Formative ICA

Roll Number: XXXX

The Functions

No.	Function
1	GC content calculator
2	Complementary DNA strand calculator
3	DNA to mRNA convertor
4	mRNA to protein
5	DNA to polypeptides calculator

The fifth function will compute the number of polypeptides produced by a user-specified DNA.

The Agenda

The following is our plan:

1. Write pseudocodes and real codes assigned to each member. After the codes are generated, they will be executed by all members as a double check.
2. When preparing the poster, make use of the pseudocodes and add necessary details to make the explanations more readable.
3. A meeting will be held to decide the design of the poster.

DDL 2020-05-07

Tools, Skills and Functions

The tools we need are as follows:

- An active GitHub repository to which we push our commit: https://github.com/YU-Zhejian/IBI-ICA-G6/
- An active Git installation on each collaborator: Git-for-Windows, GitKraken or SmartGit.
- An active Python 3 installation on each collaborator: Anaconda 3.
- A Python IDE on each collaborator: Spyder, Visual Studio Code or PyCharm.
- A Markdown editor on each collaborator: Typora, Visual Studio Code or PYCharm.
- WeChat for collaborating.

- Adobe Photoshop for editing poster.

The skills we need are as follows:

- How to manipulate string, etc. under Python 3.
- How to use Python IDE.
- How to interact to each other by Git, locally and remotely.
- How to make posters under Adobe Photoshop.

The functions we need are as follows:

- Regular expression facilitates: can be imported from module `re`.
- `if` statements.
- `for` loops.
- File operations: `open`, `close`, etc.

The Git Workflow

Our Git workflow is modelled after the centralized workflow (Chacon, S. and Straub, B., 2018). There's only one remote repository, and all collaborators push their commits to this repository and pill other's commits from this repository.

Branching is encouraged offline but not online unless necessary (e. g. Huge disagreement within our team). Those who encountered a merging conflict is responsible to solve this conflict. Fork is not recommended.

Efforts to Avoid Collisions

1. We have made a detailed specifications on Python and other scripts on indentation, line endings and encodings. The detailed specifications are listed in the appendix.
2. We have made sure that everyone's familiar with the workflow by practice.

Appendix

Python Specifications

The following specifications are made to make our code more readable:

1. All `python` file should be ended with extension name `py` and under `python 3`.
2. Use `# TODO: blablabla` when pending to do something.
3. The way of indenting pseudocode should be the same as those uncommented codes.
4. Use **FOUR WHITE SPACE** or **ONE TAB CHARACTER** as indentation.

Rules for Git and Other Scripts

1. Within numerous line of codes it is comment that appears the first. Your colleagues will weep without readable comments.

2. This public projects prefers **MERGEING** than **REBASING** and **NEVER PREFORM FORCE PUSH**. If you accidentally committed and pushed something wrong, fix it ASAP and commit & push it again.
3. Use **Markdown** instead of Word or RTF when documenting changes. This allows Git to show what was modified by `git diff` instead of telling us this is a binary.
4. If there are temporary files created by the editor, please add them to `.gitignore` to make Git ignore them.
5. Use **LF** instead of **CRLF** as the line endings. Use UTF-8 instead of ANSI as default encoding and **DO ADD A NEWLINE CHARACTER AT THE END OF A FILE.**

Reference

Chacon, S. and Straub, B. (2018) *Pro Git 2.1.87*, APress. doi: 10.1007/978-1-4842-0076-6. Available from `/docs/progit.pdf`.