用示例：学生信息管理系统
17	栈的应用示例：十进制整数转换为二进制数
18	队列的应用示例：烫手的山芋
19	二叉树的应用示例：实现四则运算
20	图的应用示例：课程学习的先后顺序与拓扑排序

教学案例（3）

第4章 算法设计及优化（18个）

1	枚举算法：求解阿姆斯特朗数
2	递归算法：求第N项斐波那契数
3	递归算法：求解汉诺塔问题
4	迭代算法：牛顿迭代法计算5的算术平方根的近似值
5	分治算法：一元三次方程求解
6	动态规划： 0/1背包问题
7	动态规划与贪心算法比较：找零问题
8	贪心算法：一个教室安排多门课程
9	贪心算法：部分背包问题

10	贪心算法：Dijkstra算法解单源最短路径问题
11	顺序查找：学生成绩查询
12	二分查找：在升序数据序列中查找任意一个关键字
13	哈希查找：在无序数据序列中查找任意一个关键字
14	简单选择排序：对随机数组升序排序
15	插入排序：对随机数组按升序排序
16	冒泡排序：对随机数组按降序排序
17	归并排序：对随机数组按升序排序
18	求解的优化：RSA算法加密与解密优化

教学案例（4）

第5章 科学计算与数据处理（28）

1	绘制交流电压随时间变化的曲线图
2	在同一张图上绘制正弦和余弦函数曲线
3	绘制多子图示例：各年级的语文平均成绩绘制
4	绘制多图表和多子图示例：绘制指数曲线、正弦函数和余弦函数曲线
5	绘制柱状图示例：学生各班各科成绩绘制
6	绘制饼图示例：按照等级绘制某班语文成绩分布饼图
7	绘制雷达图示例：某学生各科成绩的雷达图
8	最小二乘法多项式拟合示例
9	非线性最小二乘拟合示例
10	一维插值示例：绘制正弦波曲线
11	一维插值案例：温度测量曲线插值
12	二维插值案例：平板表面温度插值
13	Label 组件的简单应用
14	Label组件的高级应用

教学案例（5）

第5章 科学计算与数据处理（28）

15	Button组件的简单应用
16	Checkbutton组件的简单应用：业余爱好
17	Radiobutton组件的应用：课程问卷调查
18	创建顶级菜单示例
19	创建下拉菜单示例
20	创建弹出菜单示例
21	Text组件的简单应用
22	Text组件的高级应用
23	Scrollbar组件的应用
24	标准对话框模块的应用：打开任意路径下任意文件
25	Listbox组件的应用：问卷调查
26	画布的应用：游戏初始界面设计
27	pack方法的应用
28	grid方法的应用：用户登录界面

二、基于Python的实验体系

实验体系

- 围绕**程序设计**、**数据结构**、**算法**、**科学计算**、**可视化**这几个课程核心内容，构建了一个层次化、循序渐进的实验体系
- 通过以问题求解为导向的Python编程实践，使学生更好理解运用“计算思维”求解问题的思想，掌握其方法

实验安排

实验安排

- 实验环节22学时
- 包括8次平时实验和1次综合设计实验（大作业）
- 期中考试（2学时）
- 综合设计实验答辩（2学时）

序号	实验名称	学时数
1	Python编程环境的使用	2
2	程序控制结构的设计	2
3	问题的描述—基本数据结构（字符串、列表、字典）	2
4	问题的描述—自定义数据结构（栈或者队列）	2
5	基本算法及其实现	2
6	复杂算法设计与实现	2
7	科学计算与绘图	2
8	插值、拟合计算与图形用户界面	2
9	综合设计实验	2

实验内容（1/5）

- **组织助教团队共开发了29个平时实验，4个综合设计实验**
  - **基础级别实验两次，共计10个实验，掌握Python基本语法**

实验名称	实验任务	实验内容	主要知识点
实验1 Python编程环境的使用	1-1	圆台体积计算	变量赋值，input函数，print函数，简单的数值计算
	1-2	小明的实验数据	内置库函数math.sqrt()的调用
	1-3	简单密码	算术运算，函数的定义与调用
	1-4	ASCII码	ASCII码，ord()函数和chr()函数
	1-5	打点计时器（选做）	一元线性回归，浮点数的计算
实验2 程序控制结构的设计	2-1	刘同学家的电费	if语句
	2-2	冰雹猜想	while循环语句
	2-3	TD线	for循环语句，range函数，break语句和continue语句，嵌套if语句
	2-4	输出菱形	for循环语句，print()语句的换行控制
	2-5	质因数分解（选做）	if语句，while语句，for语句（枚举法）

实验内容（2/5）

- **数据结构实验两次，共8个，夯实编程基础与数据结构使用和设计**

实验名称	实验任务	实验内容	主要知识点
实验3 问题的描述—基本数据结构（字符串、列表、字典）	3-1	GPA计算	列表
	3-2	学生信息	字典
	3-3	命名风格	字符串。list()函数，ord()函数和chr()函数
	3-4	简单数据管理	多重循环嵌套，for循环遍历数据结构，列表的sort()方法
	3-5	质数的和（选做）	筛法求质数
实验4 问题的描述—自定义数据结构（栈或者队列）	4-1	括号匹配	栈
	4-2	由9和0组成的数字	队列，while True语句
	4-3	矩阵运算	类的定义，类的方法

实验内容（3/5）

- **算法实验两次，共6个，夯实基本算法的使用与较复杂算法如排序算法、动态规划的设计**

实验名称	实验任务	实验内容	主要知识点
实验5 基本算法及其实现	5-1	台阶走法	递归算法
	5-2	正整数分解	正整数n的划分，递归算法
	5-3	迎春舞会之集体舞	枚举算法或递归算法
实验6 复杂算法设计与实现	6-1	NULL的奖学金	按行输入空格分隔的多个数据，多关键字排序
	6-2	合唱排队	动态规划
	6-3	分桃子	贪心算法

