Project 1 - Mancala

Due Date: Friday, November 3rd, 2023 by 11:59:59 PM
Value: 90 points

This assignment falls under the standard cmsc201 academic integrity policy. This means you should not discuss/show/copy/distribute your solutions, your code or your main ideas for the solutions to any other student. Also, you should not post these problems on any forum, internet solutions website, etc.

```
"""
File:     FILENAME.py
Author:   YOUR NAME
Date:     THE DATE
Section:  YOUR DISCUSSION SECTION NUMBER
E-mail:   YOUR_EMAIL@umbc.edu
Description:
  DESCRIPTION OF WHAT THE PROGRAM DOES
"""
```

Project General Description

For our first project, we're going to be implementing the game Mancala. Mancala has been discovered in the excavated ruins of a Roman bathhouse built in the 2nd or 3rd century AD.

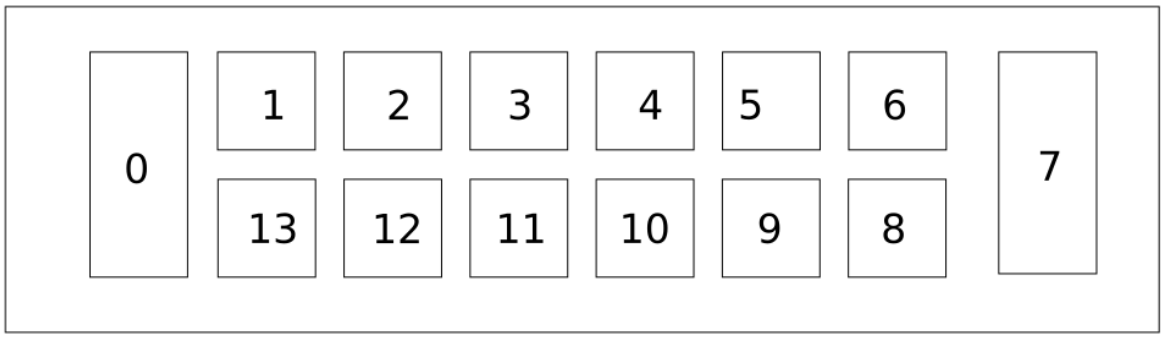
In the 21st century, however, we have Python so we'll write a little version of the game ourselves.

The game is played with 'stones' which can be glass bubbles or stones, and there are generally two rows of 6 circular "cups" or pits, and then there's a double length pit on both sides, which we will also call a "mancala."

Rules of the Game

1. The board is formed by the two long pits or cups, and 12 cups divided into two rows, like in the picture.

2. To set up the game, place four stones in each of the 12 regular cups.
3. Players alternate by taking turns.
4. A turn consists of selecting a non-empty non-mancala cup and taking all of the stones from it, and then starting in the next cup, place a stone, and then advance to the next cup and place another stone, repeating this until all of the stones have been moved.
    a. The direction of play will be clockwise.
    b. Mancalas count as a put so you must place a stone in there as you pass that space.
    c. If the last stone is placed into a mancala, then that player gets another turn.
    d. A player cannot select a mancala to move stones out.
5. The game ends when one of the rows of regular cups is empty, so when either the "top" row or the "bottom" row is empty, the game is over.
6. The winner when the game ends is the player with the most stones in their mancala.
7. The board is numbered in the following order. Cups 0 and 7 are mancalas whereas the rest are "regular."
8. Player 1's mancala is at position 7, and player 2's mancala is at position 0.



Implementation Details

This is a specification for how the project should be implemented.

1. Get the names of players.

2. Begin the cycle of turns by alternating back and forth from player 1 to player 2 and back until the game is over.
3. When a player takes a turn, move the stones for them, starting in the next cup and placing a stone in each cup until the stones run out.
4. If a user enters an invalid cup number, make them try again.
5. If a turn ends in a mancala (the last stone is placed there), then that player gets a second turn.
6. When one of the two rows (positions 1 to 6 or positions 8 to 13) are empty, then the game ends.
7. Tell the players who has won based on the number of stones in each of the mancalas.
8. After each move, reprint the board and ask for the next user input.

Design Overview

Your project should include at least the functions:

1.) This function should be the entry point for your program. Your if __name__ == "__main__": should be exactly one line, which calls this function, unless you want to create the players before you run the game.

```
def run_game():
```

   a.) You can change the parameters of the function.
   b.) This function should run the game, set up the board, alternate between players, and determine the winner at the end. You should create helper functions in order to distribute all of this functionality. This function shouldn't be a 100 line monolithic monstrosity.

