COMPOUTATION

For Data Science

Diane Woodbridge, PH.D

UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

Contents

Contents

Version Control

- Git

Programming

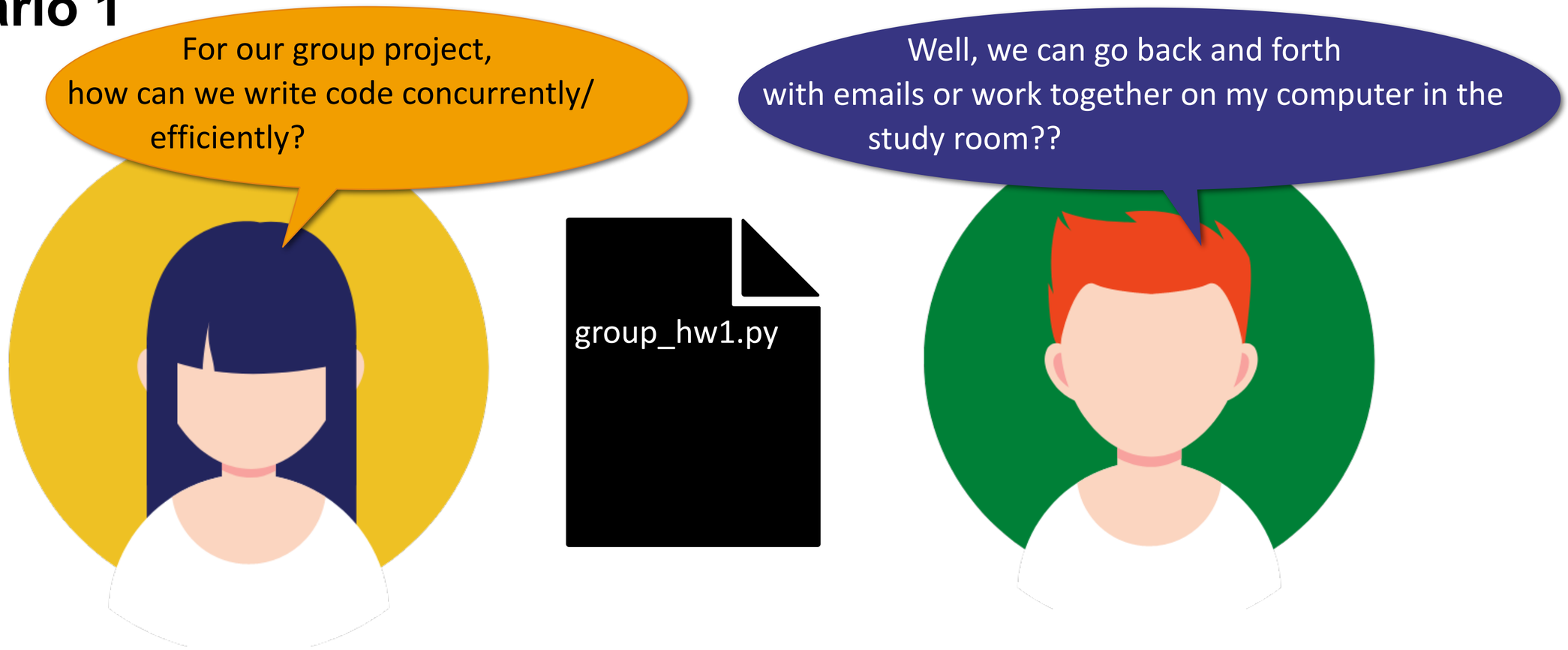
Python Integrated Development Environment (IDE)

Debugging

Error/Exception Handling

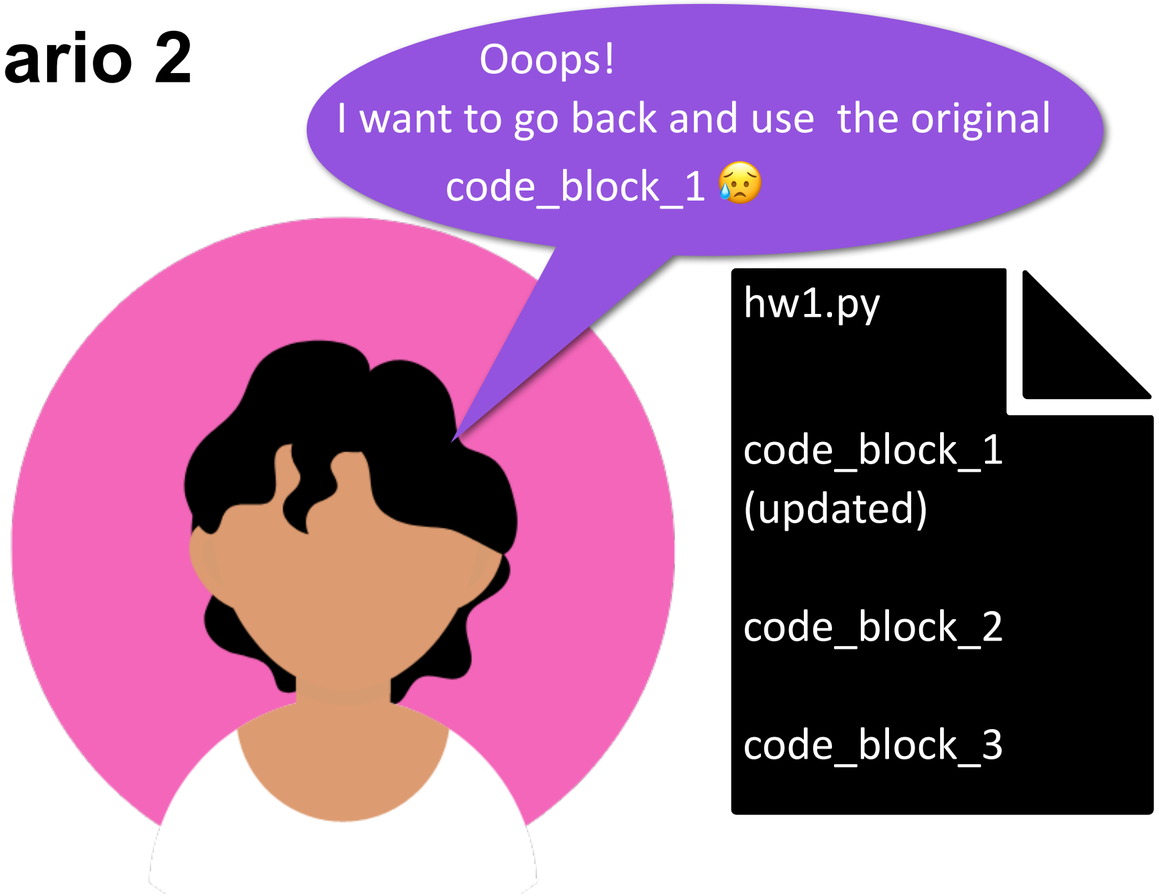
Version Control System (VCS)