实验内容（4/5）

- **科学计算与数据处理实验两次，共5个，训练绘制图表、插值、拟合和GUI设计**

实验名称	实验任务	实验内容	主要知识点
实验7 科学计算与绘图	7-1	函数图像绘制练习	matplotlib和numpy库，pylab的mpl模块解决Matplotlib无法显示中文字体的问题
	7-2	生成和管理科学数据	random库的random()函数，NumPy库的random.randn()函数，Scipy库的stats.describe()函数
	7-3	柱状图绘制练习	matplotlib和numpy库
实验8 插值、拟合计算与图形用户界面	8-1	注册界面	Tkinter库的Frame、Label、Entry（输入单行文本）和Button组件，grid方法或pack方法
	8-2	数据处理	random库的random()函数，NumPy库的random.randn()函数，scipy.interpolate的spline函数,Tkinter库

实验内容（5/5）

- **最后一次综合设计实验（大作业），只提作业要求，学生任选一个自主设计完成，要求有GUI，作为实验课程学习成果的关键考评指标**
  - ✓ 大作业题目难度系数不同，并具有一定应用场景
  - ✓ 大作业中学生将应用抽象、建模、数据结构、算法、编程实现等计算思维的核心理念

综合设计实验	作业一	天气状况统计分析
	作业二	最短路径问题
	作业三	数独
	作业四	俄罗斯方块

- 你可以充分发挥自己的想象力，创新设计，上网查阅资料，自学更多的新知识
- **大作业答辩环节**要求学生PPT演讲和现场答辩，锻炼同学们的总结、表达、思辨能力

考试范围

- **考题内容和形式**
  - 与课堂上讲过的案例或做过的实验类似
  - 考核对计算思维的掌握，用Python语言进行程序设计、解决具体问题的能力
    - ✓ 基本的数据输入和输出
    - ✓ 简单计算
    - ✓ 基本的程序结构
    - ✓ 基本的数据结构
    - ✓ 典型算法的使用
- **一共5道题（满分100分，占课程总成绩的45%）**
  - 前4道题主要考核基本能力，难度大致与平时未提前发布实验相当
  - 第5道题考核综合运用知识解决问题的能力，难度大致与平时提前发布实验相当

三、重点复习内容

- **重点复习第3章和第4章**
  - input函数
  - print函数
  - if语句
  - for语句
  - while语句
  - 函数的定义和调用

  基本语法

  - 列表
  - 字符串

  基本数据结构

  - 栈
  - 队列

  线性数据结构

  - 枚举算法
  - 查找算法
  - 排序算法

  典型算法

（一）基本语法

1、input函数

- input函数用来提供用户输入合法的Python表达式
- input函数使用一个字符串作为参数，用来提示用户应输入的内容——**一般必须有此提示语！括号中不要为空！**

格式 变量名=*input*(" <字符串> ")

使用input函数注意事项

- 但是OJ（在线评测系统）无法评测input中的字符串，故在提交到OJ上时，要求input后的括号中为空
- 如果有多个输入变量，在调试时可以分别加上提示语
- 调试通过后，再将这些语句注释掉，重新写几行input()

- **通过input函数输入的内容默认其类型为字符串**
- **如果需要进行算术运算，则必须先将其转换为int或float**

如何在一行中输入多个数据，每个数据之间用空格隔开？

- **若希望一行输入多个数据，每个数据之间用空格隔开**
  - 则可以用**a,b,c = input().split()**，按照空格将输入的多个数据分开，分别赋给等号左边的变量【参见“实验任务1-2 小明的实验数据”】
  - 当一行输入的数据很多时，可以采用**s = input().split()**，直接将输入的所有数据存入一个列表（s）中，每个数据是列表中的一个元素，用引号括起来，表示字符串
  - 当需要输入多行这样的数据时，则可以采用for语句

参见input_data.py

```
n = int(input())          #输入数据的总行数
for i in range(n):
    s = input().split()
    print("s:",s)
```

2、 print函数

- print函数用于在Shell窗口显示用户希望显示的内容（字符串或变量的值）

格式 *print*(" <字符串1>", <变量1>, " <字符串2> ", <变量2>)

- 当需要显示的内容较多时，**最好在变量名的前面加上说明其含义的字符串**，否则，难以判断输出的是哪个变量的值！

```
print("正态分布样本值为：", normal)
```

- 但是OJ（自动评测系统）无法评测print中的字符串，故在提交到OJ上时，要求print后的括号中只有变量名
- 故可以在IDLE中调试时在变量名前加上提示字符串
- 调试通过后，再将这些语句注释掉，重新写几行print()

（1）使用print函数打印中间过程

- **调试程序时，尽量使用print函数打印出中间过程（甚至包括循环变量的值），以便分析计算过程是否正确，有利于查错！**

【例：实验5-3 迎春舞会之集体舞】

- 只要求输出穿夏季校服的同学所构成的最大等腰直角三角形的边长
- 但为了观察如何遍历队形阵列可能的范围，找出所有可能满足题目要求（即全是 '-' ）的直角三角形，打印出可能的边长、起始行号、起始列号，以及不为 '-' 的元素
- 并打印符合要求的直角三角形起始行的行号、末尾行的行号，起始有效字符的列号、末尾有效字符的列号

