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.

Name of the person you collaborated with:

|  |
| --- |
| Ryan Fuller |

**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** |
| W3Schools Python Tutorials and Examples | <https://www.w3schools.com/python/> |
| Version Control Systems Explained | <https://medium.com/coders-capsule/explaining-version-control-systems-using-git-and-github-510eeb0f8a37> |
| Python Introduction | <https://docs.python.org/3/tutorial/index.html> |
| Introductory Python projects | <https://www.geeksforgeeks.org/python-projects-beginner-to-advanced/> |

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)  Python is a simple language with straightforward syntax. There are two prominent Python versions, Python 2 and Python 3. They are different from each other, but Python 3 is more commonly used. Python 3 has **print()** as a function instead of an operator. Python 2 has it as an operator that does not need parentheses.  [Variables and Types](https://www.learnpython.org/en/Variables_and_Types)  Python is an object­-oriented language. Every variable is an object. Contrary to other languages, you do not need to declare variables or their types before using them. In Python, there are integer numbers and floating-point numbers. Defining any variable in Python is as simple as **variableName = {data}** where data can be any type. Strings of characters need to be in quotes. |
| [Lists](https://www.learnpython.org/en/Lists) Review and complete exercise code:  Lists are similar to arrays in other languages. They are different from arrays in that they can contain as many different types as you want, and their size can change during runtime. This likely means a linked list is used to implement a list in Python. Elements can be removed from a list with the append() method, removed with remove(), or accessed with **[n]**. |
| [Basic Operators](https://www.learnpython.org/en/Basic_Operators) Review and complete exercise code:  Operators in Python are mostly comprised of basic arithmetic operators. You can also use \*\* for exponentiation. Python also has string arithmetic, where strings can be combined with the **+** operator. Strings are also affected by the **\*** operator, where **“hello” \* 3** would yield **“hellohelllohello”**. These operators also work with lists. |
| 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)  Python has string formatting that is very similar to C. For instance, if you had a string holding a name, you could use print(“Hello, %s!” % name) to print a greeting. The %d operator is used for integers, %f for floats, %x for hexadecimal, and %.2f would print 2 decimal places of a floating point number.  [Basic String Operations](https://www.learnpython.org/en/Basic_String_Operations)  A string’s length can be obtained with the len() method. The index() method returns the index of the first occurrence of a character in the string. The count() method returns the number of occurrences of a character in the string.  String slicing is another important part of using strings in Python that resembles the String.substring() method in Java. To slice a string called words, you can use words[a:b] where a and b are integers. The result will be the substring from index a until index b, not including index b. You can also skip over characters with words[a:b:c] where indexes a to b will be printed, with a step size of c. Python also has negative indexing, so words[-1] is the last character in the string. A string can be reversed with words[::-1], or made upper/lower case with the upper() and lower() methods. The split() method is used to split a string up into a list of strings based on a given delimiter.  [Conditions](https://www.learnpython.org/en/Conditions)  Similar to Java and C, Python uses the == operator to check equality. The **in** operator is used to evaluate if an object is in a list. For instance, the statement **if 2 in [4, 5]** would return false because 2 is not in this list. Python uses the **if**, **elif**, and **else** operators for if, else if, and else statements, respectively. The **is** operator is used to check if instances are the same, rather than checking equality. The **not** operator is used to invert statements.  [Loops](https://www.learnpython.org/en/Loops)  For loops in Python resemble for-each loops in other languages. The range() function produces a sequence of numbers that can be iterated over. The xrange() function yields an iterator that is more efficient. Like in other languages, while loops repeat for as long as a Boolean condition is true. Python does not use brackets for loops, and instead relies on indentation. A colon : must be at the end of the line with the **for** or **while** condition. The **break** keyword is used to exit a loop, and **continue** skips to the next block of the loop.  Interestingly, the **else** keyword can be used with loops. It can be used when the loop condition fails, unless the loop is terminated with a **break** statement. |
| [Functions](https://www.learnpython.org/en/Functions) Review and complete exercise code:  Functions in Python are similar to other languages. They can be called in another section of code to avoid repeating yourself, and they can be passed arguments. |
| [Classes and Objects](https://www.learnpython.org/en/Classes_and_Objects) Review and complete exercise code:  Objects are used to encapsulate variables and functions into an entity. Classes are templates for objects. This is hardly different from Java. You use the dot operator **.** to access variables in an instance of an object. The same goes for functions. The \_\_init\_\_() function assigns values in a class. |
| [Dictionaries](https://www.learnpython.org/en/Dictionaries) Review and complete exercise code:  Dictionaries are a data type similar to arrays, but they use keys and values instead of indices. Each key corresponds to a value in the dictionary. A value can be any type. A dictionary is initialized with **myDictionary = {}**, or with other notations that put values directly into it. They can be iterated over, but only in a slightly different way than lists. A dictionary doesn’t keep values in order, so you must iterate over them with something like **for name, number in phonebook.items():**. The items() method returns all the items in the dictionary. The **del** operator or the **pop()** method can be used to remove an item from the dictionary. |

# 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 are git and github? What does git provide? What does github provide?**

Git is a command-line tool installed locally. It is used for version control and code sharing. Its desktop interface is Git Gui. GitHub is a service with a graphical user interface that is hosted on the web. GitHub is used for centralized source code hosting. GitHub hosts Git repositories. Git is open-source, while GitHub has a free-tier and a pay-for-use tier.

