花式钮扣和十字架

作业 1

2023 年第 2 学期

CSSE1001/CSSE7030

截止日期：2023 年 8 月 25 日 15:00
GMT+10

1 导言

在本作业中，您将受该演示启发，实施一个基于文本的改良型九宫格游戏。该游戏与普通的九宫格游戏类似，但不同之处在于每个玩家都有不同大小的标记。游戏在两个玩家之间的 3 × 3 网格上进行。游戏开始时，双方轮流将自己的标记放在棋盘上的单元格中。玩家可以将任意标记放在空格上，也可以将较大的标记放在棋盘上已有的较小标记上。最上面的标记被认为占据了格子。当出现以下情况之一时，游戏结束

1. 一名玩家用自己的标记填满一行、一列或对角线即可*获胜*（注意，只有最顶端的标记才算获胜）。
2. 当棋盘上无法走棋，但没有棋手满足获胜条件时，对局进入*僵局*。

2 入门

从 Blackboard 下载 a1.zip - 该压缩包包含开始本作业所需的文件。解压缩后，a1.zip 压缩包将提供以下文件：

a1.py *这是您要提交*的*唯一文件*，也是您编写代码的地方。*请勿*更改任何其他文件。

constants.py *请勿修改或提交此文件*，它包含作业中使用的预定义常量。除此以外，我们鼓励您尽可能在 a1.py 中创建自己的常量。

gameplay/ 该文件夹包含一些通过使用功能完善的作业解决方案玩游戏生成的输出示例。

此文件夹中文件的目的是帮助您了解游戏的运行方式以及输出的格式。

注意：不允许在 a1.py 中添加任何 import 语句。否则将被扣最高 100%的分数。

3 游戏玩法

本节概述了游戏玩法。本节未明确提及的提示和输出，请参见第 4 节和随本作业提供的游戏/文件夹中的示例游戏。

游戏开始时，棋盘是空的，双方都有所有可用的棋子。无
(O）先下一步棋。游戏结束前，重复以下步骤：

1. 显示当前的棋局状态（包括每位棋手剩余的棋子和当前的棋盘状态）。
2. 用户会被告知轮到谁移动。
3. 提示用户移动。在此提示下，用户可以输入格式为"*{行} {列} {尺寸}*"的有效移动。*{大小}*"*的格式输入有效的移动，*或者输入 "h "或 "H "要求查看帮助信息。如果用户输入 "h "或 "H"，则应在重新提示移动之前向其显示帮助信息。否则，进入下一步。
4. 如果移动因任何原因无效，则向用户显示一条信息，告知其移动无效的原因（请参阅表 1，了解本步骤所需的所有有效性检查和信息），然后返回步骤 3。如果移动有效，则进入下一步。
5. 根据请求的移动更新棋盘，并显示新的棋局状态。
6. 如果游戏尚未结束，请返回第 2 步。
7. 如果游戏结束，则告知用户结果，并提示他们是否愿意再玩一次。在此提示下，如果用户输入 "y "或 "Y"，则创建一个新游戏（即设置一个空棋盘并将所有棋子还给玩家），然后返回步骤 1。如果棋手输入的不是 "y "或 "Y"，则优雅地终止程序。

请注意，我们不会测试一方无法下棋而另一方可以下棋的情况；也就是说，您不需要处理当前棋手无棋可下但下一个棋手可以下棋的情况。

4 实施情况

本节概述了**要求**您在解决方案中实现的函数（仅限 a1.py）。您将根据您的函数在*独立*测试时通过的测试次数获得分数。因此，如果作业不完整，但其中*的一些*函数可以正常工作，那么它的分数可能会比完整但函数有问题的作业高。您的程序必须*完全*按照规定运行。

特别是，程序的输出必须与预期输出*完全一致*。您的程序将自动打分，因此输出中的细微差别（如空白或大小写）*将*导致测试失败，该测试的*分数*为*零*。

每个函数都附有一些使用示例，以帮助您*开始*自己的测试。您还应使用其他值测试您的函数，以确保它们按照说明运行。

以下函数**必须**在 a1.py 中实现。它们按难度递增的大致顺序排列。这并不意味着前面的函数一定比后面的函数得分低。*强烈建议*不要开始后面函数的工作

用户输入问题	constants.py **中的常量**
输入的移动不包含 5 个字符，或 不包含 3 个非空格字符，每个字符之间用空格分隔	无效格式信息
第一个字符不是在 董事会	无效行信息
第二个非空格字符不是有效的 黑板上的专栏	无效列信息
最后一个字符不是有效部件尺寸 (注意，即 使 棋手已经下完棋子，棋子大小仍然有效）。有效棋子大小介于 1 和每个玩家棋子数（PIECES PER PLAYER）中存储的任何值之间。	尺寸无效信息
此举是要求将一个标记放在 棋手已将棋子置于或要求将棋子置于大小相同或更大的标记之上	无效移动信息

