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

| Justin Manaar Vincent Faragalli |
| --- |

**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’s own website provides a short but informational tutorial and introduction into the language. | https://www.python.org/about/gettingstarted/ |
| Git's very own website provides an introduction to the concept of version control. | <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control> |
| This website provides a full tutorial on how version control works and how to integrate it with GitHub. | <https://www.atlassian.com/git/tutorials/what-is-version-control> |
| You cannot learn Git without GitHub since both go hand and hand. This resource will provide how GitHub works and how it integrates with GitHub. | https://kinsta.com/knowledgebase/what-is-github/ |
|  |  |
|  |  |

Start exploring git, github, command line, and python in a virtual environment.

[1 Python and IDE](#_7a4jn11vv6wq)

[Install Python](#_79csvznoivco)

[Install VS Code IDE](#_9gomil77gszl)

[2 Pair Programming Video](#_rwvlj4hp6mc7)

[3 Version Control](#_3fp0cqgnykx1)

[Set-up git and github repository](#_bptpc7j7mx76)

[Add, Commit, Push Practice](#_27n2hu32nsae)

[Branching](#_tyjcwt)

[Version Control Concepts](#_go47xdl2sh5a)

[4 Resume and Interview Questions](#_s0jda1wrx8t6)

# 1 Python and IDE

Set up your python and IDE for your python development.

## Install Python

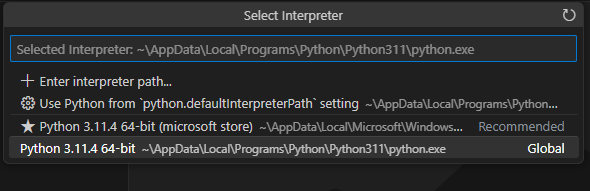
1. Open the command window and check your python version to see if you have it installed.
2. Install python version 3.11 [Download Python](https://www.python.org/downloads/). If on windows and have older version of python you should uninstall first : [How to Uninstall Python](https://www.pythoncentral.io/how-to-uninstall-python/)

## Install VS Code IDE

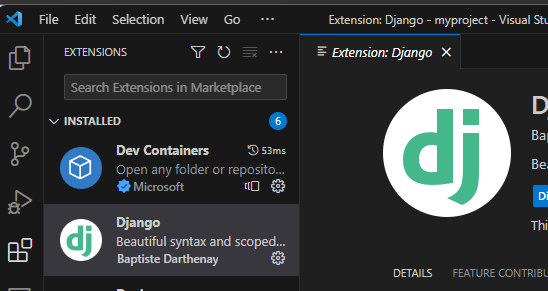
You can use a different IDE but this is what I will be using in my lectures. This has nice tools to integrate with python, django and databases.

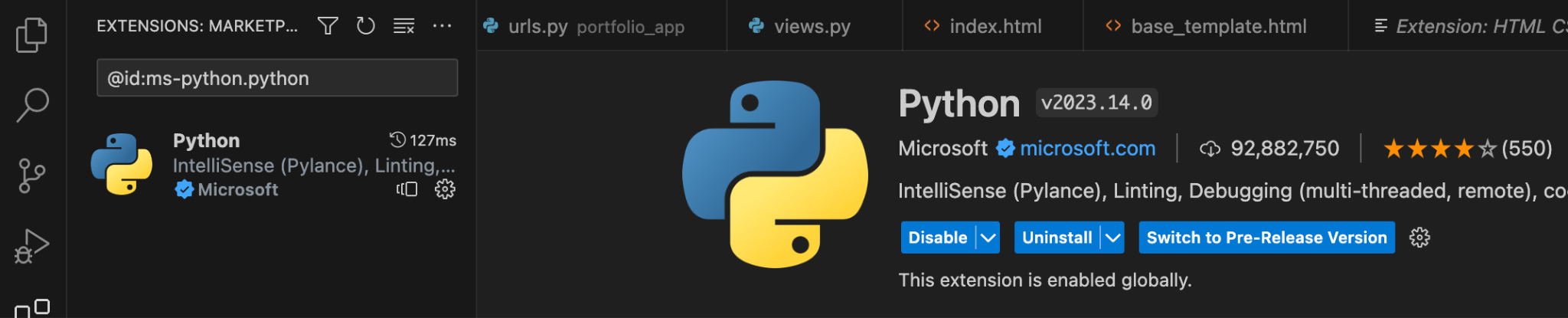
<https://code.visualstudio.com/download>

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.

