Welcome to Gitland! You were transferred here with your friend to improve your Git skills! It’s not an ordinary coding dojo. Here you have to cooperate with your partner to solve mysteries of Gitland.

This dojo has specific scenario, some task will be done in parallel some of them belongs to exact person. Initially you will work with our own laptop.

Gitland is full of small creatures called gitties. They can be taught new things by reading Python functions uploaded to GitHub. List of things that gitties need to be taught:

- Introducing themselves - *def introduce()*
- Add two numbers - *def add(a, b)*
- Tell a joke - *def joke()*
- Shout - *def shout()*

This time don't focus on the code too much. Gitties are wiser than python interpreter and are able to fix bugs by themselves. This exercise is aimed to teach you version control, not python.

To help gitties follow steps below carefully.

Section 1 - repository initialization

Decide which person will be a person A and which will be a person B.

Person A has to create a new GitHub repository. Only person A does the steps below. Person B should only watch what A is doing.

1. Create a folder on your computer using terminal. cd into that folder.
2. Initialize a git repository inside that folder.
3. Create a *gittie.py* file inside that folder. (touch gittie.py).
4. Add this file to git repository and commit it.
5. Open web browser. Log into GitHub and find a *New repository* button. Click it!
6. Fill in repository name, make sure that *Initialize this repository with a README* is NOT ticked, and then click *Create repository.*
7. Now the new remote repo is created by Github and it shows some hints to start. Look for " *…or push an existing repository from the command line".* section and paste those commands into terminal and press enter. What have those commands just done?
8. Refresh repository website on github. *gittie.py* file should show up
9. Now we'll grant acces to this repository to person B. Go to *Settings* and than to *Collaborators.*
10. Find person B and click *Add collaborator.*

Now it's time for person B:

1. Check your email. You should find a repository invitation. Accept it.
2. You should be redirected to the repository page. On this page find a *Clone or download* button. Click it!
3. Copy the given url addres.
4. Open terminal. Type *git clone*
5. What has just happened?

Section 2 - syncing changes

Now both of you have a local copy of your repository. Now you'll learn to sync your changes.

Person B:

1. Open *gittie.py* in Atom or Visual Studio Code
2. Implement *introduce* function, that'll print, "Hello, I'm Gittie!".
3. Check *git status* and *git diff.*
4. Add and commit your changes to git repository.
5. Push your changes to github. *git push origin master*

Person A:

1. Pull changes from github. *git pull origin master.*
2. Check what happened. *git log*
3. Open *gittie.py* file.
4. Both of you should have identical files now.

Section 3 - auto merging

Now it's time to work simultaneously.

Person A has implement *add* function and person B has to implement *joke* function. After doing so, each of you has to add, commit and push your changes to github.

Who pushed changes first? Was the second person able to push his/her changes successfully?

Read the *git* output carefully. You'll find instructions there how to push it.

After both of you have pushed your changes successfully, use *git pull* again. Check what happened (git log).

Section 4 - conflicts

Now you'll work independently.

Both of you have to modify *introduce* function. Modify this function body so that it's different than your partners.

Commit your changes. Try to sync your changes. What happened? Try to resolve conflict manually.

At the end of this exercise you should have identical files on both computers and git log should give the same output.

Now think for a while what happened during this exercise. Do you understand all the parts?

Section 5 - stashing & popping

This time you'll try a common scenario - pulling before committing.

Both of you have to implement *shout* function. However, only person B has to add, commit and push it. Person A does not commit anything now.

Person A has to try to pull person's B changes.

Was the pull successful? What's git's output? What's *git stash*?

Try to stash your changes and pull again.

Try to pop your changes from stash.

Make your working directory clean.

What happened during this section? Can you find any real life use case for git stash?