实验5-3程序

```
#实验5-3 迎春舞会之集体舞
#（1）用嵌套列表A存储每行的服装信息
n=int(input())                          #总行数
A=[]                                    #嵌套列表，存储每行的服装信息
for i in range(n):                      #输入n行表演者的服装信息
    A.append(list(input()))             #将输入的一行字符串转换为列表元素，添加到列表A中

print("n行表演者的服装信息A=")
for i in range(n):
    print(A[i])

#（2）判断指定范围的直角三角形是否全为'-'，i为起始行数，x,y为起始行的起始有效字符以及末尾有效字符的列号
def f(i,x,y):
    for j in range(y-x+1):              #遍历指定范围内的每一行。某一行的行号=i+j。例如i=1，x=1，y=3。则j的范围为[0,2]，指定范围内的行号为[1,3]
        for k in range(x+j,y+1):        #遍历[x+j,y]之间的每一列。对于某个j，起始有效字符的列号为x+j，末尾有效字符的列号为y
            if A[i+j][k]!='-':          #若第i+j行的第k个元素不为'-'
                print("A[%d][%d]!='-'，跳出循环" %(i+j,k))
                return False
    return True                         #若指定范围的所有元素都为'-'，则满足要求
```

实验5-3程序（Cont.）

```
#（3）遍历队形阵列可能的范围，找出所有可能满足题目要求（即全是‘-’）的直角三角形，并返回最大边长
def g(n):                                   #表示表演者一共n排
    for m0 in range(n):                     #用m0遍历0到n-1，m0表示起始行号
        m=n-m0                              #用m实现对边长从大到小(n到1)的遍历
        print()
        print("边长m=",m)
        for i in range(m0+1):               #遍历所有可能的起始行([0,n-1] )
            print("--------------------")
            print("i=",i)
            for j in range(i,m0+1):         #遍历所有可能的起始列（ [i,n-1])。第i行的起始列号为i
                print("j=",j)
                if(f(i,j,j+m-1)):   #调用f(i,x,y)函数。若该范围满足要求，则已找到最大边长，即m。起始行为i，起始行的起始有效字符的列号为j，由于边长为m，则末尾有效字符的列号为j+m-1
                    print()
                    print("符合要求的直角三角形起始行的行号i=",i," ，末尾行的行号为i+m-1=",i+m-1)
                    print("起始有效字符的列号j=",j, ",末尾有效字符的列号为j+m-1=",j+m-1)
                    return m
    return 0                                #若没有找到满足要求的m，返回0

print("符合要求的最大等腰直角三角形最大边长m=",g(n))
```

实验5-3程序运行结果

```
>>>
5
-----
 #---
  #-#
   --
    #
n行表演者的服装信息A=
['-', '-', '-', '-', '-']
[' ', '#', '-', '-', '-']
[' ', ' ', '#', '-', '#']
[' ', ' ', ' ', '-', '-']
[' ', ' ', ' ', ' ', '#']

边长m= 5
----------------------
i= 0
j= 0
A[1][1]!='-', 跳出循环
```

```
边长m= 4
----------------------
i= 0
j= 0
A[1][1]!='-', 跳出循环
j= 1
A[2][4]!='-', 跳出循环
----------------------
i= 1
j= 1
A[1][1]!='-', 跳出循环
```

```
边长m= 3
----------------------
i= 0
j= 0
A[1][1]!='-', 跳出循环
j= 1

符合要求的直角三角形起始行的行号i= 0 ， 末尾行的行号为i+m-1= 2
起始有效字符的列号j= 1 ,末尾有效字符的列号为j+m-1= 3
符合要求的最大等腰直角三角形最大边长m= 3
>>>
```

- **可能的边长m为4时**
  - 可能的起始点为0行0列、0行1列、1行1列，因为只有它们为起始点时，能够保证边长为4
  - 对每个可能的起始点所构成的三角形，调用f(i,x,y)函数，判断此范围所构成的直角三角形的每一个元素是否都是‘-’，若不是，则跳出循环，判断下一个三角形。当判断完第三个三角形后，跳出循环，m减1。继续下一个第一重循环

（2）格式化字符串

- **当需要打印的变量值为不同类型（如浮点数）时，可以采用格式化字符串的方法，来指明变量值的类型、位数**

【例：实验8-2 数据处理】

```
print("泊松分布的参数lamb=", lamb)
print("泊松分布生成的数据生成的频率在λ-0.5处插值为：%0.5f" % spline(x, y, lamb - 0.5))
```

```
泊松分布的参数lamb= 8
泊松分布生成的数据生成的频率在λ-0.5处插值为： 0.14710
```

字符串格式化操作符

- %称为**字符串格式化操作符**，在%的左侧放置一个字符串（**格式化字符串**），右侧放置**希望格式化的对象**
- 格式化字符串的%0.5f部分称为**转换说明符**，它标记了需要插入转换值的位置，以及它的格式。**%0.5f**表示该位置的值会被格式化为浮点数，保留小数点后5位
- 还可以有%s，表示被格式化为字符串；%d表示被格式化为带符号十进制整数

（3）如何使用end参数控制输出格式？

- **在print函数中，可以使用end参数控制打印内容的结尾是什么**
  - end='\t'表示输出的末尾以Tab键结束，则下一条print语句打印的内容将与刚才打印的内容的第一个字符间隔8个字符输出，不换行
  - end=' ' 表示以空格结尾，则下一条print语句打印的内容将接着刚才打印的内容末尾输出，不换行

【例：2016年考题】优雅地输出一个九九乘法表。要求：

- （1）任意两个数相乘必须单独输出，如5*2=10与5*4=20不应在同一次调用print函数中完成
- （2）行列对齐

```
>>>
1*1=1
2*1=2  2*2=4
3*1=3  3*2=6  3*3=9
4*1=4  4*2=8  4*3=12 4*4=16
5*1=5  5*2=10 5*3=15 5*4=20 5*5=25
6*1=6  6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1=7  7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1=8  8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*1=9  9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
>>>
```

输出九九乘法表的程序（方法一）

```
# 输出一个九九乘法表
#print_example1.py
#方法一

for i in range(1,10):              #i为被乘数
    for j in range(1,i+1):         #j为乘数
        print("%d*%d=%d" % (i,j,i*j), end='\t')    #end='\t'表示输出的末尾以Tab键结束
    print()                        #一行结束，换行
```

```
>>>
1*1=1
2*1=2   2*2=4
3*1=3   3*2=6   3*3=9
4*1=4   4*2=8   4*3=12  4*4=16
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
>>>
```

