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

| Luke Delzer |
| --- |

**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** |
| --- | --- |
| Basic Python Tutorial | <https://docs.python.org/3.11/tutorial/index.html> |
| Python Library Reference | <https://docs.python.org/3.11/library/index.html> |
| Python Language Reference | <https://docs.python.org/3.11/reference/index.html> |
| Github Tutorial | <https://docs.github.com/en/get-started/quickstart/hello-world> |
| Powershell Commands | <https://www.pdq.com/powershell/> |
|  |  |

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

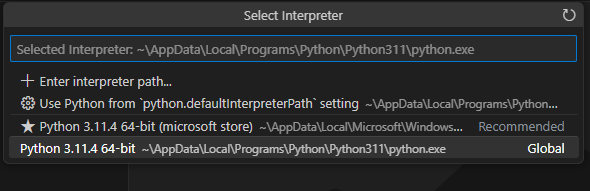
1. Open the command window and check your python version to see if you have it installed.
2. Install python version 3.11 [Download Python](https://www.python.org/downloads/). If on windows and have older version of python you should uninstall first : [How to Uninstall Python](https://www.pythoncentral.io/how-to-uninstall-python/)

## Install VS Code IDE

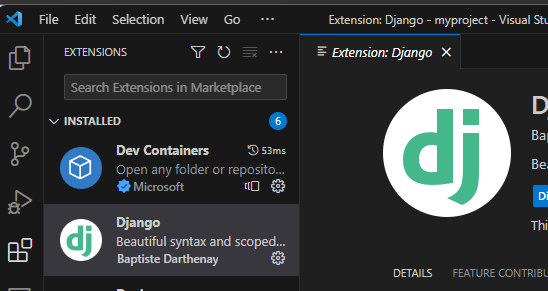
You can use a different IDE but this is what I will be using in my lectures. This has nice tools to integrate with python, django and databases.

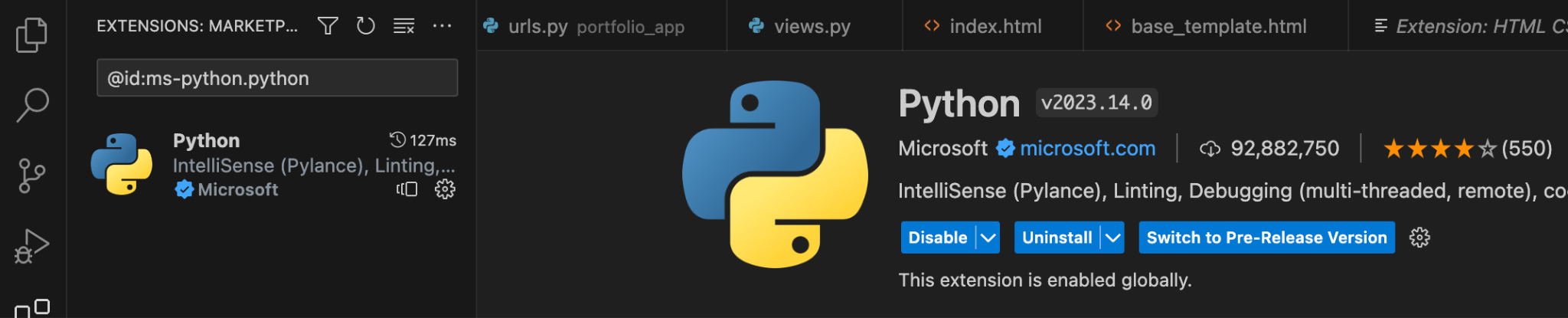
<https://code.visualstudio.com/download>

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.

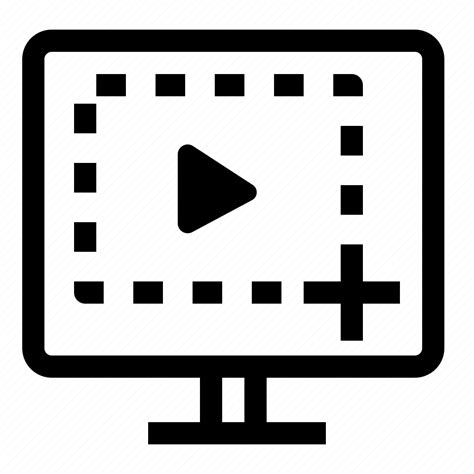
Setup:

* 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.

