GE01 Python, Pair Programming and Version Control

**Joshua Edwards**

**Effort:** Collaborative Assignment [CS3300 Academic Integrity](https://docs.google.com/document/d/1cORsFi1YrqW5ChfJu0G67Fjm8HwEMse47DVqXfEn2n4/edit#heading=h.w1yj4lpdz8sh)  (Pairs)

**REQUIREMENT: At least 20 minutes of pair programming with someone else.**

**Points:** 40 (see rubric in canvas)

**Deliverables:** DO NOT UPLOAD A ZIP FILE and submit word or pdf files.

* **Upload this document with your answers**
* **A screencast video of your pair programming activity**
* **Resume and interview questions**

**Due Date:** See Canvas

**Goals:**

* Communicate effectively in a variety of professional contexts within a team, with customers, creating oral or written presentations, and technical documents.
* Devotion to lifelong learning: Prepare to learn on their own whatever is required to stay current in their chosen profession, for example, learning new programming languages, algorithms, developmental methodologies, etc.
* Utilize pair programming to begin learning python.

Names of the person you collaborated

|  |
| --- |
| Conner C |

**Description:** Learning how to learn new technologies. This is not about getting everything working perfectly the first time but collaborating, communicating, finding resources and problem solving with others. Most of all do not panic if you run into issues. Note the issues and how you resolved them.

Think about what information is helpful to have for the next time you do this.

Find 4 or more resources that could be valuable for a new person getting started with python and version control.

|  |  |
| --- | --- |
| **Brief description** | **Resource** |
| A tutorial from the Python organization, describes basic functionality | https://docs.python.org/3/tutorial/index.html |
| Official Python documentation, useful for locating and learning about functionality with Python | https://www.python.org/doc/ |
| A Beginners Guide to Python from the Python organization, focuses on extremely entry-level installation and basic usage. | https://www.python.org/about/gettingstarted/ |
| Overview of git tags | <https://git-scm.com/book/en/v2/Git-Basics-Tagging> |
| Git Commit command summary1 | https://github.com/git-guides/git-commit |
|  |  |

Start exploring git, github, command line, and python in a virtual environment.

[1 Python and IDE](#_heading=h.7a4jn11vv6wq)

[Install Python](#_heading=h.79csvznoivco)

[Install VS Code IDE](#_heading=h.9gomil77gszl)

[2 Pair Programming Video](#_heading=h.rwvlj4hp6mc7)

[3 Version Control](#_heading=h.3fp0cqgnykx1)

[Set-up git and github repository](#_heading=h.bptpc7j7mx76)

[Add, Commit, Push Practice](#_heading=h.27n2hu32nsae)

[Branching](#_heading=h.tyjcwt)

[Version Control Concepts](#_heading=h.go47xdl2sh5a)

[4 Resume and Interview Questions](#_heading=h.s0jda1wrx8t6)

# 1 Python and IDE

Set up your python and IDE for your python development.

## Install Python

1. Open the command window and check your python version to see if you have it installed.
2. Install python version 3.11 [Download Python](https://www.python.org/downloads/). If on windows and have older version of python you should uninstall first : [How to Uninstall Python](https://www.pythoncentral.io/how-to-uninstall-python/)

## Install VS Code IDE

You can use a different IDE but this is what I will be using in my lectures. This has nice tools to integrate with python, django and databases.

<https://code.visualstudio.com/download>

1. Configure the Python interpreter: In Visual Studio Code, open the Command Palette by pressing `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (Mac). Search for "Python: Select Interpreter" and choose the Python interpreter associated with your virtual environment (e.g., `myenv`).



1. Install the Django extension developed by Baptiste Darthenay: In Visual Studio Code, go to the Extensions view and search for the "Django" extension. Install it to benefit from Django-specific features and enhancements for what we will be doing later.





1. You can use this to edit your python file for practice.
2. Take a screenshot of the ide you have set up and the python file from the repository once you edit it below.

|  |
| --- |
|  |

# 2 Pair Programming

Goal: Improve software quality by having multiple people develop the same code.

Setup:

* One shared computer, alternate roles
* Driver: Enters code while vocalizing work
* Observer: Reviews each line as it’s typed, acts as safety net + suggest next steps

Effects:

* Cooperative, a lot of talking! + Increases likelihood that task is completed correctly
* Also transfers knowledge between pairs

Start learning the basics by going through [Hello, World! - Free Interactive Python Tutorial](https://www.learnpython.org/en/Hello%2C_World%21) by following the instructions below.

* You should spend at least 20 minutes pair programming
* **** Choose video screen-recording software that you can use to capture your discussion and screen. (such as <https://obsproject.com/> )

Where it says exercise code: that means for that section you are doing the exercise at the end of the information.