2.) The function get_player should be something like:

```
def get_player():
```

   a.) You are permitted to modify the arguments of the function.
   b.) This function should get the player name or names, depending on how you decide to implement it.

3.) You should create a function:

```
def take_turn(player, game_cups):
```

   a.) You can modify the function's signature (i.e. change parameters depending on your implementation.
   b.) This function should print the board, ask for the new move, verify that it is a legal move, and then make it.
   c.) There should be a way for you to signal from this function back to run_game that you have landed on a mancala and that player should get a new turn.

4.) You must create at least 2 more functions to help you finish the project. You can choose what they do, and what exactly their signatures should be. If you ask the question "but what if I don't need 2 more functions?" then you definitely have too much functionality going on in any single function. I created about 5 helper functions in my implementation.

Provided Functions

I have provided a starter file on the GL server at:

/afs/umbc.edu/users/e/r/eric8/pub/cs201/fall23/mancala_starter.py

You should copy it to your directory using the command:

cp /afs/umbc.edu/users/e/r/eric8/pub/cs201/fall23/mancala_starter.py mancala.py

There are three functions in the file as well as some constants.

```
draw_board(top_cups, bottom_cups, mancala_a, mancala_b):
```

The draw board function is the function you should be calling. You should send data to this function in the following way:

Top_cups and bottom_cups should be a 2-d list of strings. The outer list for both of these variables should be length 6, because that's the number of cups.

The inner list should be a list of strings which is of length BLOCK_HEIGHT, because that's the number of lines.

You can use slices like this: my_string[0: BLOCK_WIDTH] to chop off any strings you print, and str.rjust(BLOCK_WIDTH) to give extra space. The draw board function will try to do this for you, but if you end up with IndexErrors, then you should use these methods to force the string to be the right length.

Mancala_a and mancala_b are lists of strings of length 2 * BLOCK_HEIGHT + 1 because there is the midline asterisk which takes up another space. The strings should all be of length BLOCK_WIDTH, but my code tries to help you out as well.

I am providing minimal documentation, i.e. nearly nothing about the following two functions. The reason for this is that you don't need to call these functions, and shouldn't modify them. You can read them of course and though they are documented, they are not terribly complex, but just call draw_board with the right data.

```
draw_mancala(fore_or_aft, mancala_data, the_board):
```

This function draws out a mancala, but you should not call this function directly. It's a helper function for my draw_board function.

```
draw_block(the_board, pos_x, pos_y, block_data):
```

Similarly, this is also a function you shouldn't have to call. It's another helper function for the draw_board function.

Submission Details

Submit the files under the following titles:
(These are case sensitive as usual.)
submit cmsc201 PROJECT1 mancala.py

Coding Standards

Coding standards can be found here.

1. At least one inline comment per function explaining something about your code.
2. Constants above your function definitions, outside of the "if __name__ == '__main__':" block.
    a. A magic value is a string which is outside of a print or input statement, but is used to check a variable, so for instance:
        i. `print(first_creature_name, 'has died in the fight. ')` does not involve magic values.
        ii. However, `if my_string == 'EXIT':` exit is a magic value since it's being used to compare against variables within your code, so it should be:
        ```
        EXIT_STRING = 'EXIT'
        if my_string == EXIT_STRING:
        ```
    b. A number is a magic value when it is not 0, 1, and if it is not 2 being used to test parity (even/odd).
    c. A number is magic if it is a position in an array, like my_array[23], where we know that at the 23rd position, there is some special data. Instead it should be
    USERNAME_INDEX = 23
    my_array[USERNAME_INDEX]
    d. Constants in mathematical formulas can either be made into official constants or kept in a formula.
3. Previously checked coding standards involving:
    a. snake_case_variable_names

b. CAPITAL_SNAKE_CASE_CONSTANT_NAMES
c. Use of whitespace (2 before and after a function, 1 for readability.)

