A.

a. `player`: String (`str`).

b. `hasWon`: Function (`function`).

c. `m`: List (`list`). (The list contains two integers. In some cases, `m` is `None`, which has the type `NoneType`. But I did not insist on this level of detail, when grading.)

d. `board`: List. (The list contains three lists, each containing three strings.)

e. `board[m[0]]`: List. (The list contains three strings.)

f. `board[m[0]][m[1]]`: String.

g. `m == None or board[m[0]][m[1]] != " "`: Boolean (`bool`).

B.

```
# Returns the sum of the given letters, preserving the case of the former.
# That is, rotates the former letter through the alphabet by the latter,
# keeping its case the same. For example, addLetters("c", "K") returns
# "m", while addLetters("W", "G") returns "C".
# Input: English letter (upper- or lower-case). English letter (upper-case).
# Output: English letter (of same case as the first input).
def addLetters(a, b):
    bIndex = ord(b) - ord("A")
    if ord("a") <= ord(a) <= ord("z"):
        offset = ord("a")
    else:
        offset = ord("A")
    aIndex = ord(a) - offset
    newIndex = (aIndex + bIndex) % 26
    return chr(newIndex + offset)
```

C.

We compute

$$\begin{aligned} 70 &= 64 + 4 + 2 = 01000110_2, \\ 117 &= 64 + 32 + 16 + 4 + 1 = 01110101_2, \\ -117 &= 10001010_2 + 00000001_2 = 10001011_2. \end{aligned}$$

Then we add 70 to -117 (showing all work) to obtain 11010001_2. What number is this? We know that it's negative, because it begins with 1. If we negate it, we get

$$-11010001_2 = 00101110_2 + 00000001_2 = 00101111_2 = 32 + 8 + 4 + 2 + 1 = 47.$$

Thus the answer was -47, which is indeed $70 - 117$.

D. Yellow is $(255, 255, 0)$, while green is $(0, 255, 0)$. Chartreuse could be halfway between yellow and green, which would be about $(128, 255, 0)$. In binary this is $(10000000, 11111111, 00000000)$, and in hexadecimal it is $(80, FF, 00)$.

E. `encryptCaesar` is a special case of `encryptSub`, in that `encryptCaesar` can be implemented using `encryptSub` with a particular substitution. The substitution looks like the ordinary alphabet, but rotated around, by an amount that depends on the Caesar cipher key. Here is some code.

```
def encryptCaesar(message, key):
    keyIndex = ord(key) - ord("A")
    sub = []
    for i in range(26):
        sub += [chr((i + keyIndex) % 26 + ord("A"))]
    return encryptSub(message, sub)
```

F.

a. There are four ways to win in each row, three ways to win in each column, 12 diagonal ways to win, and 12 anti-diagonal ways to win. So there are

$$6 \cdot 4 + 7 \cdot 3 + 12 + 12 = 69$$

ways to win. (Here, I am counting only ways to win with four contiguous pieces, rather than with more than four contiguous pieces. These ways to win exactly correspond to the tests that have to occur in part b of the problem. Part a is really just a "warm-up" for part b.)

b.

```
for i in range(6):
    for j in range(4):
        # Test row i, starting in column j.
for j in range(7):
    for i in range(3):
        # Test column j, starting in row i.
for i in range(3):
    for j in range(4):
        # Test the diagonal descending to the right from board[i][j].
for i in range(3):
    for j in range(3, 7):
        # Test the diagonal descending to the left from board[i][j].
```