04 Graduate Assignment: Tic-Tac-Toe

Write a Python program named "tictactoe.py" that accepts four named arguments:

- `-s|--state`: The state of the board (type `str`, default "." [9 dots])
- `-p|--player`: The player to modify the state (type `str`, valid "X" or "O", no default)
- `-c|--cell`: The cell to alter (type `int`, valid 1-9, default None)
- `-h|--help`: Indication to print "usage" and exit (no error)

Your program must do the following:

- Print a usage with the `-h` or `--help` argument and exit
- Ensure the `--state` of the board is exactly 9 characters (for the 9 cells) and comprised only of ".", "X", and "O"
- Print a board with the given state where each character of the `--state` is the contents of each cell; when the state for a cell is ".", print the number of the cell
- If provided only `--player` or `--cell`, print the message "Must provide both –player and –cell" and exit *with an error code*
- If provided `--player`, ensure it is only an "X" or "O"; if not, print the expected message and exit *with an error code*
- If provided `--cell`, ensure it is in 1-9; if not, print the expected message and exit *with an error code*
- If provided both `--player` and `--cell`, modify the `--state` to change the given "cell" to the value of "player"; e.g., if `--state` is "." `--cell` is 1, and `--player` is "X", the state should change to "X....." and then print the board with the new state

Expected behavior

In the following, "die" means to print an error to STDERR to and exit with an error code

Print a help message:

```
$ ./tictactoe.py -h
usage: tictactoe.py [-h] [-s str] [-p str] [-c str]

Tic-Tac-Toe board

optional arguments:
  -h, --help            show this help message and exit
  -s str, --state str   Board state (default: ---------)
```

```
  -p str, --player str  Player (default: )
  -c str, --cell str    Cell to apply -p (default: )
```

Print an "empty" (no cells selected) grid with no arguments:

```
$ ./tictactoe.py
-------------
| 1 | 2 | 3 |
-------------
| 4 | 5 | 6 |
-------------
| 7 | 8 | 9 |
-------------
```

Die if given a bad `--state` (too short, not made entirely of "-XO"). Note the feedback where the bad input from the user is repeated in the error message:

```
$ ./tictactoe.py -s abcdefghi
State "abcdefghi" must be 9 characters of only -, X, O
$ ./tictactoe.py --state XXO
State "XXO" must be 9 characters of only -, X, O
```

Die on invalid player:

```
$ ./tictactoe.py -p A
Invalid player "A", must be X or O
```

Die on invalid cell:

```
$ ./tictactoe.py -c 10
Invalid cell "10", must be 1-9
$ ./tictactoe.py -c 0
Invalid cell "0", must be 1-9
```

Die on a cell argument that is not an `int` (note that `argparse` will create this if you indicate the "type" as `int` for the argument):

```
$ ./tictactoe.py -c foo
usage: tictactoe.py [-h] [-s str] [-p str] [-c int]
tictactoe.py: error: argument -c/--cell: invalid int value: 'foo'
```

Print a valid state:

```
$ ./tictactoe.py -s XXO..X.OO
-------------
| X | X | O |
-------------
| 4 | 5 | X |
-------------
| 7 | O | O |
-------------
```

Mutate an initial state as described by `--cell` and `--player`. For instance, starting from the default state, change cell 2 to "X":

```
$ ./tictactoe.py -p X -c 2
-------------
| 1 | X | 3 |
-------------
| 4 | 5 | 6 |
-------------
| 7 | 8 | 9 |
-------------
```

From a state of "XXO..X.OO", change cell 7 to an "O":

```
$ ./tictactoe.py -s XXO..X.OO -p O -c 7
-------------
| X | X | O |
-------------
| 4 | 5 | X |
-------------
| O | O | O |
-------------
```

Die if the `--cell` is already taken:

```
$ ./tictactoe.py -s XXO..X.OO -p O -c 1
Cell 1 already taken
```

Hints

I'd highly recommend you use `new_py.py -a tictactoe` to initialize a script with examples of how to create a Python program that uses the standard `argparse` module to get named, command-line options for strings, integers, and flags. This program does not use positional parameters or flags, so remove those and modify/copy the str/int arguments to match the three options described above. Note the suggestions for "type" and "default."

Use the test suite. The tests are written in the order in which I'd suggest you attack the problem. The first tests look for "usage" which you get for free if you use `new_py.py -a` because `argparse` automatically handles the `-h|--help` flags. Note that printing help does not result in an error code exit value.

The next test looks to see if you can make a Tic-Tac-Toe grid with numbered cells when given no arguments. Using what you learned from `grid.py`, make a 3x3 grid with cells going from 1 to 9.

Next try to detect a bad `--state` argument – one that is not exactly 9 characters long and comprised just of ".XO" characters. You can use a regular expression if

you like, but you'll need to `import re` to do so. Otherwise, just look at the `len` of the state and see if all the letters in the string are in the expected range of letter similar to the `vowel_counter.py` program.

Next check that `--player` is only "X" or "O".

Then check that `--cell` is in the range 1 to 9 (inclusive).

Then handle the instance where you got one of `--player` or `--cell` but not both. Try something along these lines:

```
>>> x = None
>>> y = None
>>> any([x, y])
False
>>> all([x, y])
False
>>> x = 1
>>> y = 1
>>> any([x, y])
True
>>> all([x, y])
True
>>> x = None
>>> y = 1
>>> any([x, y])
True
>>> all([x, y])
False
>>> any([x, y]) and not all([x, y])
True
>>> x = None
>>> y = None
>>> any([x, y]) and not all([x, y])
False
```

Once you can create a default grid and accept valid values for cell and player, mutate the state for the given cell from the initial value to the new value keeping in mind that `--cell` will be 1-based and the actual position in the state will be 0-based (because it is a string/list).

Finally refused to mutate the state at a `--cell` if that position has already been set to an "X" or "O."

Remember you do get partial credit for the tests that pass!