# Python notes for Assignment 2

See also the notes for assignment 1, which we mostly won't recapitulate here.

*Beginning matter*

**tuples:** Tuples are like lightweight objects, where each item in a sequence has some different meaning. Delimited by parentheses, with values separated by commas -- for example, (2, 3). You can access individual elements of a tuple the way you would an array or list, using square brackets, like a[1] to get the 3 if a = (2,3).

*read_board*

**dictionaries:** Dictionaries are hash tables, the equivalent of HashMaps in Java. You can efficiently look up a key and get its associated value using the syntax `my_val = my_dict[my_key]`. This code shows initializing a dictionary with keys of characters that can retrieve integer-valued constants.

**dictionaries and get():** You can normally just say mydict[key] to get the associated value, but you may want some default behavior if the key is not found. This code shows how that can work - the first argument of get() is the key, and the second is the default value to return.

*find_winner*

**board == BLACK:** Various operators can apply to a whole array at once, including ==, which returns an array with 1 in the spaces where equality is true and 0 otherwise. Here, this is used to pass an array of 0's and 1's indicating matches to numpy.count_nonzero in order to succinctly count the matching spaces.

*generate_legal_moves*

**continue:** This keyword exists in Java as well, and it means "go immediately to the top of the loop; we're skipping the rest of this iteration."

*can_capture*

**for i, j in blah:** When iterating over tuples, you can automatically assign the different parts of the tuple to different iterator variables. If blah consisted of two-element tuples, "for i, j in blah" would iterate through tuples of the form (x, y) and assign x to i and y to j.

*captures_in_dir*

**friendly_color = WHITE if white_turn else BLACK:** This is the Python "ternary operator," the equivalent of friendly_color = white_turn ? WHITE : BLACK in other languages. It's a succinct way of checking a conditional and evaluating to one value if true, and another if false.

*minimax_value*

**float("-inf"), float("inf"):** These create floating point constants that behave like negative infinity and positive infinity for the purpose of comparisons. Constants like this make it easy to initialize variables to values that will definitely be beaten by the first real data they see.

*get_player_move*

**index:** A built-in list function for finding the index of a matching element. Not fast, since it needs to scan the list, so if you planned to do this a lot, you would want to consider a constant-time approach for checking membership, like a dictionary or set.

**len:** Built-in function for finding the number of items ("length") of a list, or a dictionary or set as well. Quite fast because the list remembers this quantity, turning it into a lookup instead of an iteration down the list to count.