cs12pf22

Assignment 13: Modeling a Connect Four® Game

Goals

- Define a `class` to create a customized data type in Python.
- Model the status and logic of a simple two-player game.

Prerequisites

This assignment requires familiarity with the lecture materials presented in class through week 13.

Assignment

You shall define a `class` named `ConnectFour` in a module named `connect_four`, instances of which model a playable game of Connect Four® [https://en.wikipedia.org/wiki/Connect_Four] with the following methods. Details can be found in the docstrings of the starter code below.

1. `__init__()__` constructs a new game. Player names default to `red` for player 1 and `black` for player 2, but the constructor accepts two optional arguments that give names to each player, which also serve as colors in the GUI (see below).
2. `__bool__()__` returns whether the game is still in play, i.e. whether there is not yet a winner and the game has not ended in a draw.
3. `__str__()__` returns a 6-line/7-column string representation of the board, formatted as demonstrated in the doctests below. Tokens dropped by each player are indicated by the characters `1` and `2`.

4. `drop()` allows the current player to take a turn by specifying a column (`0`–`6`) into which to drop a token. Player 1 always gets the first turn, and subsequent turns alternate player 1 and player 2.
5. `board()` returns a 6-tuple of 7-tuples representing the state of the board, where each 7-tuple represents a row in the board. Each location in the board is indicated by either `None`, `1`, or `2`, depending on whether the location is unclaimed or belonging to either player.
6. `player_names()` returns a `dict` with two items mapping the player numbers (`1` and `2`) to their respective names/colors.
7. `winner()` returns the winner of the game, or `None` if the game has not ended or has ended in a draw.

Starter Code

The starter code below is a skeletal implementation of the `connect_four` module, including numerous doctests [https://docs.python.org/3/library/doctest.html] for the methods of class `ConnectFour`.

