Contents

Overall

minor or major points seen throughout the package.

- L2 - Pep8 violation, should be two lines between a class and anything else
- L4 - L6 why empty? - these should be docstrings to explain the class as a whole.

connect4_view.py

- init - not required, it's not doing anything
- prompt_player_to_move()
  - doesn't do any type checking e.g. an input of "stuff", will through a type error when int() is called.
  - however type checking would make the view more "logical" which leads to a view with logic which isn't good, they should be "thin" IMHO
  - L32 utilizing a `while True` here allows the view to take complete control of the flow of the program. Would prefer to see the logical validation of input occur within the controller. In my humble opinion views should be really "dumb".
  - why is the param `color` required here? it's not utilized within the function and shouldn't be required.
- L61 - L62 any reason for not combining these?

connect4_model.py

- L12 - feels a bit complex for tracking players, a list of lists. might be better to use a dictionary.
- L47 - L63 place_token
  - Why do I need to pace to color of the token, L11 defines a property for the active player. wouldn't it be better to automagically place the correct color based on the active player?
  - You're calling the same function twice, `self.is_column_open()` why not store as a var and check it's value once?
  - color param seems to be case sensitive, probably should add a `.lower` on it.
- L76 slightly odd to call type(1) to get `int` why not `type(player) == int`
- L85 - L86 multiple line doc strings, first line should end with final quotes on a new line :-)

connect4_controller.py

- L41 - L46 add_new_player()
  - L46 why reassign the color vars if you initialize within the model with the correct "color".
  - moot point for this assignment, but it's probably generally dangerous to work with "raw" input from the user, might want to sanitize this, ala regex.
  - also design wise, in my humble opinion your controller should never set a property of the model instance directly. There should be a model method to set the property. It's kind of like letting the user of a soda machine to set which change it's gonna receive $1. It seems like overkill for this assignment but it's part of the design paradigm.
- L20 going back to my earlier comment about setting a piece the model could automagically set the color. There would be less to track here, if that was built it.
- same comment from view prompt_player_to_move(), no type checking, I strongly believe that a string as input will crash the program.
- L78 - L79 kind of hard to read these, might suggest each logic statement on it's own line.
- L99 - what is this for? from what I see `k` and `col` aren't used at all.
- L97 - 106 diagonal check
  - I'm not sure this checks for "negative" slopped win condition, quick test via a python shell, see iPython Shell output.

- check_connect_four() is pretty long, it could be broken down further. You could leverage your model here. via a get_row or get_column method.

iPython Shell output

```
In [3]: %cpaste
Pasting code; enter '--' alone on the line to stop or use Ctrl-D.
:game_board = [
:    ["0", "0", "0", "0", "0", "0"],
:    ["0", "0", "0", "r", "0", "0"],
:    ["0", "0", "r", "0", "0", "0"],
:    ["0", "r", "0", "0", "0", "0"],
:    ["r", "0", "0", "0", "0", "0"],
:    ["0", "0", "0", "0", "0", "0"],
:    ["0", "0", "0", "0", "0", "0"]
:]
:
:def test(game_board):
:    gb = game_board
:    for r in range(4):
:        for k, col in enumerate(game_board):
:            for i in range(3):
:                token = gb[r][i]
:                if token == "r" or token == "b":
:                    if token == gb[r + 1][i + 1] and \
:                                    token == gb[r + 2][i + 2] and \
:                                    token == gb[r + 3][i + 3]:
:                        return True
:--
In [4]: test(game_board)
# expected true
In [5]: %cpaste
Pasting code; enter '--' alone on the line to stop or use Ctrl-D.
:game_board = [
:    ["b", "b", "0", "0", "0", "0"],
:    ["b", "r", "b", "r", "0", "0"],
:    ["r", "b", "r", "0", "0", "0"],
:    ["b", "r", "0", "0", "0", "0"],
:    ["r", "0", "0", "0", "0", "0"],
:    ["b", "0", "0", "0", "0", "0"],
:    ["0", "0", "0", "0", "0", "0"]
:]
:# checking with "realistic board"
:--
In [6]: test(game_board)
```

```
# expected true
```