Common mistakes

**mode isn’t the average of the first top two values**

In other words, \( \text{mode}([1, 1, 1, 2, 2, 2, 3, 3, 3]) \) should be 2, not 1.5.

**Beware of slow code**

Using `list.count()`

Don’t use `list.count()` for more than a few elements (each call goes through the full list):

**Unnecessary looping**

This was most common in problem 4. Most people corrected their solution to be quadratic, instead of cubic:

But you can do even better (we didn’t expect you to find this for the quiz)!

**Proof:**

\[
\begin{align*}
x^2 + y^2 &= z^2 \\
x + y + z &= p
\end{align*}
\]

\[
\implies (p-x-y)^2 = x^2 + y^2 \implies y = \frac{p^2 - 2px}{2(p-x)}
\]

\[
\left( \text{verification step: } x^2 + \left( \frac{p^2 - 2px}{2(p-x)} \right)^2 = \left( p - x - \frac{p^2 - 2px}{2(p-x)} \right)^2 \right)
\]

**Useless materialization**

The `dict.keys()` method creates a list, like `range()`. Use `dict.iterkeys()` instead, or implicit iteration. Similarly, don’t create a list when an iterator is enough:
Useless copying  sorted and reversed are most useful on iterators; for regular lists, use the in-place methods:

Before
vals = sorted(vals)
vals = reversed(vals)

After
vals.sort()
vals.reverse()

Repeating dictionary lookups  Use dict.iteritems() or dict.itervalues() instead:

Before
maxfreq = max(freqs[a] for a in freqs)
modes = [num for num in freqs if freqs[num] == maxfreq]

After
maxfreq = max(freqs.itervalues())
modes = [num for (num, freq) in freqs.iteritems() if freq == maxfreq]

Don’t cargo-cult

Once a variable is a float, it stays a float. Converting to float after dividing is pointless.

Before
acc = 0
for mode in nums:
    acc = acc + mode
acc = float(acc)
return float(acc / len(nums))

After
acc = 0
for mode in nums:
    acc = acc + mode
return float(acc / len(nums))

# Or better:
return float(sum(nums)) / len(nums)

Refactorings

Here is a list of simple transformations that can be applied to many solutions:

Use list.sort() and sorted()’s optional arguments

Before
vals.sort()
vals.reverse()

After
vals.sort(reverse=True)

Use ** 2 instead of ** 0.5

Computing a square is cheaper (on small numbers) than computing a square root, and arguably clearer (plus, on large numbers, the square root test can overflow).

Before
z == (x ** 2 + y ** 2) ** 0.5

After
z ** 2 == x ** 2 + y ** 2

Use comprehensions

... instead of explicit loops

Before
out = list(secret_word)
for i, c in enumerate(secret_word):
    if c not in guessed_letters:
        out[i] = '_'

After
out = [c if c in guessed_letters else '_' for c in secret_word]
... and instead of lambdas

```
Before
return sorted(map(lambda tr: [tr[0], tr[1], p - tr[0] - tr[1]], list(out)))

After
return sorted([x, y, p - x - y] for (x, y) in out)
```