输出九九乘法表的程序（方法二）

```
# 输出一个九九乘法表
#print_example2.py
#方法二：也可以使用字符串完成不换行

for i in range(1,10):              #i为被乘数
    for j in range(1,i+1):         #j为乘数
        s=str(i) + "*" + str(j) + "=" + str(i*j) + "  "          #每个算式后面后面空2格
        if i*j<10:                  #当乘积为个位数时，在其后面添加一个空格，以便与乘
积为两位数的左对齐
            s+=" "
        print(s, end='')           #以空格结尾，则不换行，接着打印下一条
    print()                        #一行结束，换行
```

```
>>>
1*1=1
2*1=2   2*2=4
3*1=3   3*2=6   3*3=9
4*1=4   4*2=8   4*3=12  4*4=16
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
>>>
```

3、if语句（条件语句）

- **if语句（条件语句）**用于实现选择结构（分支结构 ），根据条件的判断确定应该执行哪一条分支的语句序列
- 条件语句结构分为三种形式

（1）非完整性if语句

```
if 布尔表达式:
    语句序列
```

（2）二重选择的if语句

```
if 布尔表达式:
    语句序列1
else:
    语句序列2
```

（3）多重选择的if语句

（需要检查两个以上条件时）

```
if 布尔表达式1:
    语句序列1
elif 布尔表达式2:
    语句序列2
else:
    语句序列3
```

if语句示例

【例：实验2-1 刘同学家的电费】

- 月用电量在150千瓦时及以下部分按每千瓦时0.4463元执行，月用电量在151~400千瓦时部分按每千瓦时0.4663元执行，月用电量在大于400千瓦时部分按每千瓦时0.5663元执行
- 已知用电总量，计算应缴纳的电费是多少

```
#实验任务2-1 刘同学家的电费

n = int(input())        #用电总量(单位以千瓦时计)，不超过10000

if n <= 150:
    cost = n * 0.4463
elif n > 150 and n <= 400:
    cost = 150 * 0.4463 + (n - 150) * 0.4663
else:
    cost = 150 * 0.4463 + 250 * 0.4663 + (n - 400) * 0.5663

print("%.3f" % cost)        #保留到小数点后3位(单位以元计)
```

```
>>>
255
115.906
>>>
```

if语句使用注意事项

- 当条件比较复杂、多于一个时，可以用布尔运算符and、or或not来组合真值，写成复合表达式
- 当某个条件成立时，如果还需要根据另一个条件是否成立来决定执行哪个语句块，可以嵌套使用条件语句
- 条件语句中可以使用哪些运算符？
  - **比较运算符**：<、>、>=、<=
  - **等值运算符**：==、!=
  - **成员测试运算符**：is、is not、in 、not in
- 根据题目要求选择合适的条件语句结构。注意考虑各种可能的情况，列出所有分支！

4、for语句（循环语句的一种）

- **for语句**用于实现循环结构，即对同一段程序重复执行多次
- 如果需要对一个集合（序列或其他可迭代对象）的每个元素都执行同一个代码块，适合采用for语句
  - 利用for语句，可以遍历列表、元组、字符串或字典中的每个元素（键）
  - 当循环次数已知时，适合使用for语句
  - 枚举算法中，常使用for语句进行一重或多重循环

```
for 变量 in 序列或其他可迭代对象:
    代码块
```

- for语句经常与range函数结合起来使用，使用range函数指定循环迭代的范围

for语句示例

【例：实验2-3 TD线】

- 快期末了，小天的48次TD线还没刷满。现在已经过去了N周（1<=N<=15），前N周他每周分别刷了x_1，x_2...x_N次TD，TD线第15周结束时就要关闭了。
- 已知每周最多可以刷10次TD，请你计算能否按时完成任务。如果能完成，请输出每周至少需要刷的TD次数（为一个整数）；如果不能按时完成，请输出‘gg’。
- 输入N，逐行输入前N周小天每周分别刷TD线的次数

分析

- 要求逐行输入前N周小天每周分别刷TD线的次数——采用for语句，循环体为input()
- 根据前N周小天每周刷的次数，可以计算出前N周已刷TD线总次数total——在for语句中直接将每周刷的次数累加即可
- 计算能否按时完成任务（48次）——先计算剩余的应该刷的次数remain=48-total
- 再判断在剩下的周数（15-N）内，按每周最多可以刷10次TD，能否完成任务