表 1：包含用户输入无效信息的常量。优先级是**自上而下的**（也就是说，如果用户输入存在多个问题，则只显示本表中最先出现的问题的信息）。

直到前面的每个函数*至少都*能按照所示示例运行。如果你认为其他函数有助于你的逻辑或使你的代码更容易理解，你可以执行这些函数。

这些函数中的类型提示引用了 a1.py 中定义的 Board、Pieces 和 Move 类型别名。Board 类型变量的类型是 list[list[str]]，即它是一个列表，其中每个元素也是一个列表。内部列表包含字符串。Pieces 类型的变量属于 list[int] 类型，这意味着它是一个包含字符串的列表。关于这些变量的预期内容，请参阅相关函数的使用示例。Move 的类型是 tuple[int, int, int]，表示它是一个包含 3 个整数的 tuple。

4.1 num hours() -> float

该函数应以*浮点*形式返回您估计在作业上花费（或迄今已花费）的小时数。确保该函数*尽快*通过 Gradescope 上的相关测试。如果 Gradescope 测试已经发布，您必须先确保该函数通过相关测试，然后再就后面函数的 Gradescope 问题寻求帮助。有关如何向 Gradescope 提交作业的说明，请参见第 6.3 节。

此功能的目的是让您确认自己是否了解如何尽快向 Gradescope 提交作业，并让我们评估作

业的难度，以便提供尽可能好的帮助。您不会因为在作业上花费的时间多或少而得到不同的评分。

4.2 生成初始棋子（棋子数： int） ->棋子

返回初始标记大小的列表，从 1 到 num 个（包括 num 个）。您可以假设棋子数在 5 到 9 之间（包括 9）。单个棋子内的棋子大小

在整个赋值过程中，可以假定列表是唯一的（也就是说，一个棋手不可能有两个大小相同的棋子）。

例如

```
>>> generate_initial_pieces(5)
[1, 2, 3, 4, 5] 生成初始部件
（5
>>> generate_initial_pieces(8)
[1, 2, 3, 4, 5, 6, 7, 8] 生成初
始部件 （8
```

4.3 初始状态() -> Board

返回每个单元格（即行、列位置）都包含 EMPTY 的新棋盘。示例

```
>>> initial_state()
[[' ', ' ', ' ', ' '], [' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ']]
```

4.4 place_piece(board: Board, player: str, pieces_available: Pieces, move: Move) -> None

此函数应将请求的棋子放在棋盘上的请求位置，并将其从可用棋子列表中删除。move 是一个包含（行、列、棋子大小）的元组。也就是说，move 指定了所要求的（行、列）位置，在该位置上放置棋手所要求的棋子大小的棋子。放置的标记应该是长度为 2 的字符串，分别包含棋手标记和棋子大小。请注意，此函数不会返回任何内容；它的作用是更改（即改变）可用的棋盘和棋子。

您可以假设所请求的件尺寸将出现在可用件中。

该函数无需根据游戏规则检查棋 步 是否有效。

例如

```
>>> board = initial_state()
>>> 棋子 = 生成初始棋子 （6） 棋盘
>>>
[[' ', ' ', ' ', ' '], [' ', ' ', ' ', ' '], [' ', ' ', ' ', ' ']]
>>> 作品
[1, 2, 3, 4, 5, 6]
>>> 放置棋子 （棋盘, NAUGHT, 棋子, (1, 1, 3))
```

```
>>> 董事会
[[' ', ' ', ' ', ' '], [' ', 'O3', ' ', ' '], [' ', ' ', ' ', ' ']]
>>> 作品
[1, 2, 4, 5, 6]
>>> 放置棋子（棋盘, NAUGHT, 棋子, (1, 1, 1))
>>> 董事会
[[' ', ' ', ' ', ' '], [' ', 'O1', ' ', ' '], [' ', ' ', ' ', ' ']]
>>> 作品
[2, 4, 5, 6]
```