Scenario 1



Version Control System (VCS)

Scenario 2



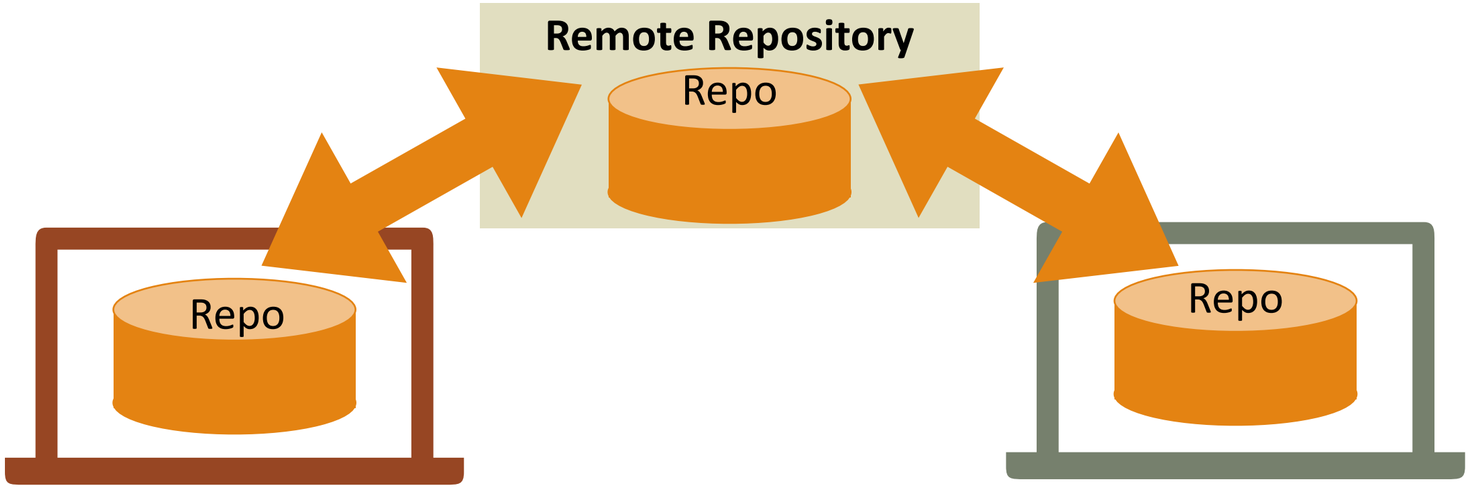
Version Control System (VCS)

System tracking and managing changes to software code

- Allows multiple people to operate on different copies of the repository without getting confused or losing changes. **Great for collaboration and backing up files and their changes.**
  - Repository : Directory subtree containing files and, optionally, directories.

Distributed VCS (DVCS)

- Each client has a local database that tracks changes and stores the copies to the remote central repository.