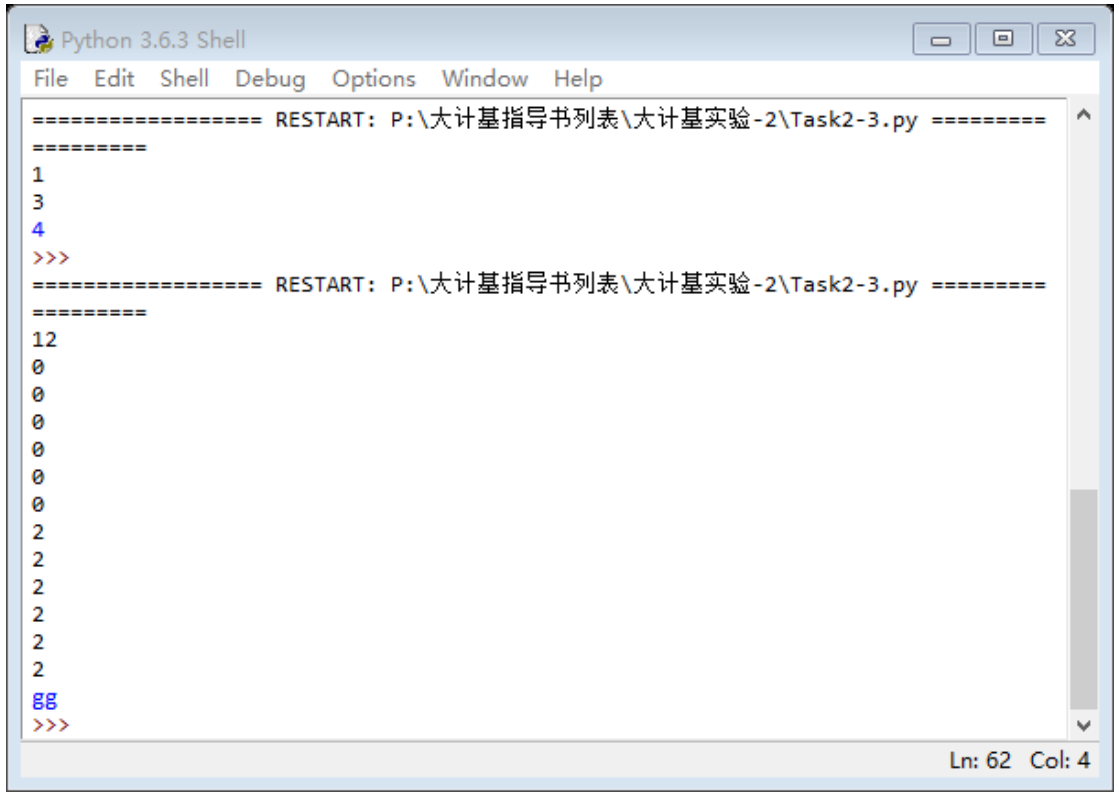
```
if remain > 10 * (15 - N):
```

实验2-3程序

```
#实验任务2-3  TD线

#（1）输入已经过去的周数N
N = int(input())                    #正整数N，表示已经过去的周数

#（2）输入前N周小天每周分别刷TD线的次数，并计算前N周已刷TD线总次数
total = 0                           #前N周已刷TD线总次数，初值为0
for i in range(N):
    total += int(input())           #输入前N周小天每周分别刷TD线的次数

#（3）计算剩余的应该刷的次数
remain = 48 - total

#（4）判断在剩下的周数内，按每周最多可以刷10次TD，能否完成任务
if remain > 10 * (15 - N):
    print('gg')
else:
    weekly = remain // (15 - N)       #在剩下的周数内，每周至少需要刷TD线的次数
    if remain == weekly * (15 - N):
        print(weekly)
    else:
        print(weekly + 1)
```

嵌套if语句

实验2-3程序运行结果




```
>>>
3
10
10
10
2
>>> ================================ RESTART =
>>>
4
5
10
8
6
2
```

5、while语句（循环语句的另一种）

- **while语句**用于实现循环结构，在任何条件为真的情况下重复执行一个语句序列
- **有条件地执行一条或多条语句。**只要循环条件表达式为真，则循环语句就重复执行
  - 利用while语句，也可以遍历列表、元组、字符串的每个元素，但不如利用for语句简洁
  - 利用while语句，不能遍历字典中的每个键，因为字典中的键值对没有顺序之分
  - 当循环次数未知时，适合使用while语句

```
while 布尔表达式:
    语句序列
```

- .在循环体中，必须有一条改变循环条件表达式的值的语句！或者采用break强制退出循环。否则，循环将无休止地进行下去

- **无限循环**适于用while语句实现，用**while True**即可
  - 需要时用**break**（打断，直接跳出循环）跳出循环
  - **continue**：继续，不再执行其后语句，但继续执行下一个循环

while语句示例

【例：使用while语句遍历列表】

- 逐行打印列表中各个元素

```
# list_traversal2.py
#使用while语句遍历列表

name=['Alice', 'Helen', 'Peter', 'John']
i=0                          #循环变量（索引号）赋初值
print('name中的所有名字是：')
while i < len(name):
    print(name[i])
    i += 1                   #改变循环条件表达式的值
```

```
name中的所有名字是：
Alice
Helen
Peter
John
>>>
```

while True语句示例

【例：实验2-2 冰雹猜想】

- 任意写出一个正整数N，并且按照以下的规律进行变换：
  - ✓ 如果是个奇数，则下一步变成3N+1；
  - ✓ 如果是个偶数，则下一步变成N/2。
- 人们发现，无论N是怎样一个数字，最终都无法逃脱回到谷底1。
- 请编写程序，输入正整数N，依据以上规则，输出每步计算的结果（整型变量），到1为止。

```
#实验任务2-2 冰雹猜想
a = input()                    #输入任意一个正整数
a = int(a)

while True:                    #无限循环
    if abs(a) == 1:            # 如果a是1, 运算结束
        break                  #跳出while循环
    elif a % 2 == 0:           # 如果是偶数，则除以2
        a = int(a / 2)
        print(a)
    else:                      # 如果是奇数，则乘3加1
        a = int(a * 3 + 1)
        print(a)
```

```
>>>
20
10
5
16
8
4
2
1
>>>
```

Python的常用内建函数

Python的常用内建函数

- abs(x)求绝对值
- float(x)将一个字符串或数转换为浮点数
- int(x)把浮点数向下取整为整数，把字符串或数转换为整数
- pow(x,y)进行指数运算，计算x的y次方
- sum：求和函数，用于对序列进行求和计算

- max(iterable[, args...][key])返回集合中的最大值
- min返回序列中最小的元素
- str([object]) 将一个数转换为string类型
- list()对一个序列（如字符串）创建一个列表
- len()返回序列中所包含元素的数量

