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

|  |
| --- |
| Andrew Stokey |

**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.org official guide | [Python Developer’s Guide](https://devguide.python.org/) |
| Initial python steps and syntax | [How to Use Python: Your First Steps – Real Python](https://realpython.com/python-first-steps/) |
| Reliable software youtube channel this video has python | [Coding Dojo | Learn Skills for Software Development in 18+ Weeks! (youtube.com)](https://www.youtube.com/watch?v=kqtD5dpn9C8) |
| Quick github guide | [Get started with GitHub documentation - GitHub Docs](https://docs.github.com/en/get-started) |
| Youtube tutorial for github | [Git and GitHub Tutorial for Beginners (youtube.com)](https://www.youtube.com/watch?v=tRZGeaHPoaw) |
|  |  |

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)  **In python variables are automatically assigned a type therefor for proper code management we would want to cast the string using**  var1 = str("Hello World!")  **We can then use a print statement to display the string directly to console.**  **Variables are objects as well so we have to be careful about the casting** |
| [Lists](https://www.learnpython.org/en/Lists) Review and complete exercise code:  **Lists are made with brackets and behave almost identical to arrays in C or java.**  **First to make a list**  list1 = ["apple", "orange"]  **Keep in mind these items must have quotes as the list contains strings in this case.**  **(Lists have to contain all the same data types)**  **List comprehensions:**  **Python has a unique feature to allow us to perform what may be many lines of code in just one singular line. For example:**  newList = [x for x in range (10) if x is less than 5]  **This would regularly take approximately 4 lines due to needing a for loop for the range of 1-5 to be added to the list.** |
| [Basic Operators](https://www.learnpython.org/en/Basic_Operators) Review and complete exercise code:  **Math operators are almost exactly like other languages such as C or Java:**  **+ adds the two variables (or can concatenate strings together)**  **-Subtracts the two variables**  **/ Divides**  **\*Multiplies**  **% is a modulo and returns the remainder**  **\*\* is the exponential operator**  **PEMDAS also applies in python therefore we must be careful with the location of our brackets and parenthesis** |
| 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)  **If you want to use the numpy library there are many built in methods such as abs(num1) which will return the absolute value of a number. Graphing, square root and many other methods are located within this library. This library can be importing by using:**  import numpy as np  **By using the as after the numpy we can customize the calling name for using the numpy method**  [String Formatting](https://www.learnpython.org/en/String_Formatting)  **As stated above, we must make sure to cast our variables as strings when necessary. In the example below I would receive an error for trying to concatenate the integer into a string:**  year = 2024  firstPart = "The year is "  sentence = firstPart + year  **I can fix this will a simple casting the number to a string:**  year = 2024  firstPart = "The year is "  sentence = firstPart + str(year)  [Basic String Operations](https://www.learnpython.org/en/Basic_String_Operations)    **The string object has plenty of methods built into python which allow it do automatic conversions such as lowercase which converts the string to all lowercase letters and isalpha which can be used to confirm that the string only contains characters**  Python  **Python is the programming language for which we will be programming in this class. It is highly praised for its versatility and combination between components of other languages. It has capabilities to be directly converted to a script for use in unix/linux coding and can even run searches through parsing HTML code through be browser access.**  [Conditions](https://www.learnpython.org/en/Conditions)  **Conditions such as true or false are enhanced within python to include truthy and falsey statements. This can be especially useful when combined with concepts such as list and dictionary comprehensions to execute code that would take other languages many extra lines. With conditions and comprehensions we can execute this in a single line of code.**  [Loops](https://www.learnpython.org/en/Loops)  **Loops function the same as many other programs except for the lack of closing and opening brackets. List comprehensions when combined with loops and concepts such as a generator can allow the user to feed many input cases/values into a single function without having to continuously enter values manually.** |
| [Functions](https://www.learnpython.org/en/Functions) Review and complete exercise code:  **Functions are different from methods in the way that they act independently from an object, therefore they can be used by any object or without objects entirely. Like loops they require only parenthesis for the arguments which will be passed to the function. In the example below you can observe how they hold onto their own indentation. A apace after the last continuous line within the function will exit the function body.**  def myFunction(int n):      for x in range(n):          print(x) |
| [Classes and Objects](https://www.learnpython.org/en/Classes_and_Objects) Review and complete exercise code:  **Like the previous examples, classes have been cut from other programing languages to be simplified. In the example through the link we were able to see that the variables are defined directly after the class that each instance of the object will contain;**  **Note: Python also features inheritance. Where the child object simply needs to have a def \_\_init\_\_ statement before its name along with the accompanying class it will be inheriting from** |
| [Dictionaries](https://www.learnpython.org/en/Dictionaries) Review and complete exercise code:  **Dictionaries are a feature of python that is not included in other programming languages. Dictionaries operate like a list but can contain objects/variables of all different data types. In this manner they are like structures. Dictionaries include keywords for each item attached to them in the place of an index. When calling for the item attached to the key you must reference the key. Dictionaries also allow you to use functions attached to their objects that allow you to remove a certain exact key, modify its value or add something to the end of the dictionary without leaving a whole in or disturbing the rest of the dictionary itself.**  **Dictionaries essentially add extra function and rules to a list, either may be used but, both have their advantages and should be used so accordingly.** |