Allowed Built-ins/Methods/etc

- Declaring and assigning variables, ints, floats, bools, strings, lists, dicts.
- Using +, -, *, /, //, %, **; +=, -=, *=, /=, //=, %=, **= where appropriate
- Comparisons ==, <=, >=, >, <, !=, in
- Logical and, or, not
- if/elif/else, nested if statements
- Casting int(x), str(x), float(x), (technically bool(x))
- For loops, both *for i* and *for each* type.
- While loops
    - sentinel values, boolean flags to terminate while loops
- Lists, list(), indexing, i.e. my_list[i] or my_list[3]
    - 2d-lists if you want them/need them my_2d[i][j]
    - Append, remove
    - **list slicing**
- If you have read this section, then you know the secret word is: createous.
- String operations, concatenation +, +=, split(), strip(), join(), upper(), lower(), isupper(), islower()
    - **string slicing**
- Print, with string formatting, with end= or sep=:
    - '{}'.format(var), '%d' % some_int, f-strings
    - Really the point is that we don't care how you format strings in Python
    - Ord, chr, but you won't need them this time.
- Input, again with string formatting in the prompt, casting the returned value.

- **Dictionaries**
    - creation using dict(), or {}, copying using dict(other_dict)
    - .get(value, not_found_value) method
    - accessing, inserting elements, removing elements.
    - Deleting elements from dictionaries using del.
- Using the functions provided to you in the starter code.
- Using import with libraries and specific functions **as allowed** by the project/homework.
- You may define your own new functions with new parameters
    - Single return values (you must return a single float, bool, string, int, list or dictionary, or None/not return).

Forbidden Built-ins/Methods/etc

This is not a complete listing, but it includes:

- Multiple returns [i.e. `var1, var2 = function_call()`]
- break, continue
- methods outside those permitted within allowed types
    - for instance str.endswith
    - list.index, list.count, etc.
- Keywords you definitely don't need: await, as, assert, async, class, except, finally, global, lambda, nonlocal, raise, try, yield
- The *is* keyword is forbidden, not because it's necessarily bad, but because it doesn't behave as you might expect (it's not the same as ==).
- built in functions: any, all, breakpoint, callable, classmethod, compile, exec, delattr, divmod, enumerate, filter, map, max, min, isinstance, issubclass, iter, locals, oct, next, memoryview, property, repr, reversed, round, set, setattr, sorted, staticmethod, sum, super, type, vars, zip
- If you have read this section, then you know the secret word is: serendipity.
- exit() or quit()
- If something is not on the allowed list, not on this list, then it is probably forbidden.
- The forbidden list can always be overridden by a particular problem, so if a problem allows something on this list, then it is allowed for that problem.