* **How can you create a github repository from a local folder?** 
  1. Create a folder for the repository to reside in.
  2. Create a new GitHub repository on GitHub’s website.
  3. Run the following in command line:
     1. git init // initializes GitHub repo
     2. git add . //
     3. git commit -m “first commit” // commit everything in the folder to the repository
     4. git branch -M main // change the default branch from “master” to “main”
     5. giit remote add origin git@github.com:user/repo-name.git // assign the local folder to the remote repository that was just created. The above URI can be obtained from the repo’s main page by clicking **<> Code**.
     6. git push -u origin main // push the local folder’s contents to the remote repository
* **What documentation could be useful to help understand the commands?**

Documentation of Git commands: <https://git-scm.com/docs/>

Neat video explaining the functions of some of these commands: <https://www.youtube.com/watch?v=WgCoEHMa6zU>

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.

|  |
| --- |
| mkdir cs3300-version-practice  cd cs-3300-version-practice  git init  git add .  git commit -m “first commit”  git branch -M main  git remote add origin [git@github.com:rmontgo/cs3300-version-practice](mailto:git@github.com:rmontgo/cs3300-version-practice)  ssh-keygen -t ed25519 -C [ryan4@startmail.com](mailto:ryan4@startmail.com) //this is the email I associated with GitHub. I created the sshkey with a passphrase.  Start-Service ssh-agent  ssh-add <ssh private key>  git push -f |

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/rmontgo/cs3300-version-practice.git>  [git@github.com:rmontgo/cs3300-version-practice.git](mailto:git@github.com:rmontgo/cs3300-version-practice.git) |

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 // checks the branch you are on  git log // prints the repository’s commit history  git branch sprint01 // creates new branch called 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 add . // add file to the staging area  git status // check git status to see changes to be committed  git commit -m “sprint01 branch” // do 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

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

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 show 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.

|  |
| --- |
| Software version control is a process used to develop software in a manner where many versions of the software can be retained at once. When an update is made to some code, a developer commits the code to their local repository folder, and then pushes it to the origin. Each pushed version of the repository is still available by viewing the repository’s commit history. This way, many old versions of the code can be accessed.  Branches are offshoot versions of a repository’s contents that can be modified without changing the contents of main. When the contents of a branch are ready to be implemented in main, they can be added with a git merge, which will apply the changes from the branch to main. Tags are human-readable labels used to identify versions of software. They can be used to easily checkout a different version of the repository.  <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control> |

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 system for managing projects. It permits the creation, deletion, switching, and merging of branches in a project. In simple terms, Git is the backbone of version control. GitHub uses Git under the covers, which is why GitHub allows you to manage repositories with such great efficiency and ease.  <https://git-scm.com/about> |

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

|  |
| --- |
| Commit: Perform new changes to the repository. Requires a commit message.  *Example: git commit -m “updated code”*  Pull: Retrieve new changes in the repository from the origin. This will not necessarily cause your local repository to be identical to the origin since there may be changes in your local repository that haven’t been pushed.  *Example: git pull*  Push: Typically run after a commit, push will update the origin with all new changes. The version history of the repository is still available.  *Example: git push*  Add: Stages contents of the local repository for a commit.  *Example: git add .*  Clone: Clone a repository to the local working directory. This includes all files, branches, and commits. Some options can be used to clone only a single branch.  *Example: git clone <url>*  Status: Show the local repository’s status. It shows the staged files that have changed from the last commit of the working branch.  *Example: git status*  Log: Show the commit logs for this repository, including author, date and time of the commit, and the commit message.  *Example: git log*  <https://git-scm.com/docs/>  <https://docs.github.com/en/get-started/quickstart/about-github-and-git> |

3.3 Explain the difference between a branch and a tag.

|  |
| --- |
| A tag points to a specific commit in the project’s history. Tags are often used to indicate the release of a new version or an important change. A branch is a new version of main (or another branch). The branch head is a pointer to the most recent commit on the branch. Both tags and branches are used for version control, but in different ways. A branch is used to fix issues and add new features without impacting the main codebase. Tags are just markers in the repository to denote milestones in the project.  <https://circleci.com/blog/git-tags-vs-branches/> |

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

|  |
| --- |
| One benefit of a version control system is that it is easy and safe to add new features. A dev branch can be created to work on new features without the risk of detrimental changes to main. New features can then be merged into main. Another benefit is being able to collaborate on a project. Multiple individuals can pull the repository, make changes, and push them. The version control system is typically very good at combining these changes into main. A third benefit is the codebase is both highly accessible and is stored with high redundancy. Repositories are usually stored in the cloud with a number of backups and can be accessed from numerous computers. You don’t have to worry about your clumsy coworker spilling coffee on his laptop and losing 6 months of work. |

# 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.