

# Re: Comparing Dictionaries

---

*Source:* <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2007-07/msg03227.html>

---

- *From:* "Martin P. Hellwig" <[mhellwig@xxxxxxxxxx](mailto:mhellwig@xxxxxxxxxx)>
  - *Date:* Sat, 28 Jul 2007 15:43:14 +0200
- 

Kenneth Love wrote:

<cut>

That should teach me not to change working code at the same time I am writing unit tests. Even so, I realize it won't be the last time I do something so silly. Yes, I know about TDD's "write the test first", but I'm not comfortable with the philosophy of these new fangled unit tests yet.

<cut>

I've been using python for about 3 years now, it was the first programming language I learned and I haven't done any other since (well that is if you dismiss sql and shell scripting on various platforms).

Though in my daily job I have some coding tasks, I do not see my self as a programmer since in my perspective a programmer is someone who can write good code regardless of the language. My code isn't any good, it is barely functional (so I really needed unit-test way of coding, for my own protection and not even because it gives more maintenance security for the project).

But the funny thing that I have seen in the development scene is that writing tests first and code later is a lot easier when you have a technical specification to base it on. A technical specification is of course based on a functional design. A functional design is written on the base of the assignment and the scope definition.

The scope and design can change in the course of the project but good practice is usually to do changes in another release. And that is what software change management and release management is for.

Still there are tons of reasons why you shouldn't follow blindly the best practice, since best practice is founded on theory molded around the most occurring reality. And the only thing that I am absolutely sure of is that reality differs enough from person to person and situation to situation, that forcing the same practice to every person/situation is a guarantee for wasting time and energy.

IMHO there are two extreme basic types of programmers, on the left are the ones that have a problem them self, solve it while happily hacking along the way, leaving the design part for a later stage, on the right are people that know beforehand on an almost bit level what a program is supposed to do, already have a design and aren't really that interested in the problem that their code should solve in the first place.

Me I am a bit schizophrenic, I move in between of them, on a hourly base. So for me t