4.5 print game(board: Board, naught pieces: Pieces, cross pieces: Pieces) -> 无

以用户友好的格式显示游戏。您的输出必须完全符合预期的格式（包括空白和语法），才能获得分数。使用下面的示例进行测试时，请确保您的输出与显示的输出完全一致。正确格式的权威是 Gradescope 测试。确保您的输出与 Gradescope 示例测试完全一致。

例如

```
>>> board = initial_state()
>>>  naught_pieces = generate_initial_pieces(5)
>>> cross_pieces = generate_initial_pieces(5)
>>> print_game(board, naught_pieces, cross_pieces) O
有：1, 2, 3, 4, 5
X 有1, 2, 3, 4, 5

   1   2   3
 -----------
1|   |   |   |
 -----------
2|   |   |   |
 -----------
3|   |   |   |
 -----------
>>> place_piece(board, NAUGHT, naught_pieces, (1, 1, 1))
>>> place_piece(board, CROSS, cross_pieces, (0, 0, 1))
>>> print_game(board, naught_pieces, cross_pieces) O
有：2, 3, 4, 5
X 有2, 3, 4, 5

    1   2   3
- - - - - - - - -
1|X1|   |   |
- - - - - - - - -
2|   |O1|   |
- - - - - - - - -
3|   |   |   |
- - - - - - - - -
```

4.6 process move(move: str) -> Move | None

尝试将移动字符串转换为代表移动的（行、列、棋子大小）的三个整数元组。如果移动无效，此函数应打印相关信息并返回 None；相关信息请参见表 1 的前 4 行。请注意，此函数无需处理表 1 中描述的最后一个问题。如果走棋格式有效，此函数将返回一个包含 3 个整数的元组，分别代表棋子的行、列和大小。

如果转换成功，该函数将返回转换后的移动结果。请注意，在字符串移动中，行和列将以 1 为索引，而在返回的元组中，行和列将以 1 为索引。

列的索引应为 0。例如

> > > process_move('apple')
移动格式无效。请重试。

>>> process_move('apple orange berry') 移
动格式无效。请重试。

>>> process_move('a o b') 无
效行。请重试。

>>> process_move('0 0 1') 无
效行。请重试。

>>> process_move('1 0 1')
栏无效。请重试。

>>> process_move('1 1 a')
部件尺寸无效。请重试。

>>> process_move('1 1 1')
(0, 0, 1)
>>> process_move('3 3 5')
(2, 2, 5)
>>> result = process_move('apple') 移动
格式无效。请重试。

>>> 结果
>>> result = process_move('2 2 2')
>>> 结果
(1, 1, 2)

4.7 get player move() -> Move

持续提示用户移动棋子，直到以有效格式输入棋子，并以整数（行、列 、大小）格式返回最终输入的有效棋子信息。如果用户在提示符下输入 "h "或 "H"，该函数应显示帮助信息（然后继续提示，直到输入有效的移动）。每次用户输入无效格式时，都应按照表 1 的前四行显示相关信息。请注意，最后一行无需在此函数中处理。

例如

```
>>> get_player_move()
输入您的移动： apple orange banana 移动
格式无效。请重试。

输入您的移动： 0 0 0
无效行。请重试。
```

输入您的移动： 1 1 0

部件尺寸无效。请重试。

输入您的移动： 1 1 1
(0, 0, 1)
>>> get_player_move()
输入你的移动： h
输入行、列和片尺寸，格式为： 行 列 尺寸

输入您的移动： 2 2 2
(1, 1, 2)

4.8 check_move(board: Board, pieces available: Pieces, move: Move) -> bool

根据游戏规则检查棋步是否是当前棋盘的有效棋步。要成为有效棋步，棋子大小必须在给定的可用棋子列表中可用，且 move 中描述的（行、列）位置必须为空，或包含一个大小小于 move 中描述的大小的棋子。如果移动有效，则返回 True，否则返回 False。

在此函数中，您可以假定这步棋是在棋盘范围内，并且指的是棋手*初始*棋子的大小。

4.9 check_win(board: Board) -> str | None

检查游戏是否已获胜。如果游戏已获胜，该函数应返回获胜者。否则，该函数应返回 "无"

