GE01 Python, Pair Programming and Version Control

**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

|  |
| --- |
| David Klunk |

**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** |
| Git cheat sheet | https://education.github.com/git-cheat-sheet-education.pdf |
| Git hub in general | https://github.com |
|  |  |
|  |  |
|  |  |
|  |  |

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) - Camilla.  [Variables and Types](https://www.learnpython.org/en/Variables_and_Types) - David  <https://uccs1.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=7851abbf-d70f-4d25-813e-b1010152eb62> |
| [Lists](https://www.learnpython.org/en/Lists) Review and complete exercise code: - David |
| [Basic Operators](https://www.learnpython.org/en/Basic_Operators) Review and complete exercise code: - Camilla |
| 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.

|  |
| --- |
| In order, I used the command cd to change directory to the folder I wanted to make a repository, then I used git init which initializes this folder as a git repository. |

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/CamillaLucero23/CS-3300-Repository |

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

|  |
| --- |
|  |

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

|  |
| --- |
| Unseen here is the git add . command to stage the commit. |

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

|  |
| --- |
|  |

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

|  |
| --- |
|  |

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 where a project is uploaded in multiple versions. This is for, if for any reason, when a commit is buggy or isn’t up to a customer’s liking, the program, software, or whatever is being worked on, can be rolled back to a commit with a previous tag. Branches are specifically meant to prevent buggy/ unfinished from reaching the main branch. One that branch is not buggy, then it can be merged and “join” the full, complete code again. |

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

|  |
| --- |
| Git is a type of version control software that can downloaded, used, and help to manage a project via version control. It allows for a project to be rolled back, managed by multiple people, and changed within a few commands. Github is a centralized cloud that based on git. It has the ability to make, control, and use git repositories in a centralized location on the web. |

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

|  |
| --- |
| Commit – Usually done after an add, commits any changed data to the local repository. For example, if you are done editing a part of a project, but not done editing another part of that project, you can commit to save a snapshot via “git commit -m ‘new feature’”  Pull – pulls all updated data from the github repository into your local one. If one of your teammates edited a certain thing and you wish to included that edited version, you can use “git pull” to pull all of their updates.  Push – Usually done after a commit, pushes all updated data into the github repository. If you’re done adding all you need to add after pulling your teammate’s code, you need to push it using “git push” to add your updated code to the repository.  Add – adds any information specified to the “stage” to be committed. Done before a commit to specify what information needs to be committed. If you want to add file.txt to the local repository, you need to “git add file.txt” to tell git to stage it for a commit  Clone – Clones the specified repository into a local repository. If you want to clone a repository, you would do “git clone <link>”  Status – Gives a status on the current branch/ master branch. Used for gaining an updated status on yhe current branch using “git status”  Log – Displays a log of commits. Used for gaining insight into what has been committed and what has been changed using “git log”  Checkout – Changes current branch to the one specified. Used for moving from branch to branch via “git checkout branch” |

3.3 Explain the difference between a branch and a tag.

|  |
| --- |
| Branches are used to create separate lines of development that don’t interfere with the current main branch. They allow devs to work on bug fixes, new additions, etc. while also keeping the main branch unaffected.  Tags mark significant milestones based on branches or whatever is committed. |

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

|  |
| --- |
| Change tracking is one of the most obvious benefits, as git/ version control allows for changes to be added, rolled back, or removed based on the company’s desires. An example of this could be adding a new feature, and using version control to roll it back to a previous state, where a software may have worked before and didn’t with the added feature.  Collaboration is another benefit. Version control and branching allows for multiple people to work on the same project without stepping on eachother’s toes. An example could be a game dev team, similar to the Resident Evil Example in Class. If you have a team of 100 or more, having a way of branching off into groups/ individual code work allows for a team to divide and conquer, then allows for the ability to put everything back together again.  Lastly, Rolling back is another. Kind of gone over in change tracking, you are able to remove code all together based on whether a customer wants or needs it anymore. This is helpful in the case of a customer changing their mind. |

# 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

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