

**Python程序设计**

**小海龟爱数学-研究生编程作业**

姓 名： 刘博文

学 号： M202221325

班 级： 协创硕2203

任课教师： 霍佳皓

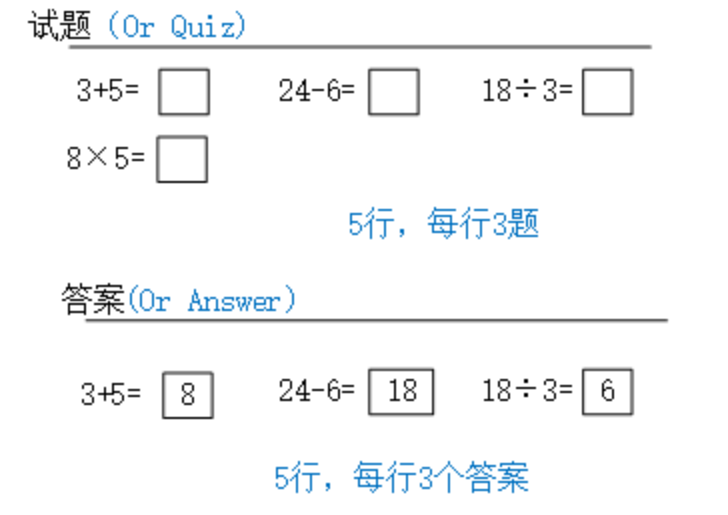
2023年 3 月18日

# 代码设计思路

本次作业的程序可以总结为以下两个部分：可视化界面部分和出题逻辑实现部分。

## 界面设计部分

首先是界面设计的思路：



**图1.1 CG系统上该作业的示例**

首先界面分为上半试题部分和下半答案部分，这两个部分大致相同。在工作开始前，我选择固定了画布的大小为800\*800，这样做的好处是可以保证每次打开程序的时候外边界是固定的，这样保证了不同用户在不同分辨率的电脑上打开后展示的大小和效果是一致的。

接着，对其内部各个部分进行设计：

1、首先，对画布的左上角试题两字的标题进行设计。我经过一些尝试找到了一个合适的坐标(-330,350)，并使用turtle的write函数打印该标题。然后，标题的下方有一条横线，这里坐标的选择是距离标题下方10个像素间距，即(-330,340)的位置，然后使用turtle的forward函数向前移动650个像素，从而绘制了一条横线。

2、接下来，对题目的出现位置进行设计。首先考虑第一个题目的出现位置，我经过尝试，发现在标题横线下方(-320,300)的位置打印第一个题目相较美观，使用pu()函数抬起画笔，goto()函数移动到这个坐标，然后使用write()函数打印出题目，题目一共5行3列，同一行的题目之间的间距为100像素，不同行之间的题目间距为50像素，共计15道题目，对于题目部分如何隐藏答案，在后面会有详细说明。

3、下半答案部分界面设计和上半部分的界面逻辑是一致的，只需要改标题即可，此处不再赘述。

## 出题逻辑实现部分

根据题目要求，需要随机给出15道试题，我最开始的想法是通过Python的字典来存储这些题目，其中字典的键为题目信息，值为答案，但后来发现这个思路在后续会出现问题，因为在下半显示答案的部分，还需要将题目都按顺序打印出来，可是字典的遍历不能保证上半试题部分和下半答案部分的遍历顺序一样，因为字典是无序的。

于是，我将所有题目的集合存在了一个列表中，这样能保证两次遍历列表的顺序一致，其中列表的每个元素是题目和答案组成的一个字符串，举例来说：question\_list[0] = "1+1=2"，当然这个字符串的保存进行了额外的设计，它的实际组成为"%2d"%(num1) + operation + "%2d = "%(num2) + "%2d" %(ans)，即由num1、符号、num2、等号、答案五个部分拼接组成，其中每一个数使用%2d即两个数据宽度进行格式化处理，保证数据宽度的一致性，这样能够使题目上下左右都是对齐的，而且对我在每个题目等号右侧的固定间距画方框也起到帮助。

下面要介绍我是如何生成其中每一个题目的，首先符号有加、减、乘、除四种，由于需要限制**运算过程和结果均为两位以内的整数，**而不同符号形成答案范围的逻辑是不同的，因此我对这四种操作进行了分别的逻辑实现，确保每一种符号产生的过程和结果都不会越界。