。

例如

>>> board = [['O1', 'O2', 'O3'], [EMPTY, EMPTY, EMPTY], [EMPTY, EMPTY, EMPTY]] [EMPTY, EMPTY, EMPTY]]
>>> check_win(board)
'O'
>>> board = initial_state()
>>> check_win(board)
>>> 板 = [['X1', 'O2', EMPTY], [EMPTY, 'X2', 'O3'], ['O4', EMPTY, 'X8']]]
>>> check_win(board)
'X'

4.10 check_stalemate(board: 棋盘, naught_pieces: 棋子, cross_pieces: 棋子) -> bool

如果这盘棋已经没有可走的棋步，则返回 True，否则返回 False。需要注意的是，棋手可以在自己的标记上放置标记，前提是现有标记小于新标记。

例如

```
>>> board = [['X3', 'O3', 'X2'], ['O4', 'X4', 'O2'], ['O5', 'X6', 'O6']] 板
>>> check_win(board)
>>> check_stalemate(board, [1], [1]) True
```

```
>>> check_stalemate(board, [1], [1, 5])
False
```

4.11 main() -> None

文件运行时应调用 main 函数，该函数负责协调整个游戏过程。第 3 节描述了游戏的玩法。主函数应利用您编写的其他函数。为了缩短主函数，您应该考虑编写额外的辅助函数。在提供的 a1.py 文件中，已经提供了 main 的函数定义，而 if name == main：代码块将确保在运行 a1.py 文件时运行 main 函数中的代码。不要在此代码块之外调用 main 函数，也不要在此代码块之外调用任何其他函数，除非是在其他函数的主体中调用。主函数的输出（包括提示）必须与预期输出完全一致。通过运行示例测试，您可以很好地了解提示和其他输出是否正确。有关主函数如何运行的示例，请参见本作业提供的 gameplay/ 文件夹。

5 CSSE7030 任务：更改网格大小

本节介绍了CSSE7030学生为获得满分而必须完成的附加任务。学习CSSE1001的学生不需要完成该任务，也不能因此获得分数。

在这项任务中，请确保当在 constants.py 中将 GRID SIZE 更改为不同值时，您的程序仍能正常运行。我们将只在 GRID SIZE 值范围 [2, 8] 内测试你的代码。不过，你最好不要硬编码，使你的解决方案只在 2 到 8 的范围内工作；这些值只是作为我们将测试的值范围的保证。写得好的解决方案很可能适用于其他网格大小。有关不同网格大小的输出更新示例，请参阅 gameplay/ 中的 7030 示例 9 件 6 网格大小.py 和 7030 示例 5 件 2 网格大小.py 文件。

您可以假设网格始终是正方形的（即 GRID SIZE 既描述了网格的行数，也描述了网格的列数）。

6 评估和评分标准

本作业评估课程学习目标：

1. 应用变量、选择、迭代和子程序等程序构造、
2. 阅读和分析他人编写的代码

3. 阅读和分析设计，并能将设计转化为可运行的程序，以及

4. 应用测试和调试技术。

6.1 功能性

程序的功能性将在总分 6 分中打分。您的作业将通过一系列测试，您的功能分数将与通过测试的次数成正比。您将在作业截止日期前获得功能测试的 *子集*。

任务。

在每个部分中，如果您只使用了部分功能，或只执行了几个功能，您可能会得到部分分数。

您需要*自己*对程序进行测试，以确保它符合作业中给出的*所有*规范。仅依靠提供的测试很可能会导致程序在某些情况下失败，并丢掉一些功能分数。注意：功能测试是自动进行的，因此字符串输出必须与预期*完全一致*。

您的程序必须在使用 Python 3.11 的 Gradescope 中运行。部分解决方案将被标记，但如果您的代码中存在错误，导致解释器无法执行您的程序，您的功能性分数将为零。如果您的代码中有一部分导致解释器无法执行，请注释掉该部分代码，以便剩余部分能够运行。您的程序必须使用 Python 3.11 解释器运行。如果程序在其它环境 (如 Python 3.8 或 PyCharm) 而不是 Python 3.11 解释器中运行，则功能分数为零。