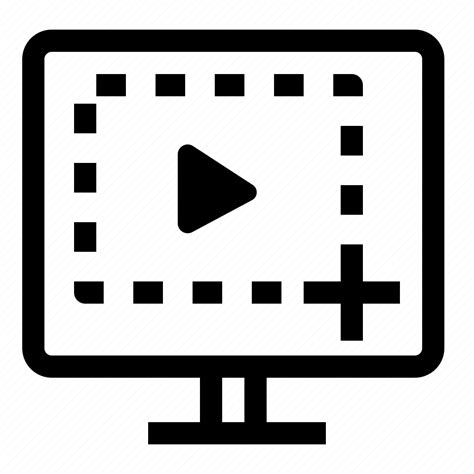
Setup:

* 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.

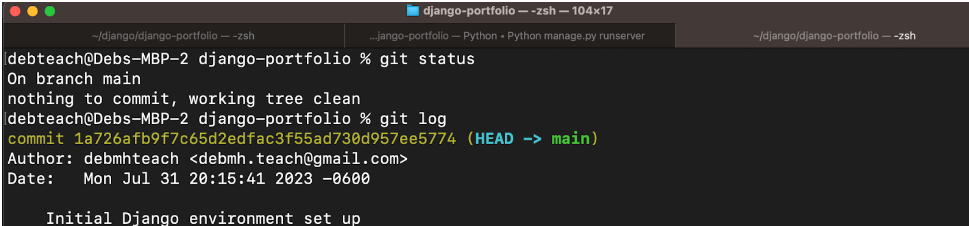
| Scan the following sections before pair programming. Take turns summarizing each section to the other. Add any brief notes or examples.  [Hello, World!](https://www.learnpython.org/en/Hello%2C_World%21)  Hello world is a simple sentence that shows off the basic of printing into the console. Print is a function that gets called and whatever is inside the parentheses, is what is printed.  [Variables and Types](https://www.learnpython.org/en/Variables_and_Types)  There are various variables and types, such as int that holds integer values, floating point variable which holds decimal values, characters which hold a single character whether its a letter or a number, and a string which is essentially a character array. |
| --- |
| [Lists](https://www.learnpython.org/en/Lists) Review and complete exercise code:  numbers = []  numbers.append(1)  numbers.append(2)  numbers.append(3)  strings = []  strings.append("Hello")  strings.append("World")  names = ["John", "Eric", "Jessica"]  # write your code here  second\_name = names[1]  # this code should write out the filled arrays and the second name in the names list (Eric).  print(numbers)  print(strings)  print("The second name on the names list is %s" % second\_name) |
| [Basic Operators](https://www.learnpython.org/en/Basic_Operators) Review and complete exercise code:  x = object()  y = object()  # TODO: change this code  x\_list = [x] \* 10  y\_list = [y] \* 10  big\_list = [x, y] \* 10  print("x\_list contains %d objects" % len(x\_list))  print("y\_list contains %d objects" % len(y\_list))  print("big\_list contains %d objects" % len(big\_list))  # testing code  if x\_list.count(x) == 10 and y\_list.count(y) == 10:  print("Almost there...")  if big\_list.count(x) == 10 and big\_list.count(y) == 10:  print("Great!") |
| 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)  There are simple arithmetic operators that can be used in Python. Add, subtract, multiply, divide, modular etc. These arithmetic operators can also be used with strings for example adding two strings together can combine them to make a sentence and multiplying a string will display it however many times it was multiplied.  [String Formatting](https://www.learnpython.org/en/String_Formatting)  When it comes to printing stuff to the console, there are string formats to follow. Each data type has their own string format. Integer’s format is %d, float is %f, strings are %s. When printing out a statement, it is important to put the format then at the end of the statement put a percent sign then the value it is formatting. For example, print(“%s is my name” % name)  [Basic String Operations](https://www.learnpython.org/en/Basic_String_Operations)  For strings, they too have their own operations inside of Python. You can check to see the length, see how often a certain character shows up in the string, check the index of a character in a string, slice the string into pieces, etc.  [Conditions](https://www.learnpython.org/en/Conditions)  Conditions in Python are evaluated using True or False expressions. The “and” and “or” operators are good ways to check for conditions. “And” indicates that all requirements are met in order to be true and “or” requires only one of the conditions to be true. The “in” operator will check if the object exists in a iterable object container for its condition. “If” statements are also a good way to check conditions along with “is” to see if a condition is true.  [Loops](https://www.learnpython.org/en/Loops) |
| [Functions](https://www.learnpython.org/en/Functions) Review and complete exercise code:  # This function makes a list with strings inside that list benefits  def list\_benefits():  strings = []  strings.append("More organized code")  strings.append("More readable code")  strings.append("Easier code reuse")  return strings  # This function take in a list with a string then add that string to a sentence to print out  def build\_sentence(benefit):  return benefit + " " + "is a benefit of functions"  def name\_the\_benefits\_of\_functions():  list\_of\_benefits = list\_benefits()  for benefit in list\_of\_benefits:  print(build\_sentence(benefit))  name\_the\_benefits\_of\_functions() |
| [Classes and Objects](https://www.learnpython.org/en/Classes_and_Objects) Review and complete exercise code:  # define the Vehicle class  class Vehicle:  name = ""  kind = "car"  color = ""  value = 100.00  def description(self):  desc\_str = "%s is a %s %s worth $%.2f." % (self.name, self.color, self.kind, self.value)  return desc\_str  # Making 2 cars with their own name, kind, color, and value then printing them out  car1 = Vehicle()  car2 = Vehicle()  car1.name = "Fer"  car1.kind = "Convertible"  car1.color = "Red"  car1.value = 60000.00  car2.kind = "Van"  car2.name = "Jump"  car2.color = "Blue"  car2.value = 10000.00  # test code  print(car1.description())  print(car2.description()) |
| [Dictionaries](https://www.learnpython.org/en/Dictionaries) Review and complete exercise code:  phonebook = {  "John" : 938477566,  "Jack" : 938377264,  "Jill" : 947662781  }  # Adding Jake to the phonebook with his number and remove Jill and her number  phonebook["Jake"] = 938273443  phonebook.pop("Jill")  # testing code  if "Jake" in phonebook:  print("Jake is listed in the phonebook.")    if "Jill" not in phonebook:  print("Jill is not listed in the phonebook.") |