# 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.

|  |
| --- |
| Used the terminal and commands such as ls and cd to get to the place I would be placing my repository. Then I used mkdir to create a new folder. Finally, I copied the document to the new folder after making a folder labled documnetation |

Paste a screenshot of your local directory code

|  |
| --- |
|  |

Paste a screenshot of your github repository code

|  |
| --- |
| I had already made and linked my repository in step 1 but, I would use these steps if not from the github main page |

Paste the url to you github repository code

|  |
| --- |
| [ddomke-class/CS3300 (github.com)](https://github.com/ddomke-class/CS3300) |

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  Git checkout  Git 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

|  |
| --- |
|  |

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 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 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.

|  |
| --- |
| I decided to use the tutorial I copied above before starting most of the github portion of this assignment. To my delight, most of the commands were simple to guess if they weren’t explained in this video [Git and GitHub Tutorial for Beginners (youtube.com)](https://www.youtube.com/watch?v=tRZGeaHPoaw)  Software Version control is important for maintaining code that works during a portion of time where updates are being made. It allows a group of people to simultaneously work on one project and easily allow others to access the updates as they are successfully tested and committed.  **Branches** allow us to make an update to the code without directly affecting the main branch or branches where the code works well. This will be useful when we accidentally go down the path of attempting to implement a feature and deciding later that we don’t want that feature. **Commits** are a way of publishing the updated code that we wish to adapt to the version of a branch we are in on the server rather than just our local repository. **Merge** allows us to adapt the changes made in a separate branch to the current branch we are working on. **Tags** allow us to give notes on branches for others and are a helpful way of tracking versions. |

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

|  |
| --- |
| Git is a version control software that allows us to control files that change over a system at a different rate. It is different from other software because of its capabilities in allowing multiple engineers to work on different components of the same software at the same time and all being able to individually commit their changes. Github is the cloud based storage of these individual repositories and works to be a one spot storage for all these changes being made. It is useful from its UI in the online and desktop versions along with linking to the capabilities of git through command prompts.  [Git - What is Git? (git-scm.com)](https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F) |

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

|  |
| --- |
| Log- git log is a way to pull a record of all the recent changes to the repository currently being worked on  Status – git status is the way you can view what items are in the staging area along with what files may be actively uploading or downloading changes from the cloud  Add- git add “insert file here” is a way to put your updated files from the local system to the staging area, waiting to be pushed to the local repository  Push- git push is the way to push your changes in the staging area to the local repository  Pull- pull “branch” is a way to pull the latest version or a specific branch from the remote repository  Commit- git commit is the way you push the changes made in your local system and repository to be permanent changes for that branch or the changes made for those branches |

3.3 Explain the difference between a branch and a tag.

|  |
| --- |
| A branch is the actual version you are working on which may have separate committed changes from the main version of the software. Branches can be merged to commit those changes on other branches. The tag is simply a note put on branches to relay information to others on changes made in that branch. This would be useful in notifying someone who oversees the project or someone who needs to make their software work hand in hand with your software of bug fixes or reasons your changes were made. |

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) less contact over what has changed in fixes. In industry we may have 3 different people working on an update, all 3 people can make changes to their code independently, then have their versions committed before the branches were merged. I also would not have to talk to them before waiting for their portion to be done, I could simply monitor the repository for an update.  2) when mistakes are made. If we had developed software that had made many updates over a month before realizing the customer doesn’t want them or the previous version worked better, I can simply go back to a previous working version/branch.  3)Individualized changes without affecting what is in the cloud. While I make changes on my system that were correcting bug fixes from the last version, a coworker could be creating the next version, all without modifying the main software while it is incomplete. |

# 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.