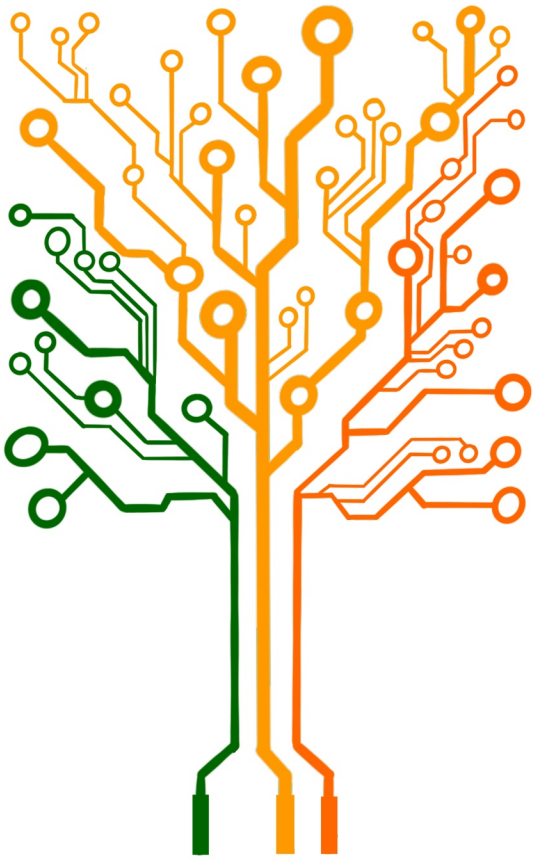
张续飞

熟练掌握 Python 编程

小学数学编程学习手册



高效学习小学数学

包含人教版【小学数学】教材三至五年级

全部 49 个单元 | 共 60 个 Python 程序

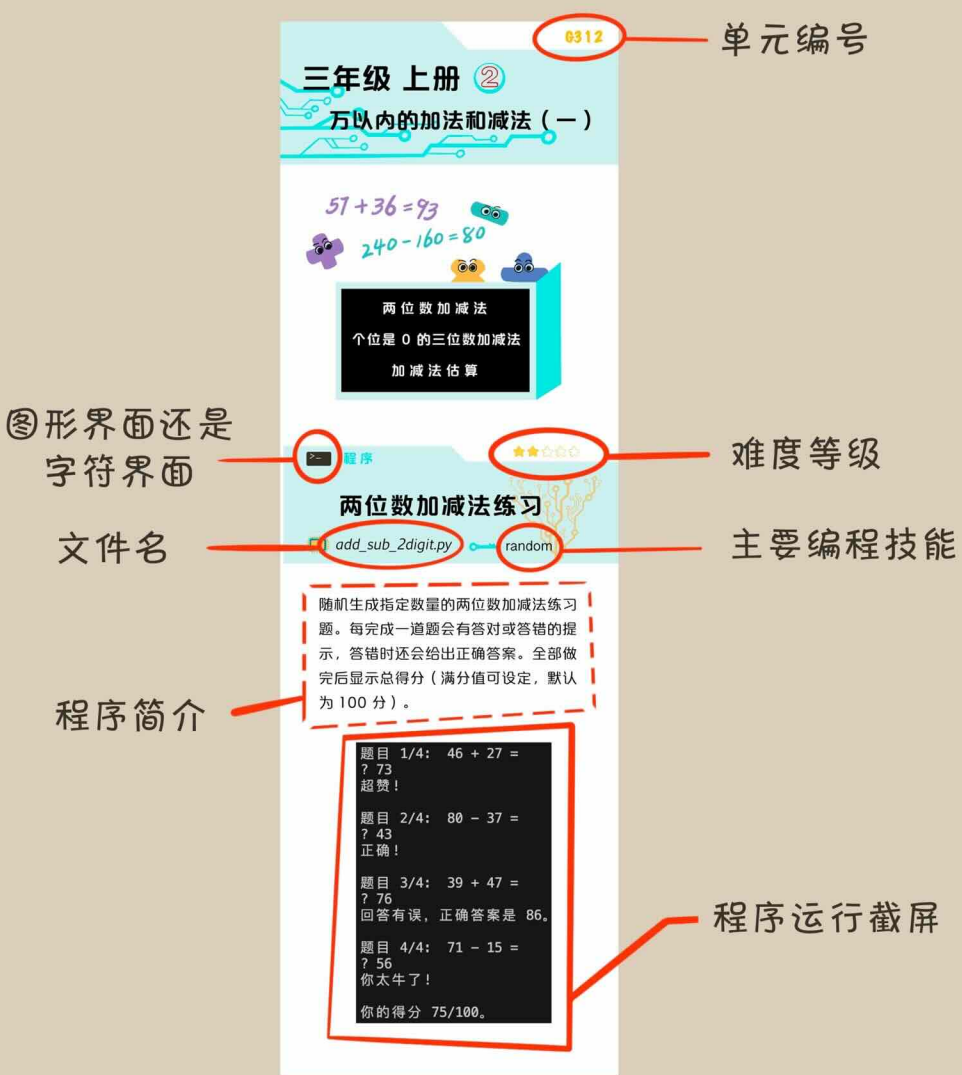
手册简介和使用指南

《小学数学编程学习手册》紧密配合人教版小学数学教材，包含小学三至五年级全部 49 个单元，共 60 个 Python 程序（22 个图形界面，38 个字符界面）。“数学+编程” 使学习者在高效学习数学的同时，还能熟练掌握一门编程语言。

小学数学编程项目既适合在学校进度下想同时学好数学和编程的学习者，也适合想按照个人进度加速完成小学数学的学习者。利用项目中的学习方法，笔者高中时曾带 6 岁的弟弟用一年时间完成了小学三至五年级的数学学习，关于项目的的详细介绍请参见项目网站 https://feli10.github.io/math-coding。

代码下载及使用说明请参见 GitHub 项目代码库 https://github.com/feli10/math-coding。

手册中每个单元的页面结构如下图所示：



单元编号

手册的编排顺序完全按照人教版小学数学教材，每个单元都有一个以 “G” 开头的三位数字编号，例如 G311。编号中各位数字的含义如下：

- 开头的 “G” 和第一位数字合起来代表年级，“G3” 为三年级 (Grade 3)；
- 第二位数字可能是 “1” 或 “2”。“1” 代表第一学期或数学教材的上册，“2” 代表第二学期或数学教材的下册；
- 第三位数字为第几单元。

所以 “G311” 表示三年级上册第一单元。

程序信息

每个单元有 1-2 个 Python 程序，用于解决与本单元数学内容紧密联系的编程问题。在每个程序介绍中都包含以下内容：

- 文件名
- 图形界面还是字符界面
- 难度等级（一至五颗星）
- 程序中用到的编程技能
- 程序简介
- 程序运行截屏

目录

三年级上

三年级下

目录

四年级上

四年级下

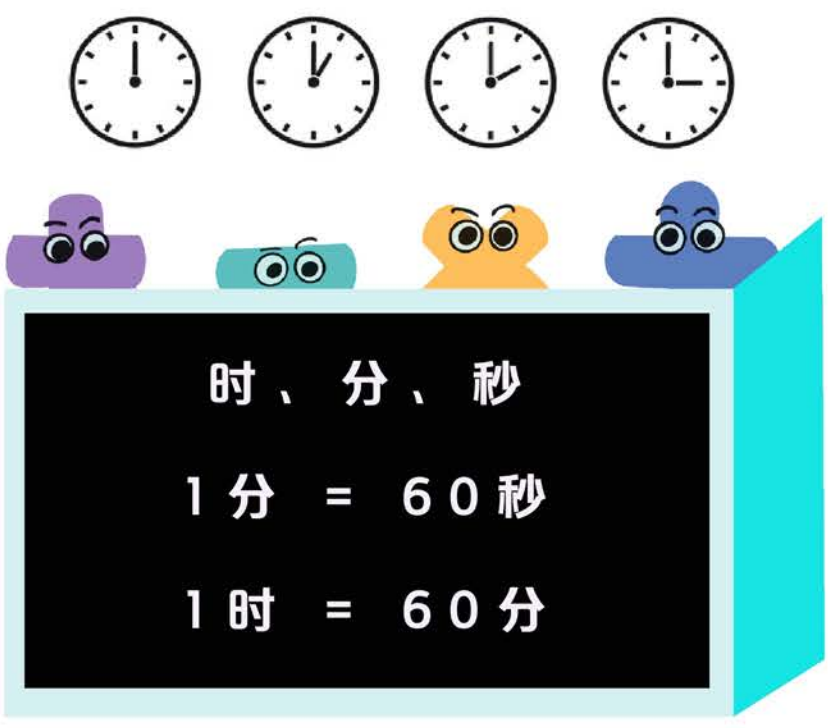
目录

五年级上

五年级下

三年级 上册 ①

时、分、秒



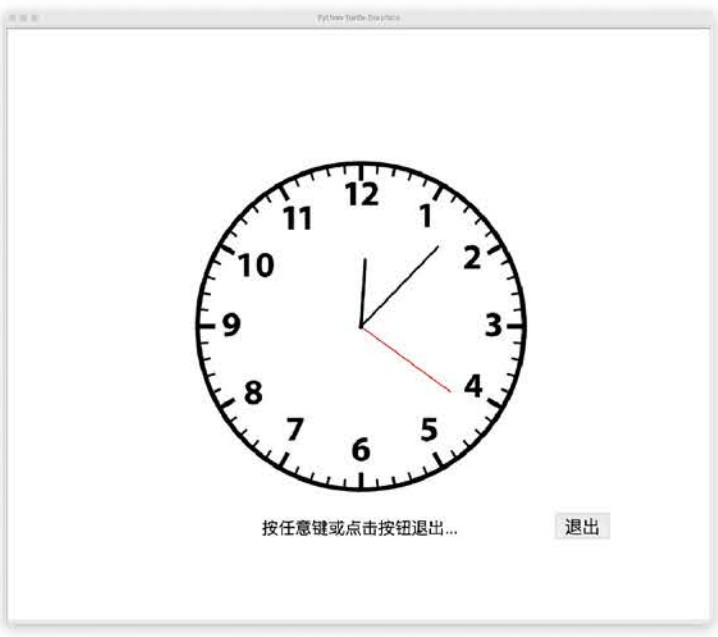
程序一

表盘时钟

clock.py turtle, tkinter, exception

第一个程序模拟了一个带有指针和表盘的时钟，随着时间的推移，秒针、分针和时针做出相应的转动。程序中使用了 turtle 和 tkinter module，在图形界面中实现时钟的动画。

程序运行后，可以按任意键或点击界面上的退出按钮结束程序。程序默认按照实际的时间变化准确计时，可以调整参数，使时钟变快或变慢。



程序二

数字时钟计时器

digital_clock.py exception

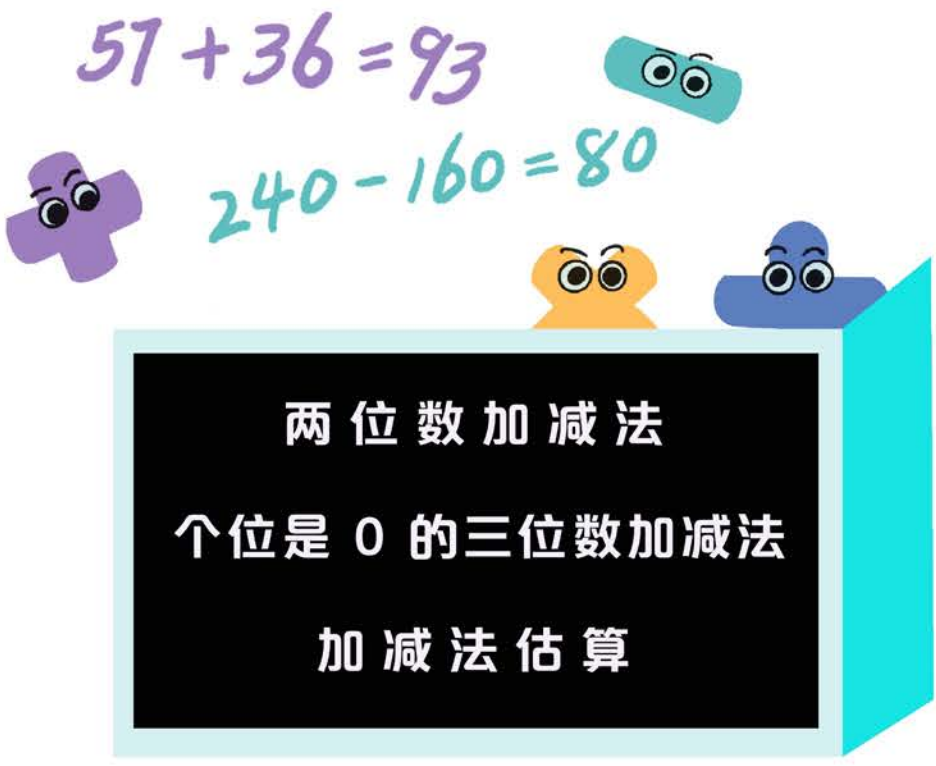
第二个程序在命令行界面中显示如 00:00:00 格式的数字时钟，随着时间的推移，时、分、秒各数字位进行相应的变化。

程序开始运行时，会先让用户输入计时器的时间长度（以秒为单位），程序会在计时器到期时结束，或在计时器未到期时，按 Ctrl-C 终止程序。程序默认按照实际的时间变化准确计时，为了能够快速考察分钟和小时的数字变化，可以调整参数，大幅加速时间变化。

```
请输入计时器时间长度（单位秒）：80
00:01:20
计时器时间到！
```

三年级 上册 ②

万以内的加法和减法（一）



程序

★★☆☆☆

两位数加减法练习

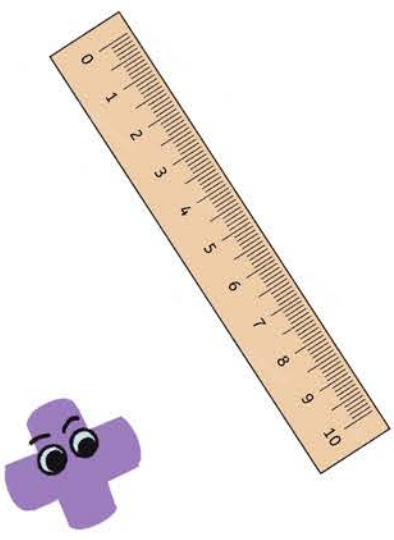
add_sub_2digit.py　　random

随机生成指定数量的两位数加减法练习题。每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。

```
题目 1/4:  46 + 27 =
? 73
超赞！

题目 2/4:  80 - 37 =
? 43
正确！

题目 3/4:  39 + 47 =
? 76
回答有误，正确答案是 86。

题目 4/4:  71 - 15 =
? 56
你太牛了！

你的得分 75/100。
```

三年级 上册 ③

测量



长度单位：毫米、分米、千米

（厘米和米在二年级已经学过了）

1 厘米 = 10 毫米

1 分米 = 10 厘米

1 米 = 10 分米

1 千米 = 1000 米

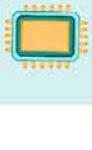
质量单位：吨

（克和千克在二年级已经学过了）

1 吨 = 1000 千克

程序

单位换算练习



unit_conversion.py

random

随机生成指定数量的长度和质量单位换算练习题。由于尚未学习分数和小数，题目全部是由较大单位向较小单位进行换算，例如 1dm = __cm，而不会出现 1cm = __dm。每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。

```
题目 1/4:  1t = __kg
? 10
回答有误，正确答案是 1000。

题目 2/4:  1dm = __cm
? 10
不错！

题目 3/4:  1kg = __g
? 1000
超强！

题目 4/4:  1dm = __mm
? 100
你真牛！

你的得分 75/100。
```

三年级 上册 ④

万以内的加法和减法（二）

程序一

加法竖式

add_vertical.py　string

程序会将用户输入的两个自然数用加法竖式求和，并将结果以竖式的形式显示在屏幕上。程序在求和时真实模拟了加法竖式的运算过程，而不是使用编程语言内置的"+"运算符直接得到结果。这样可以使学习者在程序编制过程中，加强对加法竖式运算过程的理解和掌握。

```
输入第一个自然数：298
输入第二个自然数：745
   2 9 8
+  7 4 5
---------
  1 0 4 3
```

程序二

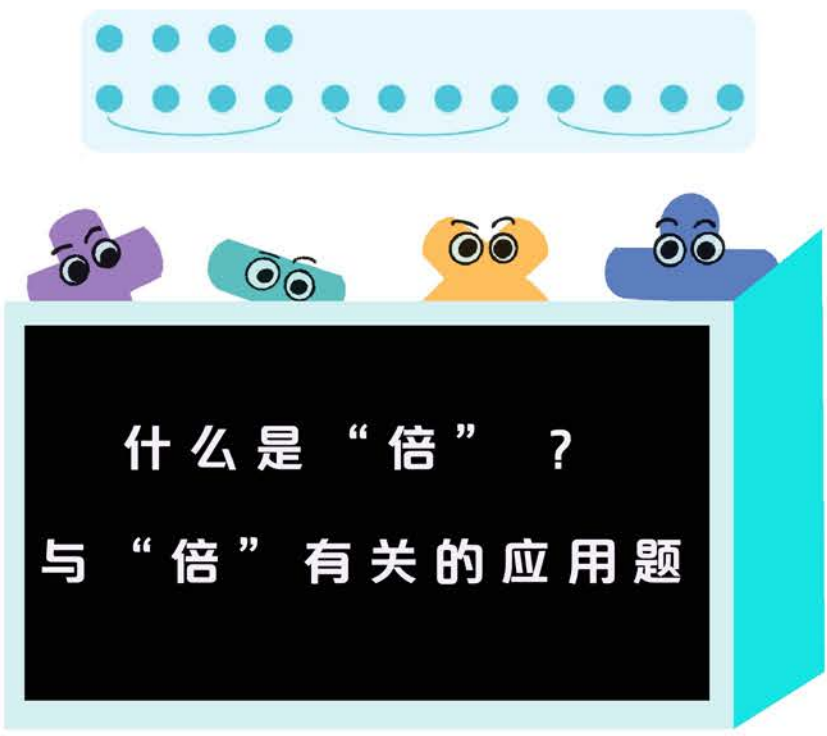
减法竖式

sub_vertical.py　string

程序会将用户输入的两个自然数用减法竖式求差（大数减小数），并将结果以竖式的形式显示在屏幕上。程序在求差时真实模拟了减法竖式的运算过程，而不是使用编程语言内置的"-"运算符直接得到结果。这样可以使学习者在程序编制过程中，加强对减法竖式运算过程的理解和掌握。

```
输入第一个自然数：435
输入第二个自然数：86
  4 3 5
-   8 6
-------
  3 4 9
```

三年级 上册 ⑤ 倍的认识



程序

与“倍”有关的文字题

a_times_b.py random

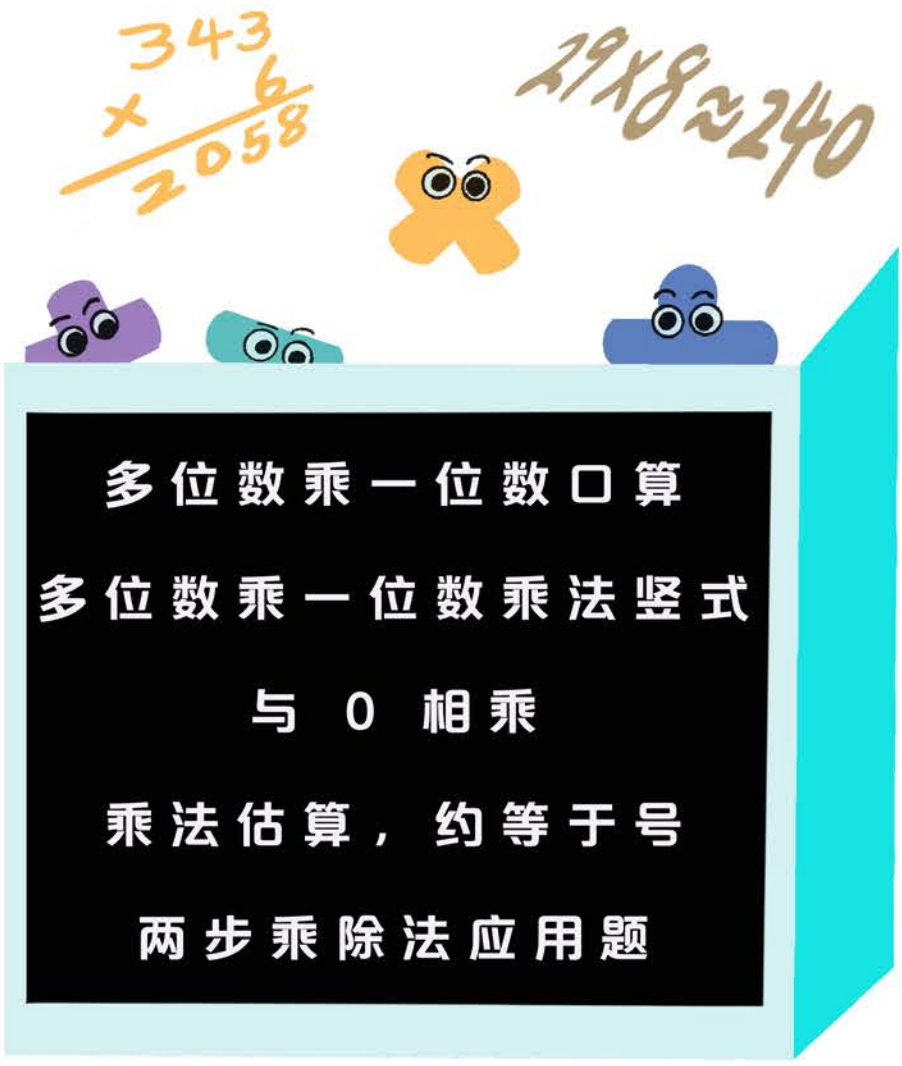
随机生成指定数量的与“倍”有关的文字题。题目包含三种简单类型，各举例如下：

- 4 的 3 倍是几？
- 12 是 4 的几倍？
- 一个数的 3 倍是 12，这个数是几？

每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。

```
题目 1/4:  4 的 7 倍是几？
? 28
优秀！

题目 2/4:  45 是 9 的几倍？
? 5
你答对了！

题目 3/4:  5 的 7 倍是几？
? 35
太神了！

题目 4/4:  一个数的 2 倍是 6，这个数是几？
? 12
回答有误，正确答案是 3。

你的得分 75/100。
```