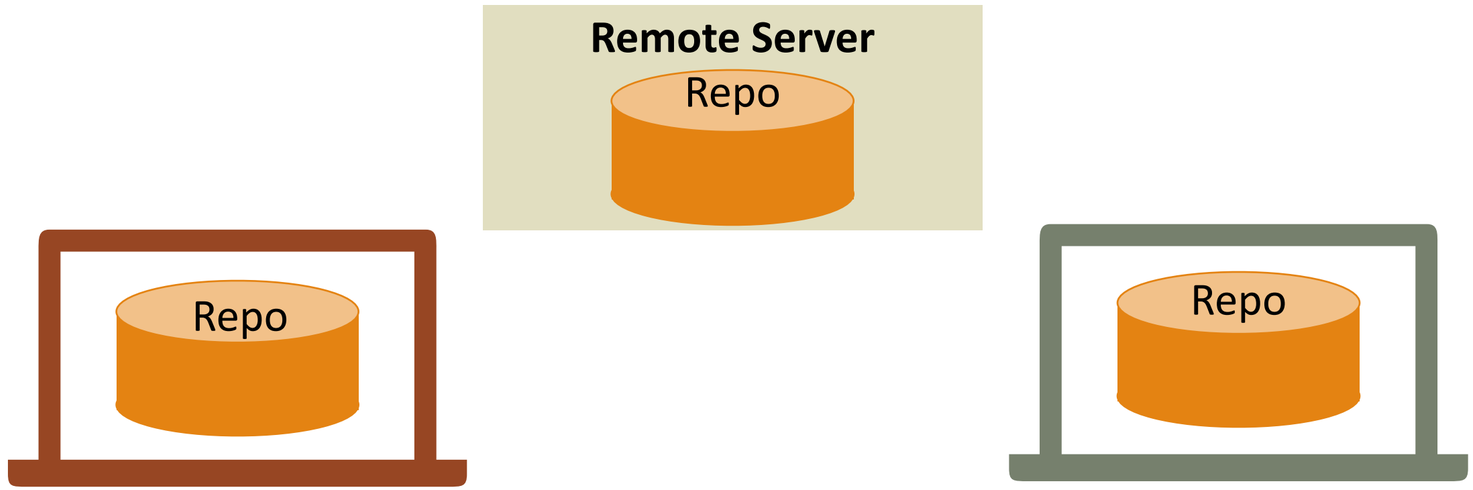


https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Git

DVCS that tracks changes to files and directories within a repository

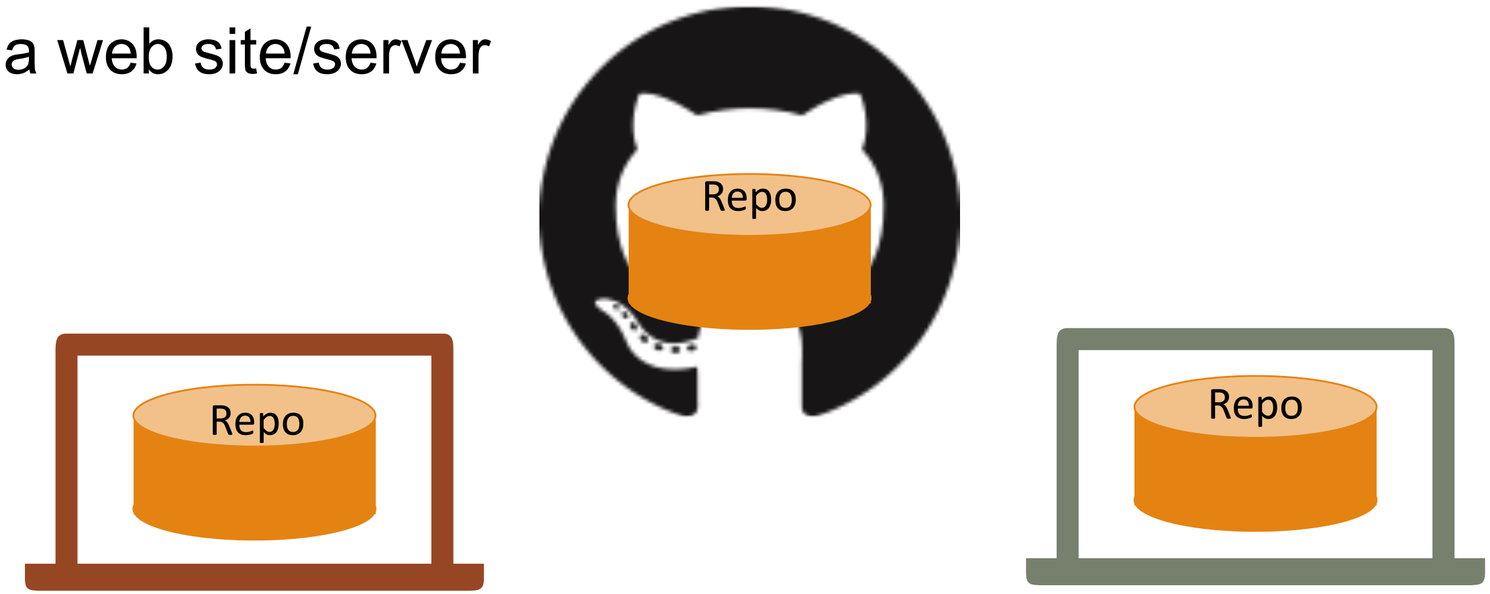
- Workers push/pull changes from a repo on one machine to a repo on a collaborator’s machine
- Git is a program runs on your local machines and remote server.



Git vs Github

github.com

- A remote server that hosts repositories, making collaboration much easier
- Note: git != github.com
  - git is program
  - github is a web site/server



Git

Motivation

- Every developer uses it, as it enhances efficiency and collaboration capacities at work or for open-source projects.
- Every company uses it

Git

First time set up

- Installation
  - `$conda install -c anaconda git`

```
ML-ITS-901885:Day2 dwoodbridge$ conda install -c anaconda git
```

- For our classes, we will use GitHub
  - Create an account at https://github.com/signup
- Configuration - stored in ~/.gitconfig
  - `$git config --global user.name "user_name"`
  - `$git config --global user.email "user_email"`

```
ML-ITS-901885:2022_msds501_example dwoodbridge$  git config --global user.name "dianewoodbridge"
ML-ITS-901885:2022_msds501_example dwoodbridge$  git config --global user.email "dwoodbridge@usfca.edu"
```

Exercise 1

Install Git on your machine.

Create Github account via `https://github.com/signup`.

Git

If you want to see available git commands and their details,

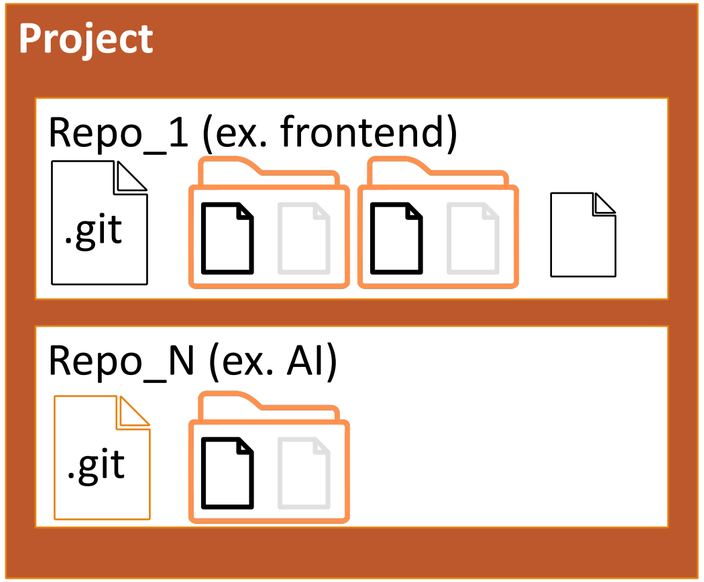
- `$git –help [command]`

```
start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local branch
   push              Update remote refs along with associated objects
```

Git

Repository (Repo)

- The set of files to track is called a repository, your computer will have multiple repositories
  - Each project you work on will be in a separate repo(s)
    - A git repository instance is just a directory but it also has a **.git** (hidden) subdirectory, with a database of all changes
  - All files associated with a repo sit somewhere in or below a directory
  - Not only we have to tell git **when** to take a snapshot, we also tell it **which** files to pay attention to
- To remove a repo, just **rm** the whole repo directory; there is no central server to notify (this would not delete repo from github.com)



Git

Create repo

1. Using an existing GitHub repository

- `$git clone repository_url path_to_clone_repo`
  - If you don't provide path_to_clone_repo, it will clone to your current working directory with the same name

```
ML-ITS-901885:2022_MSDS501 dwoodbridge$ git clone https://github.com/dianewoodbridge/2019-msds694-example ../2019_MSDS694/2019-msds694-example_new
Cloning into '../2019_MSDS694/2019-msds694-example_new'...
remote: Enumerating objects: 110, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 110 (delta 11), reused 9 (delta 9), pack-reused 95
Receiving objects: 100% (110/110), 5.07 MiB | 12.95 MiB/s, done.
Resolving deltas: 100% (58/58), done.
```

