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

|  |
| --- |
| Chandler Waller, Tyler Andrews |

**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** |
| Quick explanation of what VS code is, found this on my youtube home page and clicked because I did not know what VS code was coming into this semester, very helpful introduction | https://youtu.be/KMxo3T\_MTvY?si=F2tglSXuGpgfpDUG |
| I searched this before meeting in group for pair programming, helped me learn basics of python | https://youtu.be/kqtD5dpn9C8?si=eG5AfvHtKMKGilR6 |
| The python website that helps beginners(like I was) get up and running | https://www.python.org/about/gettingstarted/ |
| I know this was linked in the exploration but this whole page is also a very useful resource for programmers who understand concepts from other languages trying to translate those skills to python | https://www.learnpython.org/ |
| Git website explaining who they are and what they can do | https://docs.github.com/en/get-started/using-git/about-git |
| Creating github repo from local source | https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github |
| Git commands cheat sheet | https://education.github.com/git-cheat-sheet-education.pdf |

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.

|  |
| --- |
| Edits done for exercise 3 |

# 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)  Python syntax is the most relaxed one I have worked with so far. Printing just requires print(thing to print), and the weirdest part is no semicolon.  print(hello world)  [Variables and Types](https://www.learnpython.org/en/Variables_and_Types)  You do not need to declare variables before using them, python automatically knows what type of variable it is working with. Python does numbers and strings.  variable and types:  mystring = "hello"  myfloat = 10.0  myint = 20 |
| [Lists](https://www.learnpython.org/en/Lists) Review and complete exercise code:  Python is object oriented, and interacting with lists feels very similar to interacting with the object oriented structures we used in java. However, unlike java, elements can be conveniently accessed by standard array notation.  numbers.append(1)  numbers.append(2)  numbers.append(3)  strings.append("hello")  strings.append("world")  second\_name = names[1] |
| [Basic Operators](https://www.learnpython.org/en/Basic_Operators) Review and complete exercise code: Mostly standard to other programming languages. Differences include doing intuitive string math and using \*\* to raise a number to a power.  x\_list = [x, x, x, x, x, x, x, x, x, x]  y\_list = [y, y, y, y, y, y, y, y, y, y]  big\_list = x\_list + y\_list |
| Scan the following sections. Take turns summarizing each section to the other. Add any brief notes or examples.  [String Formatting](https://www.learnpython.org/en/String_Formatting)  This felt very similar to c with the format specifiers.  data = ("John", "Doe", 53.44)  format\_string = "Hello %s %s. Your current balance is $%s."  print(format\_string % data)  [Basic String Operations](https://www.learnpython.org/en/Basic_String_Operations)  Python by far is the most capable language of manipulating string I have ever ran into. You can do intuitive array like operations on them, as well as manipulate them like objects with functions.  \*this solution gave us trouble. Also reading the instructions of the assignment, it looks like we didn’t need to post any solutions for this section.  [Conditions](https://www.learnpython.org/en/Conditions)  Aside from the relaxed syntax, if elif and else statements are the same. Evaluate a condition and execute code if that condition is met. Python has extra capabilities such as checking if something is in something, and then typing the words and, or, not, is instead of their logical operators.  number = 16  second\_number = 0  first\_array = [1,0,0]  second\_array = [1,2]  \*\*This is where group work ended because we ran out of time\*\*  [Loops](https://www.learnpython.org/en/Loops) |
| [Functions](https://www.learnpython.org/en/Functions) Review and complete exercise code:  You have to define your function and then write it below after a colon. I do not think this is great for code organization, curly braces are much easier to see blocks.  return ("More organized code", "More readable code", "Easier code reuse", "Allowing programmers to share and connect code together")  return "%s is a benefit of functions!" % benefit |
| [Classes and Objects](https://www.learnpython.org/en/Classes_and_Objects) Review and complete exercise code:  The classes are very much like java, just with python syntax of defining functions. Otherwise they are the same. The object variables in this case were public and easily changed from the main program, unlike what I have been doing throughout my time a UCCS.  car1 = Vehicle()  car1.name = "Fer"  car1.kind = "convertible"  car1.color = "red"  car1.value = 60000.00  car2 = Vehicle()  car2.name = "Jump"  car2.kind = "van"  car2.color = "blue"  car2.value = 10000.00 |
| [Dictionaries](https://www.learnpython.org/en/Dictionaries) Review and complete exercise code:  I am not sure what the capabilities of the dictionary function is beyond this exercise, but as of right now that just seemed like a simple interface for dealing with a 2D array.  del phonebook ["Jill"]  phonebook["Jake"] = 9382733443 |

# 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 changed my working directory to the directory of the version practice folder. Then I initialized that folder as the repository. I added the file to be committed. Then I made an initial commit.  git init  git add cs3300-version-practice.py  git commit -m ‘initial commit’ |

Paste a screenshot of your local directory code

|  |
| --- |
|  |

Paste a screenshot of your github repository code

|  |
| --- |
|  |
| I took these screenshots after the lab |

Paste the url to you github repository code

|  |
| --- |
| https://github.com/zach-snyder/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

|  |
| --- |
| git branch sprint01  git branch  git checkout sprint01 |

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 status  git add cs3300-version-practice.py  git commit -m 'Branch push'  git push origin sprint01 |

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.

