practical_exercise_4, Methods 3, 2021, autumn semester

Lau Møller Andersen

5/10/2021

Exercises and objectives

The objectives of the exercises of this assignment are:

1) (First hour) Find someone who is **not** in your study group and take turns doing *over the shoulder* peer review for one another https://smartbear.com/learn/code-review/what-is-code-review/

2) (Second hour) Carry on with your assignments 1 and 2

3) (At the very end). Evaluation

Background

Code review is an important skill and practice in both academic and business settings.
It can save time and money - there is also academic articles comparing the different kinds of code review: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-sen.2020.0134

It can be an informal way of learning code quickly and by being forced to explain your code, you will yourself realise whether your code makes sense or not.

Today, we are doing the *over the shoulder* type, but there are others as shown here: https://smartbear.com/learn/code-review/what-is-code-review/
I can specifically recommend *pair programming*, see also here: https://www.rabitse.com/blog/peer-code-review/

Here are some general guidelines for things to keep in mind https://www.springboard.com/blog/software-engineering/code-review-checklist-for-2021/

Exercise 1 - code review

Take turns going through the assignments (1 and 2, part 1) covered so far:

One of you start out as the reviewee and the other as the reviewer.
The reviewee explains why they implemented their code the way they did.
You can choose to let the reviewee complete one exercise at a time. While the reviewee explains their code, the reviewer should thus mostly stay silent. When the reviewee has gone through an exercise, the reviewer offers their thoughts on it.

You can also choose to let the reviewer comment as the reviewee goes along

Remember as the reviewee:

- Be explicit and explain what each line of code does in everyday language

  - check why explicitness matters: https://miguelgfierro.com/blog/2018/python-pro-tips-understanding-explicit-is-better-than-implicit/

- Be open to the reviewer's suggestions
- If something is hard to explain, there may be a better way to code it
- Explain what your plots show
  - e.g. what is on the x-axis, what is on the y-axis, and what is the most important message of the plot

Remember as the reviewer:

- Be specific about what your changes you suggest (e.g. "change this function on line 21", not "your code is very hard to read")
- Suggest annotation where needed or explicit naming if that improves readability
- Be diplomatic in your language
- Remember that some matters may just be style preferences (e.g. "<-" vs. "=")
- Keep the Zen of Python in mind

```
import this
```

```
## The Zen of Python, by Tim Peters
##
## Beautiful is better than ugly.
## Explicit is better than implicit.
## Simple is better than complex.
## Complex is better than complicated.
## Flat is better than nested.
## Sparse is better than dense.
## Readability counts.
## Special cases aren't special enough to break the rules.
## Although practicality beats purity.
## Errors should never pass silently.
## Unless explicitly silenced.
## In the face of ambiguity, refuse the temptation to guess.
## There should be one-- and preferably only one --obvious way to do it.
## Although that way may not be obvious at first unless you're Dutch.
## Now is better than never.
## Although never is often better than *right* now.
## If the implementation is hard to explain, it's a bad idea.
## If the implementation is easy to explain, it may be a good idea.
## Namespaces are one honking great idea -- let's do more of those!
```

Exercise 2 - your assignments

Use the feedback from one another to start doing your revisions of the portfolio assignments
You can also ask me for advice

Exercise 3 - evaluation

Would you like to do this again for one of the exercises?
Advice for the reviewer and reviewee roles?