CSUN®

COLLEGE OF
ENGINEERING AND
COMPUTER SCIENCE

Github x Python Workshop

Introduction to Git and Github

Hosted by IEEE CSUN
Student Branch

Contents

1 Introduction

Git is a distributed version control system that helps developers track changes in code, collaborate, and manage projects. This guide will cover all the basics of utilizing git. GitHub is a web platform that uses Git for version control and adds collaboration features. This guide will cover how Github is used to back up but also use it as a collaboration platform.

2 Setup

We need to first install a Git-bash:

Windows: https://github.com/git-guides/install-git#install-git-on-windows

Mac: Install Github on macOS from the App Store or using Homebrew: `brew install git`

Linux: https://github.com/git-guides/install-git#install-git-on-linux

3 Verifying git Installation

Each Operating system has its own method to running its git client.

- Windows: Press the windows key and search up **Git Bash** and open the Git Bash application.
- Mac: Open up the terminal using: `CMD+Space` type in `Terminal`
- Linux: Open up the terminal using `Ctrl+Alt+T`.

In the terminal run `git --version` to verify the installation of git.

4 Configuring git

From now on all the steps for every operating system should be the same. Git requires its users to confirm people's identity to see who wrote the code.
In the terminal type in

- `git config --global user.email "<your_email_address_here>"`

Following that set up your username:

- `git config --global user.name "<your_user_name_here>"`

Typing `git config --list` can help you verify if you set it up properly.

Note: Use your Github email and username to have a seamless transition when working with Github.

5 Creating a Repository

After all this info dump of setting up git. Here is where we can get started. Log in to your GitHub account. Then, on the home page, click the **New** button. Give the repo the name **Python-RPS** and add a README file.

6 Setting up Authentication

Now that you created the repository, you need a way to let Github know that the person acessing your repository is you. This is done by using private and public key pairs.

In the terminal enter the follow command:
`ssh-keygen -t ed25519 -C "<your_GitHub_email_here>"`

There will be a prompt asking you to choose where to save the key pair, leave it at its default location by pressing **Enter.** A second prompt will ask to create a password to further encrypt the keypair, you can press enter to not add a password if you choose so.

Note: Having a password will ask you to enter the password every time you boot up your PC.

For Windows Only: Enter following to start you ssh-agent: `eval "$(ssh-agent -s)"`

Add your private key to the ssh-agent using this command: `ssh-add ~/.ssh/id_ed25519`

Next step is to copy the public ssh key to Github. Firstly run `cat /.ssh/id_ed25519.pub` to print out the key and then head over to https://github.com/settings/keys and add a new SSH key. To verfiy you have done all these steps correctly run `ssh -T git@github.com`

7 Cloning a Repository

Head over to your Github Repository that you have created for this Python Project. There you will find a green button labeled "Code". Click this button and select the SSH key menu. Here you can copy the SSH key for your GitHub repo.

Syntax: `"git clone <URL_here>"`

Using the above you can clone the GitHub repo into your personal system. From here it is simply a matter of opening your programming IDE and creating a python file in the created directory.

8 Storing Changes

When you have a git repo you are bound to make changes that you want to save and log so that you can revert changes later if something goes wrong. How do you save snapshots of your code? In git, we use `git commit`. To start off we run the command `git status`

`-u` to see if there are any changes that need to be saved. If there are unsaved changes we add the files to a staging area where the files are waiting to be commited. We do that by either using `git add <file_name>` or `git add -A` to commit all files. To commit our changes we do `git commit -m "a message about what you did"`. Final step is to store those changed made to the cloud, in our case to Github. We do that by running `git push` where we push the changes.

Note: Doing `git add -A` may add unwanted files. To avoid such senario from ever happening we create a file called `.gitignore` where we can enter the file type, location, file name or all three combined should be ingored.

The following code is what you will be working to do today.

```
import random

# Print game rules
print('Winning rules of the game ROCK PAPER SCISSORS are:\n'
      + "Rock vs Paper -> Paper wins \n"
      + "Rock vs Scissors -> Rock wins \n"
      + "Paper vs Scissors -> Scissors wins \n")

while True:
    print("Enter your choice \n 1 - Rock \n 2 - Paper \n 3 - Scissors \n")

    # TODO: Get and convert user input
    # HINT: Use input() to read from the user and wrap it with int()
    choice = ___

    # TODO: Keep asking until the input is valid (between 1 and 3)
    # HINT: Use a while loop to check if choice is not in the valid range
    while ___:
        choice = ___

    # TODO: Map the user's choice to its name
    # HINT: Use if-elif-else to assign choice_name based on choice number
    if choice == ___:
        choice_name = '___'  # HINT: This should be "Rock"
    elif ___:
        choice_name = '___'  # HINT: This should be "Paper"
    else:
        choice_name = '___'  # HINT: This should be "Scissors"

    print('User choice is:', choice_name)
    print("Now it's Computer's Turn...")

    # TODO: Computer makes a random choice
    # HINT: Use random.randint() with range 1 to 3
    comp_choice = ___

    # TODO: Map computer's choice to its name
    if ___:
        comp_choice_name = '___'
    elif ___:
        comp_choice_name = '___'
    else:
        comp_choice_name = '___'

    print("Computer choice is:", comp_choice_name)
    print(choice_name, 'vs', comp_choice_name)
```

```
    # TODO: Determine the winner
    # HINT: First handle tie, then combinations where Paper, Rock, or Scissors win
    if ___:
        result = "DRAW"
    elif (choice == ___ and comp_choice == ___) or (choice == ___ and comp_choice ==
        result = '___'  # HINT: Paper wins in this case
    elif (___ and ___) or (___ and ___):
        result = '___'  # HINT: Rock wins in this case
    elif (___ and ___) or (___ and ___):
        result = '___'  # HINT: Scissors wins in this case

    # TODO: Print game result
    # HINT: Compare result with "DRAW" and with user choice_name
    if ___:
        print("<== It's a tie! ==>")
    elif ___:
        print("<== User wins! ==>")
    else:
        print("<== Computer wins! ==>")

    # TODO: Ask to play again
    # HINT: Use input() and convert to lowercase
    print("Do you want to play again? (Y/N)")
    ans = ___.___()
    if ans == 'n':
        break

print("Thanks for playing!")
```