三年级 上册 ⑥
多位数乘一位数



程序

多位数乘一位数乘法竖式

short_multiplication.py　　string

程序会将用户输入的一个多位自然数和一个一位自然数用乘法竖式求积，并将结果以竖式的形式显示在屏幕上。程序在求积时真实模拟了乘法竖式的运算过程，而不是使用编程语言内置的"*"运算符直接得到结果。这样可以使学习者在程序编制过程中，加强对乘法竖式运算过程的理解和掌握。

```
输入第一个自然数：343
输入第二个自然数（一位数）：6
    3 4 3
x       6
-----------
  2 0 5 8
```

三年级 上册 ⑦
长方形和正方形

四边形

长方形和正方形

长方形和正方形的周长

长方形的周长 = (长 + 宽) × 2

正方形的周长 = 边长 × 4

程序

创建长方形类

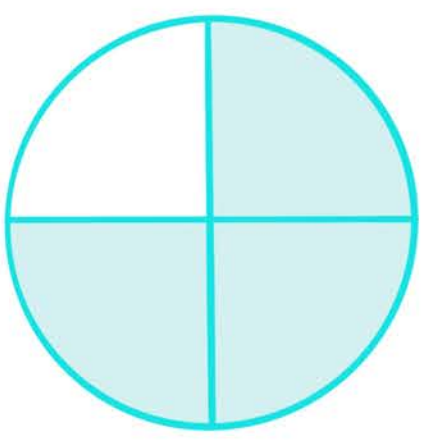
rectangle.py　class

程序创建了一个长方形类，实例化一个长方形对象后，可以查看它的长和宽、计算周长、判断是否是正方形，还可以通过 print() 函数把长方形对象的相关信息都显示在屏幕上。

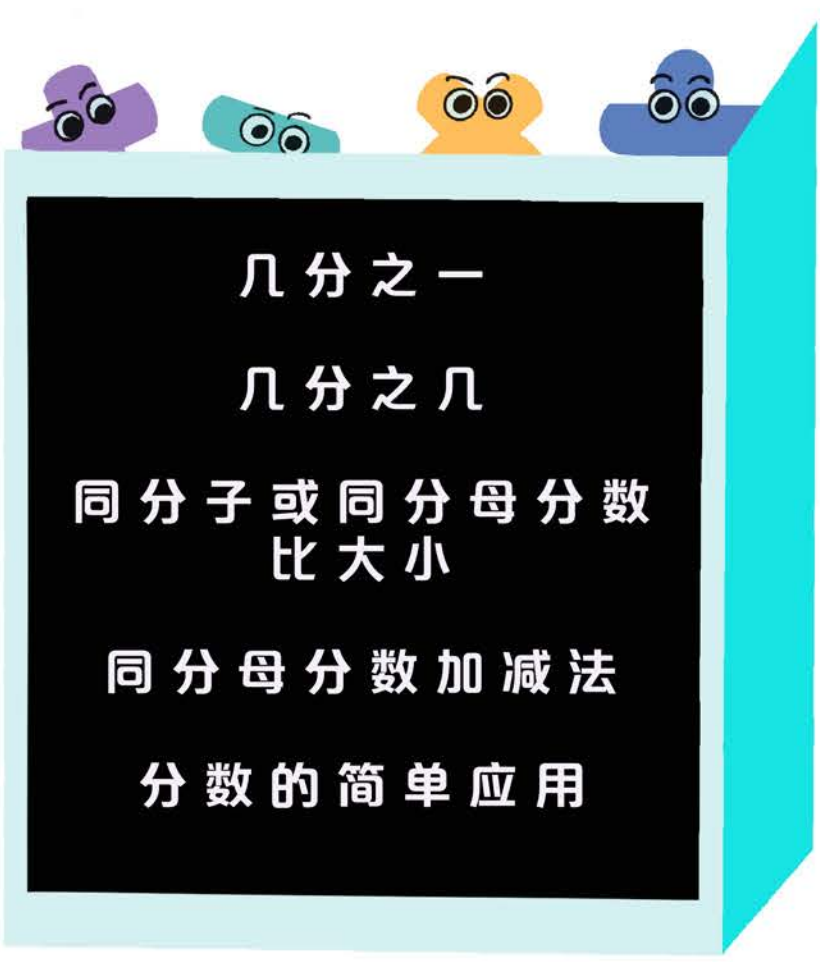
程序逻辑并不复杂，主要目的是让学习者以长方形的长、宽、周长等简单知识点为例，学习类 (class) 和对象 (object) 的概念，以及初步体会面向对象的程序设计 (OOP, Object-Oriented Programming)。

```
>>> rect1 = Rectangle(5, 2)
>>> print(rect1)
长方形
长: 5
宽: 2
周长: 14

>>> rect2 = Rectangle(3)
>>> print(rect2)
正方形
边长: 3
周长: 12

>>> rect2.length = 4
>>> print(rect2)
长方形
长: 4
宽: 3
周长: 14

>>>
```

三年级 上册 ⑧
分数的初步认识



$$1-\frac{1}{4}=\frac{3}{4}$$



程序

分数比大小练习

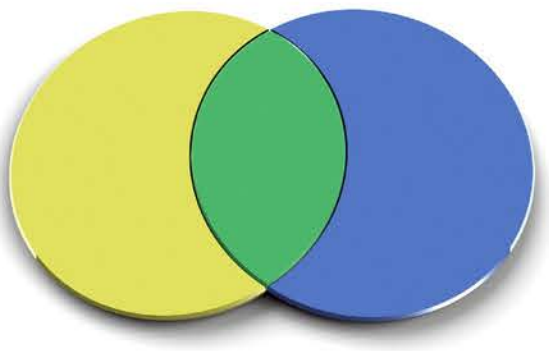
compare_fractions.py　　random

随机生成指定数量的同分子或同分母分数比大小练习题。每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。

```
题目 1/4:  1/3 __ 2/3
(> 或 <) ? <
牛！

题目 2/4:  1/5 __ 1/7
(> 或 <) ? >
棒！

题目 3/4:  3/5 __ 1/5
(> 或 <) ? >
超强！

题目 4/4:  3/10 __ 3/9
(> 或 <) ? >
回答有误，正确答案是 <。

你的得分 75/100。
```

三年级 上册 ⑨

数学广角——集合



集合

两个集合的公共部分
（交集）

两个集合合并在一起的总体
（并集）

程序

集合运算

sets.py　　set, random

程序使用两种方法获取两个集合的交集和并集：第一种是使用 list 数据类型表示集合，根据定义自行编程得到交集和并集（仍是 list 形式）；第二种是使用 Python 内置的 set 数据类型表示集合，直接使用 set 运算得到交集和并集。

程序的主要目的有两个：其一是通过多个随机生成的例子，让学习者体会两个集合的元素数之和减去公共元素数等于全体元素数；其二是让学习者了解 Python 有一种与 list 类似的数据类型 set，专门用于集合，list 使用 []，set 使用 {}，与 list 不同的是 set 内没有重复的元素且元素是无序的。

```
使用 list 自行编程：
A: [8, 4, 9, 0, 5, 3], 6 个元素。
B: [5, 7, 3, 2, 9, 0, 6], 7 个元素。
公共: [5, 3, 9, 0], 4 个元素。
全体: [8, 4, 9, 0, 5, 3, 7, 2, 6], 9 个元素。
6 + 7 - 4 = 9

使用内置 set 运算：
A: {0, 3, 4, 5, 8, 9}, 6 个元素。
B: {0, 2, 3, 5, 6, 7, 9}, 7 个元素。
公共: {0, 9, 3, 5}, 4 个元素。
全体: {0, 2, 3, 4, 5, 6, 7, 8, 9}, 9 个元素。
6 + 7 - 4 = 9
```

三年级 下册 ①

位置与方向（一）

程序

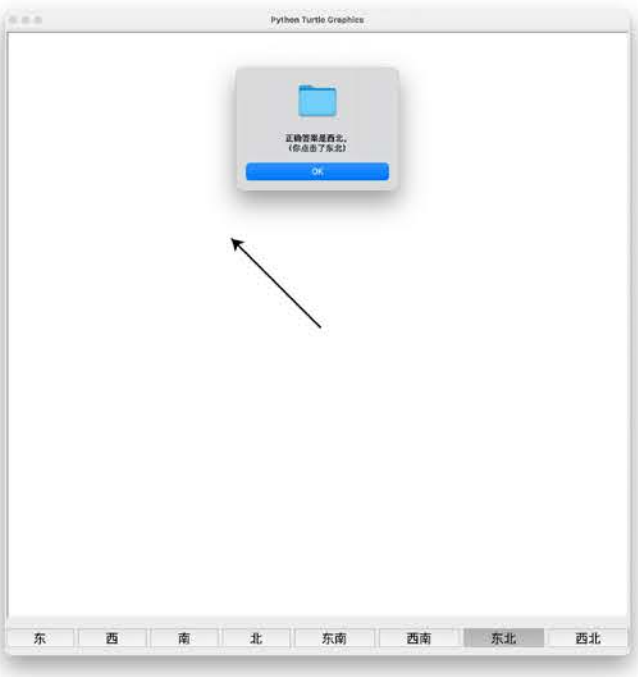
辨认方向练习

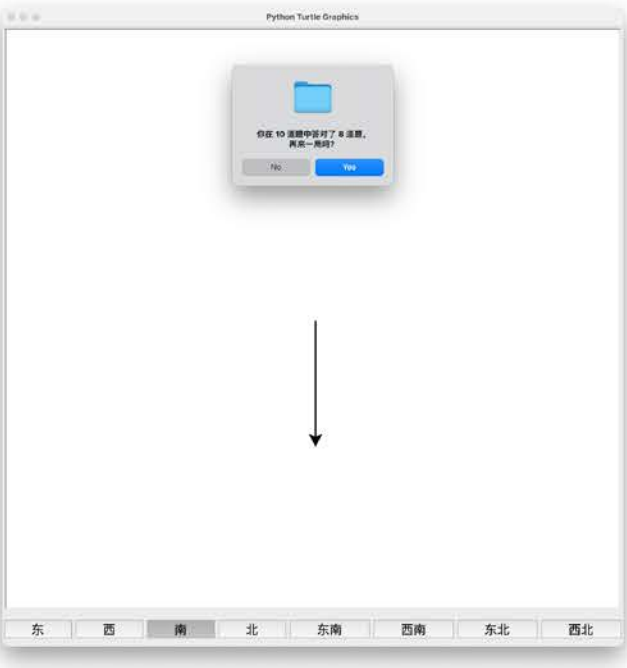
orientation.py tkinter, turtle, threads

第一个程序模随机生成指定数量的辨认方向的练习题。在练习辨认方向时，为了增加一些紧迫感和趣味性，程序为答题添加了时间限制。用户可以根据需要对答题时间进行设置或关闭限时答题功能。

程序运行后，学习者根据屏幕上箭头所示的方向，选择下方东、西、南、北、东南、西南、东北、西北等八个方向按钮，答对会直接进入下一题，答错会弹出消息框显示正确答案。如果长时间未作答超出了设置的答题时间，程序会自动弹出消息框显示正确答案。全部答题完成后，会在消息框中显示答对数目，并询问是否再来一局，选是重新开始，选否退出程序。

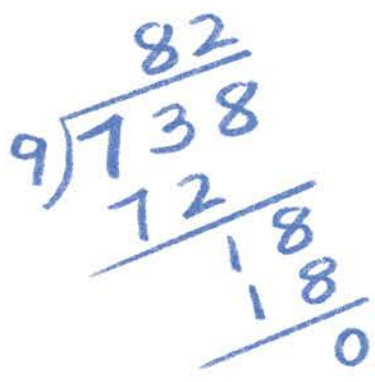
程序的计时功能使用了 tk 的after()，与常见的 sleep() 不同的是 after() 不会阻碍程序主线程的运行。



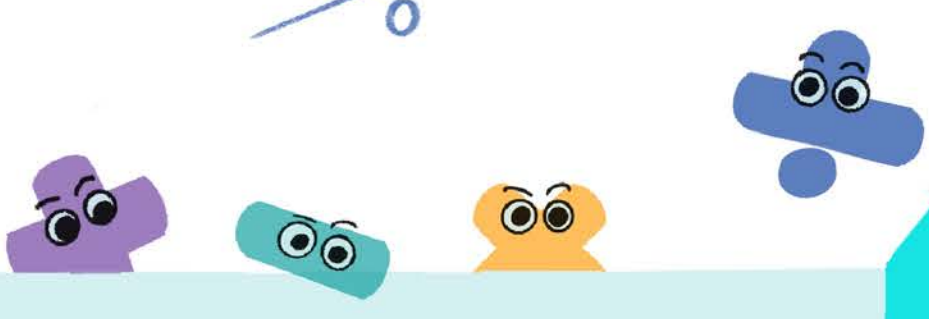


三年级 下册 ②

除数是一位数的除法



$178 \div 6 \approx 30$



除数是一位数的除法口算

除数是一位数的除法竖式

有余数的除法

除法验算

被除数是 0

除数是一位数的除法估算

程序

除数是一位数的除法竖式

short_division.py　　string

程序会将用户输入的一个多位自然数和一个一位非零自然数用除法竖式求商和余数，并将结果以竖式的形式显示在屏幕上。程序在求商时真实模拟了除法竖式的运算过程，而不是使用编程语言内置的 "//" 和 "%" 运算符直接得到商和余数。这样可以使学习者在程序编制过程中，加强对除法竖式运算过程的理解和掌握。

比较复杂的是除法竖式的显示。不像加减法竖式只有三行，除法竖式的行数是不固定的，且每一行的起始位置不断变化。在确定每一行的起始位置时，主要考虑以下两个因素：

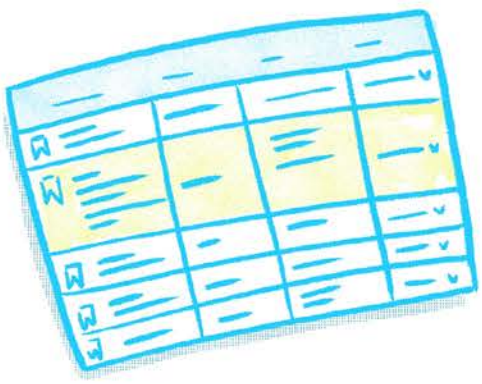
1. 除法竖式中包含若干步“小”除法，每一步的余数也是下一步被除数的最高位部分，所以在一步小除法中被除数和余数的数位差，就是下一步小除法中被除数的缩进量；
2. 如果前一步小除法的余数是 0，则下一步被除数的最高位上至少会有一个 0，在显示竖式时要将开头的 0 去掉，但 0 所占的位置要空出来并补充到缩进量中。

```
输入一个自然数作为被除数：2160
输入一个一位非零自然数作为除数：2

  1080
  ----
2/2160
  2
  ----
   16
   16
  ----
    0

2160 / 2 = 1080 ... 0
```

```
输入一个自然数作为被除数：3501
输入一个一位非零自然数作为除数：5

   700
  ----
5/3501
  35
  ----
     1

3501 / 5 = 700 ... 1
```

三年级 下册 ③

复式统计表



程序

创建表格类及在字符界面显示表格

table.py class, list, string

程序创建了一个表格类，包含行标题、列标题、行数、列数、表格数据等属性，以及数据填充（指定或随机）、数据清除、删除某行或某列等方法。让学习者在熟悉数学统计表的同时，了解编程中类 (class) 和对象 (object) 的概念，以及它们的创建和使用，体会面向对象的程序设计方式 (OOP, Object-Oriented Programming)。

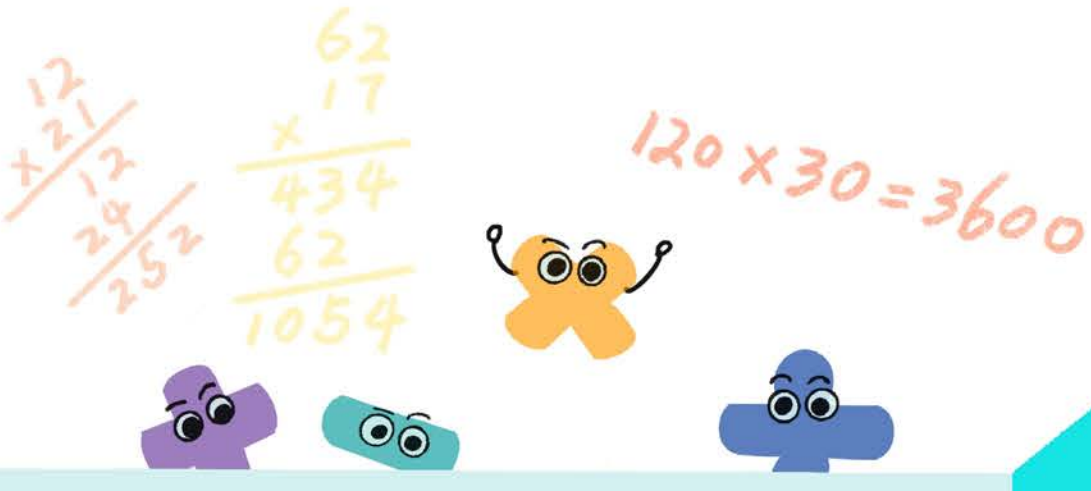
程序的一个主要功能是在字符界面中显示表格，包含行、列分割线。表格内容可以使用中西文字符，单元格的宽度会根据表格内容自动进行调整，行、列标题和数据会居中显示在单元格中。

表格的数据存储在一个二维列表中，所以程序中涉及大量对 list 的操作，特别是多次使用 list comprehension 来创建列表，以使程序更加简洁。

```
>>> vegetables = ['萝卜', '黄瓜', '西红柿', '玉米']
>>> classes = ['一班', '二班', '三班']
>>> table = Table(row_headers=classes, col_headers=vegetables)
>>> table.random()
>>> print(table)

|--------|--------|--------|--------|--------|
|        |  萝卜  |  黄瓜  | 西红柿 |  玉米  |
|--------|--------|--------|--------|--------|
|  一班  |    5   |    7   |   19   |    6   |
|--------|--------|--------|--------|--------|
|  二班  |   11   |    0   |   15   |   10   |
|--------|--------|--------|--------|--------|
|  三班  |   14   |    8   |    3   |   13   |
|--------|--------|--------|--------|--------|

>>> print(table.shape)
3 x 4
>>> table.del_row(1)
>>> table.del_col(2)
>>> print(table)

|------|------|------|------|
|      | 萝卜 | 黄瓜 | 玉米 |
|------|------|------|------|
| 一班 |   5  |   7  |   6  |
|------|------|------|------|
| 三班 |  14  |   8  |  13  |
|------|------|------|------|

>>> print(table.shape)
2 x 3
>>>
```

三年级 下册 ④

两位数乘两位数



因数末尾有 0 的多位数乘法口算

两位数乘两位数乘法竖式

两步乘除法应用题

程序

多位数乘法竖式

long_multiplication1.py　　string

程序会将用户输入的两个任意自然数用乘法竖式求积，并将结果以竖式的形式显示在屏幕上。程序在求积时真实模拟了乘法竖式的运算过程，而不是使用编程语言内置的 "*" 运算符直接得到结果。

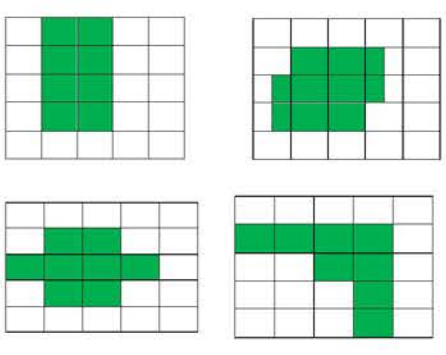
多位数乘法竖式的运算过程可以分为两步，第一步是计算多个多位数乘一位数的乘法，第二步是把第一步得到的多个乘积相加。此程序通过调用之前编写的《多位数乘一位数乘法竖式》(G316) 程序来完成第一步的工作。

long_multipy_core() 函数是模拟多位数乘法竖式运算过程计算两数乘积的代码，把它从主要负责显示乘法竖式的 long_multiply() 函数中分离出来，是为了在未来最终版本的《通用乘法竖式》(G414) 程序中，可以复用乘法竖式运算部分的代码。

```
输入第一个自然数：48
输入第二个自然数：37
    4 8
x   3 7
---------
    3 3 6
  1 4 4
---------
  1 7 7 6
```

三年级 下册 5

面积



面积的概念

面积单位
（平方厘米，平方分米，平方米）

长方形和正方形的面积

长方形的面积 = 长 × 宽

正方形的面积 = 边长 × 边长

面积单位间的进率

1 平方分米 = 100 平方厘米

1 平方米 = 100 平方分米

铺砖问题

程序

完善长方形类 —— 求面积和画长方形

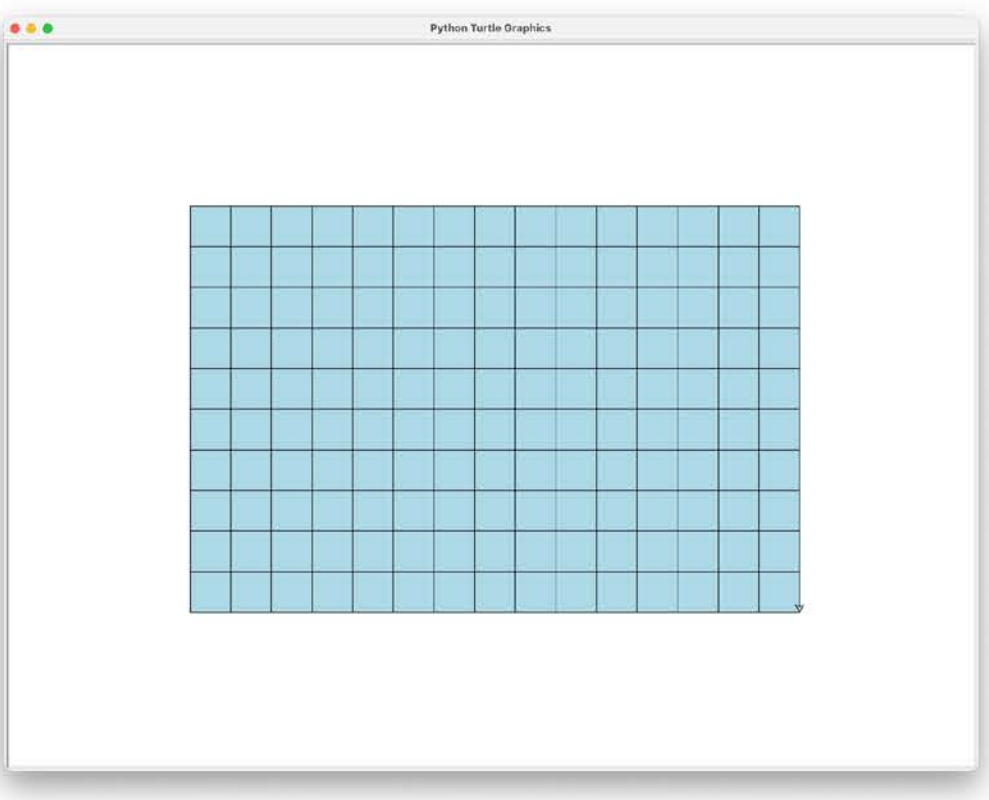
draw_rectangle.py　tkinter, turtle, class, coordinate