\*I don’t know why my main branch defaulted to master, I did not set that up.

Copy and paste the commands you used

|  |
| --- |
| Git checkout master  Git merge  Git push origin master |

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 push origin --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.

|  |
| --- |
| Version control is all about keeping track of the steps of building your working software. It saves all the versions of your code that you create, so you can go back if needed. Syncing with an online remote repository also serves as a backup server for your code. When you are ready to update your code, you commit your changes. If you want to commit changes without merging to the main program, you would do that in a branch. When you are ready to merge the changes to the main branch, you would use the merge command. In order to keep track of all the different versions of code you are committing, tags can be used as labels for snapshots in time of your software.  I used this to help me with tags: <https://intellipaat.com/community/10898/git-tag-how-to-create-a-tag-in-a-github-repository>  This to help me with branching: <https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging>  This to help me with the SSH key to connect to github: <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>  I used this resource to get started: <https://docs.github.com/en/get-started/getting-started-with-git/set-up-git>  This resource helped me when I ran into an error authenticating:  <https://docs.github.com/en/authentication/troubleshooting-ssh/error-permission-denied-publickey>  And this resource to help me understand pushing to github:  <https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github> |

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

|  |
| --- |
| Git is a tool that keeps track of previous versions of code in a repository. GitHub is an online platform that can store your repositories. Git and Github are simply tools that allow us to do versioning control. |

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

|  |
| --- |
| commit: git commit -m ‘commit description’. This will commit any outstanding changes you currently have to be the most up to date version of your repository locally.  Pull: git pull origin main. This will update the local repository with whatever version was on the github repository at the time of the pull. The main branch can be replaced with whichever branch you want to update.  Push: git push origin main. This pushes the most recent commit of the local directory to the github remote repository. Again you can push whatever branch you want to send the most recent commit of that branch to github.  Add: git add filename. By default added files and updated files will not be included in a commit. They have to manually be added to include the changes on the local machine to be included in the commit.  Clone: git clone ssh-address. This would be used to import a repository locally that already exists on github.  <https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone#:~:text=git%20clone%20is%20primarily%20used,copies%20an%20existing%20Git%20repository>.  Git status: git status. This command shows if there are any changes that have not been committed, and if there are any files changed in the directory that would not be included in the commit because they have not been added yet.  Log: git log. This shows the history of all the repository commits  Checkout: git checkout branchname. This switches the working branch to whatever the name of the branch you typed in is. |

3.3 Explain the difference between a branch and a tag.

|  |
| --- |
| A branch is like a copy of the workspace so you can add changes to the code without changing the current version of the code before you or your team is absolutely ready. A tag is just a label for a commit, and can be used as a version number. |

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

|  |
| --- |
| Github online repository: This is very useful for when multiple people on a team want to see updates and changes. When they are pushed, they can see on github and pull the changes to their local system instead of having to ask their team members to send files back and forth.  Branching for simultaneous coding: In industry there are very complex projects being created. This requires multiple people working on the same product at the same time. Branching allows this to happen while minimizing mistakes or changes not being seen due to being overwritten that alternative solutions would allow.  Rollback: When you commit changes, you can still go back to a previous version if needed. This is a very useful tool to have if a company releases a version that breaks, they can rollback and release a previous version that worked. Then they can troubleshoot off of the working code and it will be much easier to see what potentially went wrong. |

# 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.

Zach Snyder

Computer Engineer

Driven student pursuing a computer engineering major. Software engineer that will not give up on a problem until it is solved. Excited to work on a full-stack development team.

Skills:

* Programming Languages
  + C
  + C++
  + Java
  + SQL
  + Python
* Linux Command Line
* Git/GitHub repositories
* Teamwork
* Strong Determination
* Writing
* Problem solving

Relevant Experience:

* Developed a checkout system for sports charity events
* Created a game in C++ on a team

Interview Questions:

1. How did your passion about programming/software engineering come about? Does problem solving like this excite you?
2. Have you worked on any teams before? If so, how was the experience and did the task get completed?
3. Have you ever encountered any issues with a team (does not have to be a software development team)? How were you a part in fixing those issues?
4. Do you have any experience as a full-stack developer?