Sample Project Output

Here is a single quick run of the program, for more you should run the pyc file.

```
linux3[139]% python3 mancala.py
Player 1 please tell me your name: Abby
Player 2 please tell me your name: Boris
*********************************************************
*       *Cup    *Cup    *Cup    *Cup    *Cup    *Cup    *       *
*       *      1*      2*      3*      4*      5*      6*       *
*       *Stones*Stones*Stones*Stones*Stones*Stones*       *
*Boris  *      4*      4*      4*      4*      4*      4*Abby   *
*       *       *       *       *       *       *       *       *
*       ****************************************        *
*       *Cup    *Cup    *Cup    *Cup    *Cup    *Cup    *       *
*Stones*     13*     12*     11*     10*      9*      8*Stones*
*0      *Stones*Stones*Stones*Stones*Stones*Stones*0      *
*       *      4*      4*      4*      4*      4*      4*       *
*       *       *       *       *       *       *       *       *
*********************************************************
Abby What cup do you want to move? 1
*********************************************************
*       *Cup    *Cup    *Cup    *Cup    *Cup    *Cup    *       *
*       *      1*      2*      3*      4*      5*      6*       *
*       *Stones*Stones*Stones*Stones*Stones*Stones*       *
*Boris  *      0*      5*      5*      5*      5*      4*Abby   *
*       *       *       *       *       *       *       *       *
*       ****************************************        *
*       *Cup    *Cup    *Cup    *Cup    *Cup    *Cup    *       *
*Stones*     13*     12*     11*     10*      9*      8*Stones*
*0      *Stones*Stones*Stones*Stones*Stones*Stones*0      *
*       *      4*      4*      4*      4*      4*      4*       *
*       *       *       *       *       *       *       *       *
*********************************************************
Boris What cup do you want to move? 2
Your last stone landed in a mancala.
Go again please...
*********************************************************
*       *Cup    *Cup    *Cup    *Cup    *Cup    *Cup    *       *
*       *      1*      2*      3*      4*      5*      6*       *
*       *Stones*Stones*Stones*Stones*Stones*Stones*       *
*Boris  *      0*      0*      6*      6*      6*      5*Abby   *
*       *       *       *       *       *       *       *       *
```

```
*      ****************************************      *
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*Stones*    13*    12*    11*    10*     9*     8*Stones*
*0     *Stones*Stones*Stones*Stones*Stones*Stones*1     *
*      *     4*     4*     4*     4*     4*     4*      *
*      *      *      *      *      *      *      *      *
**********************************************************
Boris What cup do you want to move? 3
**********************************************************
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*      *     1*     2*     3*     4*     5*     6*      *
*      *Stones*Stones*Stones*Stones*Stones*Stones*      *
*Boris *     0*     0*     0*     7*     7*     6*Abby  *
*      *      *      *      *      *      *      *      *
*      ****************************************      *
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*Stones*    13*    12*    11*    10*     9*     8*Stones*
*0     *Stones*Stones*Stones*Stones*Stones*Stones*2     *
*      *     4*     4*     4*     4*     5*     5*      *
*      *      *      *      *      *      *      *      *
**********************************************************
Abby What cup do you want to move? 4
**********************************************************
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*      *     1*     2*     3*     4*     5*     6*      *
*      *Stones*Stones*Stones*Stones*Stones*Stones*      *
*Boris *     0*     0*     0*     0*     8*     7*Abby  *
*      *      *      *      *      *      *      *      *
*      ****************************************      *
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*Stones*    13*    12*    11*    10*     9*     8*Stones*
*0     *Stones*Stones*Stones*Stones*Stones*Stones*3     *
*      *     4*     4*     5*     5*     6*     6*      *
*      *      *      *      *      *      *      *      *
**********************************************************
Boris What cup do you want to move? 5
**********************************************************
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*      *     1*     2*     3*     4*     5*     6*      *
*      *Stones*Stones*Stones*Stones*Stones*Stones*      *
*Boris *     0*     0*     0*     0*     0*     8*Abby  *
*      *      *      *      *      *      *      *      *
*      ****************************************      *
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*Stones*    13*    12*    11*    10*     9*     8*Stones*
*0     *Stones*Stones*Stones*Stones*Stones*Stones*4     *
*      *     5*     5*     6*     6*     7*     7*      *
```

```
*      *      *      *      *      *      *      *      *
*************************************************************
Abby What cup do you want to move? 6
Your last stone landed in a mancala.
Go again please...
*************************************************************
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*      *     1*     2*     3*     4*     5*     6*      *
*      *Stones*Stones*Stones*Stones*Stones*Stones*      *
*Boris *     0*     0*     0*     0*     0*     0*Abby  *
*      *      *      *      *      *      *      *      *
*      ****************************************      *
*      *Cup   *Cup   *Cup   *Cup   *Cup   *Cup   *      *
*Stones*    13*    12*    11*    10*     9*     8*Stones*
*1     *Stones*Stones*Stones*Stones*Stones*Stones*5     *
*      *     6*     6*     7*     7*     8*     8*      *
*      *      *      *      *      *      *      *      *
*************************************************************
Abby is the winner
```

Sample Runs - PYC File

You can run my version of the project and see some sample output.

I'm going to release a compiled version of the project so that you can play the game and get a feel for what the sample output should be. You don't have to match my messages exactly but the program should flow in the same way.

The pyc file (a compiled python file) can be found at /afs/umbc.edu/users/e/r/eric8/pub/cs201/fall23/mancala.pyc

To copy a file you should use cp and then . at the end. Without the period at the end it won't copy to the current directory.

Run the file by running the command:

```
linux5[115]% cp /afs/umbc.edu/users/e/r/eric8/pub/cs201/fall23/mancala.pyc .
linux5[115]% python3 mancala.pyc
```

It will probably only work on the GL server or on linux with python3 version 3.8.

Approximate Rubric

I say approximate here to mean that this rubric doesn't break down specific requirements within each group, and may be modified by a few points here and there, but this should provide you with a good understanding of what we intend the value of each piece of the project implementation to be.

Requirement	**Points** (appx)
Get the names of the players	5
Set up the board.	15
Move the stones correctly.	15
Give an extra move for landing on the mancala space.	10
Detect end of game when one of the rows is empty	10
Detect Win/Loss conditions.	15
Print out who was the winner	5
Implement at least 2 but probably more helper functions.	5
Coding Standards, Documentation, Comments, Overall Correctness	10