程序为之前编写的《创建长方形类》(G317) 增加了求面积和画长方形的方法。为了加强对面积概念的直观印象，画长方形时可以选择在长方形中画出网格，每个正方形小格代表一个单位面积，网格的数量代表了长方形面积的大小。

Turtle 和 Tk 是编写 Python 图形界面程序最常用的两种工具。Turtle 的特点是简单直观，画图过程自带动画，无需了解平面坐标即可开始使用，适合初学者。Tk 的功能更加丰富强大，不止可以画图，还可用于编写图形用户界面下的应用软件。作为对比，程序使用了 Turtle 和 Tk 两种方法画长方形，让学习者体会两种工具的画图风格，未来可以选择适合的工具编写图形界面程序。

本程序需要学习者先简单了解一些坐标知识，需要注意的是：Turlte 的坐标原点 (0, 0) 在屏幕中心（类似 Scratch）；Tk 的坐标原点 (0, 0) 在屏幕的左上角（向下为正方向）。另外，在计算机中一般用“宽”和“高”表示屏幕的横向和纵向尺寸，为了使长方形类的属性名与之保持一致，将之前的 length 和 width 属性，改为 width 和 height。

```
>>> rect = Rectangle(30)
>>> print(rect)
正方形
边长: 10
周长: 40
面积: 100

>>> rect.width = 40
>>> print(rect)
长方形
宽: 15
高: 10
周长: 50
面积: 150

>>> rect.draw(fill='lightblue', grid=True)
```



三年级 下册 ⑥

年、月、日

每月天数

闰年

24 时计时法

时间间隔

程序

显示某年某月的日历

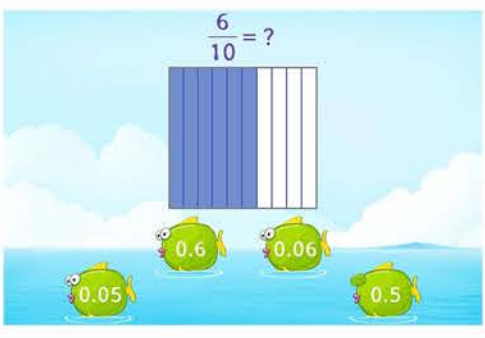
calendar.py　　string

用户输入年份和月份，程序将该月的日历显示在屏幕上。

```
输入年份 (1 - 9999): 2000
输入月份 (1 - 12): 2

   S   M   T   W   T   F   S
           1   2   3   4   5
   6   7   8   9  10  11  12
  13  14  15  16  17  18  19
  20  21  22  23  24  25  26
  27  28  29
```

三年级 下册 ⑦

小数的初步认识



认识小数

小数和分数的关系

简单的小数加减法

程序一

四类小数练习题

decimal_practice1.py　　random

随机生成指定数量的小数练习题。每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。小数练习题共有四种类型，可以一次只练习一种类型，或选择多种类型混合出题。

1. 分数转化为小数。分数的分母可能是 10，100 或 1000，分母越大出现的可能性越低。
2. 小数转化为分数。小数是 0 到 1 之间的 1-3 位小数，位数越大出现的可能性越低。因尚未学习约分，无需化为最简分数，只要分数和小数的大小相等就算回答正确。以 0.8 为例，转化为 “8/10” 即可。
3. 双向单位换算。之前在编写单位换算练习 (G313) 的时候，由于尚未学习分数和小数，题目全部是由较大单位向较小单位进行换算。此程序中的题目为双向单位换算，在由较小单位向较大单位换算时，可以输入小数或分数，例如 1 cm = 0.01 m 或 1 cm = 1/100 m。另外，此程序还在长度单位和质量单位的基础上，加入了货币单位和面积单位。
4. 简单小数加减法。题目为十以内的一位小数加减法口算练习，运算数有一定可能是整数，用以练习整数和小数的混合运算。

通过以上练习，可以体会分数和小数的内在联系，加深对小数的理解，开始从整数加减法向小数加减法过渡。

```
题目 1/4:  3/10
对应的小数？0.3
回答正确！

题目 2/4:  0.48
对应的分数？48/100
真棒！

题目 3/4:  8 - 3.9 =
？4.1
对了！

题目 4/4:  1cm² = __dm²
？0.1
回答有误，正确答案是 0.01 或 1/100。

你的得分 75/100。
```

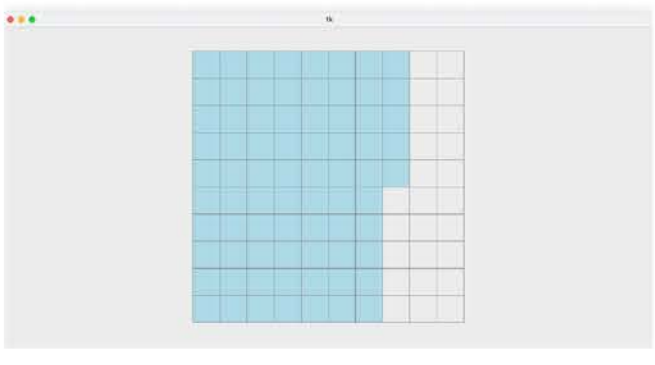
程序二

小数的形象化表示

decimal_representation.py　　tkinter, coordinate

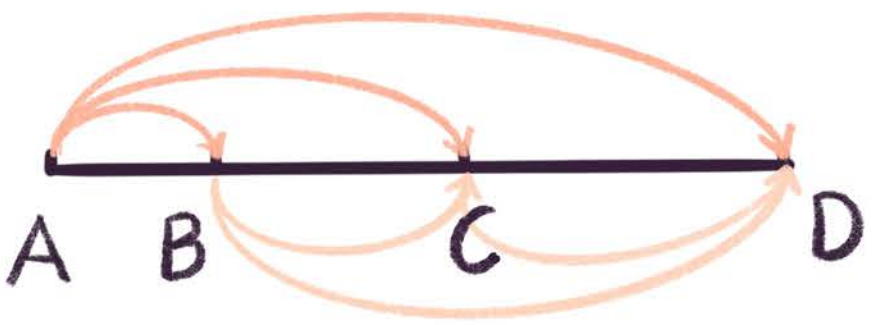
用户输入一个 0 到 1 之间的小数或分数，用这个数代表整个正方形的一部分。程序在一个带有网格的正方形中把与这个数对应的部分涂上颜色。把小数对应为更加直观的图形，可以加深对小数的意义以及小数与分数关系的理解。

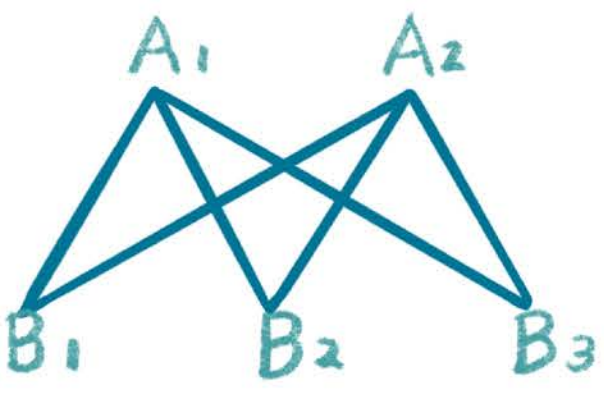
```
输入一个 0 到 1 之间的小数或分数：0.75
```



三年级 下册 ⑧

数学广角— 搭配（二）





计数原理

从两组中各选一个进行搭配

从一组中选若干个有顺序

从一组中选若干个无顺序

程序

三类常见计数问题

combination.py　itertools, nested loop

程序给出了用多重循环解决三类常见计数问题的方法。

- 第一类：从两组中各选一个进行搭配，例如上衣和裤子、荤菜和素菜，这属于搭配问题（在数学中也被称作笛卡尔积）；
- 第二类：在同一组中选择多个元素，且选择方案与选择的顺序有关，例如先选 A 后选 B 和先选 B 后选 A 算两种方案，这是排列问题；
- 第三类：在同一组中选择多个元素，且选择方案与选择的顺序无关，例如无论怎样选，A 和 B 都只算一种方案。这是组合问题。

在各类计数问题中，不重不漏地列出所有方案是一项重要能力，这正是计算机程序的优势，使用多重循环可按给定步骤，机械地把所有方案罗列出来。程序优先选择位置靠前的元素，并按第一选择的不同把所有方案分行显示，便于学习者从中发现规律。

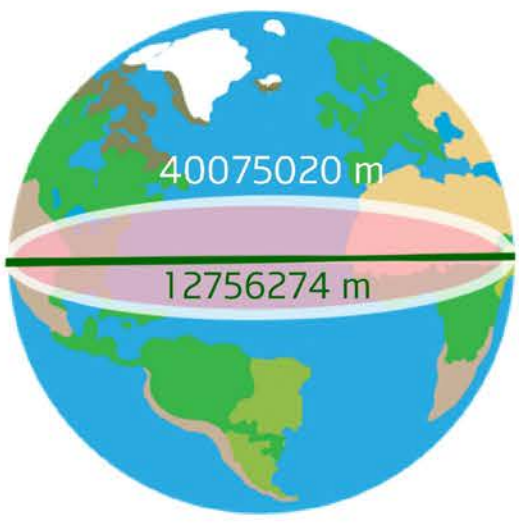
程序中的三类问题都是选择两个元素，所以都使用二重循环，只是各自使用了略有差别的二重循环。如果选择三个元素，则需使用三重循环。当选择元素较多时，多重循环就不是一个好的解决方案了，实际中常用递归。但在选择元素不多的情况下，多重循环是最能帮助学习者体会和掌握课程内容的实现方法。

Python 的标准库中提供了 itertools 模块，其中有与计数相关的多个函数，在今后需要时可以直接调用。程序最后把自行编写的函数与 itertools 相应函数的运行结果做了比较，二者是一致的。

```
搭配问题
A_:  Aa  Ab  Ac  Ad
B_:  Ba  Bb  Bc  Bd
C_:  Ca  Cb  Cc  Cd
D_:  Da  Db  Dc  Dd
从第一组（4 个元素）和第二组（4 个元素）中各选择一个元素，共有 16 种选法。

排列问题
A_:  AB  AC  AD
B_:  BA  BC  BD
C_:  CA  CB  CD
D_:  DA  DB  DC
从第一组（4 个元素）中选择 2 个元素（与顺序有关），共有 12 种选法。

组合问题
A_:  AB  AC  AD
B_:  BC  BD
C_:  CD
D_:
从第一组（4 个元素）中选择 2 个元素（与顺序无关），共有 6 种选法。

>>> print(my_product_result == list(itertools.product(GROUP1, GROUP2)))
True
>>> print(my_permutation_result == list(itertools.permutations(GROUP1, 2)))
True
>>> print(my_combination_result == list(itertools.combinations(GROUP1, 2)))
True
>>>
```

四年级 上册 ①

大数的认识



计数单位、数位

数级（个级、万级、亿级）

读、写大数

大数比大小

用“万”和“亿”做单位的数

四舍五入

自然数、十进制计数法

计算器的使用

程序

读出任意自然数

read_number.py　　string

用户输入任意自然数，按照如下教材中的读数规则，把读出来的汉字显示在屏幕上。为了方便用户数位数，输入时可以使用逗号或空格把大数进行分段。

- 先分级，从最高级读起；
- 每一级都按照个级数的读法来读，读完亿级或万级的数，要加“亿”字或“万”字；
- 每级末尾不管有几个零，都不读，其他数位上有一个零或连续几个零，都只读一个零。

```
输入一个自然数：403020010
四亿零三百零二万零一十

输入一个自然数：10 0000 0000 1000
十万亿零一千
```

四年级 上册 ②

公顷和平方千米

5100 T2000 km²

大面积单位：公顷、平方千米

1 公顷 = 10000 平方米

1 平方千米 = 100 公顷

程序

面积单位换算练习

area_unit_conversion.py　　random

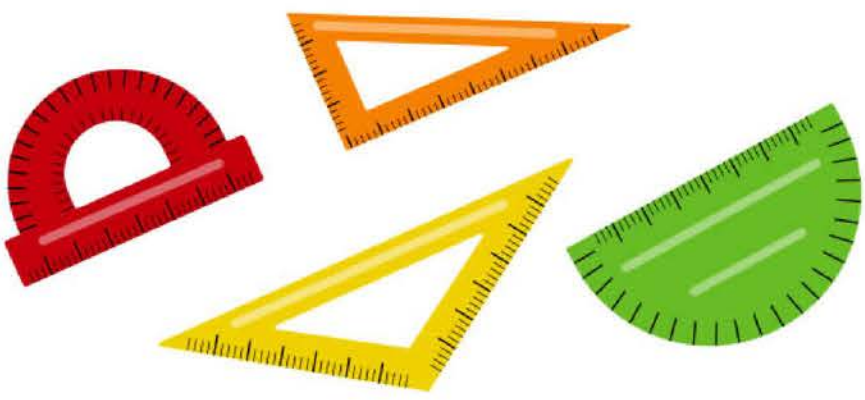
随机生成指定数量的面积单位换算练习题。每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。

《四类小数练习题》(G327) 有一类是双向单位换算练习，此程序扩充了其中单位换算函数中的面积单位，加入了公亩、公顷和平方千米。教材中并未介绍公亩，1 公亩是边长为 10 米的正方形的面积，即 100 平方米。这里加入公亩是为了确保面积单位从“平方毫米”到“平方千米”进制的延续性。在由较小单位向较大单位换算时，可以输入小数或分数，例如 1 cm^2 = 0.0001 m^2 或 1 cm^2 = 1/10000 m^2。当要输入的数位数较多时，可以采用科学计数法。例如 1e3 代表 1 后面有三个零，即 1000；1e-3 代表 1/(1e3)，即 1/1000 或 0.001。

```
题目 1/4:  1公亩 = __公顷
? 0.01
好样的！

题目 2/4:  1公顷 = __平方分米
? 1e6
正确！

题目 3/4:  1平方米 = __公顷
? 1e-6
回答有误，正确答案是 0.0001 或 1/10000。

题目 4/4:  1平方毫米 = __平方厘米
? 1/100
棒！

你的得分 75/100。
```

四年级 上册 ③

角的度量



线段、直线、射线

角、顶点、边

角的度量、量角器

直角、平角、周角
（锐角和钝角在二年级已经学过了）

用量角器画角

程序

画表盘

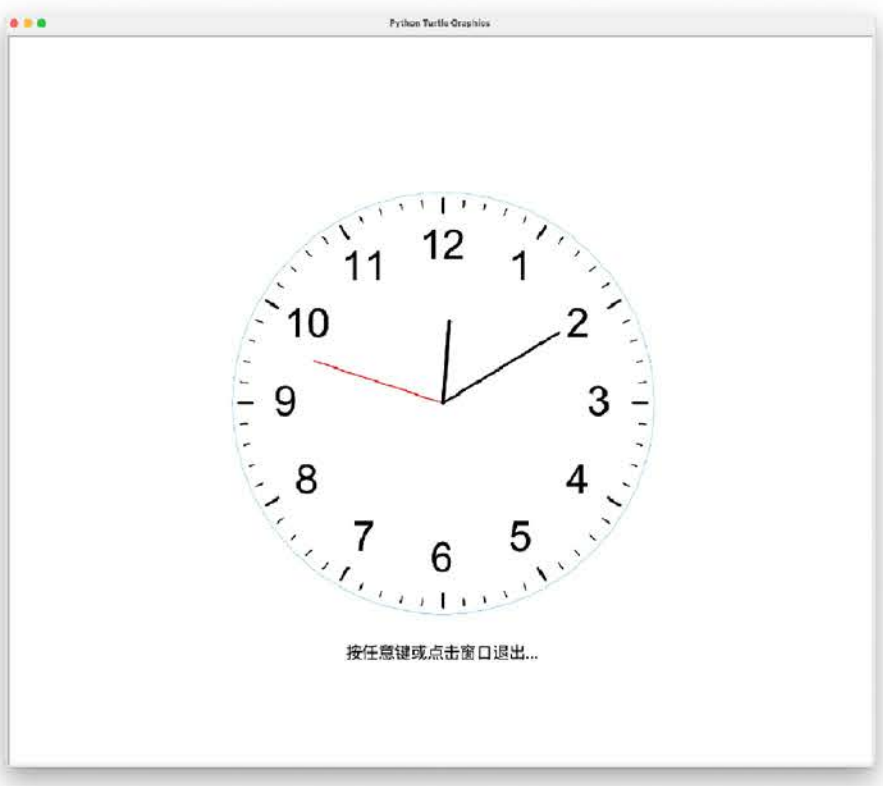
draw_clock_dial.py　turtle, tkinter, threads

此程序是一个自绘表盘的时钟，与三年级时的《表盘时钟》(G311) 程序有两个主要区别：

1. G311 的时钟使用的是表盘背景图片，此程序配合角度的学习，使用 turtle 自行绘制时钟的表盘。

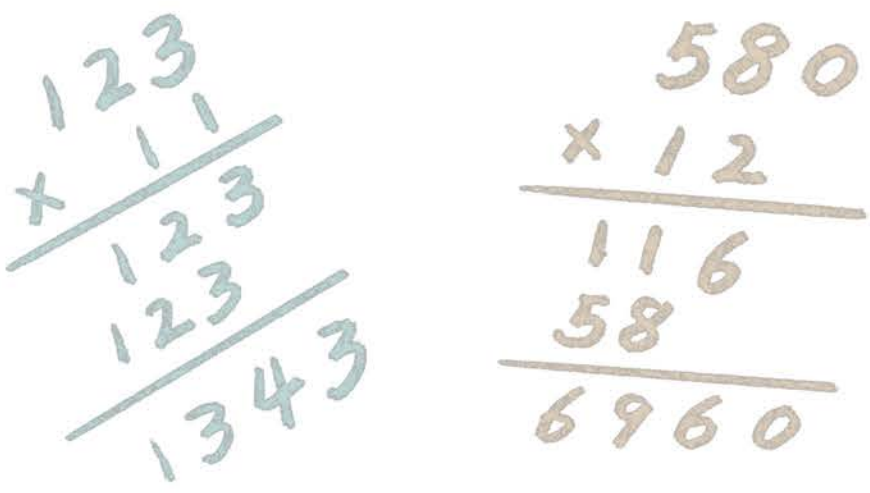
2. G311 使用 time 模块的 sleep() 进行计时，此程序使用 tk 的 after()。与 sleep() 不同的是 after() 不会阻碍程序主线程的运行。

程序运行后可按任意键退出，或在钟表指针开始转动后在窗口任意处点击鼠标退出程序。简单调整程序，还可以加速指针转动或关闭绘制表盘的动画。



四年级 上册 4

三位数乘两位数



三位数乘两位数乘法竖式

因数末尾有 0 的乘法竖式

积随因数变化的规律

单价 × 数量 = 总价

速度 × 时间 = 路程

程序

通用乘法竖式

long_multiplication2.py　　string

程序会将用户输入的两个任意自然数用乘法竖式求积，并将结果以竖式的形式显示在屏幕上。程序在求积时真实模拟了乘法竖式的运算过程，而不是使用编程语言内置的 "*" 运算符直接得到结果。

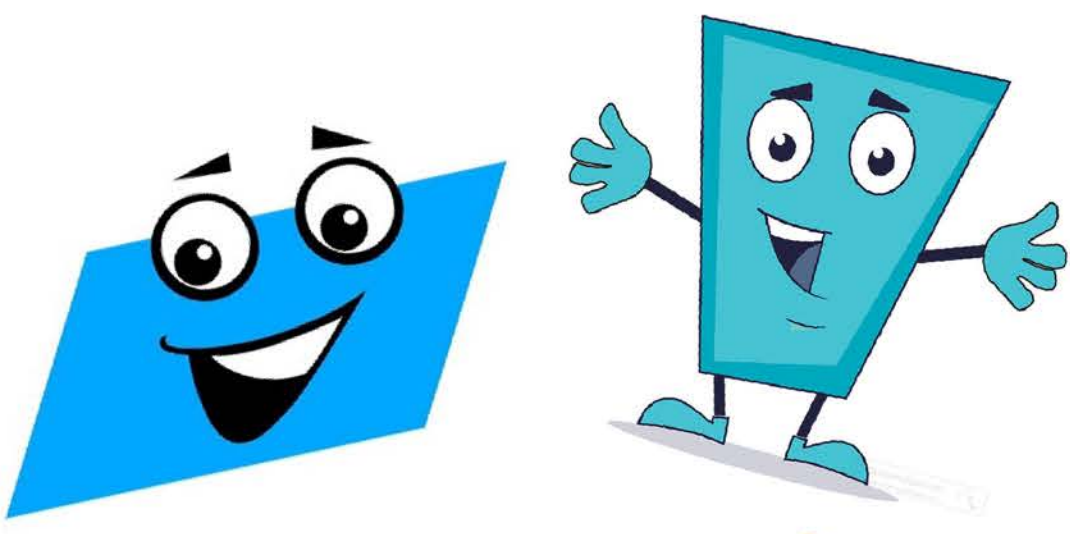
此程序是对《多位数乘法竖式》(G324) 程序的升级，对于因数末尾有零的情况，先不考虑因数末尾的零，把两个因数最右侧的非零位对齐进行乘法竖式运算，最后再把因数末尾的所有零添加到乘积的末尾。两个程序的乘法竖式核心运算过程是一致的，所以此程序仍调用 long_multipy_core() 函数计算两个因数（忽略末尾的零）的乘积。

