GE01 Python, Pair Programming and Version Control

**Tyler Andrews**

**CS 3300 001**

**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

|  |
| --- |
| Zach Snyder, Chandler Waller |

**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** |
| Background on version control and Git | https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control |
| Version control and tools | https://www.spiceworks.com/tech/devops/articles/what-is-version-control/ |
| Python libraries | https://docs.python.org/3/library/index.html |
| Python syntax | https://docs.python.org/3/reference/index.html |
| Python tutorials | https://docs.python.org/3.12/tutorial/index.html |
| Python documentation | <https://docs.python.org/3.12/index.html> |
| Initialize and Import repo with github command line | https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github |

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:  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:  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.  [Basic Operators](https://www.learnpython.org/en/Basic_Operators): +, -, /, \*, %, and \*\* (power of). “hello" + " " + "world” = “hello world”  [String Formatting](https://www.learnpython.org/en/String_Formatting): print("Hello, %s!" % name). print("%s is %d years old." % (name, age)). %s (string), %f (float), %d (integer)  [Basic String Operations](https://www.learnpython.org/en/Basic_String_Operations): length of string includes spaces, letters, and punctuation. String index starts at 0.  [Conditions](https://www.learnpython.org/en/Conditions) : Comparisons made with == not =. Use ‘and’ and ‘or’ instead of && and ||  [Loops](https://www.learnpython.org/en/Loops): while for loops. Can use else statements in loops |
| [Functions](https://www.learnpython.org/en/Functions) Review and complete exercise code:  def list\_benefits():  return ["More organized code", "More readable code", "Easier code reuse", "Allowing programmers to share and connect code together"]    # Modify this function to concatenate to each benefit - " is a benefit of functions!"  def build\_sentence(info):  return info + " is a benefit of functions!" |
| [Classes and Objects](https://www.learnpython.org/en/Classes_and_Objects) Review and complete exercise code:  Objects are variables and functions in one. Classes are a outline to create objects.  car1 = Vehicle()  car2 = Vehicle()    car1.name = "Fer"  car1.color = "red"  car1.value = 60000.00    car2.name = "Jump"  car2.color = "blue"  car2.kind = "van"  car2.value = 10000.00 |
| [Dictionaries](https://www.learnpython.org/en/Dictionaries) Review and complete exercise code:  A list but uses a key to denote the index. Each key is paird with a value.  phonebook["Jake"] = 938273443  phonebook.pop("Jill") |

# 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.

|  |
| --- |
| git init -b main  cd /desiredFilePathHere // commit all files at this file path  git add .  git commit -m "First commit"  git remote add origin REMOTE-URL  git remote -v // make sure everything is working so far  git push origin main  Had authentication error after push origin main. Resolved with:  <https://stackoverflow.com/questions/68775869/message-support-for-password-authentication-was-removed> |

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/S02441112/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 checkout sprint01  git branch  git log |

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 TestProgram.py  git commit -m "v1.0"  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.

Copy and paste the commands you used

|  |
| --- |
| git checkout main  git branch  git merge sprint01  git push origin main |

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 v1.0 |

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 method to save past and changes to reference or fall back onto if necessary. Branches allow new functionality to be created without changing the current main branch that is in production. Commits can be used on each branch in a repository to save changes for that branch (locally). Branches can be merged to keep old functionality and introduce new functionality into a single branch. Tags are added to branches to show major/minor changes to a branch at a certain time/version. |

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

|  |
| --- |
| GitHub is a GUI to host git repositories and uses git as a base to operate upon. Git is version control that racks changes to repositories. |

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

|  |
| --- |
| commit: snapshot of the repository at a specific time.  git commit -m "descriptive commit message"    pull: updates current local working branch. combination of git fetch + git merge.  git pull    push: send committed local changes to cloud  git push origin main    add: adds files in working directory to staging area  git add . // all files in working directory  git add <path> // specific path    clone: create copy of a repository  git clone <url>    status: shows state of working directory and staging area.  git status    log: show commits with time stamps and users  git log |

3.3 Explain the difference between a branch and a tag.

|  |
| --- |
| git branch is a separate version of the main repository.  git tag represents specific points in a branch's history such as a release “v1.2”. |

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

|  |
| --- |
| Accidental merging of a branch and main could be undone with version control.  Accidental deletion or overwriting of a branch can be undone.  Keeping separate branches and versions for each person in a project can be beneficial. |

# 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.

familiar