# 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?

Git is a version control system used in software development. It allows multiple developers to work and collaborate on a project and it tracks changes to the source code. GitHub is a platform that uses Git to create a remote repository and it provides a centralized hub for code repositories, enabling teams to work together on projects, manage code, and track changes. . Git provides a way to manage different versions of files, keep track of changes, and merge the work of multiple contributors seamlessly. Git provides a variety of different commands which include clone which creates a identical copy of the repository you want, commit which creates a save point, branch which is a parallel version of the branch that can allow developers to work on different bugs separately which can then be merged with the source code later, push which uploads the save point to a remote repository, and fetch which retrieves changes from the remote repository without merging them. GitHub provides the remote repository for all of the commands that Git can do.

* How can you create a github repository from a local folder?

In the command line type these commands:

git init

git add \*insert files and folders to the staging area\*

git commit -m "Initial commit"

Add GitHub Repository as a Remote:

# Copy the URL of your GitHub repository (you can find it on the GitHub repository page).

git remote add origin <repository\_url>

# Push the committed changes to GitHub

git push -u origin master

This command pushes the changes to the master branch. If you are working with a different branch, replace "master" with the branch name.

# This command pushes the changes to the master branch

* What documentation could be useful to help understand the commands?

The GitHub learning lab: <https://github.com/apps/github-learning-lab>

The ProGit Book: <https://git-scm.com/book/en/v2>

Various YouTube tutorials online

Git Immersion: <https://gitimmersion.com/>

All of these sources contain documentation or interactive activities that help with understanding the various commands.