2. Creating a new GitHub repository

- First create a repo (https://github.com/new)
- Be in the directory to track files
- `$git init`
- `$git remote add origin url`

```
ML-ITS-901885:2022_MSDS501 dwoodbridge$ mkdir 2022_msds501_example
ML-ITS-901885:2022_MSDS501 dwoodbridge$ cd 2022_msds501_example/
ML-ITS-901885:2022_msds501_example dwoodbridge$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/dwoodbridge/Class/2022_MSDS501/2022_msds501_example/.git/
ML-ITS-901885:2022_msds501_example dwoodbridge$ git remote add origin https://github.com/dianewo
```

Git

Commit Changes

- git tracks snapshots called a ***commit*** (think it as *transactions*). Perform a commit to lock in a logical chunk of work, such as the addition of a feature or fixing of a bug.
- You can check repository status by `$git status`
  - Modified : Additions, deletions, renaming are all considered changes
  - For tracking files,
    - `$git add` *file_name*
    - if all the files, `$git add *`

```
[ML-ITS-901885:2022_msds501_example dwoodbridge$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Day2/

nothing added to commit but untracked files present (use "git add" to track)
[ML-ITS-901885:2022_msds501_example dwoodbridge$ git add Day2
[ML-ITS-901885:2022_msds501_example dwoodbridge$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Day2/ex01_git_commands.sh
```

Git

Commit Changes

- In your repository, you may have tracked files which can be 1) unmodified, 2) modified, or 3) staged.
  - For staging, you can commit with messages (`-m` option)
    - Use the "-a" option on the git commit command to automatically add and commit files that have pervasively committed and updated.
  - `$git commit -a -m "messages"`

```
ML-ITS-901885:2022_msds501_example dwoodbridge$ git commit -a -m "added ex01"
[master (root-commit) 89d46fc] added ex01
 1 file changed, 8 insertions(+)
 create mode 100644 Day2/ex01_git_commands.sh
```

Git

Working with remote repo

- Pull - Fetching changes and merging with the current branch.
  - `$git pull [remote] [branch]`
    - By default, *remote*'s name is origin, and `branch`'s name is master
- Push - Sharing the commits
  - `$git push [remote] [branch]`
  - Every push ensures that the complete file set and git change database (in .git subdirectory) is mirrored at GitHub.

```
ML-ITS-901885:2022_msds501_example dwoodbridge$ git pull origin master
From https://github.com/dianewoodbridge/msds501_computation_2022
 * branch            master     -> FETCH_HEAD
Already up to date.
```

```
ML-ITS-901885:2022_msds501_example dwoodbridge$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 375 bytes | 187.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/dianewoodbridge/msds501_computation_2022/
remote:
To https://github.com/dianewoodbridge/msds501_computation_2022
 * [new branch]      master -> master
```

- Note : Always pull the changes, resolve conflicts where developers might worked on together but has different updates, and then push

Git

Commands Summary

	Commands	Explanation
Configuration	`$git config --global user.name "user_name"` `$git config --global user.email "user_email"`	Add default user name and email to be used in git
Creating Repo	`$git clone repository_url path_to_clone_repo`	Create, using an existing GitHub repository
	`$git init` `$git remote add origin url`	Create an empty Git repository Connect with a remote repo
Checking Status	`$git status`	Show the state of the current directory and files
Tracking Files	`$git add file_name`	Track files and their changes
Committing Changes	`$git commit –a –m "messages"`	Stage eiles
Pulling Changes	`$git pull [remote] [branch]`	Fetch changes and merge with the current branch
Pushing Changes	`$git push [remote] [branch]`	Sharing the commits

- We will cover other git topics including branch as the program proceeds.

Git

Some other useful commands

Command	Explanation
`$git rm filename`	Remove a file from the directory and from git repo tracking
`$git mv from_filename to_filename`	Rename a file or directory managed by git
`$git log`	Shows the commit logs including its Secure Hash Algorithm (SHA), author, date, and message
`$git reset –-hard SHA/HEAD`	Wipe out any changes you've made to managed files, resetting the repository to the previous commit (SHA)/ the most recent commit(HEAD)

What is the typical sequence of Git commands used to send your local changes to a remote repository?

- git commit => git pull => git push
- git commit => git push => git pull
- git commit => git push
- None of the above

What is the typical sequence of Git commands used to send your local changes to a remote repository?

git commit => git pull => git push

0%

git commit => git push => git pull

0%

git commit => git push

0%

None of the above

0%

What is the typical sequence of Git commands used to send your local changes to a remote repository?

git commit => git pull => git push

0%

git commit => git push => git pull

0%

git commit => git push

0%

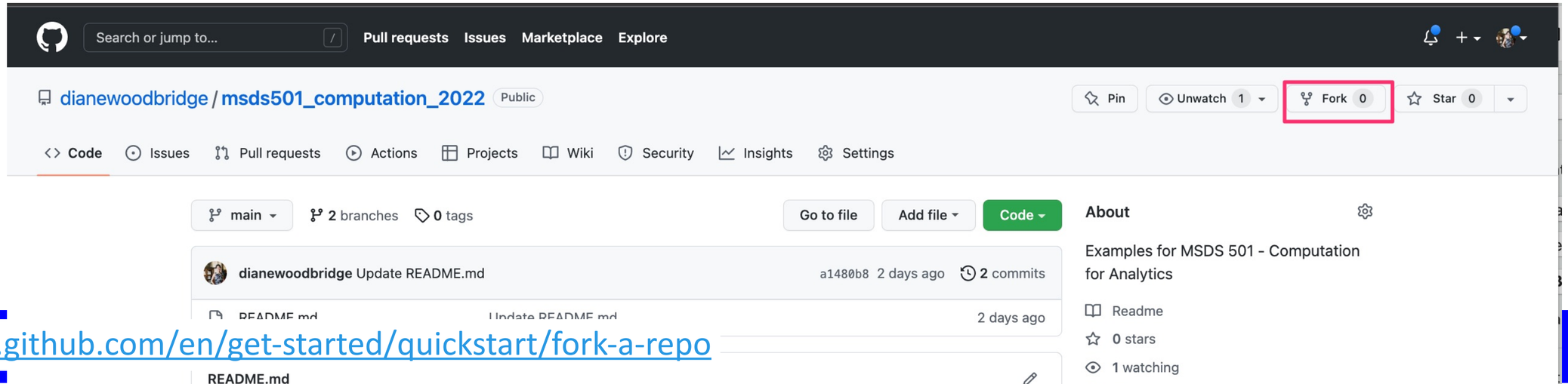
None of the above

0%

GitHub

GitHub Fork

- This is not a git command and just provided by GitHub.
- Create a connected copy of the repository and make changes without affecting the upstream repository.
  - This is for a repo that you don't have a write access or you don't want to write/update.
  - Fork creates a copy to your GitHub and your push is not going to affect to the original repo.
- Create Fork => Clone that to your local repo (instead of the original url)