6.2 代码风格

导师将对您的作业风格进行评估。风格将根据随作业提供的风格评分标准进行评分。风格评分满分为 4 分。

标记代码风格的主要考虑因素是代码是否易于理解。代码风格的几个方面会影响代码的易懂程度。在本次作业中，将根据以下标准对您的代码风格进行评估。

- 可读性
  - 程序结构：代码的布局使其易于阅读并遵循其逻辑。这包括使用空白来突出逻辑块。
  - 描述性标识符名称：变量名、常量名和函数名应清楚地描述它们在程序逻辑中的含义。标识符名称不要使用匈牙利符号。简而言之，这意味着不要在名称中包含标识符的类型，而是要让名称更有意义（如雇员标识符）。
  - 命名常量：代码中的任何非重要固定值（字面常量）都由一个描述性命名常量（标识符）来代表。
- 算法逻辑
  - 单一逻辑实例：程序中的代码块不应重复。任何需要多次使用的代码都应以函

数的形式实现。

  - 变量范围：变量应在需要使用的函数中本地声明。不应使用全局变量。
  - 控制结构：善用控制结构（如循环和条件语句），使逻辑结构简单明了。

- 文件：

  - 注释清晰：注释对代码进行有意义的描述。注释不应重复阅读代码就能明显看出的内容（如 # 将变量设置为 0）。注释不应冗长或过多，否则会导致难以理解代码。

- 翔实的文档说明：每个函数都应有一个概述其目的的文档字符串。这包括描述参数和返回值（包括类型信息），以便他人了解如何正确使用该函数。
- 逻辑说明：所有重要的代码块都应该有注释来解释逻辑是如何工作的。对于小型函数，这通常就是 docstring。对于长函数或复杂函数，函数中可能有不同的代码块。每个代码块都应有一个行内注释来说明其逻辑。

6.3 提交作业

您必须通过 Gradescope (https://gradescope.com/) 以电子方式提交作业。
您**必须**使用基于您学号的昆士兰大学电子邮件地址（如 s4123456@student.uq.edu.au）作为您的 Gradescope 提交账户。

登录 Gradescope 后，您可能会看到一个课程列表。选择 CSSE1001/CSSE7030。您将看到作业列表。选择**作业 1**。系统会 提示您选择要上传的文件。提示可能会说您可以上传任何文件，包括压 缩 文 件 。您**必须**以名为 a1.py 的单个 Python 文件（使用此文件名--全部小 写）提交作业，而*不能提交任何*其他文件。您提交的文件将自动运行，以确定功能分数。如果您提交的**文件名不同**，测试将**失败**，您的功能性得分将为**零。请勿**提交**任何**类型的压缩文件（如 zip、rar、7z 等）。

在截止日期前*至少*一周上传作业的初始版本。即使只是作业提供的初始代码，也要这样做。如果您无法访问 Gradescope，请*立即*联系课程帮助台 (csse1001@helpdesk.eait.uq.edu.au)。您无法登录或无法上传文件等借口将不被接受作为批准延期的理由。

当您上传作业时，它将对您提交的作业运行**子集**功能自动检测。它会向您显示这些测试的结果。您有责任确保上传的作业文件能够运行，并通过您期望通过的测试。

逾期提交的作业将不予批改。不要等到最后一刻才提交作业，因为上传作业的时间可能会导致作业迟交。允许并鼓励多次提交作业，因此请确保在 15:00 作业提交截止时间之前提交了基本完整的作业。您最近一次按时提交的作业将被评分。确保提交正确的作业版本。

如果出现特殊的个人或医疗情况，导致您无法按时交作业，您可以提交延期申请。有关如何申请延期的详情，请参阅课程简介。

延期申请必须**在**提交截止日期**前**提出。申请和证明文件（如医疗证明）必须通过 my.UQ 提交。您必须保留原始文件至少六个月，以便在需要时提供证明。

6.4 剽窃

本作业必须是您的个人作业。提交作业即表明这完全是你自己的作品。您**可以**与下列人员讨论有关解题方法的一般想法

其他学生。描述如何实现函数的细节或与其他学生共享部分代码被视为**串通行为**，将被视为抄袭。**不得**在作业中复制从互联网上找到的代码片段。

请阅读课程简介中有关抄袭的部分。我们鼓励您*在*开始本作业*前*完成学术诚信模块的 A 和 B 部分。提交的作业将通过电子方式检查是否存在潜在的抄袭情况。