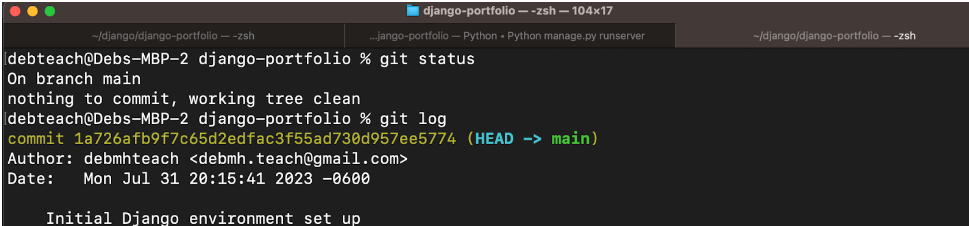
| Scan the following sections before pair programming. Take turns summarizing each section to the other. Add any brief notes or examples.  [Hello, World!](https://www.learnpython.org/en/Hello%2C_World%21)  [Variables and Types](https://www.learnpython.org/en/Variables_and_Types)  You can have apostrophes in your string if you use double quotations marks!  Not using semicolons or braces is weird. |
| --- |
| [Lists](https://www.learnpython.org/en/Lists) Review and complete exercise code:  numbers = []  strings = []  names = ["John", "Eric", "Jessica"]  # write your code here  numbers.append(1) #Can also use a for loop but I'm keeping it simple to start off.  numbers.append(2)  numbers.append(3)  strings.append("hello")  strings.append("world")  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\_list + y\_list #Why did the original script have [] here? Weird.  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)  [String Formatting](https://www.learnpython.org/en/String_Formatting)  [Basic String Operations](https://www.learnpython.org/en/Basic_String_Operations)  [Conditions](https://www.learnpython.org/en/Conditions)  [Loops](https://www.learnpython.org/en/Loops)  Notes:   * print() acts like printf, but % is used for everything. * Printing a list as part of a formatted string is weird, going to take some getting used to. * Splitting text input as a list seems useful especially with CSV’s, going to need to remember that. * If statements don’t use parenthesis or braces and conditionals are language… * A for/while loop can have an else statement?!?! |
| [Functions](https://www.learnpython.org/en/Functions) Review and complete exercise code:  # Modify this function to return a list of strings as defined above  def list\_benefits():  return ["More organized code", "More readable code", "Easier code reuse", "Allowing programmers to share and connect together"]  # Modify this function to concatenate to each benefit - " is a benefit of functions!"  def build\_sentence(info):  return "%s is a benefit of functions!" % info  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()  The website’s script runner says this is wrong but the only difference is that I put my first function’s return in an actual list and named my parameter “info” instead of “benefit”. |
| [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  # your code goes here  car1 = Vehicle()  car1.name = "Fer"  car1.kind = "convertible"  car1.color = "red"  car1.value = 60000  car2 = Vehicle()  car2.name = "Jump"  car2.kind = "van"  car2.color = "blue"  car2.value = 10000  # test code  print(car1.description())  print(car2.description())  Note: \_\_init\_\_ is what’s used as a constructor |
| [Dictionaries](https://www.learnpython.org/en/Dictionaries) Review and complete exercise code:  phonebook = {  "John" : 938477566,  "Jack" : 938377264,  "Jill" : 947662781  }  # your code goes here  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.")  Note: Braces mean dictionaries |

# 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.

| After moving to my main git repo folder, I used the “git init cs3300-version-practice” command to create the folder and initialize it with git. Then I used the powershell command “ni” to create the python file and documentation 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/NathanLindeman93/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

| git add test-file.py  git commit -m “Updated Python File” |
| --- |

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 -a v1.0 -m “Test version 1.0”  git push CS3300-version-practice 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 way to let multiple people work on the same codebase at the same time and provide ways of tracking changes to the codebase and overseeing those changes. By creating branches of the main code, someone can essentially make a separate version of a codebase that they can make changes to without affecting the working code. They can then commit that code to save the changes to their local repository, and merge the code branch back into the code the branch was created from. When the codebase reaches a point of release, it can be tagged which references a release or iteration point which can function while branches are made for updating/testing. |
| --- |

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

| Git is a software version control system that has a focus on branching and merging. It allows you to create branches to work on code without affecting the main codebase. GitHub is a cloud service that hosts repositories, which allows people to clone/pull repositories and then push their code changes back. |
| --- |

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

| Commit is a command that essentially saves the current state of the code and catalogs the changes made, allowing you to comment on the changes and possibly push the changes.  git commit “name/description”  Pull is a command that grabs changes made from the repository/branch and applies those changes to the local repository.  git pull  Push is a command that updates the remote repository with changes made by a local commit.  git push <remote branch> <local branch>  Add is a command that adds modified files to a potential commit. Does not actually commit.  git add <file1>  Clone is a command that creates a local version of a codebase that exists in a remote repository.  git clone <remote repository address>  Status is a command that gives the status of potential changes to the local repository.  git status  Log is a command that lists all of the commits to a repository.  git log  Checkout is a command that lets you switch between branches.  git checkout <branch> |
| --- |

3.3 Explain the difference between a branch and a tag.

| When a tag is created, a snapshot of the codebase is taken at that point in time, and no further commits can be added to it. A branch is similar, as it is also a snapshot, but changes can be made to the branch and committed to eventually update the main branch. |
| --- |

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

| * A version control system makes it possible to rollback and revert changes if necessary.   + If an update was pushed through but it had a devastating bug that wasn’t caught in testing, you can roll-back to a previous version to give time to patch the bug. * A version control system lets multiple people work on the same codebase without requiring constant updating.   + More people can work on a single codebase without it affecting each person * A version control system can allow branching, which makes testing new implementations and changes easier.   + If someone wants to try adding a feature to a product, they can use a branch to add it and test it without changing the main code, and can then merge the branch into the main code if it passes all testing. * A version control system keeps track of changes to the code.   + In industry this allows for more traceability and opens the door to facilitating more code review. |
| --- |

# 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.