https://docs.github.com/en/get-started/quickstart/fork-a-repo

GitHub

GitHub Fork

- Once forked,

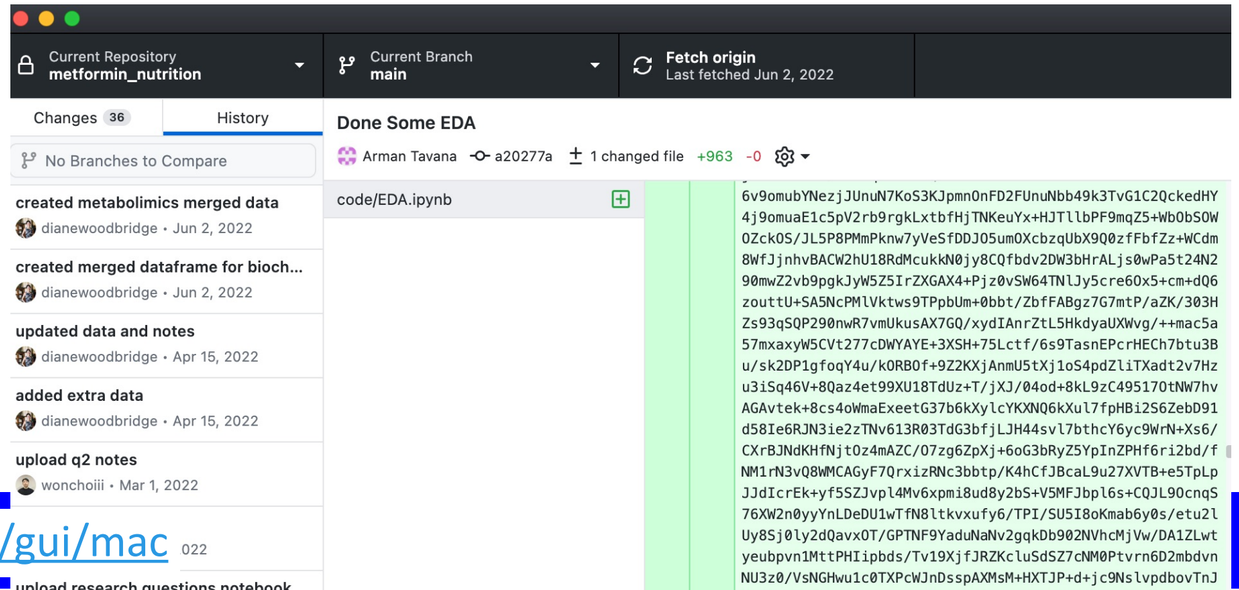
```
git clone [forked_repo_url]
# Clones your forked repository to your local machine
cd [forked_repo_url]

git remote add upstream https://github.com/dianewoodbridge/msds501_computation_2025
# (Optional but recommended) Adds the original repo as the "upstream" remote
# If you don't add upstream,
# when the original repo (dianewoodbridge/msds501_computation_2025) gets updated, you won’t be able to sync with it (unless you explicitly add it as upstream).

git pull upstream main
# Pulls the latest updates from the original (upstream) repository into your local branch
```

https://docs.github.com/en/get-started/quickstart/fork-a-repo

GitHub

GUI Tools

- I found that it is easy to start with GUI to familiarize with Git and see the differences/updates when you're working with multiple people.
  - If you're using GitHub, you can use GitHub Desktop
  - In general, Source Tree and other GUI tools are also useful.



https://git-scm.com/download/gui/mac

Example 1

- Create a fork from https://github.com/dianewoodbridge/msds501_computation_2025 (to your GitHub)
- Clone the forked repo to your local machine

After you fork a repository and clone your fork to your local machine, it is automatically connected to the original remote repository without any additional configuration.