1、加法部分：

首先num1使用random.randint(0, 99)生成0-99区间内的整数，而为了保证num2+num1不产生越界，因此num2的范围是(0, 99 - num1)，这样进行相加，可以使相加的结果最多为99，而不产生越界。

2、减法部分：

首先num1使用random.randint(0, 99)生成0-99区间内的整数，num2的范围为(0, num1)，这样num1-num2的结果最低为0，不产生越界。

3、乘法部分：

首先num1使用random.randint(1, 99)生成1-99区间内的整数，num2的范围为(0, 99 // num1)，这样两者相乘的结果最大为99，不会产生越界。

4、除法部分：

除数num2设计为(1, 99)内的随机数，被除数num1为num2的具有上限的整数倍，因为是整数倍关系，所以num1除以num2得到的一定还是整数，即num1 = num2 \*倍数。当然这会产生一个新问题，num1有可能会超过99，因此对倍数加一个上限限制，这个上限为99 // num2，展开来说，即num1 = num2 \* random.randint(1, 99 // num2)，这样既保证了被除数，除数和商都是整数，也保证了被除数不会超过99，满足题目要求。

# 关键代码解释

首先是每个题目右侧的都有一个方框，这里是用turtle连续转4次90度并前进一段距离来进行实现：

1. **def** draw\_box():
2. t.pendown()
3. **for** \_ **in** range(4):
4. t.forward(30)
5. t.left(90)
6. t.penup()
7. t.forward(30)

生成题目列表，使用random.choice()函数选择符号，对每个符号生成题目进行额外函数封装来进行分别的逻辑处理，这些函数返回的结果是包含题目和答案组成的字符串(在代码设计部分已经说明)，并赋给question保存，每次向题目列表添加元素，判断question是否存在列表中，不存在则添加，存在则重新生成，直至列表长度包含了15道题目。

1. **def** generate\_question\_list():
2. # 为了保持顺序，必须用列表类型
3. question\_list = []
4. **while** len(question\_list) < 15:
5. operation = random.choice(["+", "-", "×", "÷"])
6. **if** operation == "+":
7. question = generate\_addition\_question()
8. **elif** operation == "-":
9. question = generate\_subtraction\_question()
10. **elif** operation == "×":
11. question = generate\_multiplication\_question()
12. **else**:
13. question = generate\_division\_question()
14. **if** question **not** **in** question\_list:
15. question\_list.append(question)
16. **return** question\_list

对不同符号产生的题目逻辑进行分别实现，num1和num2的赋值的思路在前面代码设计部分已经详细说明，其中generate\_question()函数是根据传入的参数来拼接成题目字符串，并返回给question。

1. **def** generate\_addition\_question():
2. num1 = random.randint(0, 99)
3. num2 = random.randint(0, 99 - num1)
4. ans = num1 + num2
5. question = generate\_question("+", num1, num2, ans)
6. **return** question
8. **def** generate\_subtraction\_question():
9. num1 = random.randint(0, 99)
10. num2 = random.randint(0, num1)
11. ans = num1 - num2
12. question = generate\_question("-", num1, num2, ans)
13. **return** question
15. **def** generate\_multiplication\_question():
16. num1 = random.randint(1, 99)
17. num2 = random.randint(0, 99 // num1)
18. ans = num1 \* num2
19. question = generate\_question("×", num1, num2, ans)
20. **return** question
22. **def** generate\_division\_question():
23. num2 = random.randint(1, 99)
24. num1 = num2 \* random.randint(1, 99 // num2)
25. ans = num1 // num2
26. question = generate\_question("÷", num1, num2, ans)
27. **return** question

generate\_question()函数将operation, num1, num2, ans四个传入的参数拼接成格式化字符串，然后将生成的题目字符串进行返回。

