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

|  |
| --- |
| Hanh Nguyen |

**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** |
| Python standard library | https://docs.python.org/3/library/index.html |
| Beginner Python tutorial | https://www.learnpython.org/en/Hello%2C\_World%21 |
| How to add local code to github through 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 |
| How to set up git | https://docs.github.com/en/get-started/getting-started-with-git/set-up-git |
| How to login | https://cli.github.com/manual/gh\_auth\_login |
| Working with remote repos | https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories |

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) -- Just print statements, very easy, very much like c  [Variables and Types](https://www.learnpython.org/en/Variables_and_Types)  Also easy, not having to declare a data type is weird, also the default print statement does formating |
| [Lists](https://www.learnpython.org/en/Lists) Review and complete exercise code:  Lists share some syntax with arrays and they are both data structure but that is about all they have in common. Lists are dynamically sized and polymorphic by nature. Everything being an object is strange to me |
| [Basic Operators](https://www.learnpython.org/en/Basic_Operators) Review and complete exercise code:  Interesting, everything is an object, and every operator is overloaded. I suppose this makes sense when we do not have to work with primitive data types. I can see why people like python, although I still miss pointers and braces |
| 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)  -- As above  [String Formatting](https://www.learnpython.org/en/String_Formatting)  Works similarly to c, looks easier to use though  [Basic String Operations](https://www.learnpython.org/en/Basic_String_Operations)  Strings seem very versatile, there are so many ways to manipulate them built into the object  [Conditions](https://www.learnpython.org/en/Conditions)  Logical operators and conditional statements, made more interesting by the fact that everything is an object  [Loops](https://www.learnpython.org/en/Loops)  Interesting, Loops in python look more like bash loops than c, very reminiscent of scripting in Unix but this time with an actually good text editor and debugger |
| [Functions](https://www.learnpython.org/en/Functions) Review and complete exercise code:  We have to use a keyword to specify that we are making a function interesting. Other than that, we have parameters and we have return values so nothing too complicated. It looks like functions can return multiple values, and when this is done it returns a list containing the returned objects. The elements of a list can include other lists. I am surprised how polymorphic lists are. I suppose this is what is meant by a high level language |
| [Classes and Objects](https://www.learnpython.org/en/Classes_and_Objects) Review and complete exercise code:  Interesting. I had to look up how to create a constructor. Seems strange that it uses \_\_init\_\_ instead of something more intuitive, also interesting that objects have to be passed a reference to themselves to access their own data members in their member functions. I am sure there is a good reason for that but is strange coming from say java |
| [Dictionaries](https://www.learnpython.org/en/Dictionaries) Review and complete exercise code:  From my understanding dictionaries are just a way to link together information in a data structure without having to create a new class. Seems useful. Other than that, they can be used like lists in many ways. |

# 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.

Paste a screenshot of your local directory code

|  |
| --- |
|  |

1. Create a github account if you do not have one.
2. Create a github repository that is public from the local folder.

Explain what you did

|  |
| --- |
| I used the gh repo create command to create a remote repository by cloning a local repository |

Paste a screenshot of your github repository code

|  |
| --- |
|  |

Paste the url to you github repository code

|  |
| --- |
| https://github.com/ILaffely/cs3300-class-code |

## Add, Commit, Push Practice

Use command line for the following.

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

Use the command line for the following.

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 checkout -b 'sprint01'  Git log  Git branch  I actually messed this up and now the branch name has single quotes around it |

1. Switch to sprint01 branch to check out code:

git checkout 'sprint01'

git branch

git status

1. Modify the 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 .  git commit -m "Trying to add this to a different branch"  git push --set-upstream origin 'sprint01'  Got a little ahead of my self |

1. Upload the changes to 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 merge 'sprint01' |

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  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 the processes of maintaining organization and documenting changes as a piece of software is under development. Using git it begins with a local repository which is linked to a directory on a local machine. When a file in that directory is changed, that change can be committed which then updates the repository. That local repository can then be pushed to a remote repository as well.  A branch is an offshoot of development it is normally used as a way to test new features. A branch works by creating a copy of a repository, changes can then be made to this copy without effecting the main branch. When the branch is sufficiently developed it can then be merged with the main branch. This causes the changes from the branch to be integrated into the main repository. This should result in a new tag. A tag is a snapshot of a project. It represents one specific iteration. Whenever the release version of a project is updated a new tag should also be pushed. Tags are useful for troubleshooting and compatibility. |

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

|  |
| --- |
| Git is a piece of version control software that saves snapshots of a project not changes, per their website. It tracks the current state of a repository and will update this snapshot repository with any changes. It is a piece of software that runs locally. Github is a hosting service that works with git to maintain remote repositories and easily share code between multiple users. Github gets some kind of authentication from git and then allows the managing of these remote repositories from the local machine. |