此程序是自然数乘法竖式系列程序 (G316, G324, G414) 的最终版本。

```
输入第一个自然数: 5890
输入第二个自然数: 6500
     5 8 9 0
x      6 5 0 0
--------------------
    2 9 4 5
  3 5 3 4
--------------------
  3 8 2 8 5 0 0 0
```

四年级 上册 5

平行四边形和梯形



平行线、互相平行

互相垂直、垂线、垂足

三角板画垂线、画长方形

点到直线的距离

平行四边形、底、高

平行四边形不稳定

梯形、上底、下底、腰、高

等腰梯形、直角梯形

程序

数梯形

count_trapezoids.py

random, nested loop, tkinter, coordinate

程序在两条平行线间随机生成指定数量的线段，并保证所有线段互不平行。以两条平行线为上、下底，以任意两条不相交的线段为腰，即可构成一个梯形。

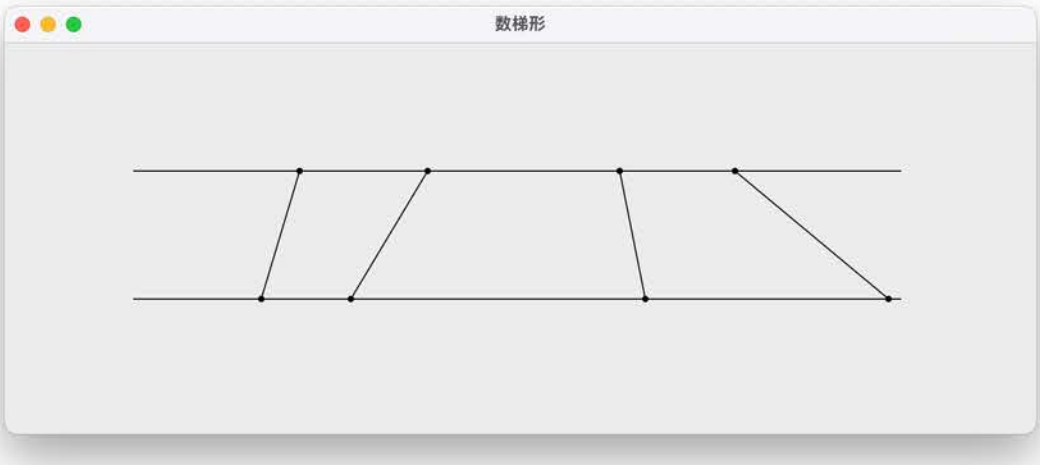
程序开始时，首先让用户输入生成线段的数量，随后会询问生成的线段是否可以相交，除非用户给出肯定回答，否则默认是所有线段互不相交。接下来用户在随机生成的图中数出共有多少个梯形，输入答案后会有答对或答错的提示，答错时还会给出正确答案。

数梯形时，为保证不重不漏，可遵循以下步骤：

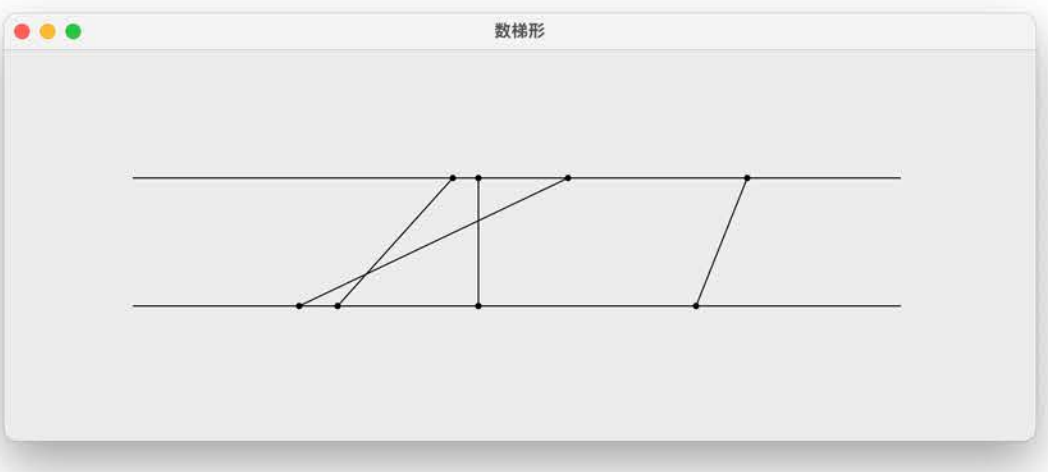
- 按照一定次序，每次固定一条线段作为梯形的一条腰；
- 向后数出所有与这条线段不相交的线段作为另一条腰（只向后寻找第二条腰，是为了避免相同的两条腰被重复计数）。

这也是程序中数梯形的步骤，本质上与之前《三类常见计数问题》(G328) 中的组合问题是一致的，即在多条线段中选出两条作腰，且选择方案与选择顺序无关，所以程序实现采用了类似的二重循环。

```
生成线段的数量 (2 < n < 10): 4
线段能相交吗？(y/n) n
图中共有多少个梯形？ 6
正确！
```

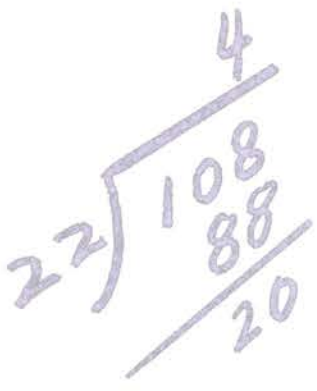


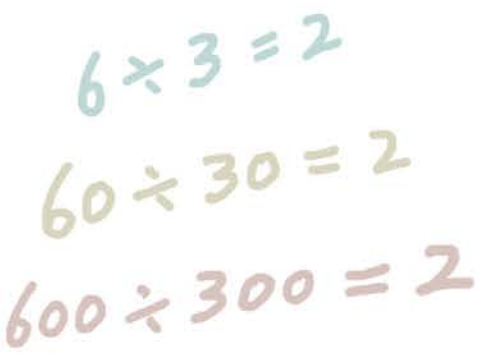

```
生成线段的数量 (2 < n < 10): 4
线段能相交吗？(y/n) y
图中共有多少个梯形？ 3
回答有误，正确答案是 4。
```



四年级 上册 6

除数是两位数的除法





除数是两位数的除法
口算、估算

除数是两位数的除法竖式

试商，余数小于除数

商的变化规律
（含商不变规律）

被除数和除数
末尾都有 0 的除法竖式

程序

通用除法竖式

long_division.py　string

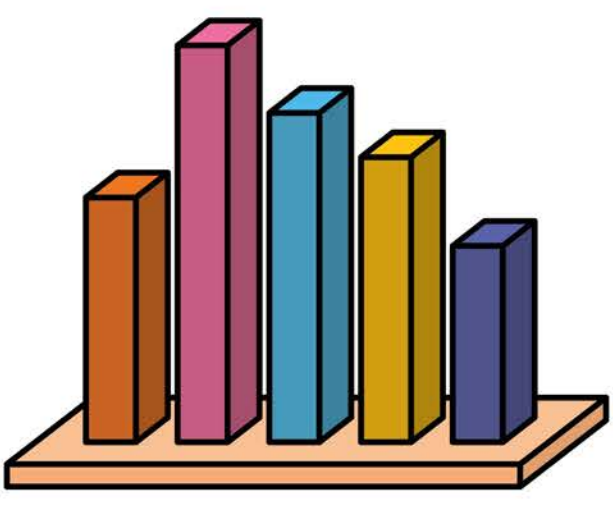
程序会将用户输入的两个任意自然数（除数不能为零）用除法竖式求商和余数，并将结果以竖式的形式显示在屏幕上。程序在求商时真实模拟了除法竖式的运算过程，而不是使用编程语言内置的 "//" 和 "%" 运算符直接得到商和余数。

无论除数是一位数或多位数，除法竖式的基本计算方法都是一样的，所以在《除数是一位数的除法竖式》(G322) 程序中，只需解除对除数位数的限制，即可实现除数是任意位数的除法竖式。根据商不变规律，当被除数和除数的末尾都有零时，可以同时消去末尾相同数目的零，使竖式计算更加简便（如果有余数的话，要在余数末尾补相同数目的零）。此程序在 G322 的基础上，为除法竖式补充了上述消零的步骤。

此程序是自然数除法竖式系列程序 (G322, G416) 的最终版本。

```
输入一个自然数作为被除数：5480
输入一个非零自然数作为除数：360

    15
   ---
36/548
   36
   ---
   188
   180
   ---
     8

5480 / 360 = 15 ... 80
```

四年级 上册 ⑦
条形统计图



条形统计图

为条形统计图
选择适当的标注间隔

程序一

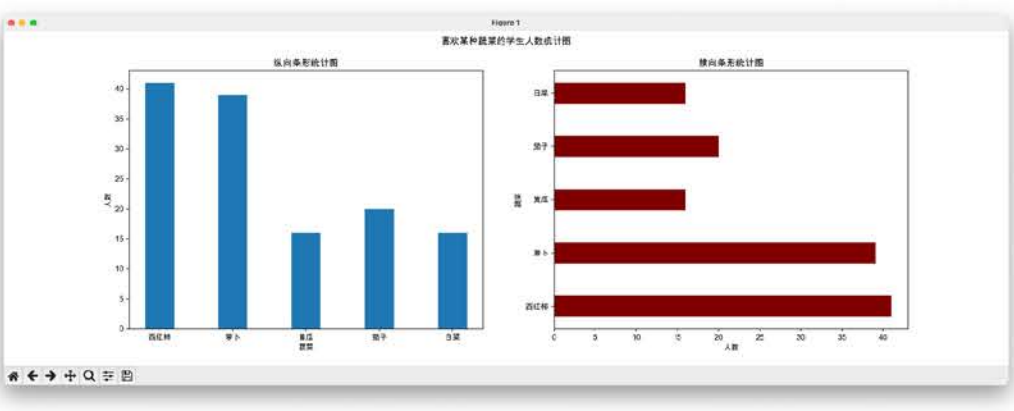
应用 Matplotlib 绘制条形统计图

bar_chart.py　matplotlib

此程序使用流行的 Python 绘图库 Matplotlib 绘制纵向和横向条形统计图。在 Matplotlib 中，一个窗口内的所有绘图称作一个 Figure，一个 Figure 可能包含一个或多个绘图，每个绘图称作一个 Axes。Matplotlib 有两种代码风格：

- 面向对象 (OO) 风格：显式创建 Figure 和 Axes，并调用它们的方法进行绘图。
- pyplot 风格：使用 pyplot 隐式创建和管理 Figure 和 Axes，并使用 pyplot 的函数进行绘图。

一般建议使用面向对象风格，特别是对于复杂的绘图项目，而 pyplot 风格对于快速交互绘图来说非常方便。作为对比，程序分别使用两种代码风格各在一个 Figure 上绘制条形统计图，绘图效果完全相同。每个 Figure 有横向排列的两个 Axes，一个是纵向条形统计图，另一个是横向条形统计图。



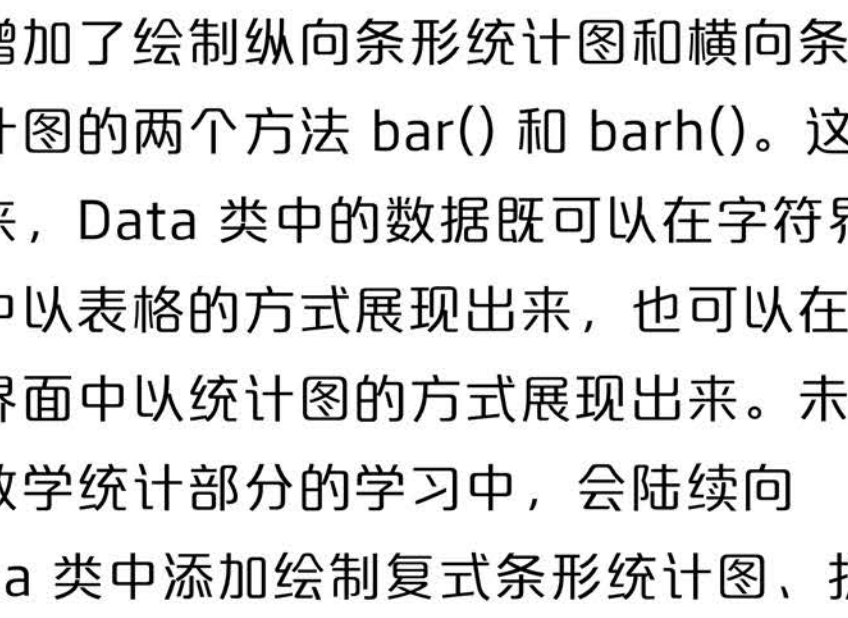
程序二

创建带有绘图功能的表格类的子类

data1.py　matplotlib, class

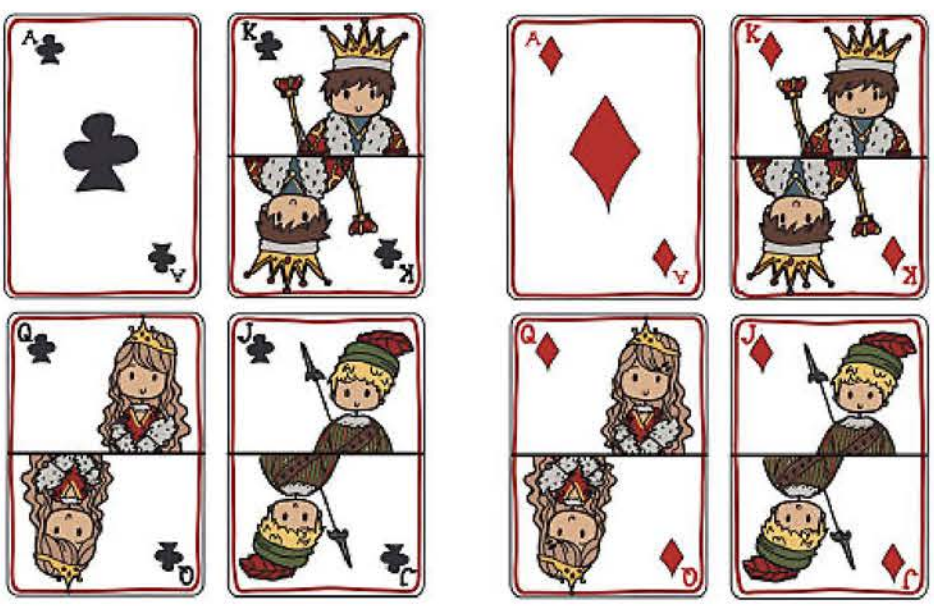
此程序基于《创建表格类》(G323) 程序，创建了 Table 类的子类 Data 类。Data 类继承了 Table 类的属性和方法，并增加了绘制纵向条形统计图和横向条形统计图的两个方法 bar() 和 barh()。这样一来，Data 类中的数据既可以在字符界面中以表格的方式展现出来，也可以在图形界面中以统计图的方式展现出来。未来在数学统计部分的学习中，会陆续向 Data 类中添加绘制复式条形统计图、折线统计图等更多方法。

	西红柿	萝卜	黄瓜	茄子
喜欢的人数	19	13	11	17



四年级 上册 8

数学广角—优化



统筹优化

必胜策略

程序

报数游戏

counting_game.py　　random

报数游戏的规则如下：两人轮流报数，每次在给定范围内选择一个加数（例如给定范围是 1 到 3，则可以选择 +1、+2 或 +3）累加到公共计数器中，谁报数后使计数器加到约定的数（例如 21），谁就获胜。

程序编写的报数游戏是由玩家对电脑。游戏开始时会随机给出加数的选择范围和和胜利条件，然后由玩家首先开始报数，如果玩家输入的数超出了可选择范围，会给出提示并让玩家重新输入。之后电脑和玩家轮流报数，直到一方获胜。

报数游戏是有必胜策略的，以加数范围 1 到 3，目标 21 为例。要想抢到 21，就需要抢到 17，因为无论对方 +1、+2 还是 +3，你在下一次都可以加到 21。而想要抢到 17，就要抢到 13，以此类推，就必须抢到 9、5 和 1 这些“必胜数”。所以只要 0 不是“必胜数”，先报数的玩家就一定能抢到第一个“必胜数”进而赢得比赛。电脑程序也是按照这个策略编写的，如果玩家在报数过程中出现失误，电脑就会抢到“必胜数”并取得最终胜利。

```
每回合在 [1, 2] 中选择一个数累加到公共计数器中。
谁报数后使计数器加到 10, 谁就获胜。

当前计数器: 0 -> 目标累加值: 10
你的选择: 1

当前计数器: 1 -> 目标累加值: 10
计算机选择: 2

当前计数器: 3 -> 目标累加值: 10
你的选择: 2

当前计数器: 5 -> 目标累加值: 10
计算机选择: 2

当前计数器: 7 -> 目标累加值: 10
你的选择: 1

当前计数器: 8 -> 目标累加值: 10
计算机选择: 2

当前计数器: 10 = 目标累加值: 10
计算机赢了!
```

四年级 下册 ①

四 则 运 算

$$96\div[(12+4)\times2]$$
$$=96\div[16\times2]$$
$$=96\div32$$
$$=3$$

加法

和 = 加数 + 加数

加数 = 和 - 另一个加数

减法

差 = 被减数 - 减数

减数 = 被减数 - 差

被减数 = 减数 + 差

乘法

积 = 因数 × 因数

因数 = 积 ÷ 另一个因数

除法

商 = 被除数 ÷ 除数

除数 = 被除数 ÷ 商

被除数 = 商 × 除数

四则混合运算、括号（）、[ ]、{ }

有括号的混合运算的顺序

程序

有括号的四则混合运算

eval_expressions.py string

用户输入任意有括号的四则混合运算算式，程序把该算式的计算结果显示在屏幕上。

程序可以解析含有加减乘除和括号的算术表达式，所用算法模拟了人们实际做计算的步骤：

1. 扫描算式字符串，获取运算数、运算符，以及按照本单元所学的运算顺序规则，为每个运算符设定运算优先级。
2. 再按照优先级的大小顺序，从最高优先级的运算做起，优先级相同时，从左至右依次计算，直至算出最终结果。

某些堆栈或递归算法，对算式字符串边扫描边计算，完成一次扫描即可得出结果，效率更高。程序中的算法是先扫描一次，确定运算顺序后再计算，这和我们在脱式中的计算过程基本一致，所以更容易理解。

程序在扫描用户输入的算式字符串时，会忽略所有 “0123456789+-*/()” 以外的字符，多重括号仅使用小括号。

```
输入有括号的四则混合运算算式:
96/((12+4)*2)
3

输入有括号的四则混合运算算式:
96 / ((1 2+ 4)*2 )
3
```

四年级 下册 ②

观察物体（二）

立体图形的三视图

从前面看：正视图

从上面看：俯视图

从左面看：左视图

程序

几何体三视图

cubes_3view.py

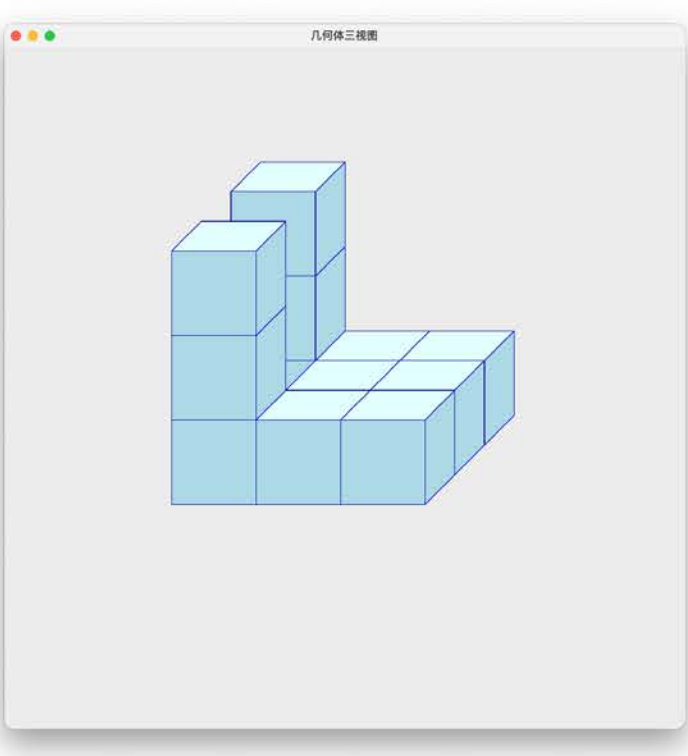
tkinter, coordinate, 2d list, random, class

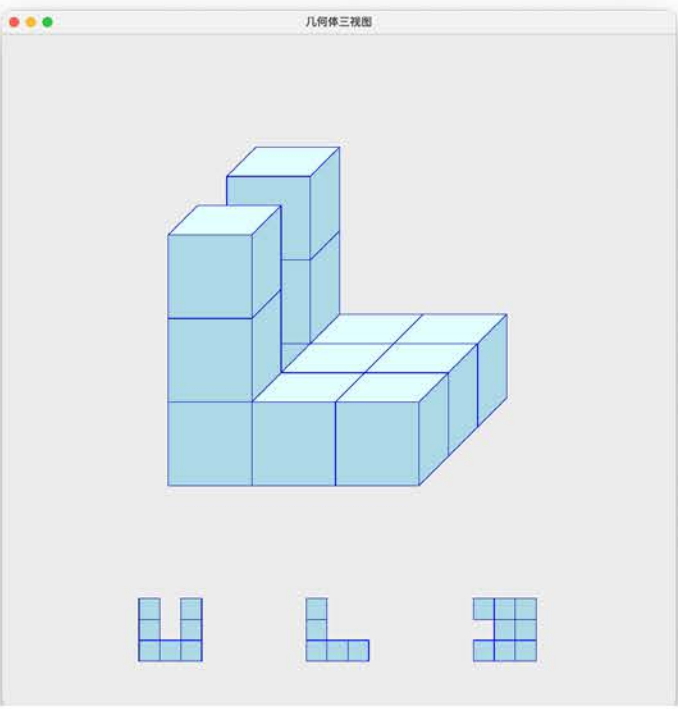
程序随机生成由立方体积木块拼成的三维几何体，并在窗口正中显示该几何体，按空格键在窗口下方显示该几何体的三视图，从左至右依次为：左视图、正视图、俯视图。再按一次空格键重新生成新的几何体。