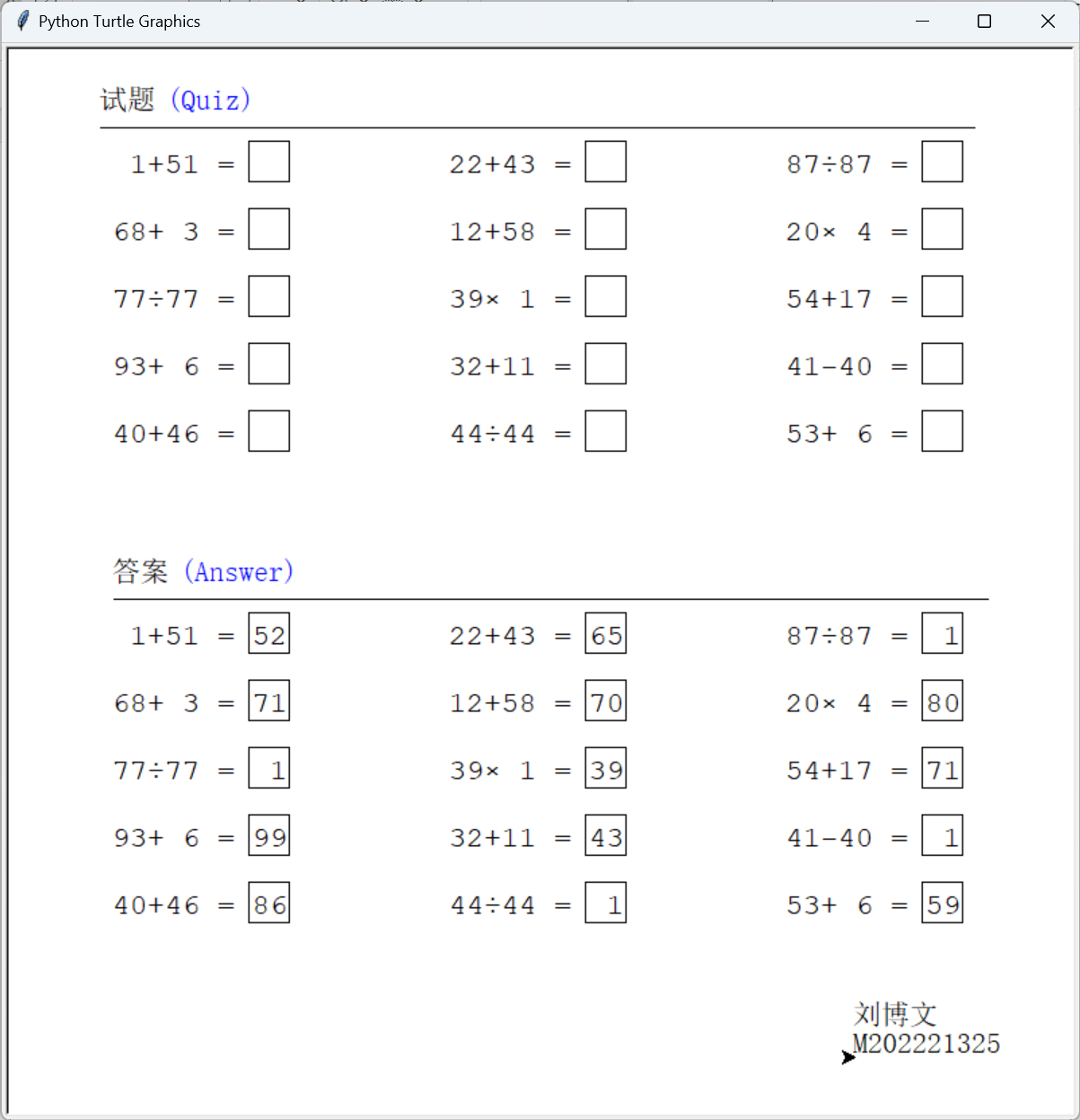
1. **def** generate\_question(operation, num1, num2, ans):
2. question = "%2d"%(num1) + operation + "%2d = "%(num2) + "%2d" %(ans)
3. **return** question

主函数中打印各个题目，逻辑是先生成题目列表，然后使用for循环配合迭代器进行逐个打印，并且在题目右侧偏移100个像素的位置画一个方框，每打印一个题目，需对下一个题目的x和y坐标产生适当的偏移，题目的字体设置为Courier New，这是一种等宽字体，确保题目之间以及题目和右侧方框的相对位置都是固定的。

1. question\_list = generate\_question\_list()
2. it1 = iter(question\_list)
3. it2 = iter(question\_list)
4. x,y = -320,300
5. **for** i **in** range(5):
6. **for** j **in** range(3):
7. t.pu()
8. t.goto(x,y)
9. t.write(next(it1)[:-2],font=("Courier New", 16, "normal"))
10. x+=100
11. t.goto(x,y)
12. draw\_box()
13. x+=150
14. x = -320
15. y -= 50

另外需要说明的是next(it1)[:-2]表示往后迭代一个题目字符串进行打印，且将最后两位舍去(答案的数据宽度一定为2，详见之前生成题目字符串的说明)，这样就可以做到隐藏答案的效果，而当下半部分给出各个题目答案的时候，仅需要写这样一段相似的代码，迭代器改成it2使得重新迭代各个题目，且去掉最后的切片[:-2]，就可以将答案进行展示。