connect_four.py

```
#!/usr/bin/env python3
"""
Module connect_four contains the ConnectFour class, instances of which model Connect 4 games.
(Seven columns, six rows.)
"""


class ConnectFour:
  """
  Models a game of Connect Four. Players can be given colors, but are still represented as 1 and 2
  on the board. Player 1 has the first turn. Turns are taken by calling the drop() method.
  """

  def __init__(self, p1='red', p2='black'):
    """
    Initializes a Connect Four game with the disc colors for players 1 and 2 specified by
    arguments `p1` and `p2`, respectively. Turns alternate, with player 1 getting the first turn.
    Colors can be any two different strings, but if you want them to show up appropriately in the
    GUI, choose one of the color names specified in file `/srv/datasets/rgb.txt`, or an
    equivalent hexadecimal color code, if you know what that means.

    >>> g = ConnectFour()
    >>> print(g)
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    >>> g.player_names()
    {1: 'red', 2: 'black'}
    >>> bool(g)
    True
    >>> g.winner()
    >>>
```

```
    """
    pass  # TODO

  def __bool__(self):
    """
    A Connect Four game is considered Boolean True if it is still in play, i.e. there is no winner
    and possible moves remain.

    >>> g = ConnectFour(p1='yellow', p2='blue')
    >>> flag = True
    >>> while g:
    ...   g.drop(0 if flag else 1)
    ...   flag = not flag
    ...
    >>> print(g)
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |1|_|_|_|_|_|_|
    |1|2|_|_|_|_|_|
    |1|2|_|_|_|_|_|
    |1|2|_|_|_|_|_|
    >>> g.winner()
    'yellow'
    """
    pass  # TODO

  def __str__(self):
    """
    Returns a string representation of the game board, formatted as in the following:

    >>> g = ConnectFour()
    >>> print(g)
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    >>> g.drop(0)
    >>> print(g)
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |1|_|_|_|_|_|_|
    >>> g.drop(0)
    >>> print(g)
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |2|_|_|_|_|_|_|
    |1|_|_|_|_|_|_|
    >>> g.drop(1)
    >>> print(g)
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |_|_|_|_|_|_|_|
    |2|_|_|_|_|_|_|
```

```
    |1|1|_|_|_|_|_|
    """
    pass  # TODO

  def board(self):
    """
    Returns a 6-tuple of 7-tuples representing the state of the board, where each 7-tuple represents
    a row in the board. Index 0 is the top row, and index 5 is the bottom row. Each 7-tuple shall
    consist of either the int value `1` or `2` to represent a location claimed by player 1 or 2, or
    the value `None` for a location that has not been claimed by either player.

    >>> import pprint
    >>> g = ConnectFour()
    >>> pprint.pprint(g.board())
    ((None, None, None, None, None, None, None),
     (None, None, None, None, None, None, None),
     (None, None, None, None, None, None, None),
     (None, None, None, None, None, None, None),
     (None, None, None, None, None, None, None),
     (None, None, None, None, None, None, None))
    >>> g.drop(0)
    >>> g.drop(0)
    >>> g.drop(1)
    >>> pprint.pprint(g.board())
    ((None, None, None, None, None, None, None),
     (None, None, None, None, None, None, None),
     (None, None, None, None, None, None, None),
     (None, None, None, None, None, None, None),
     (2, None, None, None, None, None, None),
     (1, 1, None, None, None, None, None))
    """
    pass  # TODO

  def drop(self, col: int):
    """
    Takes a turn by dropping a token into the given column (0-6) for the current player.
    Turns alternate, with player 1 getting the first turn. Raises a ValueError if the turn is
    invalid, i.e. if the column is already full, or the index is out of bounds, or the game is no
    longer in play.

    >>> g = ConnectFour()
    >>> g.board()[5][0]  # bottom row, first column
    >>> g.drop(0)  # drop token in first column
    >>> g.board()[5][0]
    1
    >>> g.board()[4][0]  # second to last row, first column
    >>> g.drop(0)  # drop token in first column
    >>> g.board()[4][0]
    2
    """
    pass  # TODO

  def player_names(self):
    """
    Returns a dict with two items mapping the player numbers (1 and 2) to their respective colors.

    >>> g = ConnectFour()
    >>> g.player_names()
    {1: 'red', 2: 'black'}
    >>> g = ConnectFour(p1='yellow', p2='blue')
    >>> g.player_names()
```

```
    {1: 'yellow', 2: 'blue'}
    """
    pass  # TODO

  def winner(self):
    """
    Returns the winner of the game, or None if the game has not ended or has ended without a winner.

    >>> g = ConnectFour(p1='yellow', p2='blue')
    >>> flag = True
    >>> while g:
    ...   g.drop(0 if flag else 1)
    ...   print(g.winner())
    ...   flag = not flag
    ...
    None
    None
    None
    None
    None
    None
    yellow
    """
    pass  # TODO

# feel free to define other methods and/or a main block, if you'd like
```

Optional GUI

The doctests in the starter code above demonstrate how a `ConnectFour` object should behave in the REPL. Since the `ConnectFour` class offers an interface to view and change the game state through its methods, you could view and control it from a GUI. The code below offers a simple GUI with clickable buttons. If your `ConnectFour` class works correctly, you should be able to play a game through this GUI, where clicking on any column will take a turn for the current player.

Note that this GUI uses the `tkinter` [https://docs.python.org/3/library/tk.html] library module, which offers an interface to the Tcl/Tk [https://en.wikipedia.org/wiki/Tcl] toolkit. If you want to run this, you have two major options:

1. Run it on your own machine, assuming you have Python installed and it supports `tkinter`.
2. Run it on the server, with X11 forwarding [https://en.wikipedia.org/wiki/X_Window_System#Remote_desktop] enabled such that the GUI code will run on the server, but you can interact with the GUI from your own machine.
    I. MobaXterm supports this automatically, or at least should.
    II. Add the `-Y` flag when connecting using the `ssh` command: `ssh -Y lethorington@jeff.cis.cabrillo.edu`
        a. On macOS, you will first need to install XQuartz [https://www.xquartz.org/] in order for this to work.
        b. On GNU/Linux, this should work fine without additional configuration.
        c. This should probably work on Windows Terminal as well, if you have the appropriate Linux subsystem services installed.