Resources:

<https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>

<https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github>

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.

| To create the directory, I used the command “git init cs3300-version-practice”. I then created a file inside of the terminal using “vim practice.py”. I then downloaded this document and moved it into the documentations folder. |
| --- |

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/AleIsPale435/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 log  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

| Vim practice.py # to modify the file  Git add practice.py  Git status  Git commit -m sprintCommit |
| --- |

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 master  Git merge sprint01  Git push –set upstream 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 -a v1.0  Git show v1.0  Git push -set-upstream origin master  (For some reason it shows the tag on my terminal but not in GitHub) |
| --- |

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.

| Branches, commits, merges, and tags are all important in software version control because it is important to have multiple versions. In case of an emergency and a certain version is corrupted or no longer accessible, Having different versions of code in different branches allows for multiple developers to work on the code and then commit a brand new version of that code. Tagging each different version is also useful and then merging the different branches of code together to get the best possible version of the code. |
| --- |

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 used for source code development. It is a free and open-source version control system that allows developers to handle projects. It is mostly used to keep track of source code and allow multiple developers to work together. Git works with GitHub because GitHub is where the source code is stored. Each developer has their own repositories that can be uploaded to GitHub where other developers can access different versions and branches of the source code. Git tracks all of the changes you make to the source code and stores it in GitHub. GitHub serves as a location for uploading copies of a Git repository. |
| --- |

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

| Git commit records the changes happening in the branch. Committing a change is essentially a save point that I can go back to if I find a bug or need to make a change. For example, git commit -m “First release” displays the changes that were made and saves it. Git pull pulls all changes from a remote repository into the current branch you are working on. Git pull \*insert file\* will pull the file from a repository and put it in the current branch you are on. Git push will push any changes you have already committed to your GitHub repository so everyone can see your changes. Git push master will push all of the changes committed on the master branch into the remote repository on GitHub. Git add will add a file or directory to the branch allowing it to be committed and pushed. Git add text.txt will add this file to the remote repository so it can be pushed and committed. Git clone will just create a full copy of a repository, including all of the logs and files with it. Git status when on a branch will display will tell you if any or all files are part of the repo along with the current branch, untracked files, and commit status and whether it needs to be pushed to the remote repository. Git log will display a full log of commits inside of the branch. Let’s say you committed two times inside your directory, if you type git log, it will display the two commits of the branch, the time, and the description. Git checkout will switch branches inside of the depository. If you have two branches named master and test, if you do git checkout test, it will switch to that branch and you will need to add your files to the current branch, commit your files, etc. |
| --- |

3.3 Explain the difference between a branch and a tag.

| When it comes to tags, when they are created they are used to create significant points in the project timeline and have no history of commits. Branches create separate lines of development that can be eventually merged back into the main code of the project. |
| --- |

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

| * Using a version control system can have multiple developers working on a project and can work on it alone thanks to branching. When multiple developers are working on a source code, they can combine the code they worked on and can keep several streams of work separate while still having the option to merge them back together to ensure that their changes don’t conflict. For example, let's say there are 70 people working on a project. All 70 of those people can work on the code simultaneously in their own branch so the main branch is not affected. Once they are done updating the code they can merge it with the main branch and update the whole thing. * Another benefit of using a version control system is to keep track of its history. With each change made, it will be kept on file. Other developers may find it easier to comprehend how a specific code section came to be. This is important for working efficiently with historical code and allowing developers to predict future work with accuracy. It also allows for code to be reused for other projects. For example, when I finish a project and I use a particular function or method, I can simply visit the source code of the project and reuse the function or method for another project. That way I do not have to rewrite the code, I can simply just reuse it and change it to fit the needs of the new project. * The last benefit of using a version control system is that it will create regular automated backups so the source code is never lost. When it comes to working on software, errors may occur and sometimes software can crash. With a version control system, developers can access the version they were working on before. This also allows for developers to work on the source code offline and they can resolve bugs with a single change-set increasing work productivity. For example, I could be working on a project when all of a sudden the software shuts down and I lose all of my progress. I could load up the latest version I saved on the branch and pick up where I left off without any trouble and without any internet. Once the internet goes back up I can upload my new saved changes and have that backup. |
| --- |

# 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.

What are some of your strengths and weaknesses?

What is your experience working with teams?

If the team is failing to get the job done, what is one way you approach that conflict?

What is your ideal definition of a team player?