Stop doing this when making dictionaries

Online I see people using the following piece of python code, all over the place:

```
freq = {}
for c in string:
    if c in freq:
        freq[c] += 1
    else:
        freq[c] = 1
```

They use this because it somewhat makes sense in python and is very explicit. The problem is, is that it is messy. It's that simple, it's just messy code.

Take the following C++11 code:

```
std::unordered_map<std::char, int> freq;
for(char& c : str) {
    freq[c]++;
}
```

You may question the above code if you are a python programmer and say, "but what if the character hasn't been seen yet, surely and error will be thrown?". That's the magic of a map, it just lets you create a new "key" if it doesn't already exist!

This is similar to the following in python:

With a list:

```
x: list = []
x[1] = 1 # IndexError: list index out of range
```

With a dict:

```
x: dict = {}
x[1] = 1 # Entirely legal and allowed
```

So if you can do that with a `dict`, surely you should be allowed to do it with a `+=` sign.

Don't fear, because you actually can!

```
from collections import defaultdict

x: defaultdict = defaultdict(int)
x[1] += 1
```

This can now be implemented in a near mimic of the C++.

Let's implement character counting like we did earlier and do a speed test too!

In [1]:

```python
def dictCount(s: str):
    freq = {}
    for c in s:
        if c in freq:
            freq[c] += 1
        else:
            freq[c] = 1
    return freq
```

In [2]:

```python
from collections import defaultdict

def defaultDictCount(s: str):
    freq = defaultdict(int)
    for c in s:
        freq[c] += 1
    return freq
```

In [3]:

```python
s = """Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Sed dapibus dignissim lectus, at tristique odio scelerisque non.
Vestibulum scelerisque rhoncus elit, nec blandit libero eleifend id.
Etiam sapien justo, viverra eu tristique vel, facilisis vitae ligula.
Sed nulla risus, molestie non tellus a, consectetur placerat mi.
Pellentesque a feugiat diam, quis blandit libero.
Donec eu metus purus.
Nullam vel aliquet odio."""
```

In [4]:

```python
%timeit -n 50000 dictCount(s)
```

```
45.7 µs ± 294 ns per loop (mean ± std. dev. of 7 runs, 50000 loops each)
```

In [5]:

```python
%timeit -n 50000 defaultDictCount(s)
```

```
40.2 µs ± 377 ns per loop (mean ± std. dev. of 7 runs, 50000 loops each)
```

As you can see, `defaultdict` is marginally faster, but also (arguably) cleaner too. I personally prefer it because it also forces static typing like in C++. Some people don't like static typing, and I have to admit, I didn't either for a long time, but it genuinely makes a lot of sense and it makes debugging code an order of magnitude easier. It is also extremely useful when linting programs.