connect_four_gui.py

```
#!/usr/bin/env python3
"""
Quick and dirty example of using the tkinter module to make a simple GUI (graphical user interface)
for the ConnectFour class. Note that all the game logic is in your ConnectFour class. This GUI is
simply a "view" and "controller" of a ConnectFour object, calling its __bool__(), drop(), board(),
player_names() and winner() methods as appropriate.
"""
__author__ = 'Jeffrey Bergamini for CS 12P, tw // historical trauma jeffrey.bergamini@cabrillo.edu'

import itertools
import tkinter

from connect_four import ConnectFour


class ConnectFourGui(tkinter.Tk):
  """A Tk instance containing a ConnectFour instance and clickable buttons for each board square."""

  _WINDOW_TITLE = 'CS 12P Connect Four'

  def __init__(self):
    super().__init__()
    self.title(ConnectFourGui._WINDOW_TITLE)
    self.game = ConnectFour()  # the internal game object
    self.buttons = []
    for row in range(6):
      for col in range(7):
        frame = tkinter.Frame(master=self, relief=tkinter.FLAT, borderwidth=1)
        frame.grid(row=row, column=col, sticky='ew')
        button = tkinter.Button(master=frame,
                                command=self.click(col),
                                font=('TkFixedFont', 100),
                                width=2)
        button.pack(expand=True, fill='both')
        self.buttons.append(button)

  def click(self, col):
    """Returns a handler for a button click."""

    def handler():
      if not self.game:  # re-initialize game if not in play
        self.game = ConnectFour()
        for button in self.buttons:
          button.configure(fg=None, text='')
      try:
        self.game.drop(col)  # try to make a move in this column
      except ValueError:
        self.title(f'{ConnectFourGui._WINDOW_TITLE} — Invalid move!')
        return
      self.title(f'{ConnectFourGui._WINDOW_TITLE} — ' +
                 ('Game in Progress' if self.game else f'Winner: {self.game.winner()}'))
      for (button, status) in zip(self.buttons, itertools.chain.from_iterable(self.game.board())):
        if status:
          button.configure(text='●', fg=self.game.player_names()[status])
        else:
          button.configure(text=None, fg=None)
        if not self.game and button['text'] and button['fg'] != self.game.winner():
          # hollow out the losing tokens if the game is over
          button.configure(text='○')

    return handler
```

```
if __name__ == '__main__':
  # fire up the GUI!
  gui = ConnectFourGui()
  gui.mainloop()
```

Leaderboard

As submissions are received, this leaderboard will be updated with the top-performing fully/nearly functional solutions, with regard to execution speed.

(Last updated: Thu Dec 15 2022 12:13:54 GMT-0800 (PST))

Rank	Test Time (s)	Memory Usage (kB)	SLOC (lines)	User
1	26.8500	12060	97	gehult
2	26.8800	11944	56	mcornish
3	27.1700	11848	87	ecweiler
4	28.2400	11708	92	nmpanec
5	29.3700	11744	57	altorresmoran
6	30.5800	11836	106	tomott

Submission

Submit `connect_four.py` via turnin.

Feedback Robot

This project has a feedback robot that will run some tests on your submission and provide you with a feedback report via email within roughly one minute.

Please read the feedback carefully.

Due Date and Point Value

Due at 23:59:59 on the date listed on the syllabus.

`Assignment 13` is worth 60 points.

```
Possible point values per category:
-------------------------------------
Properly implemented methods,        60
  split roughly evenly as long
  as __init__() is functional
Possible deductions:
  Style and practices            10–20%
-------------------------------------
```

Discussion

Jeffrey Bergamini, 2022-11-25 22:30

Hi all,

Hope you had a good Thursday of remembering whatever it is you wanted to remember.

Assignment 13 (our penultimate!) is ready for you, and this time you'll be defining a class to model the state and logic of a simple tabletop game, viz. Connect Four®. There is also an optional GUI that will use your class to display a graphical interface for the game, if you want to try it out.

Have a look!

Nathan E. Pedrotti, 2022-12-03 16:41

Hey Jeff,

I'm having a lot of trouble finding out how to start this assignment. Been looking at everything for a while and its just not clicking. Do you have any advice on where to start?