True

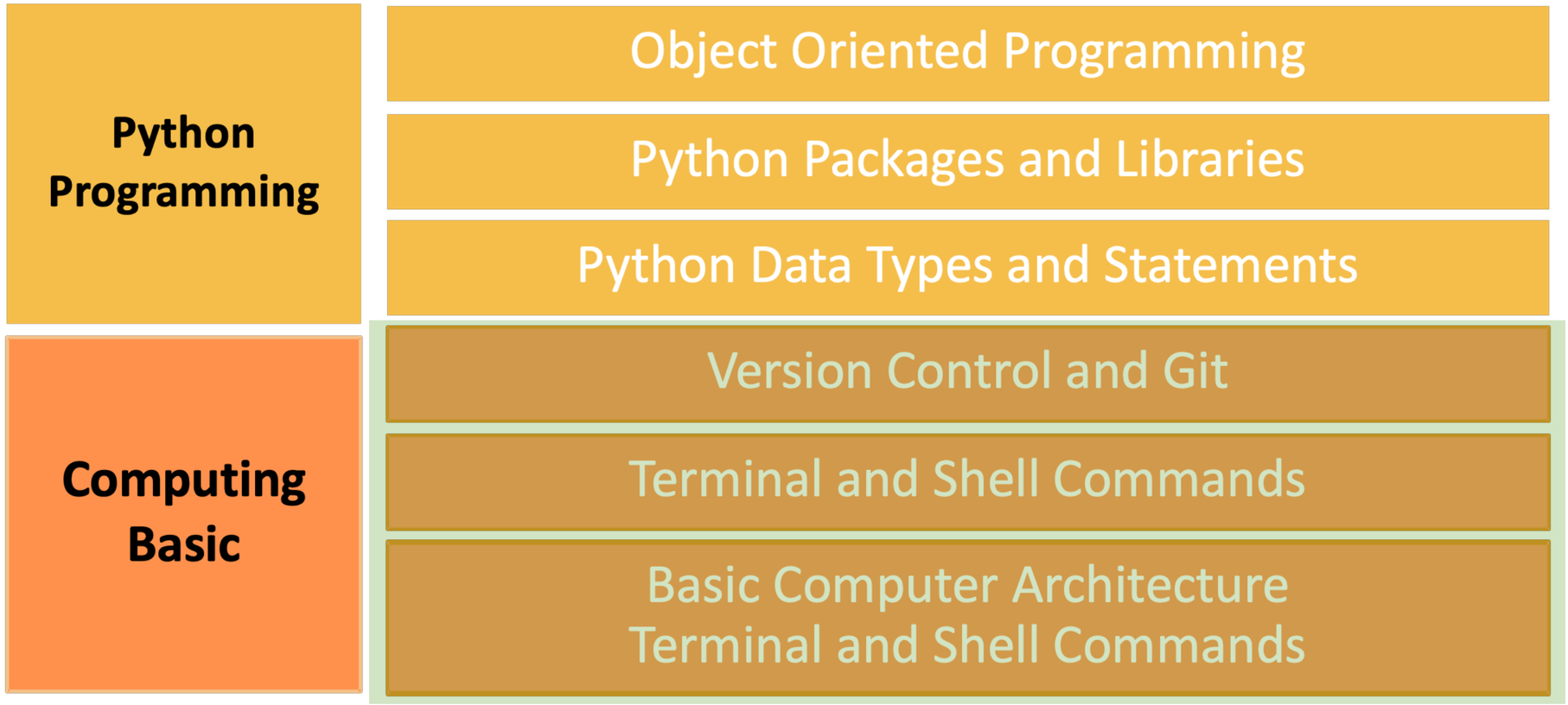
False

After you fork a repository and clone your fork to your local machine, it is automatically connected to the original remote repository without any additional configuration.

After you fork a repository and clone your fork to your local machine, it is automatically connected to the original remote repository without any additional configuration.

Class Summary

Class Overview



Contents

Programming

A process of designing and implementing computer programs

Computers don't inherently understand Python or any other programming language ☹️

- A programmer has to create a program in some language X, called an interpreter, that understands statements in language Y (Python's interpreter is written in the C language)

Contents

Python

A high-level, general-purpose and the most popular Scripting language

- The language that data scientists use

https://www.python.org/

Python Interface

Integrated Development Environment (IDE)

- Please download and install VSCode (Or your preferred IDE).

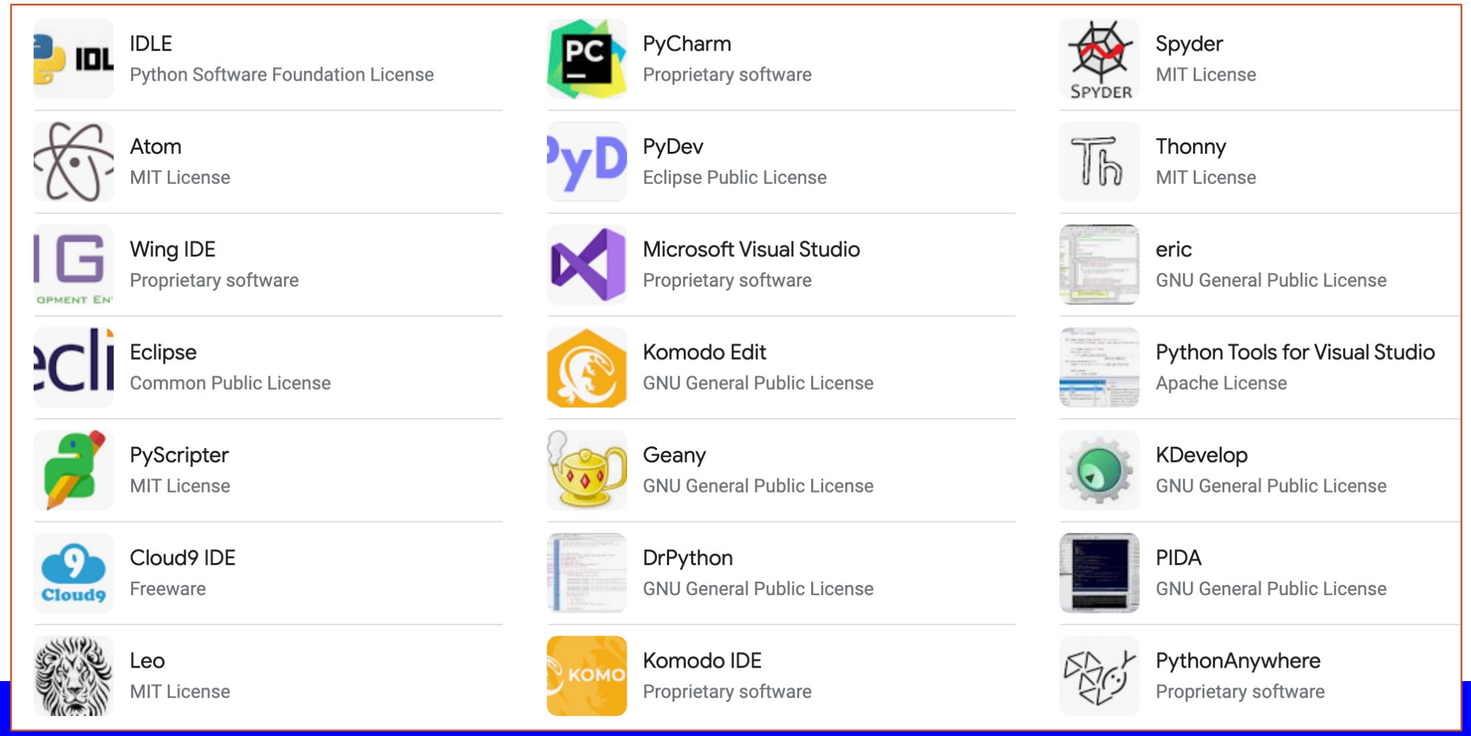
Writing and executing the codes

- For writing codes in .ipynb and .py, I will use VSCode.
- For executing .ipynb in the class, I will use VSCode.
- For executing .py in the class, I will use terminal/VSCode.