Python提供的数学类函数库math

函数名	数学表示	含义	示例	说明
pi	π	圆周率π的近似值，15位小数	>>> math.pi 3.141592653589793	
e	e	自然常数e的近似值，15位小数	>>> math.e 2.718281828459045	
ceil(x)		向上取整	math.ceil(32.9) 33	
floor(x)		对浮点数向下取整	>>>math.floor(32.9) 32	
log(x)	lgx	对数，以e为基	>>> math.log(10) 2.302585092994046	
log10(x)	$\log_{10}x$	对数，以10为基	>>> math.log10(10) 1.0	
exp(x)	e^x	x次幂，以e为底	>>> math.exp(3) 20.085536923187668	
sqrt(x)	$\sqrt{x}$	计算一个浮点数的平方根	>>> math.sqrt(9) 3.0	该函数不能计算复数或虚数的平方根
sin(x)	sinx	正弦函数	>>> math.sin(math.pi/2) 1.0	

6、函数的定义和调用

- 使用def（函数定义）语句来创建（或定义）函数
- 函数的定义一般包括函数名、参数、函数体和返回值等4部分，其中函数名和函数体是必不可少的部分，参数和返回值可根据需要进行定义
  - 参数称为**形式参数（形参）**。形式参数之间用逗号分开。有时也可以没有参数
  - 一般函数会有返回值，使用return语句把计算结果返回给调用函数的位置

```
def 函数名(参数):
    函数体
    return 返回值
```

函数的调用

- **调用函数**：在程序内或其他程序中使用已定义的函数实现特定的功能
  - 如果函数在定义时有形参，则在调用函数时必须给函数提供实际参数（**实参**）
  - 如果函数在定义时**没有形参**，则在调用函数时函数名的**后面也必须写出括号**

切记！

- 在用Tkinter进行GUI设计时，Button、Checkbutton组件的回调函数定义时一般没有参数
  - 则在创建Button、Checkbutton组件时，参数command后面直接赋回调函数名，后面不要有括号，否则出错！
- 当回调函数有参数时，则必须使用匿名函数。即在回调函数名之前使用“lambda:”，在回调函数名的后面必须要跟括号，括号内提供实际参数

```
#计算器程序中创建等号按钮
btn_equal = Button(frame, text = '=', width = 12, command = lambda: calcu(expression))
                                                    #calcu为回调函数
```

函数的调用——两种方法

- **方式一**：如果函数是在同一程序中定义和调用，则调用时格式如下：

  格式 函数名(<实参>)

- **方式二**：如果函数是分别在**不同**的程序中定义和调用，则在顶层文件中首先要**导入定义函数的模块**；然后再调用该模块中的函数
- 调用时格式如下：

  格式 模块名.函数名(<实参>)

  定义函数的程序名称

函数的定义和调用示例

【例】定义一个函数，计算圆面积。然后在同一程序中调用该函数，计算任意半径的圆的面积。

```
# 例3.16-circle_one.py
#函数的定义和调用在同一程序内
import math

#定义函数，计算圆面积
def area(radius):
    s=math.pi*(radius**2) #圆面积
    return s

r=float(input('请输入半径：'))
my_s=area(r)  #调用本程序中的area函数求圆面积

print ('圆面积=%0.1f' % my_s)  #采用格式化字符串的方法，结果只显示一位小数
```

函数的定义

调用函数

```
>>>
请输入半径：4
圆面积=50.3
```

（二）基本数据结构

1、列表

- **序列的通用操作**
  - 索引（indexing）：通过元素编号访问（获取）序列中的某个元素
  - 分片（slicing）：访问序列中的一定范围（间隔一定步长）内的元素
  - 加（adding）：使用加号连接两个或两个以上的序列成为一个序列
  - 乘（multiplying）：将原序列重复若干次，连接成一个新序列
  - 检查成员资格：使用成员资格运算符in检查一个值是否在某个序列中
- **列表的基本操作**
  - 元素赋值
  - 元素删除
  - 分片赋值

列表的遍历和列表方法

- **列表的遍历**
  - 使用for语句遍历最简单
- **列表方法**
  - **append**
  - count
  - extend
  - index
  - insert
  - **pop**
  - remove
  - **reverse**
  - **sort**

- 方法的调用：**对象.方法(参数)**

- 使用POP方法可以实现一种常见的数据结构——**栈**（Stack），利用列表的**pop方法**使栈中元素**出栈**；
- 利用列表的**append方法**将元素压入栈中
- reverse()将列表中的元素反向存放

sort方法与排序

- sort方法用于在原位置对列表进行排序，这意味着经过排序，原列表被改变了
- **但如果用户需要一个排好序的列表副本，同时又保留原列表不变，怎么做？**

- **方法**：先采用分片（调用x[:]）复制整个列表给另一个变量（y），再对该变量（y）排序，则原列表不变

```
# -*- coding: utf-8 -*-
#利用sort方法排序，同时保留原列表不变
#sorting.py

x=[4,6,2,1,7,9]          #原始数据
y=x[:]                   #复制整个列表x，并赋值给y
y.sort()                 #对y按值从小到大排序

print('x中所有元素是：',x)
print('排序后的y中所有元素是：',y)
```

```
x中所有元素是：  [4, 6, 2, 1, 7, 9]
排序后的y中所有元素是：  [1, 2, 4, 6, 7, 9]
```

不能写为y=x

2、字符串

- **字符串**是用双引号" "或单引号' '括起来的一系列字符
- **序列的通用操作**也适于字符串
- **字符串的遍历**
- **有用的内置函数**
  - **list()函数**：将一个序列（如字符串）转换为列表（列表中每个元素是一个字符）
  - **ord()函数**：获取某个字符的ASCII码
  - **chr()函数**：将某个字符的ASCII码转化为该字符

字符串常用方法

- **字符串常用方法**
  - **find**
  - **join**
  - **lower**
  - **upper**
  - **replace**
  - **split**

字符串示例

【期中考试第2题：回文数】

- 回文数是从左到右和从右到左念都一样的数，例如1234321。输入一个数，如果它是回文数则输出True，否则输出False
- 输入限制（不作为测试点）：输入一个非负整数不超过10^20，如果不为0则首位不为0