> Jeffrey Bergamini, 2022-12-03 17:16
>
> Hi Nathan,
>
> I think maybe a place to start would be getting the first couple doctests in `<nowiwki>init() </nowiki>` to pass:
>
> ```
> >>> g = ConnectFour()
> >>> print(g)
> |_|_|_|_|_|_|_|
> |_|_|_|_|_|_|_|
> |_|_|_|_|_|_|_|
> |_|_|_|_|_|_|_|
> |_|_|_|_|_|_|_|
> |_|_|_|_|_|_|_|
> >>> g.player_names()
> {1: 'red', 2: 'black'}
> >>> bool(g)
> True
> ```

Consider what variables you need to establish as part of the `ConnectFour` object, i.e. what you should add to `self` in the constructor. I have five, for example:

1. `self._players` is a `dict`
2. `self._board` = is a `list` of `list`s
3. `self._turn` is an `int`
4. `self._in_play` is a `bool`
5. `self._winner` is initially `None`

Nathan E. Pedrotti, 2022-12-04 13:49, 2022-12-04 13:50

Thanks! that helped a lot

I'm now having a problem with the str method, my return value for the initial state is,

'|_|_|_|_|_|_|/n|_|_|_|_|_|_|/n|_|_|_|_|_|_|/n|_|_|_|_|_|_|/n|_|_|_|_|_|_|/n|_|_|_|_|_|_|/n'

and when I print it, that's exactly what it prints out. How do you make it so the newline characters make new lines instead of just printing them as is?

Tyler Green, 2022-12-04 14:01

I would try switching which type of slash you are using, it looks like you are using forward slashes "/n" instead of "\n". Hopefully that helps!

Nathan E. Pedrotti, 2022-12-04 14:06

haha yeah that would do it huh. Thanks!

Thom Mott, 2022-12-03 23:14, 2022-12-04 00:07

Is there a sample connect four program to debug against? Having a hard time with debugging at the moment.

EDIT: Upon rereading some of the doc string's prompts I've realized where I got tripped up. The line "Index 0 is the top row, and index 5 is the bottom row." confused me on first read. Thought it meant the output should be like: (None, None, None, None, 2, 1), not (1, 2, None, None, None, None) like is intended.

Thom Mott, 2022-12-04 00:25

Something I am having issues with is "Unexpected return value from board()". I'm pretty stumped on this one since my output looks like the example given in the docstring.

```
>>> pprint.pprint(g.board())
((None, None, None, None, None, None),
(1, None, None, None, None, None),
(None, None, None, None, None, None),
(2, None, None, None, None, None),
(1, None, None, None, None, None),
```

```
(None, None, None, None, None, None),
(None, None, None, None, None, None))
```

But my submission will be flooded with errors like

> Unexpected return value from board() after dropping in columns [6, 2, 2, 1, 6] in a game constructed via ConnectFour()
> Unexpected return value from board() after dropping in columns [3, 4, 3, 3, 2, 4, 5, 5, 5] in a game constructed via ConnectFour(x='SlateBlue', o='MediumPurple2')

I'm unsure what I'm missing here 🥺

Jeffrey Bergamini, 2022-12-04 02:42

Hi Thom,

Is your code passing the doctests?

Thom Mott, 2022-12-04 10:23, 2022-12-04 10:24

EDIT: Never mind I see it

Jeffrey Bergamini, 2022-12-04 11:44

Remember, you can run those tests:

```
python3 -m doctest connect_four.py
```

Thom Mott, 2022-12-07 21:18

demostudent got kicked off the leaderboard 🤣

Jeffrey Bergamini, 2022-12-07 22:07

Finally!

