## Answer to Q1:

Without source code control it is extremely difficult to collaborate in a project within a development team. It is the best and easiest way to keep track of everyone's codes, create separations in developers' works, and revert changes if necessary. It also provides a unified management system for all codes from all team members. However, its use is not limited to team work. Even in personal projects, source code control is useful to manage the project, keep track of changes, revert them when necessary.

## Answer to Q2:

Branching in Git is very useful when, for example, we want to add a new feature to the main (previous) version or fix some bugs in the codes. It is the best way to keep track of the project's versions. When we create a new branch from the project, we can make as many changes in the code as we want. Therefore, it is possible to keep the main branch intact, while having other branches for different tests.

## Answer to Q3:

Git commit saves the added changes to the local repository. In fact, it applies the updates from the staging area to the repository. Then, these committed changes (or updates) can be made (pushed) to the remote (cloud) repository using Git push.

## Answer to Q4:

I personally try to use Git commit as much as possible. I like to use it often, but not too often. The way I think about it is to identify the checkpoints in writing code or making updates. For example if I write a small function, I make a commit at the end before doing anything else, so I can go back to the code before that function. But I do not commit in the middle of writing that small function, since it does not make any sense and creates too many checkpoints. However, if the function is something big with multiple sections, I may commit a few times when I am writing the function.

```
In [ ]:
```