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

|  |
| --- |
| Commit updates the local repository following a change to files in the local directory. For example just hitting save does not update the repository, you instead have to commit those changes. When you check the status of a repository it should tell you if you have any uncommitted changes. Pull gets the latest version of a remote repository and updates the local repository, it is how developers get code from their team. Push takes the local repository and updates the remote repository, it is how developers share code with their team. Status informs the user what branch they are in and checks to see if there are any uncommitted changes as mentioned above. Log gives a history of commits and pushes made to a repository as well as the branch in which they occurred and the tag and developer. It is useful for finding out who did what when. Checkout switches the active branch of the repository and informs the user if the branch is up to date with the remote repository. |

3.3 Explain the difference between a branch and a tag.

|  |
| --- |
| A tag is snapshot of a project, it is used to preserve a working state of a project and give it a name so that users can refer to the specific version for compatibility or other resources. A branch is used in development to add new features while not modifying the main project, it is an offshoot. Most of the time branches will be merged back with main. |

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

|  |
| --- |
| 1: Centralization of code. If people are committing and pushing often it will mean that all the code for a project is in one place, meaning that a developer does not need to bother other teammates or go on a scavenger hunt to try to get a file shared with them in order to work with a piece of code.  2: Easy way to test new features. Branches offer a great way to conduct experimental changes without posing a risk to the release version of the product. This way a feature can be fully developed and tested without losing any uptime on the live version.  3: Tagging makes troubleshooting and compatibility easy. Clear tagging means that known issues with a certain release can be catalogued and someone can look up the specific release they are working with to find the information that is relevant to them. Additionally, third party software may take a while to catch up with new features and so users can be easily see which versions of the software they are using are compatible with the third party programs they want to run |

# 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.

**Isaac Laffely**

[Laffely.isaac@gmail.com,](mailto:Laffely.isaac@gmail.com) [ilaffely@uccs.edu](mailto:ilaffely@uccs.edu)

**Overview**

* Currently pursuing a BS in Computer Science from UCCS - 3.722 GPA: Expected Graduation, Spring 2025
* Graduate of Salve Regina University with two Bachelors of Arts in Chemistry and Economics.
* Flexible and team-oriented attitude, able to accommodate a variety of work styles and personalities

Skills

* Programing Experience: significant knowledge of C, C++, and Java.
* Familiarity with Unix scripting, Python, x86 – 64 Assembly, Matlab and R.
* Knowledge of secure programing practices. Able to implement secure, maintainable, modular code in multiple languages.
* Proficient with advanced algorithm design and implementation.
* Significant familiarity with technical writing
* Experience using Git and Github: for managing and versioning projects.

**Experience**

* Undergraduate: Chemistry and Economics, Salve Regina University
  + Completed rigorous double major at 4 year college: 128 credit hours with on-time graduation.
  + Experience working in teams of up to 5: able to effectively divide and complete complex tasks in a technical environment
* Delivery Driver: FedEx Ground
  + Independent work ethic and ability to complete tasks with no supervision
  + Ability to surge to meet seasonal deadlines and work under considerable time constraints
  + Experience training new employees
* Math Tutor: Palmer High School
  + More than hundreds of hours of one-on-one tutoring experience.

Isaac Laffely Interview questions – most of these were questions that I liked taken from interviews I have been involved in

**Section 1 – General personality**

* What do you believe is your biggest personal strength is?
* What do you view as your biggest weakness?
* What do you believe that would you bring to a team?
* What is something new you have learned this week?

**Section 2** – **Interpersonal/teamwork skills**

* Suppose that you are on a team and one of your teammates has missed multiple deadlines recently, what do you do?
* Suppose that you are on a team, and you have been given an assignment that you will not be able to complete in time, what do you do?
* Suppose you and another member of your team disagree on a course of action and that they are fully committed to an approach that you are sure is wrong, what do you do?

**Section 3 – Problem solving assessment**

* Suppose I wanted to fill this room completely with pennies, approximately how much would this cost? (To test problem solving and decomposition)
* Could you tell me a short story containing two characters in which one of the characters tells a short story containing two characters? (To test recursive thinking)
* Suppose you are given an elephant; you must try to keep it alive, and you can neither sell it nor give it away, what do you do? (A preposterous question to test divergent reasoning, based on a historical conundrum from Thailand)

Interview Notes

Hanh Nguyen

* Tue –Thurs 9 – 12
* Wed 10 –12
* Strong time management and organization
* Hard worker
* Shy introvert
* Conflict averse
* Answered very honestly ie. Aproaches interview as a discussion
* Team Cohesion mindset

Ryan Montgomery

* Tue – Thurs 10 –11
* Diligent worker
* Good knowledge base
* Sometimes stubborn
* Discussion oriented, not afraid of confrontation
* Not dishonest, but tried to answer “correctly” ie. Aproaches interview as a test.
* Problem solving mindset