在程序中，每个立方体的空间位置（列、行、层）存储在一个列表中，几何体中包含的所有立方体也存储在一个列表中。所以实际上几何体是由一个二维列表所表示。

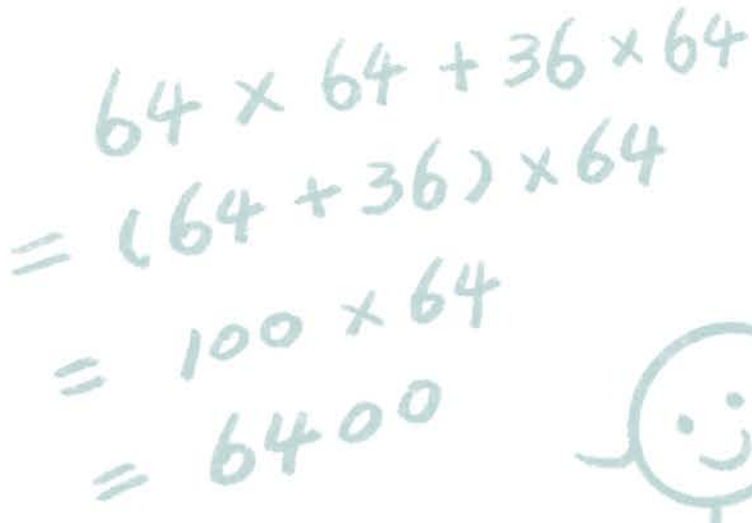
画图时，每个立方体用可见的三个面（右面、前面、上面）表示，通过颜色和角度的变化表现立体感。使用 Tk Canvas 画图时，在同一位置上新画的图形会覆盖先前的图形。所以只需遵循从左向右、从后向前、从下向上的顺序绘制立方体，就可以实现几何体各立方体积木块之间的遮挡效果。

画几何体三视图时，只需在列、行、层三个维度中忽略一个维度即可把立体图 “压扁” 为平面图。例如如果忽略 “层” 这个维度，就可以画出俯视图。





四年级 下册 ③
运 算 定 律



加法运算定律

加法交换律
a + b = b + a

加法结合律
(a + b) + c = a + (b + c)

连减
a - b - c = a - (b + c)

乘法运算定律

乘法交换律
a × b = b × c

乘法结合律
(a × b) × c = a × (b × c)

乘法分配律
(a + b) × c = a × c + b × c
a × (b + c) = a × b + a × c

连除
a ÷ b ÷ c = a ÷ (b × c)

程序

解 24 点

solve_24.py itertools, nested loops, dictionary, algorithm

24 点是一个流行的算数游戏，它的基本规则是：一副扑克牌除去 J、Q、K 和大小王，任抽四张牌，使用加、减、乘、除和括号把它们连成算式，使结果等于 24。24 点还有很多变体，例如只去掉大小王，把运算数的范围扩大到 13，或增加更多的运算符等。

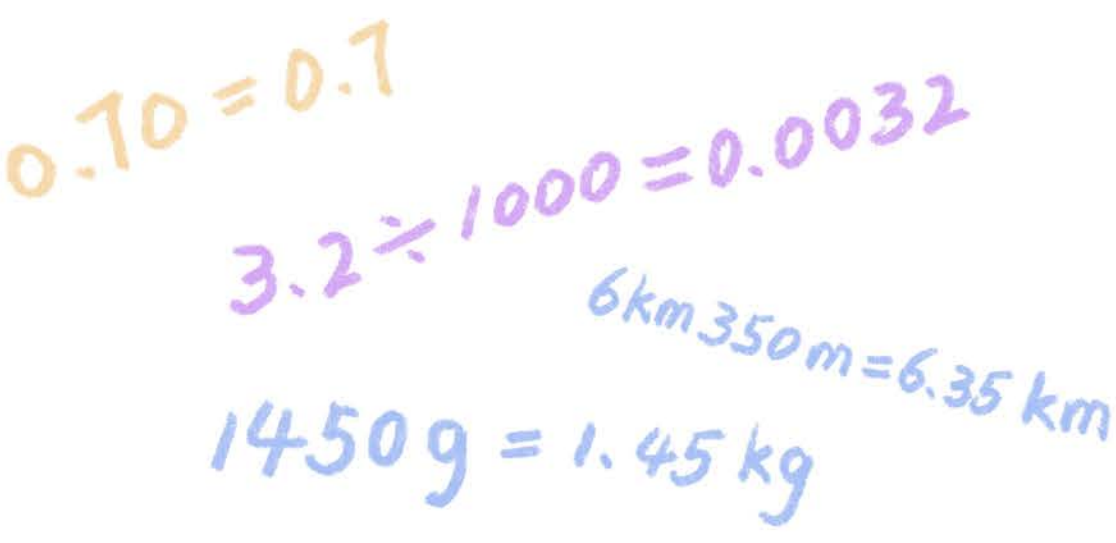
此程序对于用户输入的任意四个数，显示通过四则运算和添加括号得到 24 的全部独立解式。以下是一些具体说明：

1. 程序使用穷举法，对运算数、运算符和运算顺序的全部可能排列，让计算机逐个检验结果是否等于 24。一组运算数、运算符和运算顺序即代表了一个算式，程序在计算算式结果时，调用了《有括号的四则混合运算》(G421) 程序中的 calc() 函数。
2. 通过穷举法得到的全部解式中有很多是“等效”的重复解，例如通过交换律和结合律得到的解。为了得到更有意义的“独立”解，需要进行“去重”的操作。如果根据等效解出现的原因定出一套去重规则，再根据规则进行去重，实现起来会很复杂。程序在处理去重问题时取了个巧，不究其原因，而只是认定一个简单的事实：如果把两个解式对应的四个运算数换成任意四个数，计算结果仍然相同，就认为这两个解式是等效的。这便是程序中去重的方法。
3. 在显示结果时，简单的做法是把除最后一步外的所有运算都加上括号，但其实很多情况是不需要加括号的，例如乘法运算的前面如果是加法，就不需要再单独给乘法运算加括号。程序中使用了一组条件语句，对每步运算的运算符及其前、后运算符可能出现的所有情况进行判断，决定是否需要添加括号。
4. 程序不止可以解 24 点。用户可以输入任意数量的运算数和任意目标结果，在有解的情况下，程序会显示出全部独立解式。随着运算数的增加，程序很快会变慢，这也是穷举法的特点：虽然在一定范围内解决问题简单方便，但规模受限。

```
>>> solve(7, 8, 5, 4)

8 + 7 + 5 + 4
(8 - 7 + 5) × 4
8 × (7 + 5) / 4
(8 + 4) × (7 - 5)

>>> solve(7, 8, 5, 4, target=40)

(8 + 7 - 5) × 4
(7 + 5) × 4 - 8

>>>
```

四年级 下册 4

小数的意义和性质



小数的意义

小数的计数单位

十分之一（写作 0.1）

百分之一（写作 0.01）

千分之一（写作 0.001）

……

相邻计数单位之间的进率是 10

小数的读法和写法

小数的数位顺序表

小数的性质：小数的末尾添上“0”或去掉“0”，小数的大小不变

小数化简

小数比大小

小数点移位

每向右移动一位，相当于乘以 10

每向左移动一位，相当于除以 10

小数与单位换算

小数的近似数

四舍五入

表示近似数时，小数末尾的 0 不能去掉。

保留整数（精确到个位）

保留一位小数（精确到十分位）

保留两位小数（精确到百分位）

……

把大数改写成用“万”或“亿”作单位的数

程序

三类小数练习题

decimal_practice2.py　　random, decimal

随机生成指定数量的小数练习题。每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。小数练习题共有三种类型，可以一次只练习一种类型，或选择多种类型混合出题。

1. 求小数的近似数。程序随机给出一个小数，并随机产生保留几位小数或精确到哪一位的指令。
2. 小数点移位练习。程序随机给出一个小数，通过计算这个小数乘以或除以 10, 100 或 1000 的积或商，练习向右或向左移动小数点。
3. 引用《四类小数练习题》(G327) 中的双向单位换算。对于小数点向左移动位数较多的单位换算，三年级时可能更多采用分数形式的答案，本单元正好可以练习使用小数作答。程序给出的答案有时会出现类似 “1e-06” 的符号 (等效于 “1e-6”)，这是计算机中的科学计数法，表示将 1.0 的小数点向左移动 6 位，即 0.000001。相应地，”1e6” 是将 1.0 的小数点向右移动 6 位，即 1000000。

Python 中默认表示小数的数据类型是 float (浮点数)，float 运算效率很高，但常存在一定误差，例如 1.1 + 2.2 = 3.3000000000000003。Python 标准库的 decimal 模块里有一种专门用于表示小数的数据类型 Decimal，虽然运算效率不如 float，但可以得到小数运算的精确结果。程序中求小数的近似数和计算小数点移位的正确答案时，都使用了 Decimal 数据类型。

```
题目 1/4:  19.971 精确到十分位
? 20
回答有误，正确答案是 20.0。

题目 2/4:  84.68 / 1000 =
? 0.08468
太牛了！

题目 3/4:  98.4758 保留两位小数
? 98.48
强！

题目 4/4:  1g = __t
? 0.000001
好样的！

你的得分 75/100。
```

四年级 下册 5

三角形

三条边、三个角、三个顶点

三角形的高和底

三角形具有稳定性

两点间所有连线中线段最短，
这条线段的长度叫做
两点间的距离。

三角形任意两边的和大于第三边

三角形的分类

按角分：
锐角三角形、钝角三角形、直角三角形
（直角边、斜边）

按边分：
等腰三角形、等边三角形（正三角形）

三角形的内角和是 180°

四边形的内角和是 360°，
多边形内角和的规律

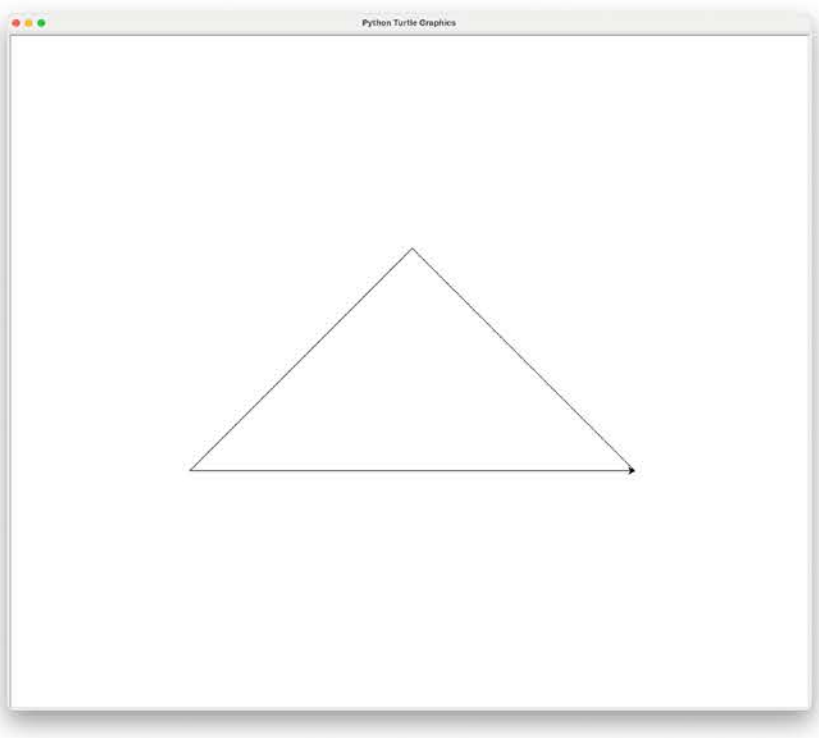
程序一

画等腰三角形

isosceles_triangle.py turtle

用户输入等腰三角形的顶角和腰长，程序画出等腰三角形，并给出底角和底边长。在不使用三角函数的情况下，底边长是用 turtle 的 distance() 方法返回底边两端点之间的距离获得的。

```
请输入等腰三角形的顶角：90
请输入腰长 (0 - 500): 500

顶角：90  底角：45  腰长：500  底长：707
```



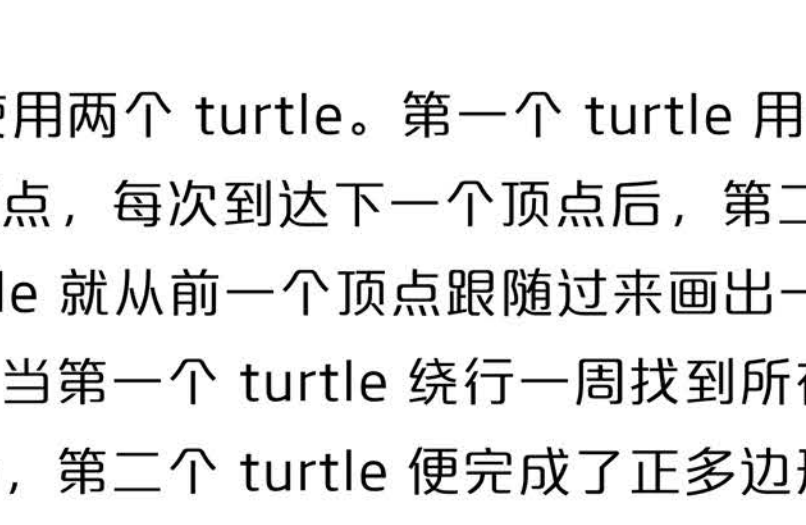
程序二

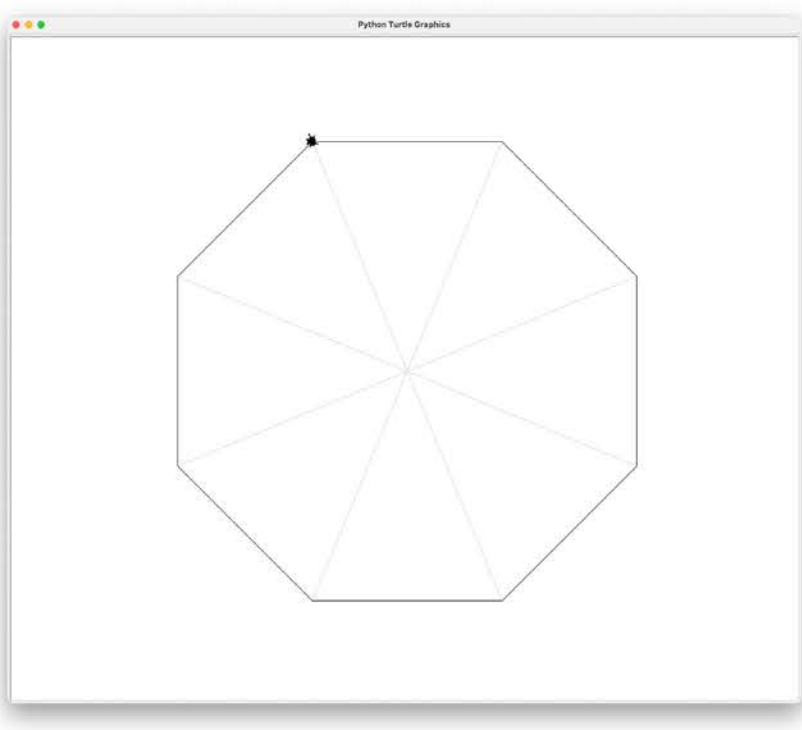
画正多边形

regular_polygon.py turtle

本单元在三角形的基础上，也把研究范围延伸到了多边形。所以第二个程序是画正多边形（已知中心到顶点的距离）。程序中使用了两种方法：

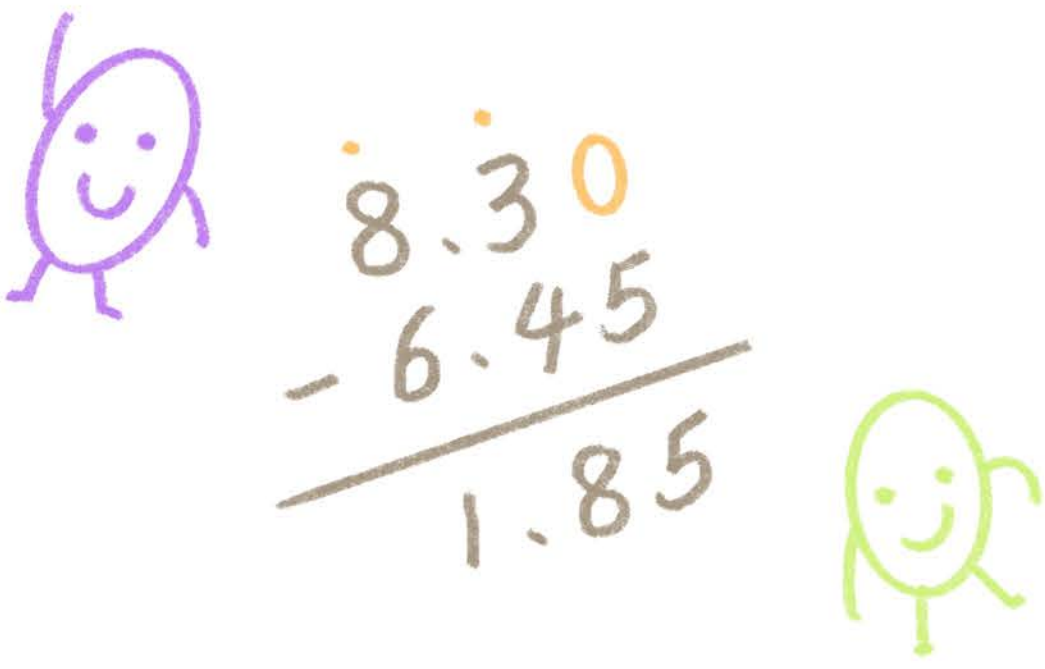
1. 只要知道边长就可以用 turtle 绕行一周画出正多边形。首先找到正多边形相邻的两个顶点，然后用 turtle 的 distance() 方法返回它们的距离，即正多边形的边长；
2. 使用两个 turtle。第一个 turtle 用于找顶点，每次到达下一个顶点后，第二个 turtle 就从前一个顶点跟随过来画出一条边。当第一个 turtle 绕行一周找到所有顶点后，第二个 turtle 便完成了正多边形的绘制。





四年级 下册 6

小数的加法和减法



小数加减法竖式
（小数点对齐）

利用加法运算定律
简便计算

程序

小数加减法竖式

add_sub_decimals.py　string

程序开始时，提示用户输入两个小数（也可以输入整数），并选择是做加法还是减法。之后程序会根据用户输入，使用相应的小数或整数的加减法竖式进行计算，并将结果以竖式的形式显示在屏幕上。程序在求和时真实模拟了小数加减法竖式的运算过程，而不是使用编程语言内置的 "+" 或 "-" 运算符直接得到结果。

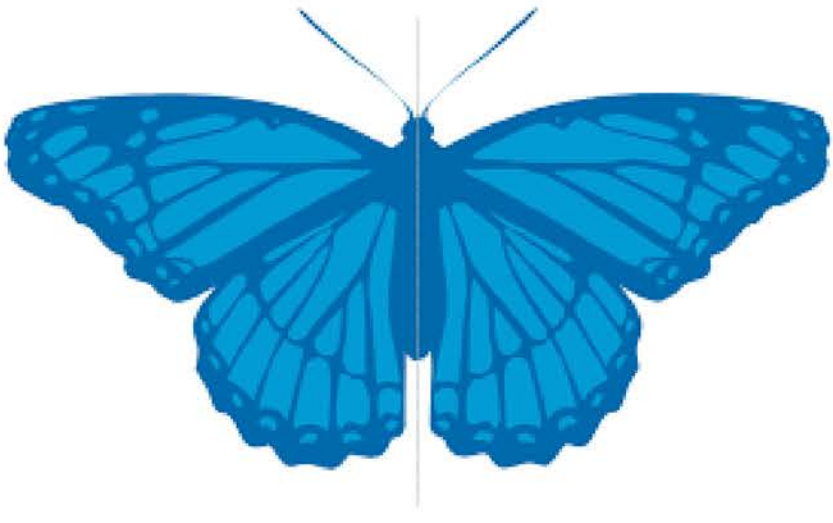
小数加减法竖式本质上就是小数点对齐的整数加减法竖式，所以程序重组了《加法竖式》(G314) 和《减法竖式》(G315) 程序，并在此基础上加入了小数加减法竖式的计算过程：

1. 在小数位数较少的小数末尾补零，使两个小数具有相同的小数位数，即小数点对齐；
2. 忽略小数点，做整数加减法竖式（同 G314/G315 的程序）；
3. 在与运算数的小数点相同的位置，为计算结果添加小数点；
4. 删除第一步中向加数或减数末尾添加的零，只保留被减数末尾的零；
5. 对齐小数点位置，显示竖式（同 G314/G315 的程序）；
6. 对计算结果进行化简，去掉末尾不必要的零，显示最终结果。

```
输入第一个小数（或整数）：23.5
输入第二个小数（或整数）：4.85
选择做加法还是减法 (+ 或 -)? +

  2 3 . 5
+   4 . 8 5
-----------
  2 8 . 3 5

化简后的结果是：28.35


输入第一个小数（或整数）：1
输入第二个小数（或整数）：0.01
选择做加法还是减法 (+ 或 -)? -

  1 . 0 0
- 0 . 0 1
---------
  0 . 9 9

化简后的结果是：0.99


输入第一个小数（或整数）：0.99
输入第二个小数（或整数）：0.01
选择做加法还是减法 (+ 或 -)? +

  0 . 9 9
+ 0 . 0 1
---------
  1 . 0 0
```

