Functions

Estimated time for completion: 30 minutes

Overview:

It's time to right a wrong that has been committed upon the developer community at large. The code we wrote for the python language exercise (the tic-tac-toe game), which had no functions, was seriously bad. Let's rewrite it to be more correct by encapsulating much of it within functions.

We will also explore some more advanced function usage not captured in that simple tic-tac-toe game.

These instructions assume you have **Python 3** installed. You can use any OS you choose (OS X, Windows, or Linux).

Goals:

- Encapsulate reusable functionality within functions
- Return multiple values from functions
- Pass arguments to functions
- Cleanup monolithic code by refactoring to functions

Part 1 – Implement main()

Open the **game.py** file (or project if you are using PyCharm) in **Functions/Before/functions_before**. This code is identical to the solution code from the language basics lab. It went to great pains to avoid using functions and other data structures. Your job is to define a **main** method. Move the entirety of the code into that method, and then refactor it to look like the code listed below. Each method bolded below will be a method you write.

Note:

- **main()** will not be called until you call it. You should do using the import friendly syntax of

  if __name__ == "__main__": main()
- You can rewrite the **check_for_winner** code to be much cleaner by making a list of the sums and asking simply:
  - if 3 in sums: x is winner
  - if -3 in sums: o is winner
  - else no winner

game.py refactoring target (new method details not shown):

```
def main():
    # Print out cheesy welcome
    print_welcome_message()

    x, o, e = get_placeholder_values()
    game = build_empty_board(e)

    print("The board is set:")
    print_board(game, x, o)
```

```
turn = 1
active_value = None
winner = None


while not find_winner(game):
    # print who's turn it is (X or O)
    print_current_player(turn)


    # get the value of the player who's turn it currently is.
    active_value = get_value_for_current_player(turn, x, o)
    row, col = query_placement_from_user()


    # verify that entry is not already chosen
    if cell_already_played(row, col, game, e):
        print("Sorry, that position is already occupied")
        print("try again")
        print_board(game, x, o)
        print()
        continue


    # play the cell
    play_cell(game, row, col, active_value)


    print("Excellent choice:")
    print_board(game, x, o)
```

```
        print()


        # check for a winner
        print("We're checking for a winner...")
        winner = find_winner(game)
        if not winner:
            print("No winner yet, keep going!")
            print()


        turn *= -1


    # Announce winner and show the winning board.
    print("We have a winner! It's the " + winner + "'s!")
    print("Way to win the day")
    print()
    print_board(game, x, o)
```

Play a few rounds to make sure you can still play the game successfully.