设计思路

- 先利用list函数将字符串转换为列表
- 再复制该列表的一个副本（为了不影响原列表）
- 使用**reverse方法**将副本中的元素反向存放
- 若原列表与反向存放的列表相等，则打印"True"，否则打印"False"

字符串示例程序

```
#期中考试第2题_回文数_方法1_amj.py
a=input()                          #输入一个非负整数

if 0<int(a)<=10**20:
    list1=list(a)                  #将字符串转换为列表
    copy1=list1[:]                 #复制list1的一个副本，不能用“copy1=list1”，否则下面的语句也将改变list1的值
    copy1.reverse()                #将列表copy1中的元素反向存放
    print("list1=",list1)          #为了观察list1中各元素的值。提交到OJ上时注释掉该语句
    print("copy1=",copy1)          #提交到OJ上时注释掉该语句

    if list1==copy1:
        print("True")
    else:
        print("False")
```

```
>>>
123421
list1= ['1', '2', '3', '4', '2', '1']
copy1= ['1', '2', '4', '3', '2', '1']
False
>>> ================================ RESTART =
>>>
12321
list1= ['1', '2', '3', '2', '1']
copy1= ['1', '2', '3', '2', '1']
True
>>>
```

方法3：依次判断第i个和倒数第i个元素是否相等。用到负数索引，序列从右往左数，最右边的索引为-1，然后依次是-2、-3……。

（三）线性数据结构

1、栈

- **栈**（Stack）是元素的有序集合，是限定在表尾进行添加或者移除元素操作的线性表
- 可以采用**列表**来存储栈的各元素
  - **压栈（入栈）**：利用列表的append方法，每次在列表末尾增加一个新的元素
  - **弹栈（出栈）**：利用列表的pop方法，移除列表中最后那个元素

【例】利用栈的思想实现将十进制整数转换为二进制数。
参见教材

栈的应用示例

【期中考试第3题：车站】

P 城有一个火车站，每辆火车都从 X 方向驶入车站，再从 Y 方向驶出车站。

为了调度火车，火车站设有停放轨道，可存放 5 辆火车。已知从 X 方向进入车站编号顺序为 1、2、3……。现在给你一个调度方案，判断是否可行。如果可行，输出出站顺序。

有以下几种调度方法：

A： 将 X 方向的头一辆车驶出 Y 方向；

B： 将 X 方向上的头一辆车停入暂停轨道；

C：将暂停轨道上最外面的车驶出 Y 方向。

- **设计思路**
  - 采用栈来作为临时停放轨道。采用一个列表final存储车辆依次出站的编号

（1）首先采用for循环，决定三种调度方法对当前车辆应进行的操作。关键是期间采用了标志位flag标识循环是否正常结束：

0表示正常结束，说明该调度方案可行；1表示中断循环，说明该调度方案不可行。

（2）当flag=0时，输出"Yes"，并顺序打印车辆依次出站的编号。

栈的应用示例程序（1/2）

```
#期中考试第3题_车站_amj.py
from Stack_def import Stack
s = Stack()                    #创建栈的实例，作为临时停放轨道

n = int(input())               #输入一个整数，表示调度方案数目
seq = input()                  #输入操作序列seq

final = []                     #存储车辆依次出站的编号
curr = 1                       #X方向当前车辆编号，初值为1
flag = 0                       #标识循环是否正常结束，0表示正常结束；1表示中断循环

#（1）决定三种调度方法对当前车辆应进行的操作
for i in range(n):
    if seq[i] == 'A':
        final.append(curr)     #将X方向的头一辆车驶出Y方向
        curr += 1

    elif seq[i] == 'B':
        if (s.size() >= 5):    #判断是否轨道已满【有同学没有判断，则程序运行出错】
            print("No")        #若已满，则该方案不可行，不可能有车驶入暂停轨道
            flag = 1
            break
        else:
            s.push(curr)       #若未满，则当前车辆入栈
            curr += 1
```

栈的应用示例程序（2/2）

```
 elif seq[i] == 'C':
        if (s.isEmpty()):       #判断是否临时停放轨道已空【有同学没有判断，则程序运行出错】
            print("No")         #若为空，则该方案不可行，不可能有车驶出暂停轨道
            flag = 1
            break
        else:
            final.append(s.pop())#若不为空，则将暂停轨道上最外面的车辆驶出Y方向

#（2）当flag=0时，输出"Yes"，并顺序打印车辆依次出站的编号
if(flag == 0):                  #若循环正常结束
    print("Yes")
    for i in final:             #则顺序打印车辆依次出站的编号
        print(i)
```

```
>>>
6
ABBCCA
Yes
1
3
2
4
>>> ============
>>>
5
BACAC
No
>>>
```

2、队列

- **队列**（Queue）是元素的有序集合，向队列的一端（队尾rear）添加新的元素，而在另一端（队头front）移除现有元素
- 采用列表来存储队列的各元素，按照人们的习惯，列表的位置0存放队首元素，末尾存放队尾元素
  - **入队**：利用列表的append方法，每次向队尾添加新的元素
  - **出队**：利用列表的pop(0)方法，每次从列表的位置0（即队首）移除元素

【例】利用队列的思想实现游戏“烫手的山芋”。
参见教材

（四）典型算法

1、枚举算法

- **枚举算法（穷举算法，Enumeration）**就是按问题本身的性质，通过多重循环——列举出该问题所有可能的解（不能遗漏，也不能重复），并在逐一列举的过程中，检验每个可能的解是否是问题的真正解，若是，我们采用这个解，否则抛弃它
- **枚举算法设计技巧**