* Do not copy the solution code. Instead copy your code and paste below. Add any notes that would be helpful.
* Do not worry if you do not finish all the parts when pair programming but you should start pair programming and videoing with lists.
* Complete on your own after the pair programming ends.

|  |
| --- |
| Scan the following sections before pair programming. Take turns summarizing each section to the other. Add any brief notes or examples.  [Hello, World!](https://www.learnpython.org/en/Hello%2C_World%21)  [Variables and Types](https://www.learnpython.org/en/Variables_and_Types) |
| [Lists](https://www.learnpython.org/en/Lists) Review and complete exercise code: |
| [Basic Operators](https://www.learnpython.org/en/Basic_Operators) Review and complete exercise code: |
| Scan the following sections. Take turns summarizing each section to the other. Add any brief notes or examples.  [Basic Operators](https://www.learnpython.org/en/Basic_Operators)  [String Formatting](https://www.learnpython.org/en/String_Formatting)  [Basic String Operations](https://www.learnpython.org/en/Basic_String_Operations)  [Conditions](https://www.learnpython.org/en/Conditions)  [Loops](https://www.learnpython.org/en/Loops) |
| [Functions](https://www.learnpython.org/en/Functions) Review and complete exercise code: |
| [Classes and Objects](https://www.learnpython.org/en/Classes_and_Objects) Review and complete exercise code: |
| [Dictionaries](https://www.learnpython.org/en/Dictionaries) Review and complete exercise code: |

# 3 Version Control

## Set-up git and github repository

Use the command line tool of your preference in your environment. I ended up using command prompt on my windows but also have used windows powershell.I use the generic command tool on my mac.

Here is an example of using the default command prompt



Research

* What is git and github? What does git provide? What does github provide?
* How can you create a github repository from a local folder?
* What documentation could be useful to help understand the commands?

Include resources in the table above.

1. Create a python file in a local folder cs3300-version-practice
2. Create a folder called documentation in cs3300-version-practice that contains this document.
3. Create a github account if you do not have one.
4. Create a github repository that is public from the local folder.

Explain what you did and the commands you used.

|  |
| --- |
| I created a new folder called “cs3300-version-practice” and opened the git gui in that folder.  I then ran these commands:  git init  echo “# test” >> README.md  git add README.md  git commit -m “Initial commit”  git branch -M main  git remote add main <https://github.com/BlackBeltJ/cs3300-version-practice>  git push  Then I copied and pasted this GE01 Word document into the new repo. I used “git status” to check that the changes were found. Then I used “git add .” to add all the changes to a commit. Then I used “git commit -m “Added GE01 documentation folder and Word file”. Then I used “git push” to push the local changes to the remote repository on GitHub. |

Paste a screenshot of your local directory code

|  |
| --- |
|  |

Paste a screenshot of your github repository code

|  |
| --- |
|  |

Paste the url to you github repository code

|  |
| --- |
| https://github.com/BlackBeltJ/cs3300-version-practice |

1. You may need to generate an SSH Key pair to configure remote access to your repositories. Github’s instructions for this process can be found [here](https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent).
2. You may need to set

git config --global user.email "you@email" (email associated with repository)

git config --global user.name "Your Name

## Add, Commit, Push Practice

1. You can just work with updating a python file.
2. Check the git branch and status

git branch

git status

1. Update the file. Before you can commit the version you must add the new file to the index (the staging area)

git add .

git status

1. Record changes to the local repository with a description but first you might need to include the author identity. Then check the status

git commit -m ‘add description’

git status

1. You will add your code, commit and push. Then explore the repository on the remote server, github

git push

git status



## Branching

1. From the command line in your repository on your computer check the log and what branch you are on.
2. Create a branch called sprint01 and check the log and branch

Copy and paste the commands you used

|  |
| --- |
| <- to check the current branches  git branch sprint01 <- to create a branch  <- to check the branches again  git switch sprint01 <- to switch branches  <- to verify switched branch |

1. Switch to sprint01 branch to check out code:

git checkout 'sprint01'

git branch

git status

1. Modify python file and Add the file to the staging area and update the version in your local directory.

Copy and paste the command(s) you used

|  |
| --- |
| git add .  git commit -m “Modified GE01.py on sprint01 branch” |

1. Share the changes with the remote repository on the new sprint01 branch. Go to your github and you will see you now have two branches. Click to view the branches. Now others working on the branch could pull your updates from the sprinto1 branch.

git push --set-upstream origin sprint01

git status

git log



1. Switch to the main branch and update the remote main branch repository with the change from sprint01 branch. Then go to github to see the versioning.

Copy and paste the commands you used

|  |
| --- |
| git push --set-upstream main sprint01  git branch  git status  git log |

1. Tag the main branch ‘v1.0’ then view the tag and push to the remote repository. When you go to the remote repository you should see the tag listed.