Python Interface

Integrated Development Environment (IDE)

- You can use IDEs such as PyCharm, Visual Studio, etc. for writing codes and debugging



Python

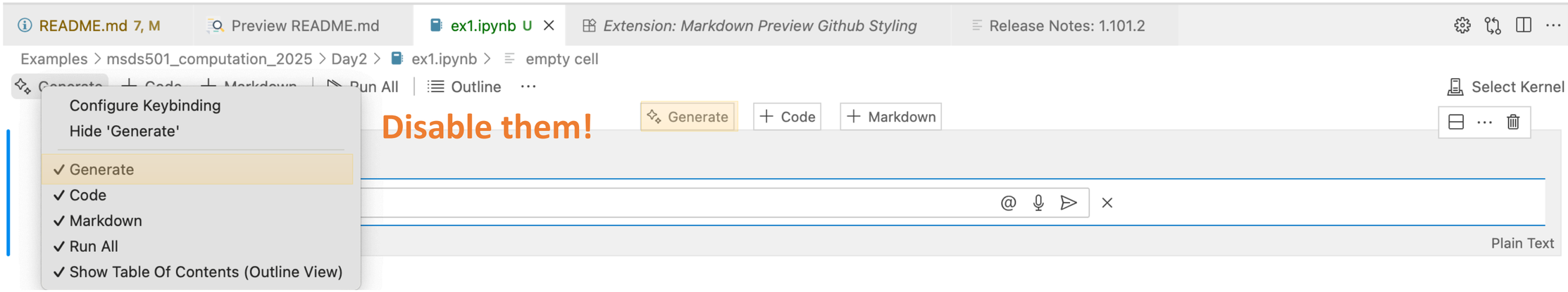
.ipynb and .py file extensions

- `.ipynb`
  - iPython Notebook file extension (has its own format)
  - You can run this on Jupyter notebook / lab
- `.py`
  - Regular python extension (plain text that you can open on text editors)
  - You can execute it on your terminal `$python your_code.py`

Python

Autocomplete/Copilot

- In this class, please disable autocomplete, autocorrection, copilot (especially), etc. to enhance your learning and not to rely on AI.



I completed installing Python and VSCode (Or your preferred IDE)

Yes

0%

No

0%

Contents

Debugging

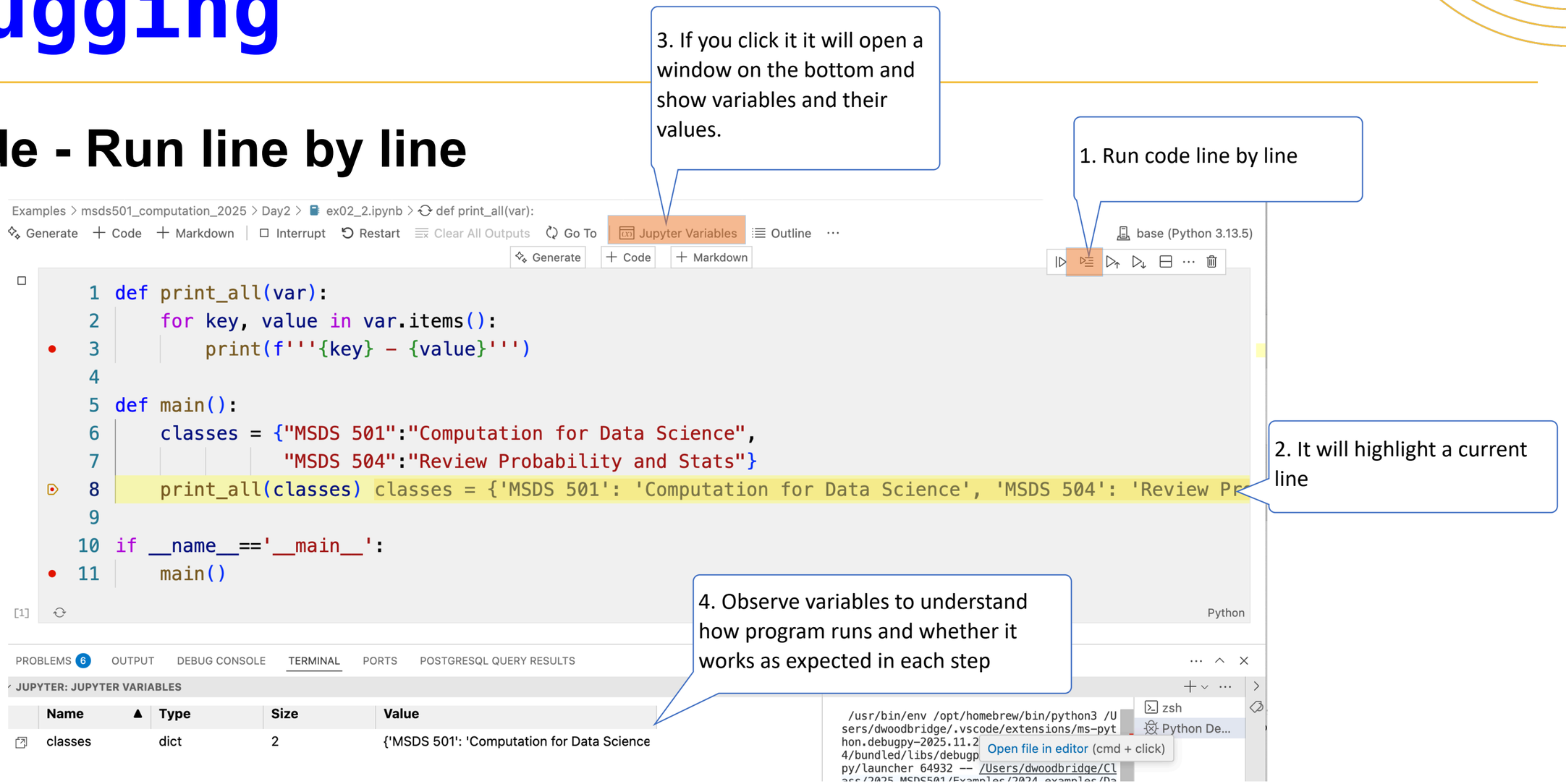
Process of finding causes of bugs/errors and fixing them.

