BEST PRACTICES OF DATA/ CODE ORGANIZATION

Automating everything

- Little automation is better than no automation
- It's more work to do things properly, but it could save you a ton of aggravation down the road

What to automate?

- what you're trying to do
- what you're thinking about
- what you're seeing
- what you're concluding and why

Good vs. bad programming

“Any fool can write code that a computer can understand”

“Good programmers write code that humans can understand”

-Martin Fowler, 2008

Bad programmer explains him/herself with comments, good programmer explains him/herself with code

https://en.wikiquote.org/wiki/Martin_Fowler

Good code = Clean code

- Follow coding conventions
  - **PEP-8** for Python, **PSR-2** for PHP, google "<language_name> coding conventions" for more
  - **Google's R Style Guide**, **R style guide by Hadley Wickham**

- Clean code is
  - Understandable at first glance
  - Neat and elegant
  - Unambigious
  - Not necesserily computationally efficient
  - Self-explanatory
  - Maintainable

http://www.cbs.dtu.dk/courses/27610/clean_code_index.html#clean-code-and-refactoring
https://www.python.org/dev/peps/pep-0008/
http://www.php-fig.org/psr/psr-2/
https://google.github.io/styleguide/Rguide.xml
http://adv-r.had.co.nz/Style.html

Bad code

- Full of “magic” – variables/values noone can understand
- Cluttered, or too loose
- Redundant
- Poorly commented
- Does not follow conventions
- Hardly maintainable

Code represents you – don’t write a bad code

Good vs. Bad code

BAD

```
for i in data:
  if i.status() == "premium":
    dis = 20
  elif len(i.basket()) >= 3:
    dis = 15
  else:
    dis = 0
```

GOOD

```
ITEMS_DISCOUNT_COUNT = 3

for customer in customers:
  discount = 0.0
  if customer.status() == "premium":
    discount = 0.20

  items_in_basket = len(customer.basket())
  if items_in_basket >= ITEMS_DISCOUNT_COUNT:
    discount = 0.15
```

Good variable names

Variable names – nouns

- informative
- unambigious
- descriptive
- choose and follow conventions
  - `underscore_convention`
  - `camelCaseConvention`
  - `dot.convention`
- variables are in lower case, constants are in UPPER case

BAD

```
# elapsed time in days
d = 12
delay = 10000
name_string = "Jerry"
```

GOOD

```
delay_ms = 10000
elapsed_time_in_days = 12
name = "Jerry"
```

Good function names

- Function names – verbs
  - "verb first" rule, e.g., **`print_full_name`**
  - informative, unambigious, descriptive, etc., as for variables

Question: good or bad?

BAD

```
def print_full_name(a, b, c):
  print(a, b, c)
```

GOOD

```
def print_full_name(first_name, middle_name, surname):
  print(first_name, middle_name, surname)
```

Principles of good code development

DRY

- Don't Repeat Yourself
- Do everything to avoid code repetition!

WET

- Write Everything (more than) Twice
- The first time you write a code, you are writing it for the solution, second for comprehension, third for efficiency and last for your sake

KISS - Keep It Small and Simple. Simplicity over complicity, shorter over longer

https://en.wikipedia.org/wiki/Don%27t_repeat_yourself

Refactoring

Refactoring – making better code

- Make code understandable by other developers. Here we ask ourselves a question; If I would give the code to my grandma, would she understand it?

- Increase readability of the code = reduce cluttering of the code. Make code loose in tight places and tight in loose places

- Globally search-and-replace bad variable/function names

http://refactoring.com/

Project organization principles

- One project = one folder
  - Create readable names for subfolders/code. E.g. "**`00_raw_data`**", "**`01_raw_data_QC`**" etc. [My folder structure]
  - Choose file names carefully. Don't put spaces in file names!

- Be sure to get and keep any/all data and meta-data possible

- Get the data in the most-raw form possible. Keep the original files, names intact. (gzipped) CSV Text format is the most preferable. Convert Excel files to CSV https://github.com/dilshod/xlsx2csv

- Separate data from code. Use relative paths in code. Create multiple **`README.md`**

Project organization principles

- **Script everything**. All analysis steps, including data cleaning (removal of outliers, correcting numbers, typos, renaming columns etc.) should be scripted

- **Scalability and universality** - ask yourself a question, if the data are updated (e.g., additional subjects) or you find some artifact that needs fixing, can you just “press a button” to update? If you work on a similar project, can you reuse your existing scripts with minimal modifications?

- **Document everything**. Text format, human readable. Explicitly tie files together. Have a plan to organize, store and make your work understandable by others