Nathaniel A. Newman (Nate), 2022-12-08 20:43

Hi, when I submit my assignment I get this message

✗ Passed 44556 of 267336 tests: Unexpected exception calling bool() after dropping in columns [4, 2, 6, 6, 0, 3, 3, 3, 5, 4, 3, 1, 3, 2, 0, 2, 5, 3, 5, 4, 0, 1, 1, 5, 5] in a game constructed via ConnectFour(): Traceback (most recent call last):, TypeError: bool should return bool, returned NoneType, Unexpected exception calling bool() after dropping in columns [3, 6, 5, 3, 3, 3, 6, 0, 2, 1, 1, 1, 2, 5, 1, 5] in a game constructed via ConnectFour(): Traceback (most recent call last):, TypeError: bool should return bool, returned NoneType, Unexpected return value from board() after dropping in columns [3, 0, 1, 0, 3, 6] in a game constructed via ConnectFour() Unexpected return value from str() after dropping in columns [5, 0, 4] in a game constructed via ConnectFour(x='grey62', o='DarkSeaGreen1') Unexpected return value from str() after dropping in columns [1, 1, 2, 3, 6, 2, 2, 5, 3, 3, 2, 4, 5, 5, 4, 1, 3, 4, 3, 4] in a game constructed via ConnectFour() Unexpected exception calling bool() after dropping in columns [0, 4, 4, 1, 0, 2, 2] in a game constructed via ConnectFour(x='aquamarine4', o='gray51'): Traceback (most recent call last):, TypeError:

bool should return bool, returned NoneType, Unexpected exception calling bool() after construction via ConnectFour(x='grey15', o='turquoise4'): Traceback (most recent call last):, TypeError: bool should return bool, returned NoneType, Unexpected exception calling bool() after dropping in columns [1, 0, 4, 1, 0, 2, 1, 3, 6, 0, 4, 5, 3, 2, 2, 4] in a game constructed via ConnectFour(x='SteelBlue2', o='SeaGreen3'): Traceback (most recent call last):, TypeError: bool should return bool, returned NoneType, Unexpected return value from str() after dropping in columns [5, 3, 6, 6, 4, 1, 5, 5, 4, 0, 6, 5, 1, 1, 2, 1, 0, 0, 5, 2, 5, 6, 6, 1, 2, 6, 1, 2, 6, 0, 2, 1, 1, 2, 4, 4, 2, 4, 1, 1, 3, 2, 0, 0, 5, 4, 2, 5, 6, 5, 6, 2, 1, 0, 3, 2, 5, 6, 5, 2, 2, 2, 3, 1, 4, 1, 0, 6, 4, 5, 0, 0, 4, 0, 1, 2, 6, 5, 4, 2, 1, 3, 5, 4, 0, 5, 5, 6, 5, 5, 6, 0] in a game constructed via ConnectFour() Unexpected return value from str() after dropping in columns [6, 1, 0, 3, 0, 5, 5, 3, 3, 0, 5, 1, 4, 6, 3, 0, 0, 6, 1, 0, 5, 2, 6, 5, 6, 6, 6] in a game constructed via ConnectFour() (sorry for bad formatting)

wasn't sure if my program was properly finding the winner or not, but it doesn't seem like the bots complaining about it. Also, do you have any suggestions for how we can use .join() in the str function? When I tried I got the error TypeError: sequence item 0: expected str instance, list found

Nathaniel A. Newman (Nate), 2022-12-09 14:30

I've been getting this error message when I try to submit my assignment

../robot: line 37: ((: Command 124: syntax error in expression (error token is “124”)

I'm not sure how to fix this, unfortunately

> Jeffrey Bergamini, 2022-12-12 19:03
>
> Hi Nate,
>
> Just a robot typo, but thanks for letting me know.

Isabella L. Turner (Bella), 2022-12-10 16:53

Thanks for a good semester, hopefully will finish this assignment next week

Nathan E. Pedrotti, 2022-12-12 09:36

Hey Jeff, I have a few questions.

When I run the gui it doesn't work and I get this error.

File “/home/nepedrotti/connect_four.py”, line 29, in bool

```
  possiblewins.append(self._board[self._slot[0]][self._slot[1] + i])
```

AttributeError: 'ConnectFour' object has no attribute '_slot'

self._slot is the list I use to log the last slot that was filled. It works fine in the repl.

Also a few of the errors I get when turning in the program say it gets an unexpected value from board(), when I create a game and drop those inputs I can't find anything wrong with it. Could you check this out for me?

Jeffrey Bergamini, 2022-12-12 19:15

Hi Nathan,

Looks like you don't establish the `_slot` attribute until `drop()` is called at least once. You should probably give it an initial value in the constructor.

---

assignments/assignment_13.txt · Last modified: 2022-11-25 22:27 by Jeffrey Bergamini