四年级 下册 ⑦

图形的运动（二）



轴对称

平移

利用平移求面积

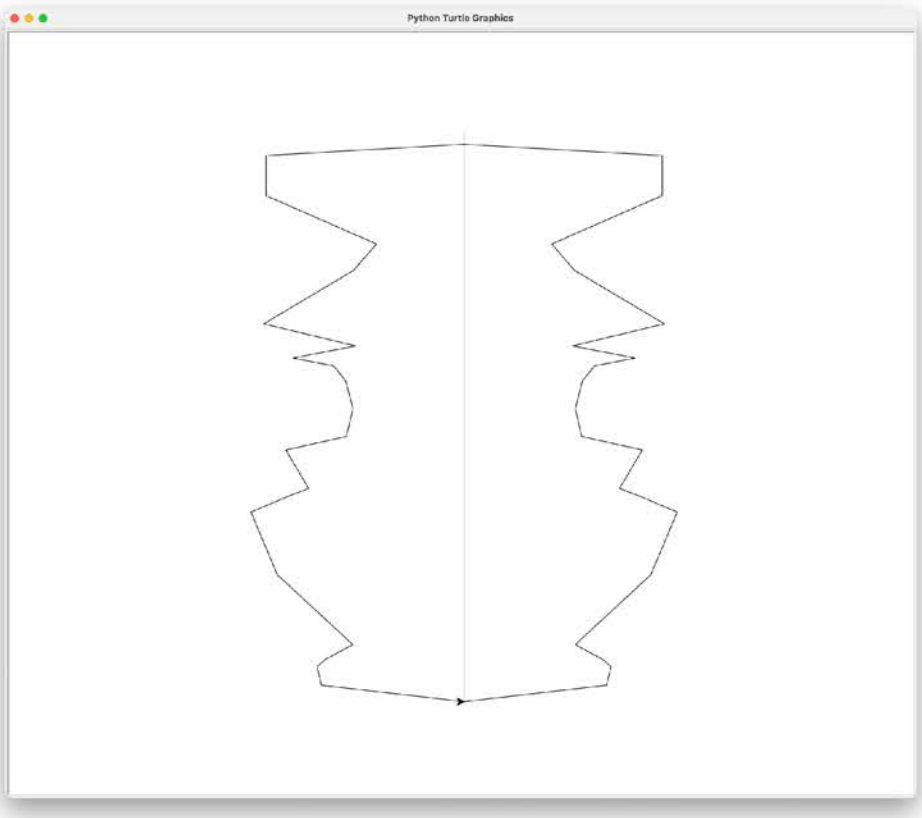
程序

随机生成轴对称图形

reflective_symmetry.py　　random, turtle

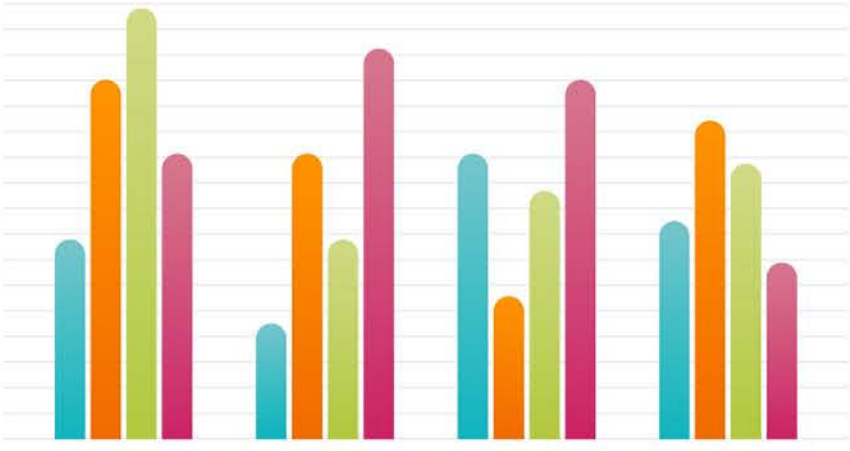
程序随机生成一个左右对称的轴对称图形，对称轴位于窗口中间。在对称轴的一侧按照用户输入的点数，随机生成若干个点，并把这些点镜像到对称轴的另一侧。把所有点连接起来就形成了一个轴对称图形。

```
在对称轴一侧随机生成多少个点（1-50）？ 20
```



四年级 下册 8

平均数与条形统计图



平均数

复式条形统计图

程序

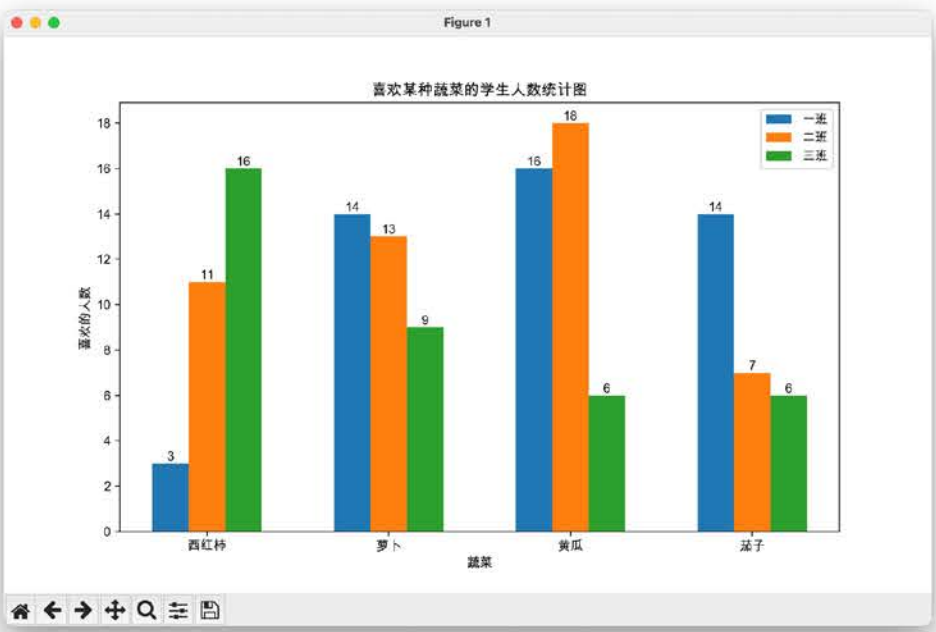
复式条形统计图和平均数

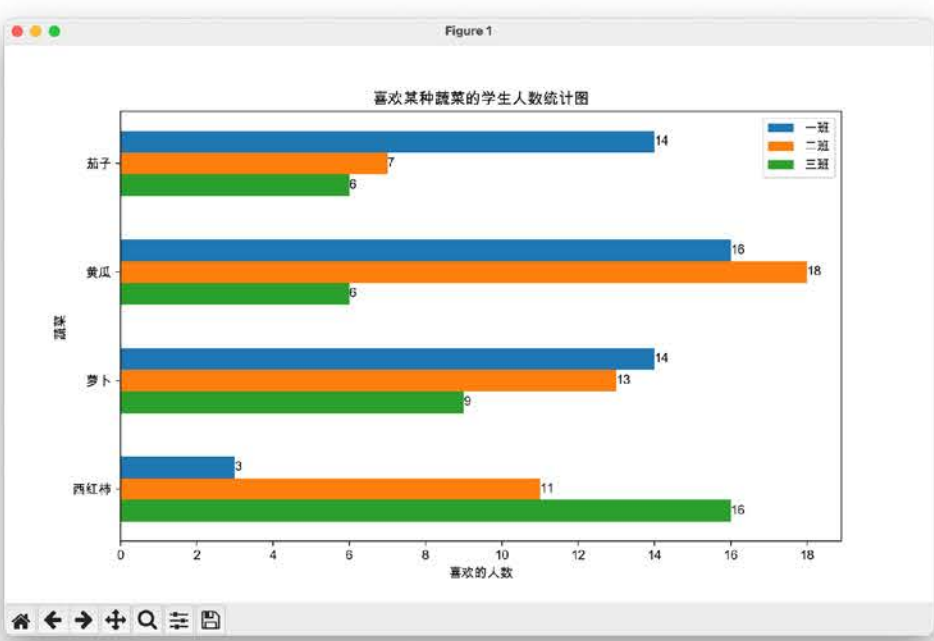
data2.py matplotlib, class, 2d list

程序改进了 data1.py (G417) 中的 Data 类，使绘制条形统计图的方法 bar() 和 barh() 现在可以绘制表现多行数据的复式条形统计图。程序还为 Data 类增添了两个新的方法分别用于求指定行或列数据的平均数。

	西红柿	萝卜	黄瓜	茄子
一班	3	14	16	14
二班	11	13	18	7
三班	16	9	6	6

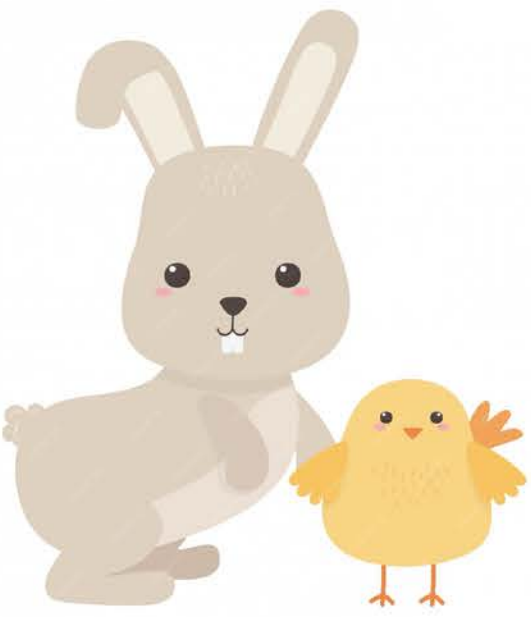
```
每行的平均数是: [12, 12, 9]
每列的平均数是: [10, 12, 13, 9]
```





四年级 下册 9

数学广角—鸡兔同笼



解鸡兔同笼问题

列表法

假设法

抬腿法

程序

鸡兔同笼

chicken_rabbit.py　　tkinter, random

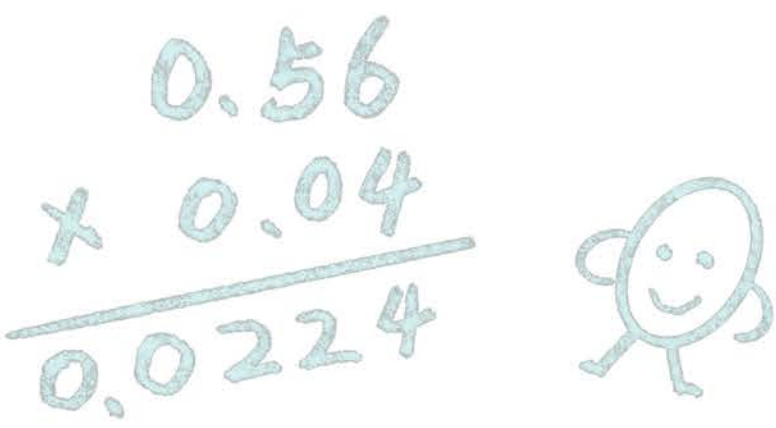
“鸡兔同笼”是中国古代（大约 1500 年前）一道著名的数学问题。笼中有若干只鸡和兔，从上面可以数出有多少个头，从下面可以数出有多少只脚，求鸡和兔各有几只。

程序随机生成一道鸡兔同笼题，用户输入答案后，会给出答对或答错的提示，并显示正确答案。程序还会在窗口中显示同正确答案数量的鸡和兔子的图片，以更直观的方式，增强学习者对解题思路的理解。

```
从上面数，有 4 个头，从下面数，有 14 只脚。
有多少只鸡？ 1
有多少只兔子？ 3
正确！笼中有 1 只鸡，3 只兔子。
```

五年级 上册 ①

小数乘法



小数乘法竖式

积的近似数

利用乘法运算定律简便计算

估算

程序

小数乘法竖式

multiply_decimals.py　　string

程序会将用户输入的两个任意小数（或整数）用乘法竖式求积，并将计算结果显示在屏幕上。程序在求积时真实模拟了小数乘法竖式的运算过程，而不是使用编程语言内置的 "*" 运算符直接得到结果。小数乘法的计算过程如下：

1. 先忽略因数的小数点，按照整数乘法算出积；
2. 看两个因数中一共有几位小数，就从积的个位起数出几位，点上小数点；
3. 积的小数位数不够时，要在前面用 0 补足，再点小数点；
4. 结果化简，去掉小数末尾的 0。

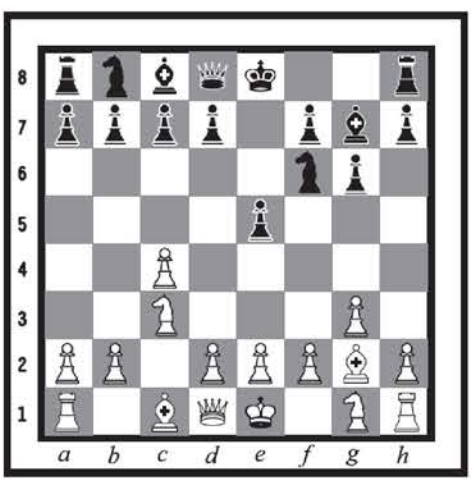
程序在实现第一步的整数乘法竖式时，调用了《多位数乘法竖式》(G324) 程序中的 long_multiply_core() 函数返回计算结果。

本单元的重点是对小数乘法过程的理解和掌握，由于显示小数乘法竖式在显示层面需要考虑的因素较多，所以程序只给出小数乘法竖式的结果，感兴趣的学习者可以挑战一下小数乘法竖式的显示。

```
输入第一个小数（或整数）：5.5
输入第二个小数（或整数）：6
5.5 × 6 = 33

输入第一个小数（或整数）：0.55
输入第二个小数（或整数）：0.6
0.55 × 0.6 = 0.33
```

五年级 上册 ②

位 置



用数对表示位置

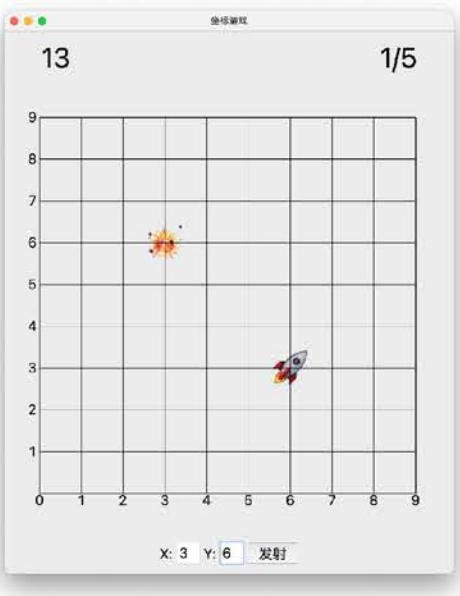
程序一

坐 标 游 戏 —
根 据 位 置 输 入 坐 标

input_coordinate.py　　tkinter, random

游戏会在随机位置放置一个火箭，玩家的任务是输入火箭所在位置的坐标，发射炮弹将其击落。如果在限定时间内击落了规定数目的火箭，任务成功，否则失败。限定时长和火箭数目可以在程序中进行设置。

输入坐标时，可以使用鼠标或 TAB 键切换 X、Y 坐标输入框和发射按钮的焦点。为了加快发射速度，程序为坐标输入设置了键盘快捷键：在 X 输入框内按 RETURN 键可以切换焦点到 Y 输入框，再次按 RETURN 键发射。

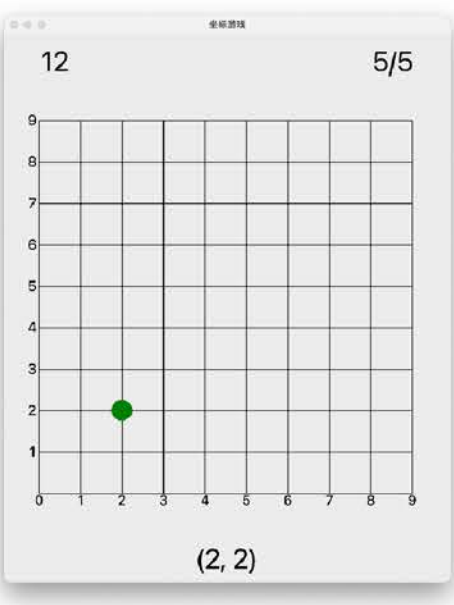


程序二

坐 标 游 戏 —
根 据 坐 标 点 击 位 置

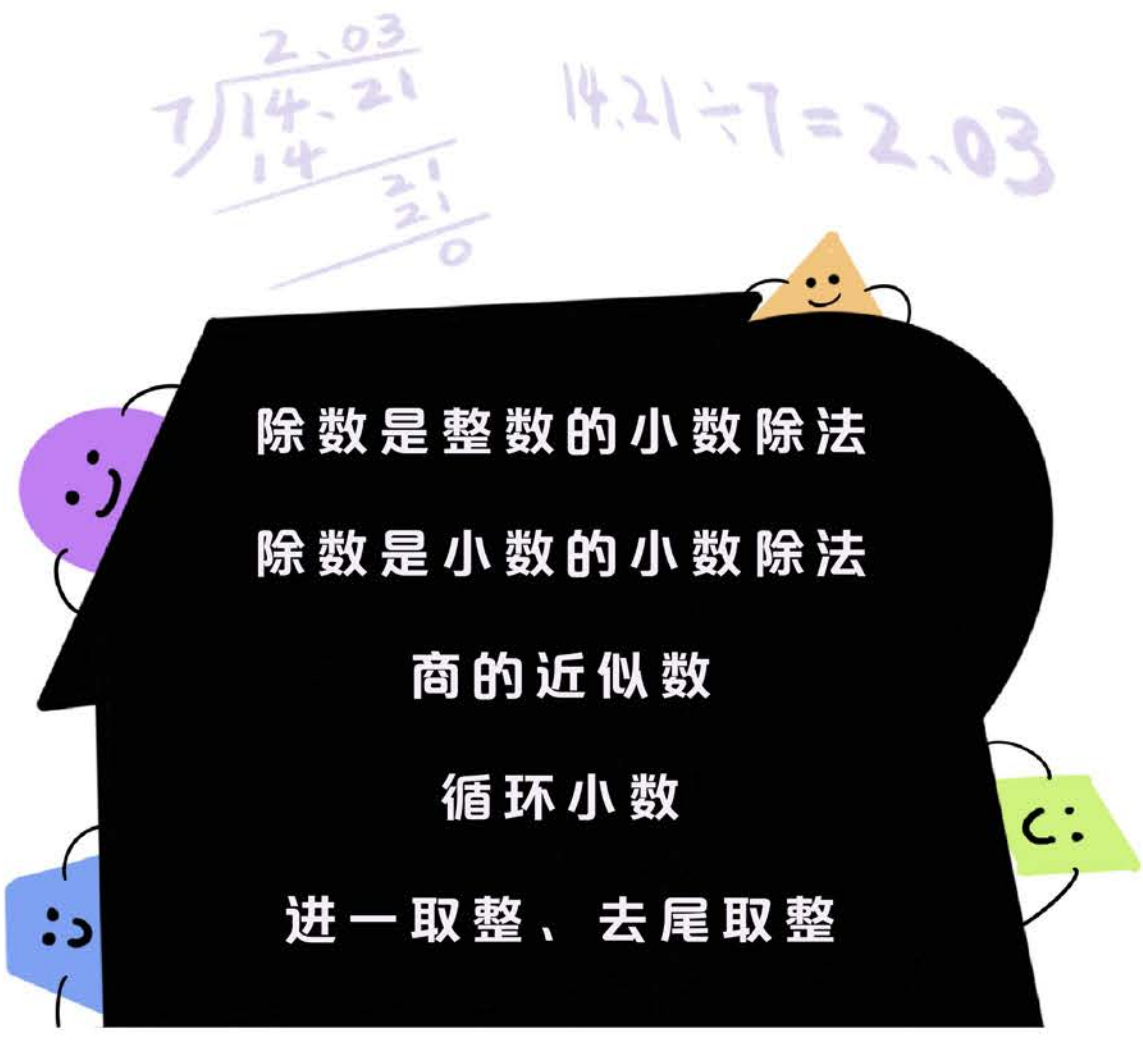
click_position.py　　tkinter, random

游戏随机生成一组坐标数对，玩家需要根据数对中的坐标，用鼠标点击相应的位置，点对了会显示一个绿色圆圈，点错了会显示红色。如果在限定时间内完成规定数目的题目，任务成功，否则失败。限定时长和题目数量可以在程序中进行设置。



五年级 上册 ③

小数除法



程序一

小数除法竖式

divide_decimals.py　string, decimal

程序会将用户输入的两个任意小数（或整数）用除法竖式求商。如果商是有限小数，一直除到余数为 0，得到商的精确值；如果商是无限循环小数，一直除到找到循环节，并将商以含循环节的方式表示出来（小括号中的部分为循环节）。程序在求商时真实模拟了小数除法竖式的运算过程，而不是使用编程语言内置的 "/" 运算符直接得到结果。小数除法的计算过程如下：

1. 先移动除数的小数点，使它变成整数；
2. 除数的小数点向右移动几位，被除数的小数点也向右移动几位（位数不够的，在被除数的末尾用 0 补足）；
3. 按除法是整数的小数除法进行计算，商的小数点和被除数的小数点对齐；
4. 除到被除数的最后一位还没有除尽，添 0 继续除，一直除到余数为 0（商为有限小数）或相同的余数再次出现（商为无限循环小数且找到循环节）。

本单元的重点是对小数除法过程的理解和掌握，由于显示小数除法竖式在显示层面需要考虑的因素较多，所以程序只给出小数除法竖式的结果，感兴趣的学习者可以挑战一下小数除法竖式的显示。

```
输入被除数（小数或整数）：7.86
输入除数（非零小数或整数）：1.3
商： 6.0(461538)
```