# 效果图



# 附录

1. '''''
2. Author: 刘博文
3. Date: 2023-03-17 22:03:41
4. LastEditTime: 2023-03-18 23:29:40
5. FilePath: \turtle\Turtle\_math.py
6. Description:
7. https://github.com/touchfisher
8. Copyright (c) 2023 by ${git\_name\_email}, All Rights Reserved.
9. '''
10. **import** turtle as t
11. **import** random
13. **def** draw\_box():
14. t.pendown()
15. **for** \_ **in** range(4):
16. t.forward(30)
17. t.left(90)
18. t.penup()
19. t.forward(30)
21. **def** generate\_question(operation, num1, num2, ans):
22. question = "%2d"%(num1) + operation + "%2d = "%(num2) + "%2d" %(ans)
23. **return** question
25. **def** generate\_addition\_question():
26. num1 = random.randint(0, 99)
27. num2 = random.randint(0, 99 - num1)
28. ans = num1 + num2
29. question = generate\_question("+", num1, num2, ans)
30. **return** question
32. **def** generate\_subtraction\_question():
33. num1 = random.randint(0, 99)
34. num2 = random.randint(0, num1)
35. ans = num1 - num2
36. question = generate\_question("-", num1, num2, ans)
37. **return** question
39. **def** generate\_multiplication\_question():
40. num1 = random.randint(1, 99)
41. num2 = random.randint(0, 99 // num1)
42. ans = num1 \* num2
43. question = generate\_question("×", num1, num2, ans)
44. **return** question
46. **def** generate\_division\_question():
47. num2 = random.randint(1, 99)
48. num1 = num2 \* random.randint(1, 99 // num2)
49. ans = num1 // num2
50. question = generate\_question("÷", num1, num2, ans)
51. **return** question
53. **def** generate\_question\_list():
54. # 为了保持顺序，必须用列表类型
55. question\_list = []
56. **while** len(question\_list) < 15:
57. operation = random.choice(["+", "-", "×", "÷"])
58. **if** operation == "+":
59. question = generate\_addition\_question()
60. **elif** operation == "-":
61. question = generate\_subtraction\_question()
62. **elif** operation == "×":
63. question = generate\_multiplication\_question()
64. **else**:
65. question = generate\_division\_question()
66. **if** question **not** **in** question\_list:
67. question\_list.append(question)
68. **return** question\_list
70. **if** \_\_name\_\_ == "\_\_main\_\_":
71. # 设置画布大小和画笔速度
72. t.setup(800,800)
73. t.speed(0)
75. # 移动初始位置到左上角
76. t.pu()
77. t.goto(-330,350)
79. # 写标题
80. t.write("试题",font=("宋体",16,"normal"))
81. t.fd(50)
82. t.pencolor('blue')
83. t.write("(Quiz)",font=("宋体",16,"normal"))
85. # 画标题线
86. t.goto(-330,340)
87. t.pencolor('black')
88. t.pd()
89. t.fd(650)
91. question\_list = generate\_question\_list()
92. it1 = iter(question\_list)
93. it2 = iter(question\_list)
94. x,y = -320,300
95. **for** i **in** range(5):
96. **for** j **in** range(3):
97. t.pu()
98. t.goto(x,y)
99. t.write(next(it1)[:-2],font=("Courier New", 16, "normal"))
100. x+=100
101. t.goto(x,y)
102. draw\_box()
103. x+=150
104. x = -320
105. y -= 50
107. # 写答案标题
108. y-=50
109. t.pu()
110. t.goto(x,y)
111. t.write("答案",font=("宋体",16,"normal"))
112. t.fd(50)
113. t.pencolor('blue')
114. t.write("(Answer)",font=("宋体",16,"normal"))
116. # 画答案标题线
117. y-=10
118. t.goto(x,y)
119. t.pencolor('black')
120. t.pd()
121. t.fd(650)
123. # 答案
124. x=-320
125. y-=40
126. **for** i **in** range(5):
127. **for** j **in** range(3):
128. t.pu()
129. t.goto(x,y)
130. t.write(next(it2),font=("Courier New", 16, "normal"))
131. x+=100
132. t.goto(x,y)
133. draw\_box()
134. x+=150
135. x = -320
136. y -= 50
138. x += 550
139. y -= 50
140. t.pu()
141. t.goto(x,y)
142. t.write("刘博文\nM202221325",font=("宋体",16,"normal"))
143. t.done()