① 有几个求解变量就对应几个for循环

② 根据求解变量特性确定枚举范围（整型、取值大小）

【例】采用枚举算法求解阿姆斯特朗数。

参见教材

2、查找算法

- 从给定的集合中查找一个给定的值，称为**查找**，也称为**搜索**
- 常用的查找算法
  - **线性查找**——直接从头到尾搜索集合的查找键，直到找出与查找键相同的数据为止
  - **二分查找**——必须首先将集合按照降序或升序排序，然后利用折半技术搜索集合的查找键
    - 当集合是有序的时候，使用折半查找效率高、速度快
  - **哈希查找**——以线性表中每个元素的关键字key为自变量，计算哈希函数值，以该值作为地址，将元素存入哈希表中；查找某元素时，先计算其哈希函数值，再在哈希表中查找该地址对应的存储单元

掌握一种即可

3、排序算法（1/2）

3、排序算法

- **掌握一种即可**
- **如果题目要求实现排序，则不允许使用sort方法！**

- **排序（Sort）**：将一组原始数据按照某种规则进行有序排列的过程
- **常用的排序算法**
  - **选择排序——**从最左边的一个元素开始处理，每次从待排序的数据元素中**选出最小**（或最大）的一个元素，存放在**待排序**序列的**起始**位置；直到全部待排序的数据元素排完序
  - **插入排序——**把数组A中的n个元素看做由一个有序空间和一个无序空间组成，每次从无序空间中取出第一个元素，将其插入到有序空间中的合适位置，使之成为一个新的有序空间；经过n-1次插入后，有序空间包含了所有n个元素，而无序空间变为空，排序完毕

3、排序算法（2/2）

- **冒泡排序——**在每一轮次中依次对待排序数组元素中**相邻**的两个元素进行**比较**和**交换**，若是递减排序，则将大的放前，小的放后。当第一轮比较结束时，数组中最小的元素就会被提升到这组元素的尾部；对剩余的n-1个待排序元素执行此过程，直到全部排好序
- **归并排序——**将大的数据集切分为N个子集合，对每一个子集合进行排序；再将（排好序的）“小”数据集合并，并使合并后的数据集仍然保持有序，即可实现对整个数据集的排序

四、期末备考方法

- **根据今天讲座，围绕重点复习内容，逐个知识点认真复习**
  - 弄清楚语法，需要传递的参数，用法等
  - 读懂教材和课件上相关案例，并会自己独立编写程序
- **把期中考试试题自己做一遍**
- **把与重点复习内容相关的平时实验自己做一遍**
- **总结编程技巧，打印出来（今天的PPT也可以打印出来）**
- **熟记各重点复习内容在教材上的章节，以便于考试时迅速查阅**

考试攻略（1）

千万勿忘！

- **必带资料和用具：教材，你整理打印的资料，笔**
- **按要求提前20分钟达到考场，确保计算机能够登录OJ系统**
- **沉着冷静，先把5道题快速浏览一遍，大致了解题目**
  - 题目一般是按照先简单后难的顺序编排
- **总的答题策略**
  - 快速完成第1题和第2题，尽快做第3题和第4题，第5题比较综合，最后做

考试攻略（2）

- **对每道题认真审题和构思**
  - 编程前先**仔细审题**，弄清楚输入是什么（有什么限定范围），要做什么，输出是什么
  - 在脑子里构思，采用什么**程序控制结构**（if，for或while），采用什么**数据结构**（列表，字符串还是栈，队列？），采用什么**算法**（枚举算法或排序算法）
  - 对于比较复杂的题，可以先在纸上写下大致思路

考试攻略（3）

- **编程和调试**
  - **变量命名**：变量名尽量**简洁**且**有意义**，不要用a、b、c、d或x、y、z等作为变量名
  - 循环变量一般用i、j、k、m等表示。如果有多个循环结构，则每个循环结构最好用不同的循环变量，以便查错
  - 每道题**添加必要的注释**（对变量、算法、数据结构等）
  - **添加必要的空行**，使程序清晰
  - **调试时尽量使用print函数打印出中间过程（甚至包括循环变量的值），以便分析计算过程是否正确，有利于查错！**

考试攻略（4）

- **如果程序出错，不要慌，仔细阅读出错信息**
  - ✓ 如果提示**语法错误**，很可能是使用了中文标点，或是多了一个或多个括号，或是少了一个或多个括号，或是少了一个冒号（定义函数或if语句或循环语句的后面）。**光标所在位置不一定就是出错的地方！**
  - ✓ 如果提示变量**超出范围**，很可能是for循环的范围超出了，将上限减一试试
  - ✓ 如果提示**不能进行算术运算**，很可能输入的是字符串，没有转换为int或float

考试攻略（5）

OJ评测

- 先在Python的集成开发环境IDLE中运行程序；通过后，再提交到OJ
- 如果没有通过，根据评测结果（ WA 、CE、 TLE ）判断是什么问题
  - ✓ 首先输入的内容和格式（包括大小写、标点等）必须与“输入样例”完全一样！否则测试结果为0分， **WA**（Wrong Answer，答案错误)
  - ✓ 输出结果的内容和格式也必须与“输出样例”完全一样！否则测试也会得0分；如果不一样，则修改源程序
  - ✓ **CE**（Compile Error，编译错误）可能是程序语法有错
  - ✓ **TLE**（Time Limit Exceed，超时）可能程序太复杂，算法复杂度过高
- 程序不要轻易改动，先**保留一个在IDLE中通过的版本**（文件名中加上标注）；再另起名保存，再继续修改

OJ评测结果含义

- "AC": "Accepted“
- "WA": "Wrong Answer“
- "CE": "Compile Error“
- "RE": "Runtime Error“
- "REG": "Runtime Error (SIGSEGV)“
- "REP": "Runtime Error (SIGFPE)“
- "WT": "Waiting“
- "JG": "Running“

- "TLE": "Time Limit Exceed“
- "MLE": "Memory Limit Exceed“
- "PE": "Presentation Error“
- "ERR": "Judge Error“
- "IFNR": "Input File Not Ready“
- "OFNR": "Output File Not Ready“
- "EFNR": "Error File Not Ready“
- "OE": "Other Error"


祝同学们期末考试
取得优异成绩！