Theory: Avoiding bad comments

6 minutes 4 / 7 problems solved

Start practicing

Current topic:

Avoiding bad comments 3

Topic depends on:

Stage 1

Comments 17

As you already know, a Python feature (beloved by all Python programmers) is its **well-readable syntax**. However, apart from the syntax itself, there are other important things that contribute to the readability of your program. We assume that you are already familiar with **comments** and how they help in learning a new language.

In real programs, comments become especially important as the program gets bigger and more complicated. Without them, things may get confusing even for you within a few months after writing the program, not to mention other developers who see your code for the first time. However, there is also a downside of making comments, meaning that more is not necessarily better, and we will discuss it below as well.

§1. When not to write comments?

This may sound strange but sometimes it's better not to write comments at all. Carefully written, they can indeed contribute to the readability of your program, but it doesn't mean you should include them wherever you can. On the contrary, many programmers are convinced that a good piece of code doesn't require any comments in the first place, for it is so transparent and accurate. That is what we all should aim at. So, if code can be made self-explanatory, comments are unnecessary, and it's better to change the code. Let's highlight cases when developers need to comment less.

- If a comment **explains a variable/function**, you can usually delete the comment and **explicitly name** the variable/method itself. Compare the following lines of code:

```
n = 0  # number of participants
participants_num = 0
```

- Avoid writing **obvious comments**, like the one below. They make your code redundant and even harder to read. You should always seek following the D.R.Y. (don't repeat yourself) principle, not W.E.T. ("wrote everything twice" or "wasted everyone's time" for more cynical ones).

```
age = 20  # age is 20
greeting = 'hello'  # assign 'hello' to greeting
```

- If you see a way to alter your code so that **comments** would become **unnecessary** - you should do that.

That is, if you can avoid commenting — you'd better do. Then your code would be clean, wouldn't be overloaded with unnecessary details and wouldn't become more complicated rather than clearer for readers to understand.

§2. How to write good comments?

Now, let's turn to cases when you decide to write a comment. The main thing to remember then is that comments should be easily **understandable** by anyone, be that Future You or some other programmer. Here are some tips on how to achieve it:

- Generally, comments should answer the question "**why**" as opposed to "**what**". However, it may be **useful for beginners** to write comments for themselves explaining ***what*** the code does, especially when using a newly learned syntax expression, e.g.:

```
result = 15 % 4  # % is used for division with remainder
```

- Make sure that your comments do not **contradict the code** (that happens more often than you can imagine!). Wrong comments are worse than no comments at all.

```
# decrease the counter
counter += 1
```

- Do not forget to **update** the comments if you modify the code. It will only confuse the reader instead of helping them. In the example below the variable "counter" used to be designated as "i"; the programmer changed its name in the code but not in the comment.

```
# i is incremented in case of errors
counter += 1
```

Following these pieces of advice, you can write code that is clean, organized, easy to understand, and pleasant to read.

§3. Conclusion

When annotating the code, it's important to know where to draw the line. Both **over-** and **undercommented programs** can be difficult to understand, resulting in wasted and unpleasant time spent working with such pieces of code. So, you should always try to write comments carefully and **only when necessary**.

The simplest way to learn to do so is just by doing it. It's a good idea to start practicing when you only start coding because you will get used to it, and by the time some more complex problems should be solved, you will know how to write comments properly. From now on, try to **include simple comments** in your code to explain difficult moments that took you a while to understand. It is also useful to get back to **review** your **older programs** and see how they (including comments) could be enhanced.

Report a typo

1179 users liked this theory. **17** didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Table of contents:

Comments (3) Hints (0) Useful links (0) Show discussion