程序二

常见分数转化为小数练习

fraction_to_decimal.py　random

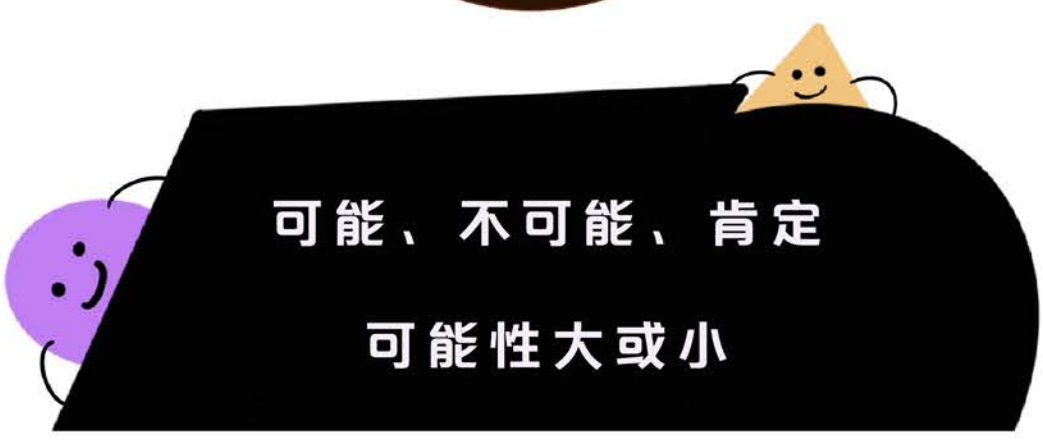
随机生成指定数量的分数转化为小数练习题，分数为分母在十以内的常见分数。每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。

将分数转化为小数时，直接用分子除以分母即可。练习的主要目的是让学习者熟悉分母在十以内的常见分数和小数之间的对应关系。对于分母是 3, 6, 7, 9 的分数，在转化为小数时是无限小数，此时答案保留三位小数（四舍五入）。

```
分数转化为小数。
(无限小数精确到千分位)

题目 1/4:  3/4
对应的小数？ 0.75
不错！

题目 2/4:  1/5
对应的小数？ 0.5
回答有误，正确答案是 0.2。

题目 3/4:  2/9
对应的小数？ 0.222
牛！

题目 4/4:  1/6
对应的小数？ 0.167
答对了！

你的得分 75/100。
```

五年级 上册 4

可能性



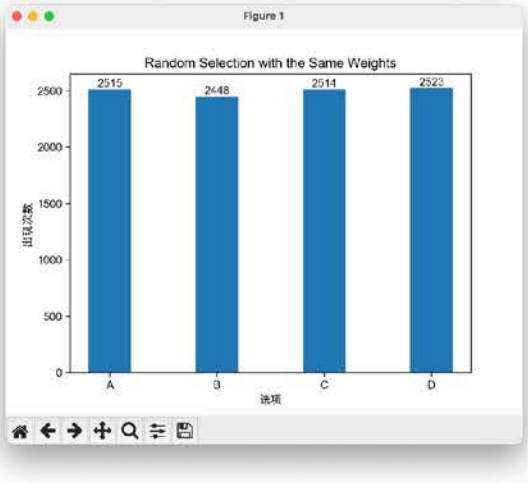
程序一

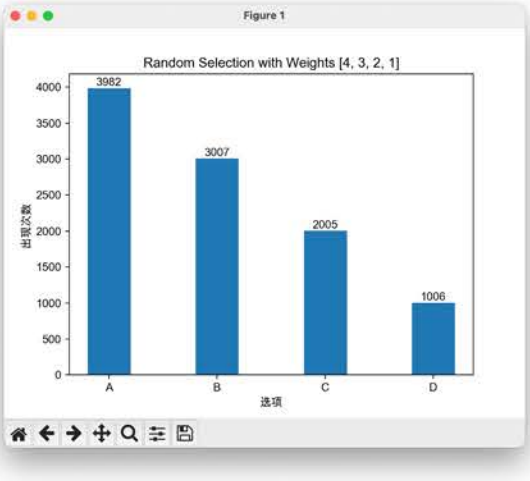
随机选择可能性不同的选项

random_with_weights.py random, dictionary, matplotlib

使用 random()、randint()、choice() 等随机函数在给定范围内进行选择时，所有选项都具有相同的可能性。如果选项被选中的可能性不同呢？程序给出了四种使用常见随机函数选择可能性不同的选项的方法。程序中使用“权重”（Weight）来表示可能性的大小，权重越大，被选中的可能性也越大。

程序对每种方各进行了 10000 次随机试验，用字典（dict）数据类型存储统计数据，并使用 Matplotlib 把结果绘制成条形统计图。从统计图中可以看出每个选项被随机选中的次数与它们被选中的可能性（权重）是一致的。



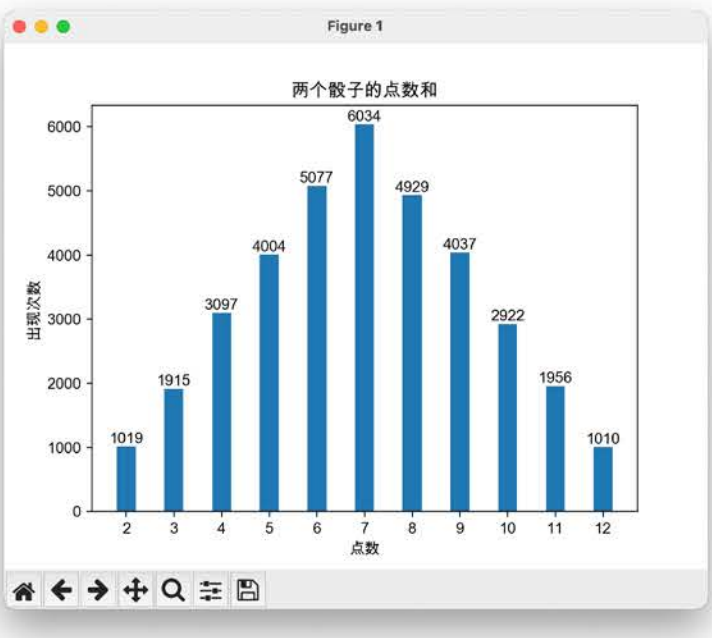


程序二

两个骰子的点数和

two_dice.py random, dictionary, matplotlib

两个骰子的点数和可能是 2 到 12，但各种结果出现的可能性是不同的。程序随机模拟了 36000 次掷出两个骰子，用字典存储统计数据，并使用 Matplotlib 把点数和的各种可能情况绘制成条形统计图。从统计图中可以看出掷出 7 的次数最多，大约 6000 次左右；掷出 2 和 12 的可能性最小，大约各 1000 次左右。



五年级 上册 5

简易方程

$8\cdot(5X-12)=24$

用字母表示数

用字母表示定律、公式

含字母的式子

方程是含有未知数的等式

等式的性质

1. 等式两边加上或减去同一个数，左右两边仍然相等

2. 等式两边乘同一个数，或除以同一个不为 0 的数，左右两边仍然相等。

方程的解

解方程

列方程解应用题

程序

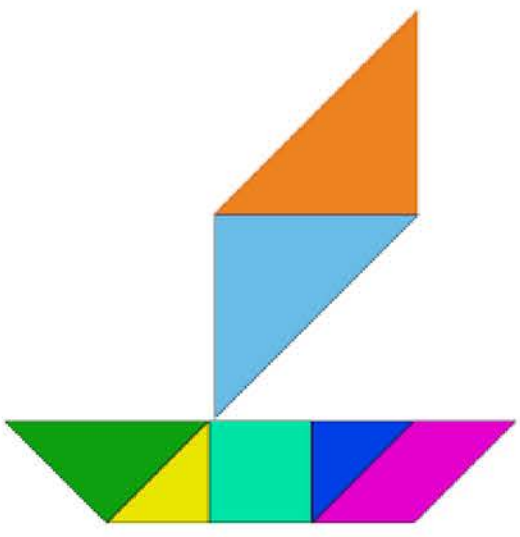
列方程求解鸡兔同笼问题

chicken_rabbit_equation.py string

程序根据用户输入的头和脚的数量，在屏幕上显示出列方程解应用题的全部步骤，包括：解、设、列方程、解方程、答等。输入头和脚的数量时，脚必须是偶数，且介于头数的二到四倍之间，如果输入不合理，会提示重新输入。

```
从上面数，有多少个头？ 36
从下面数，有多少只脚？ 126

解：设有 x 只兔子，则有 36 - x 只鸡。

     4x + 2·(36 - x) = 126
      4x + 2·36 - 2x = 126
             4x - 2x = 126 - 72
                  2x = 54
                   x = 54 / 2
                   x = 27

    鸡：36 - 27 = 9（只）

答：有 9 只鸡，27 只兔子。
```

五年级 上册 6
多边形的面积



平行四边形的面积 = 底 × 高

$$S = ah$$

三角形的面积 = 底 × 高 ÷ 2

$$S = ah \div 2$$

梯形的面积 =
(上底 + 下底) × 高 ÷ 2

$$S = (a + b)h \div 2$$

组合图形的面积

估算不规则图形的面积

程序

多边形类的面积属性

polygon_classes.py　　class

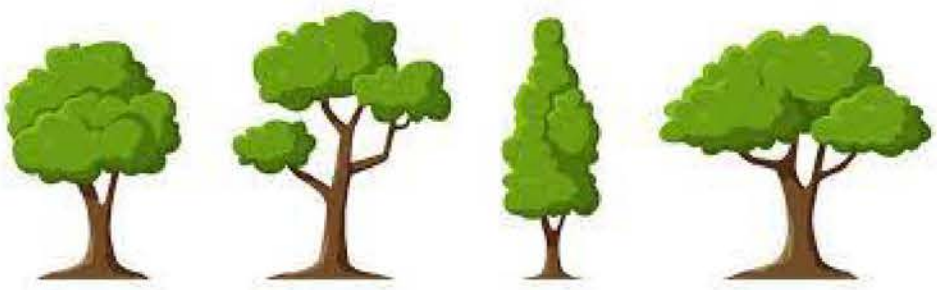
程序分别定义了平行四边形、三角形和梯形类，在每个类中都有一个面积属性 (area)。与一般的属性不同，area 的取值是由底和高等其它属性值决定的，其它属性值改变，area 的值也要随之改变，而且 area 也不允许被直接改写。像 area 这种特殊的属性，在 Python 中被称为 Property。在类中定义 Property 时，是在同名函数前加“@property” 装饰符。访问 Property 同访问其它属性一样，但对 Property 的访问在内部实际上是对函数的调用，所以可以对读、写进行更多控制。

程序中多边形类的 area 属性只能查询、不能改写（即只有 Getter 没有 Setter），在《长方体类》(G523) 的程序中将会见到设置了 Setter 的 Property。

```
>>> parallelogram = Parallelogram(4, 3)
>>> print(parallelogram.area)
12
>>>
>>> parallelogram.area = 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: can't set attribute 'area'
>>>
>>> parallelogram.base = 5
>>> print(parallelogram)
平行四边形
底：5
高：3
面积：15

>>>
```

五年级 上册 ⑦

数学广角—植树问题



树的棵数和间隔数的关系

直路（开放图形）

环路（封闭图形）

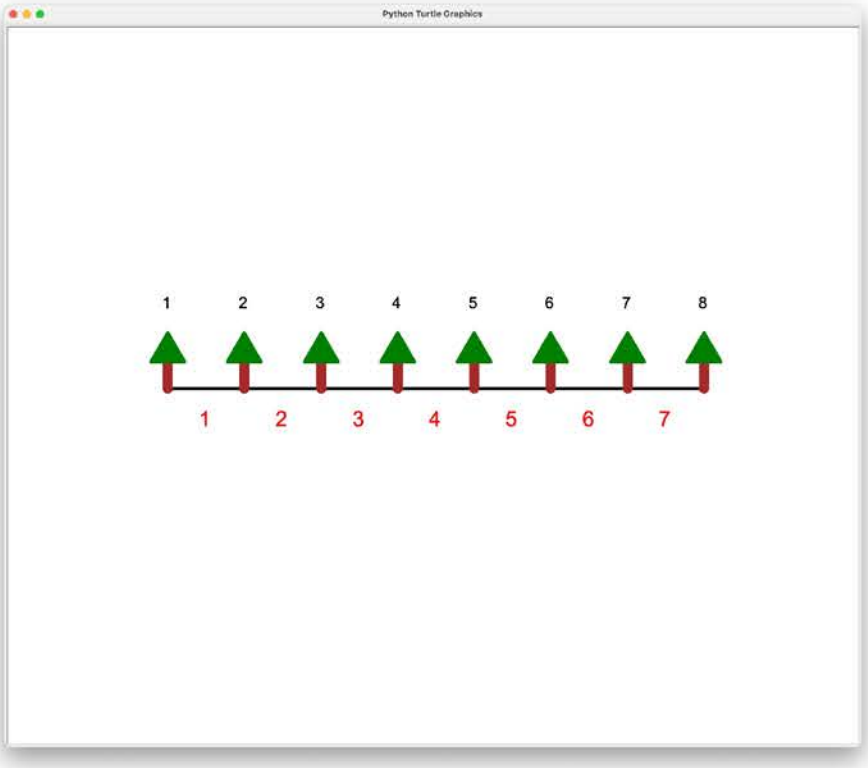
程序

植 树 问 题

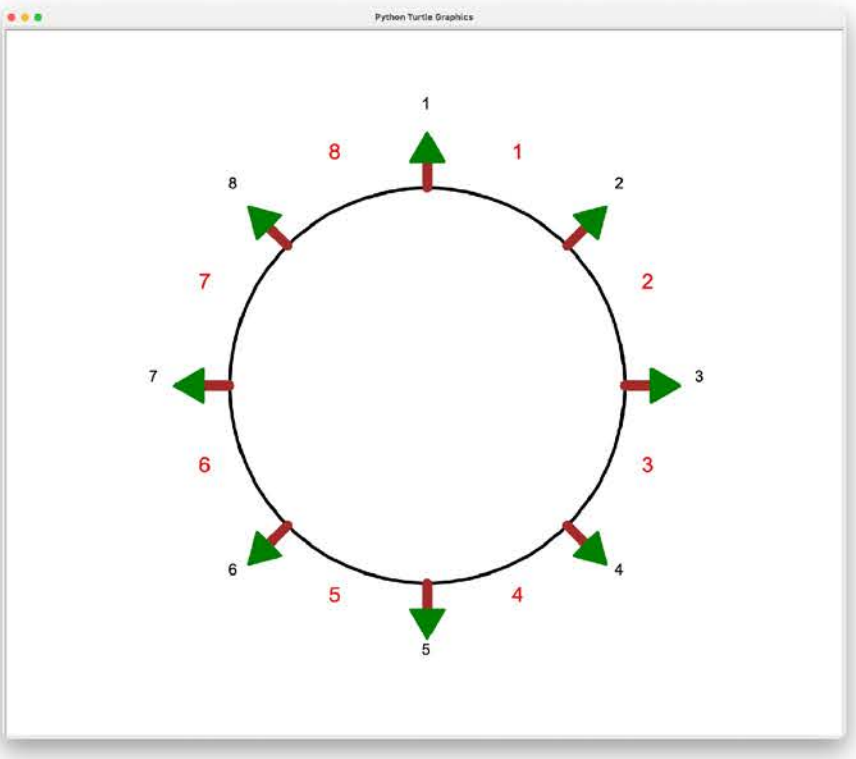
plant_trees.py　　turtle

用户输入植树的数量，并选择是直路还是环路（分别代表开放图形和封闭图形）。根据用户输入，程序用 Turtle 画出路和树，并在图中标出树的棵数和间隔数，方便学习者体会二者之间的关系。

```
植多少棵树 (2-20)? 8
是否是闭合的环路 (y/n)? n
```




```
植多少棵树 (2-20)? 8
是否是闭合的环路 (y/n)? y
```



五年级 下册 ①

观察物体（三）

由三视图推导几何体的可能摆放情况

程序

★★★★★

几何体三视图 v2

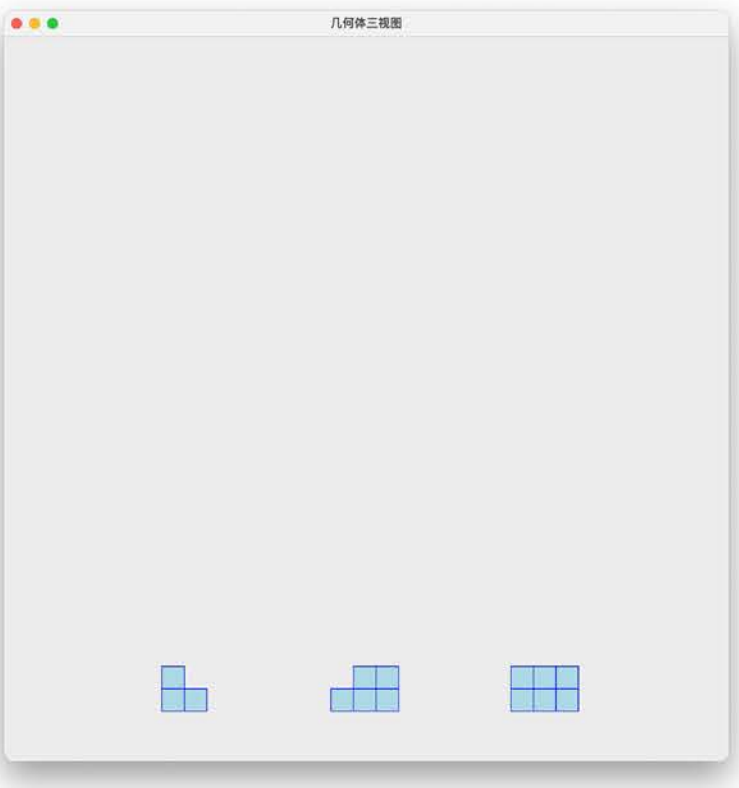
cubes_3view_v2.py

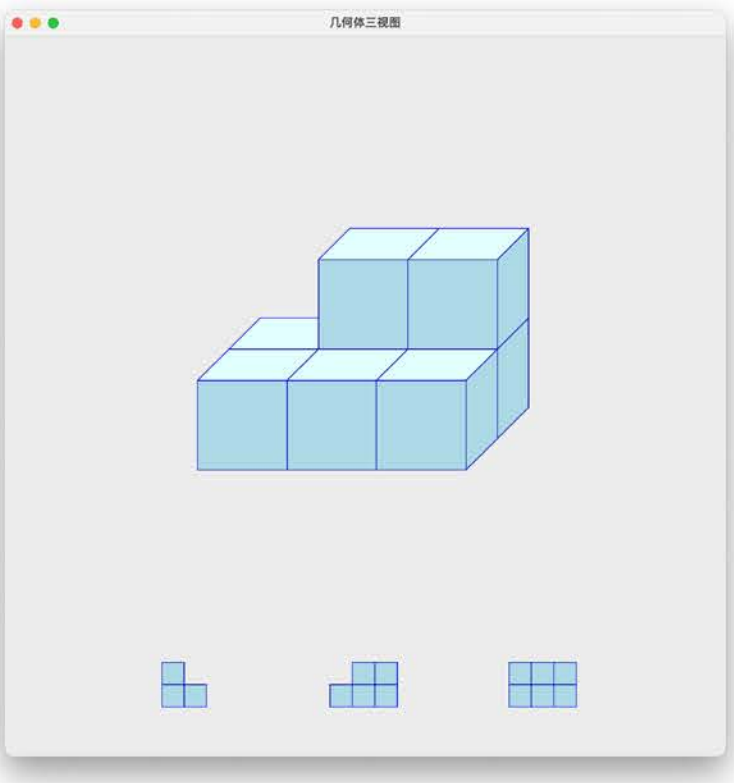
tkinter, coordinate, 2d list, random, class

程序随机生成由立方体积木块拼成的三维几何体，并在窗口下方显示该几何体的三视图，从左至右依次为：左视图、正视图、俯视图。按空格键在窗口正中显示该几何体，再按一次空格键重新生成新的几何体。

此程序是《几何体三视图》(G422) 程序的扩充。之前的程序主要锻炼由给定几何体绘制三视图的能力，所以首先显示几何体，按空格键后在窗口下方显示三视图。本单元的主要学习内容是由三视图推导几何体的可能摆放情况，所以程序首先在窗口下方显示三视图，学习者先思考一下几何体的空间构成（答案可能不唯一），然后按空格键在窗口正中显示几何体。