Copy and paste the commands you used

|  |
| --- |
| git tag v1.0  git tag -l <- to verify the tag was created  git push main –tags |

For example



## Version Control Concepts

Individually answer each question in your own words, **including any resources you used to help you above.** This will be helpful when you keep technical documentation with your team. **You can use AI to help you understand but answer in your own words.**

3.1 Explain software version control. Address in your description branches, commits, merges, tags.

|  |
| --- |
| Software version control is a technology that allows you to check changes before they are applied, work on a single file from multiple devices, have separate branches for different production lines, and add tags. Branches allow developers to work on separate work which then allows a product manager to review the code before it all comes together and possibly breaks. With version control, you stage your local changes by “committing” them and then you “push” them to the remote repository which updates it on all other computers linked to the repo. Merges happen when you take two branches and merge the code or documents into one of the branches, thus making the other branch obsolete. Lastly, tags are ways to “tag specific points in a repository’s history as being important” (<https://git-scm.com/book/en/v2/Git-Basics-Tagging>). Tags allow developers to locate significant tags which could correspond to versions or big leaps in the code. They can revert back to a previous version if need be. |

3.2 Research what Git is and what its relationship is to software version control. Include how GitHub integrates with git.

|  |
| --- |
| Git is the version control software that allows user to manage their projects’ code and documents. Git is the heart of all other VC hosts like GitHub. GitHub is a web-based program that hosts Git functionality to allow users a visual GUI and a remote cloud-based location to store their repositories. |

3.2 Explain the following commands and include examples: commit, pull, push, add, clone, status, log, checkout.

|  |
| --- |
| commit: “Git commit creates a commit, which is like a snapshot of your repository. These commits are snapshots of your entire repository at specific times. You should make new commits often, based around logical units of change. Over time, commits should tell a story of the history of your repository and how it came to be the way that it currently is. Commits include lots of metadata in addition to the contents and message, like the author, timestamp, and more. (<https://github.com/git-guides/git-commit>) ” (<https://github.com/git-guides/git-commit>). Commit does not update the remote repo, it only stores your changes for future use.    pull: Git pull updates your local code with the most recent version from the remote (cloud) repository.  push: Git push takes the code on your local machine and joins it to the remote repository. This makes the local changes available for everyone to see on the branch you committed from.  add: Git add is used to add specific files to a commit command. “Git add .” adds all changes which is most common, but “git add [file name]” can also be used to add specific files.  clone: Git clone is used to duplicate the contents in a remote repo onto a local machine but it does not link your local machine with the remote repo (use git add remote for that).  status: Git status is used to see if there are any changes to your local repository which would tell you if a commit is needed.    log: Git log is used to view a history of past commits. |

3.3 Explain the difference between a branch and a tag.

|  |
| --- |
| A branch is a collection of code and files that is separate from the main branch and thus can be modified and tinkered with without endangering the production code in the main branch. A tag is a specifier that can be put on branches or commits. Tags allow easy labelling of significant moments in the program’s code history for rollback or checkpoints. |

3.4 Describe at least three benefits of a version control system and include an example for each that would be related to industry.

|  |
| --- |
| If there is ever a need to roll back the production code because of a critical bug, version control makes that easy with tags. You could simply go back in your version history and find a version that you know was working and then restore your code. Then you can troubleshoot the error and hopefully avoid it as you continue in production.  If there is an error that was patched and production continued, but then a similar error comes up, you could use version control to go back and see what was changed that fixed the error.  You can use version control and different branches to test different aspects of a program before full compatibility is introduced. This means that the dev team can focus on making small portions of the program compatible with the rest of the program as opposed to constantly implementing new features that have to be completely tested with the rest of the code.  Version control with Git is a great way to collaborate on the same codebase. With Git, multiple people across multiple teams can all access the same codebase and can work at the same time (hopefully) without disrupting others. |

# 4 Resume and Interview Questions

Create a document that contains the following parts

Part 1: Create a resume to use to interview to be a full stack developer intern that only includes these sections

1. Summary
2. Skills
3. Relevant Experience

Resume -> <https://docs.google.com/document/d/1MJFLq3CFk-aLbqaYs5IHm-SD0-EMgYO2RtNnJW4khEw/edit?usp=sharing>

Part 2: Interview questions you would ask to see if someone would be a good fit on your team. Include at least 4 questions.

**Given a deadline of three weeks, what would your timeline look like for working on and completing a project during those three weeks? What is your work style?**

**When you run into a problem or bug in your code, what is your first response? Where do you go first to attempt to find a solution?**

**Have you worked on a team before? What was the experience like? What is your picture of a perfect team?**

**What is your opinion on the concept of daily team meetings?**

**What are your strengths? What are your weaknesses?**

**How do you handle stress?**