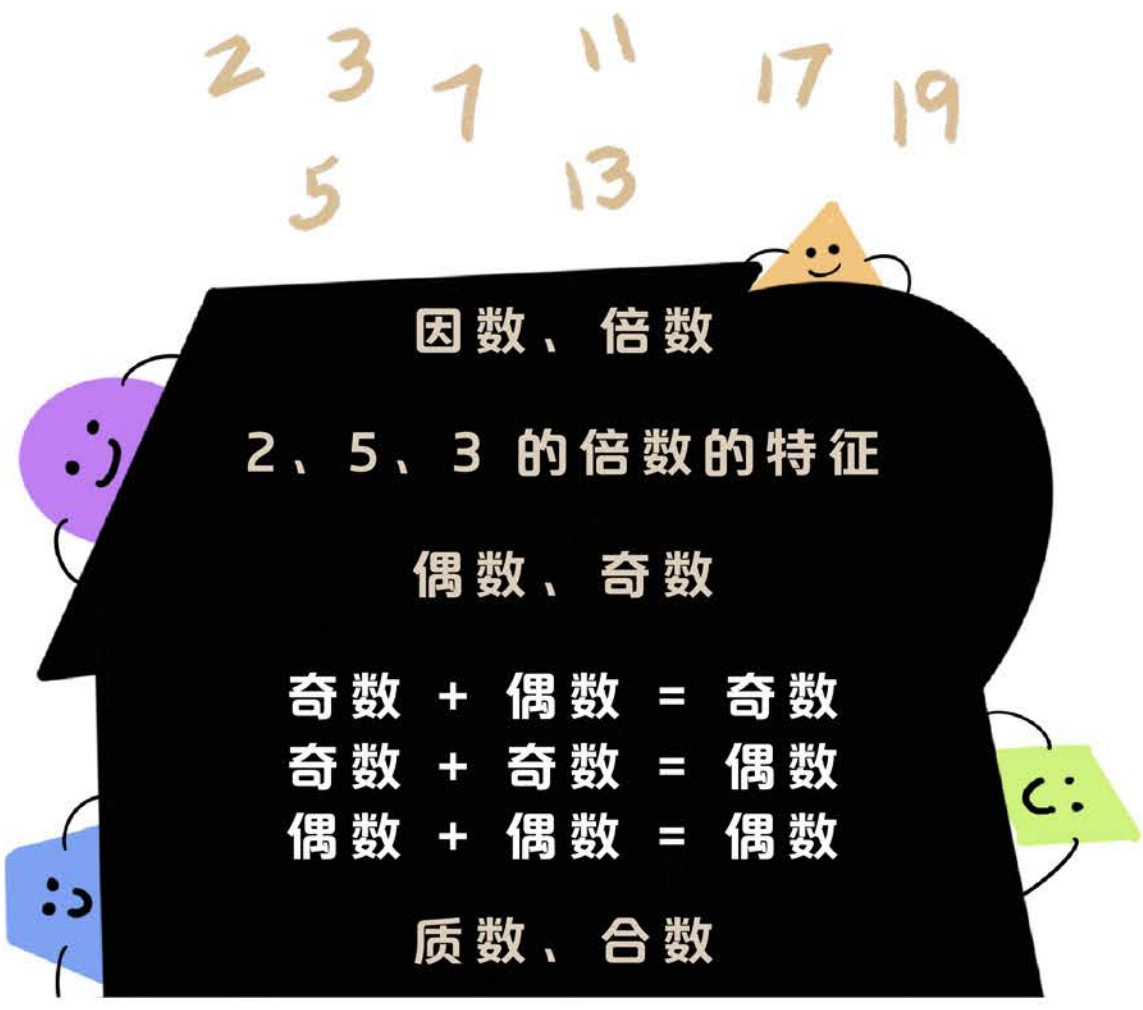
程序改进后，用户可以方便地在两种练习模式间进行切换。





五年级 下册 ②

因数与倍数



>_ 程序一

获取 n 以内的所有质数

get_prime_numbers.py　algorithm

程序中编写了与因数和质数有关的四个函数：

1. get_factors(n) 获取 n 的所有因数。采用循环依次检验小于 n 的数是否能整除 n，以判断该数是否是 n 的因数。从 1 循环到 n 肯定是可以的，能否减少一些不必要的循环呢？从 1 循环到 n/2 也是可以的，因为除了 n 自身以外，大于 n/2 的数肯定不能整除 n。再进一步，因数一般都是成对出现的，如果 i 是 n 的因数，那么 n/i 也是 n 的因数。以 36 的因数为例，1 和 36、2 和 18、3 和 12、 4 和 9，直到最后一个因数 6（6 的平方是 36，6 被称作 36 的平方根，在 Python 中用 sqrt(36) 表示）。换言之，找到一个小于 sqrt(n) 的因数，也就同时找到了一个大于 sqrt(n) 的因数，因此从 1 循环到 sqrt(n) 就可以找到 n 的全部因数。当 n 很大时，循环次数对程序运行时间的影响会非常显著。例如 n 为 1 亿时，n/2 是 5 千万，运行时间缩短一倍；sqrt(n) 是 1 万，运行时间缩短了几个数量级。

2. is_prime_number(n) 判断 n 是否为质数。质数只有 1 和自身两个因数，所以与寻找因数的函数类似，只要在 2 到 sqrt(n) 的范围内找不到 n 的因数，那么 n 就是质数；否则只要找到 1 个因数，就可以断定 n 不是质数，不需要等到循环全部结束。

3. 用两种方法获取 n 以内的所有质数。两种方法虽然可以获得同样的结果，但运行时间存在较大差异。通过对比，学习者可以体会算法对程序设计的影响。

1. get_prime_numbers_slow(n)。利用 is_prime_number() 函数依次检验每一个小于 n 的数是不是质数。这种方法简单直观，但运行时间较长；

2. get_prime_numbers(n)。采用了数学教材中寻找 100 以内质数的方法：在数表中（除去 1），先划掉所有 2 的倍数、再划掉所有 3 的倍数……一直划到 sqrt(n) 的倍数，剩下没有被划掉的数就是 n 以内的全部质数。之所以只需要划到 sqrt(n) 的倍数，是因为对于 sqrt(n) 的 k 倍，当 k < sqrt(n) 时，在前面划 k 的倍数时已经被划掉了；而当 k > sqrt(n) 时，sqrt(n) 的 k 倍又超出了 n 的范围。所以划完 sqrt(n) 的倍数，数表中所有是其它数倍数的数就都已经被划掉了。这种方法虽然没有第一种方法直观，运行时间却少得多。

```
100 的所有因数：[1, 2, 4, 5, 10, 20, 25, 50, 100]

100 以内的所有质数：[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

10000000 以内质数的个数：664579
较慢方法的运行时间：28.06s

10000000 以内质数的个数：664579
较快方法的运行时间：1.11s
```

>_ 程序二

哥德巴赫猜想

goldbach.py

哥德巴赫猜想是：任一大于 2 的偶数都可写成两个质数之和。对于用户输入的大于 2 的偶数，程序将其表示为两个质数之和。

```
输入一个大于 2 的偶数：98
98 = 19 + 79
```

五年级 下册 3

长方体和正方体

长方体、正方体（立方体）

8 个顶点、12 条棱、6 个面

长方体：长、宽、高，正方体：棱长

长方体和正方体的表面积

长方体的表面积 = (长 × 宽 + 长 × 高 + 宽 × 高) × 2

$$S = 2(ab + ah + bh)$$

正方体的表面积 = (棱长 × 棱长) × 6

$$S = 6a^2$$

体积、体积单位（立方厘米、立方分米、立方米）

长方体和正方体的体积

长方体的体积 = 长 × 宽 × 高

$$V = abh$$

正方体的体积 = 棱长 × 棱长 × 棱长

$$V = a^3$$

底面积

长方体（或正方体）的体积 = 底面积 × 高

$$V = Sh$$

体积单位间的进率

1 立方分米 = 1000 立方厘米

1 立方米 = 1000 立方分米

容积、容积单位（升、毫升）

1 升 = 1000 毫升

1 升 = 1 立方分米

1 毫升 = 1 立方厘米

用排水法测量不规则物体的体积

程序一

带有单位属性的长方体类

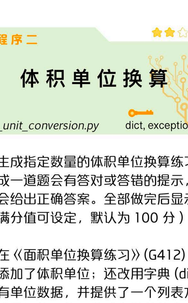
cuboid.py — class, exception, decimal

类似于《多边形类的面积属性》(G516)，长方体类的表面积和体积属性，也是取值随长、宽、高变化，且不能被直接改写的 Property。Property 的定义可以包含两个函数，查询属性值的函数称为 Getter，改写属性值的函数称为 Setter。长方体类的表面积和体积属性只有 Getter 没有 Setter，所以只能查询不能改写。长方体类的单位属性 unit 存储棱长的单位（例如 “m”），当 unit 改变时，长、宽、高也要做出相应的改变，这只能在 unit 的 Setter 函数中完成。unit 的 Getter 和 Setter 的函数名同为 unit，定义函数时，在 Getter 前面加 “@property” 装饰符，在 Getter 前面加 “@unit.setter” 装饰符。

此程序中长方体类的长、宽、高也是 Property，这主要是为了在它们的 Setter 中把用户输入的 int 或 float 类型数据转换为 Decimal 类型，以在小数计算中获得表面积和体积的精确值。

在创建长方体对象设置棱长时，可以给出 0、1 或 3 个参数，分别对应棱长为 1 的正方体对象、棱长为给定参数的正方体对象和长、宽、高为给定参数的长方体对象。如果给出的棱长参数数量不是 0、1 或 3，或者参数值不是一个有效的数（例如字符串或负数），因为不是语法错误，程序可以通过语法检查，但会在运行阶段出现问题，这类问题被称为 Exception (异常)。Exception 可以被程序捕捉和处理，未经处理的 Exception 会终止程序并在屏幕上显示错误信息。Python 中有很多内置的 Exception 类型，也可以在程序中自定义新的类型，并在需要时用 raise 语句人为抛出 Exception。

程序中自定义了一个 “参数数量有误” 的 Exception。当棱长参数或各属性设置出现问题时，会抛出相应的内置或自定义的 Exception 终止程序并显示错误信息。



程序二

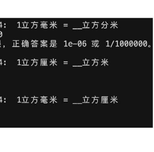
体积单位换算

volume_unit_conversion.py — dict, exception, random

随机生成指定数量的体积单位换算练习题。每完成一道题会有答对或答错的提示，答错时还会给出正确答案。全部做完后显示总得分（满分值可设定，默认为 100 分）。

程序在《面积单位换算练习》(G412) 的基础上添加了体积单位：还改用字典 (dict) 存储所有单位数据，并提供了一个列表方便学习者设置要练习的一种或多种单位类型，列表中默认只有体积单位。

在由较小单位向较大单位换算时，可以输入小数或分数，例如 1 cm^3 = 0.0001 m^3 或 1 cm^3 = 1/10000 m^3。当要输入的数位数较多时，可以采用科学计数法。例如 1e3 代表 1 后面有三个零，即 1000；1e-3 代表 1/(1e3)，即 1/1000 或 0.001。



五年级 下册 4

分数的意义和性质

分数的意义

单位 “1”、分数单位

分数与除法 $a \div b = \frac{a}{b}\ (b \neq 0)$

真分数和假分数，
假分数可以转化为整数或带分数

分解质因数、短除法

分数的基本性质：分数的分子和分母同时乘或除以相同的数（0 除外），分数的大小不变。

公因数、最大公因数、互质

约分、最简分数、分数化简

公倍数、最小公倍数

通分、异分母分数比较大小

分数和小数

分数和小数的互化

程序一

最大公因数和最小公倍数

gcd_lcm.py　　algorithm

程序用两种方法求两个数的最大公因数（较大的数是 a，较小的数是 b）。

第一种方法是采用循环依次检验比 b 小的数是不是两数的公因数，并找到所有公因数中最大的一个。优化的循环次数是从 1 到 sqrt(b)，类似于 get_prime_numbers.py (G522) 中的 get_factors() 函数。但即便采用优化的循环次数，求解大数时程序的运行时间依旧较长。

第二种方法是采用欧几里得算法，又称辗转相除法。算法基于如下原理：两个整数的最大公因数等于其中较小的数和两数相除余数的最大公因数。算法如下：
1. 把两个数做除法，大的是被除数，小的是除数，为求被除数和除数的最大公因数，现在只要求除数和余数的最大公因数即可；
2. 然后再把除数和余数当做新的被除数和除数做除法，重复第一步的过程；
3. 直到某一步的余数为 0，则当时的除数即为所求的最大公因数。

欧几里得算法可以使两个大数快速变小，所以用于求最大公因数效果极好。通过对比，学习者可以体会算法对程序设计的影响。

由于两个数的乘积等于它们最大公因数和最小公倍数乘积，所以程序还利用求最大公因数的函数，实现了求最小公倍数的函数。

```
1219326309891784 80 和 1219326311126352 69 的最大公因数是 123456789。
较慢方法的运行时间： 13.52s

1219326309891784 80 和 1219326311126352 69 的最大公因数是 123456789。
较快方法的运行时间： 0.0s

36 和 48 的最小公倍数是 144。
```

程序二

小数转化为最简分数

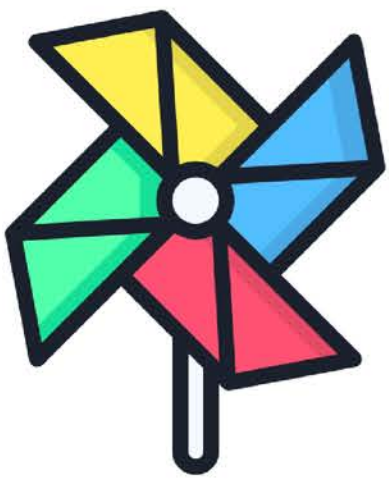
decimal_to_fraction.py　　string

用户输入一个小数，程序将其转化为最简分数并显示在屏幕上。分数化简时使用了程序 1 中求最大公因数的函数得到分子和分母的最大公因数，再进行约分。

```
输入一个小数： 0.48
0.48 = 12/25
```

五年级 下册 5

图形的运动（三）



图形绕一点顺时针或逆时针旋转

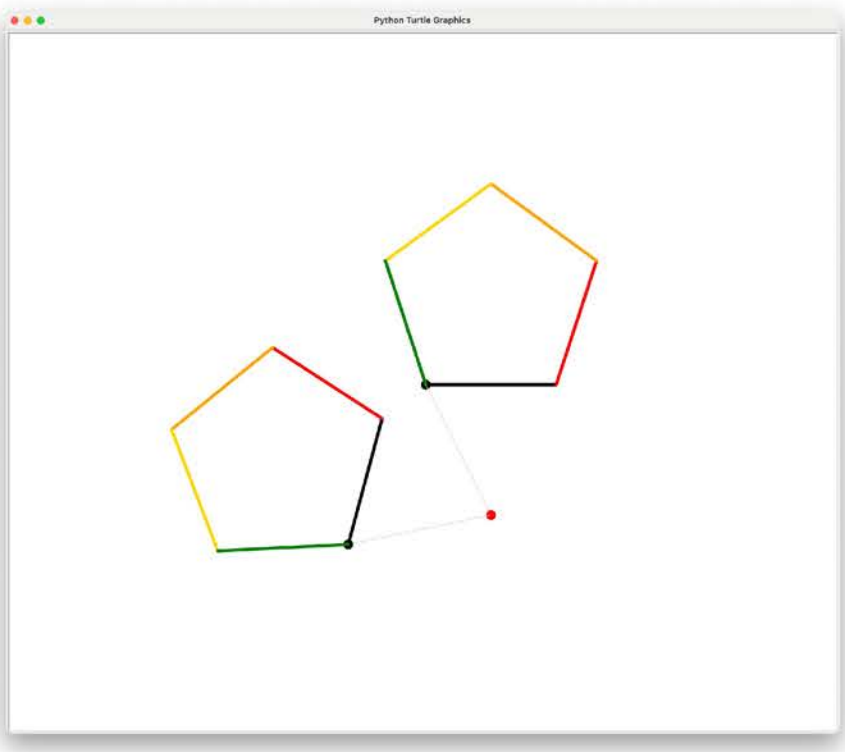
程序

★★☆☆☆

图形绕一点旋转

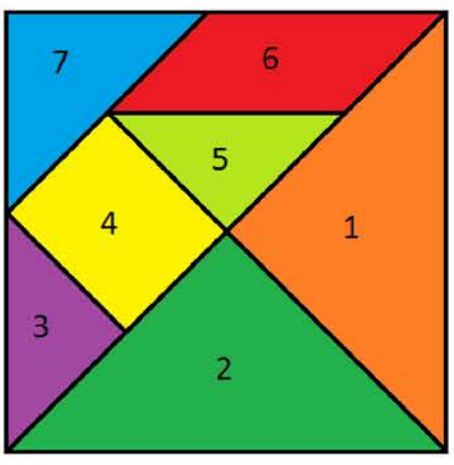
rotation.py　turtle, coordinate, random

程序随机生成一个正多边形，然后将图形按给定的旋转中心和旋转角度进行旋转，并画出旋转后的图形。为了方便区分图形旋转前后的对应边，程序为图形的每条边设置了不同的颜色。



五年级 下册 6

分数的加法和减法



$$\frac{1}{4}+\frac{1}{4}+\frac{1}{16}+\frac{1}{8}+\frac{1}{16}+\frac{1}{8}+\frac{1}{8}=1$$

同分母分数加、减法

异分母分数加、减法

分数加减混合运算

利用加法运算定律
简便计算

程序

★★★☆☆

分 数 加 减 法

add_sub_fractions.py　　string

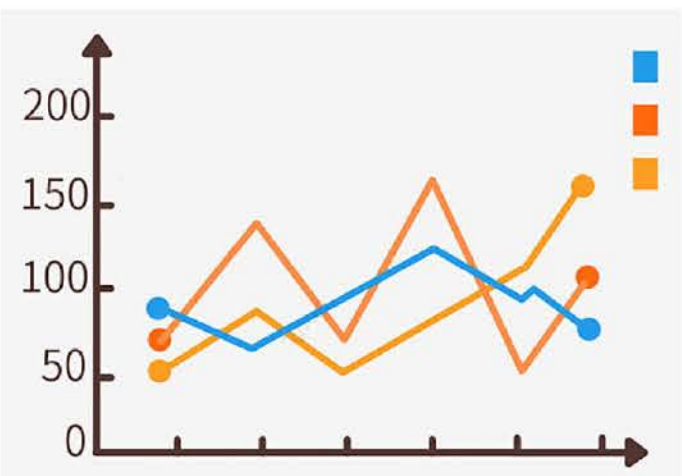
程序开始时，提示用户输入两个分数（也可以输入整数），并选择是做加法还是减法，之后程序会将计算结果以最简分数的形式显示在屏幕上。

在计算分数加减法时，程序调用了《最大公因数和最小公倍数》(G524) 程序中求最小公倍数的函数，先将异分母分数通分为同分母分数。完成同分母分数加减法后，再调用《小数转化为最简分数》(G524) 程序中分数化简的函数，将计算结果约分为最简分数。

```
输入第一个分数（或整数）：1/3
输入第二个分数（或整数）：1/6
选择做加法还是减法 (+ 或 -)? +
1/3 + 1/6 = 1/2

输入第一个分数（或整数）：3/5
输入第二个分数（或整数）：1
选择做加法还是减法 (+ 或 -)? -
1 - 3/5 = 2/5
```

五年级 下册 7

折线统计图



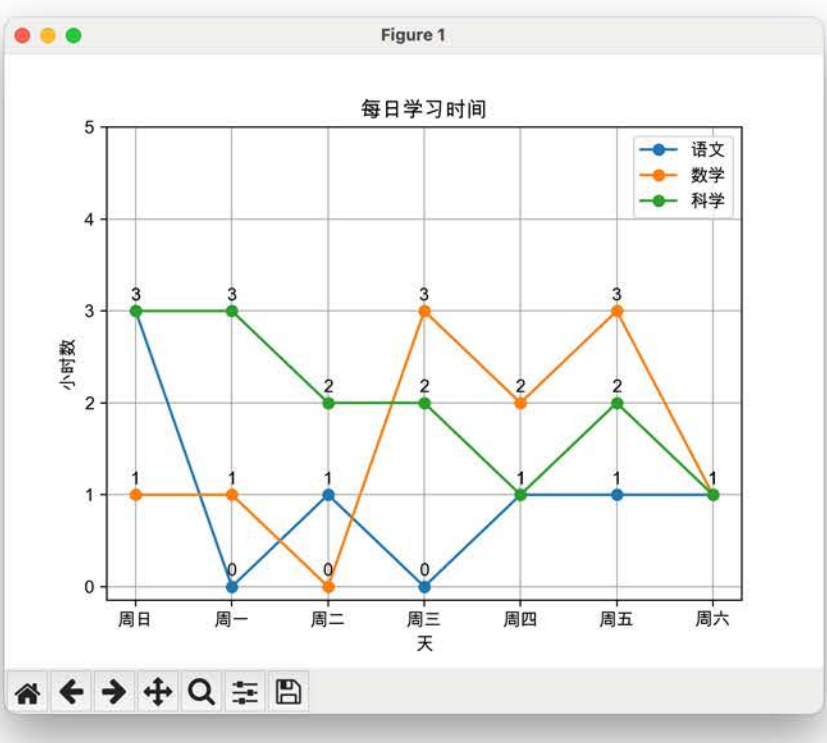
程序

data3.py

matplotlib, class, 2d list

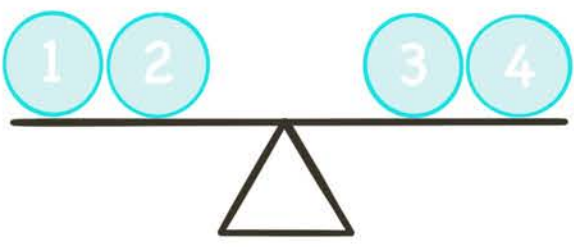
程序在 data2.py (G428) 的基础上，向 Data 类中添加了绘制复式折线统计图的 line() 方法。除新添加的方法外，Data 类的其它代码保持不变。

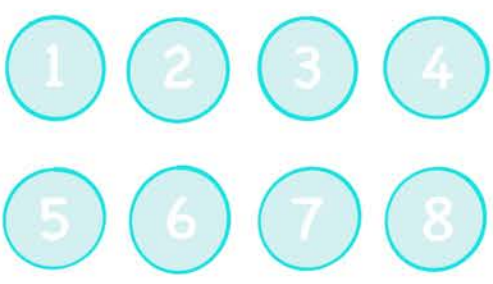
	周日	周一	周二	周三	周四	周五	周六
语文	3	0	1	0	1	1	1
数学	1	1	0	3	2	3	1
科学	3	3	2	2	1	2	1



五年级 下册 8

数学广角 — 找次品





用最少次数找次品的方法

求出最少称量次数

分情况列出找次品的过程

程序

找 次 品

identify_outlier.py recursion, algorithm, string

用户首先输入产品数量，程序在屏幕上显示从这些产品中寻找次品的全过程。程序的难点在于每次称量都有平衡和不平衡两种情况，在每种情况下再次称量又有平衡和不平衡两种情况……，所以整个过程呈现出嵌套的、树杈状的结构。这与我们常见的顺序、条件、循环或调用函数的程序流程（语句的执行顺序）都不一样。程序中使用了“递归”（Recursion），递归是一种在函数中调用自身的结构，可以实现复杂的程序流程，将找次品的全过程重复分解为规模更小的相同子过程。使用递归时一定要在递归函数中设置终止条件，否则函数会无休止的调用自身直到超出最大允许的递归深度（Python 默认为 1000）。

```
从多少件产品中找次品？ 10
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

第 1 次：[1, 2, 3] vs [4, 5, 6]
如果平衡：
....第 2 次：[7] vs [8]
....如果平衡：
........第 3 次：[9] vs [10]
........有问题的是次品。
....如果不平衡，有问题的是次品。
如果不平衡，假设 [1, 2, 3] 有问题：
....第 2 次：[1] vs [2]
....如果平衡，3 是次品。
....如果不平衡，有问题的是次品。

至少称 3 次，肯定能找出次品。
```

致谢

所学隔山海，山海皆可平

感谢小续在整个学习项目中的积极配合，项目中的代码和注释是由我和他共同完成的。

感谢在各学科领域指导或帮助过我的师长和朋友，特别是在数学和编程上帮我启蒙的爸爸。

感谢我的妈妈提供本手册插图的绘制。

联系方式：math-coding@hotmail.com

GitHub 代码库：https://github.com/feli10/math-coding