- Important to read the error messages
  - Error message could be detailed enough that you can google to find high-level solutions.
  - You might still want to see where it is being caused and how to fix it by “debugging” your code.
    - Using print statements to see the status/value could be helpful.
    - You can also use your IDE’s debugging features.

https://jupyterlab.readthedocs.io/en/stable/user/debugger.html

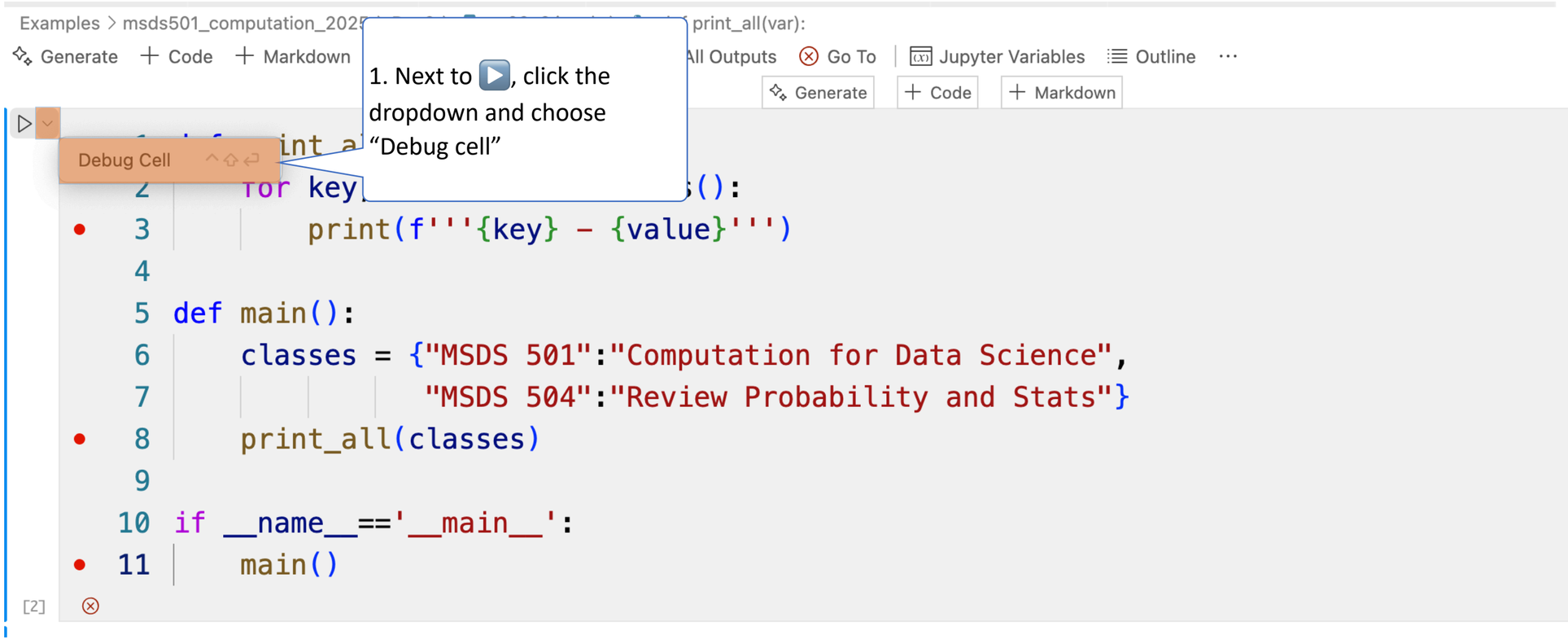
Debugging

VSCode - Run line by line



Debugging

VSCode - Use a debugger

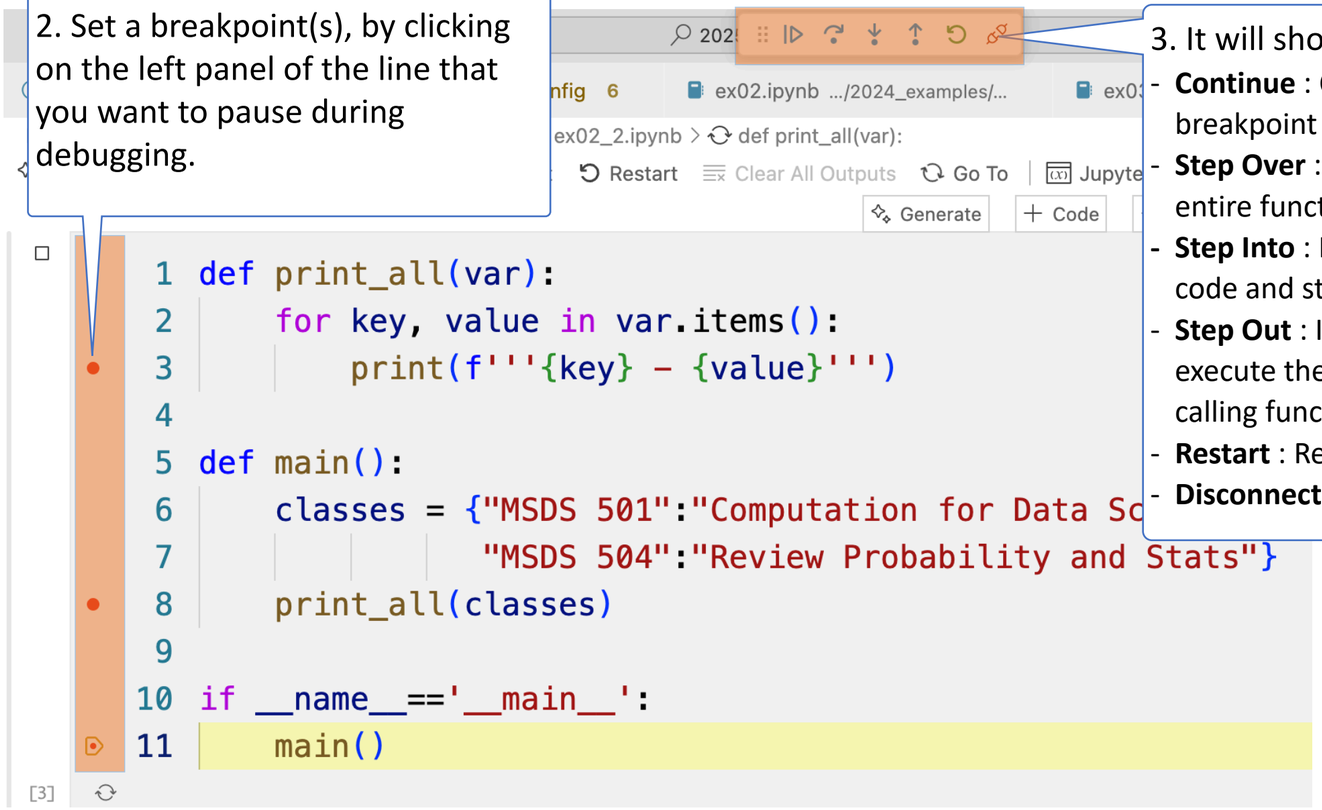


https://code.visualstudio.com/docs/datascience/jupyter-notebooks
https://code.visualstudio.com/docs/python/debugging

Debugging

VSCode - Use a debugger

2. Set a breakpoint(s), by clicking on the left panel of the line that you want to pause during debugging.



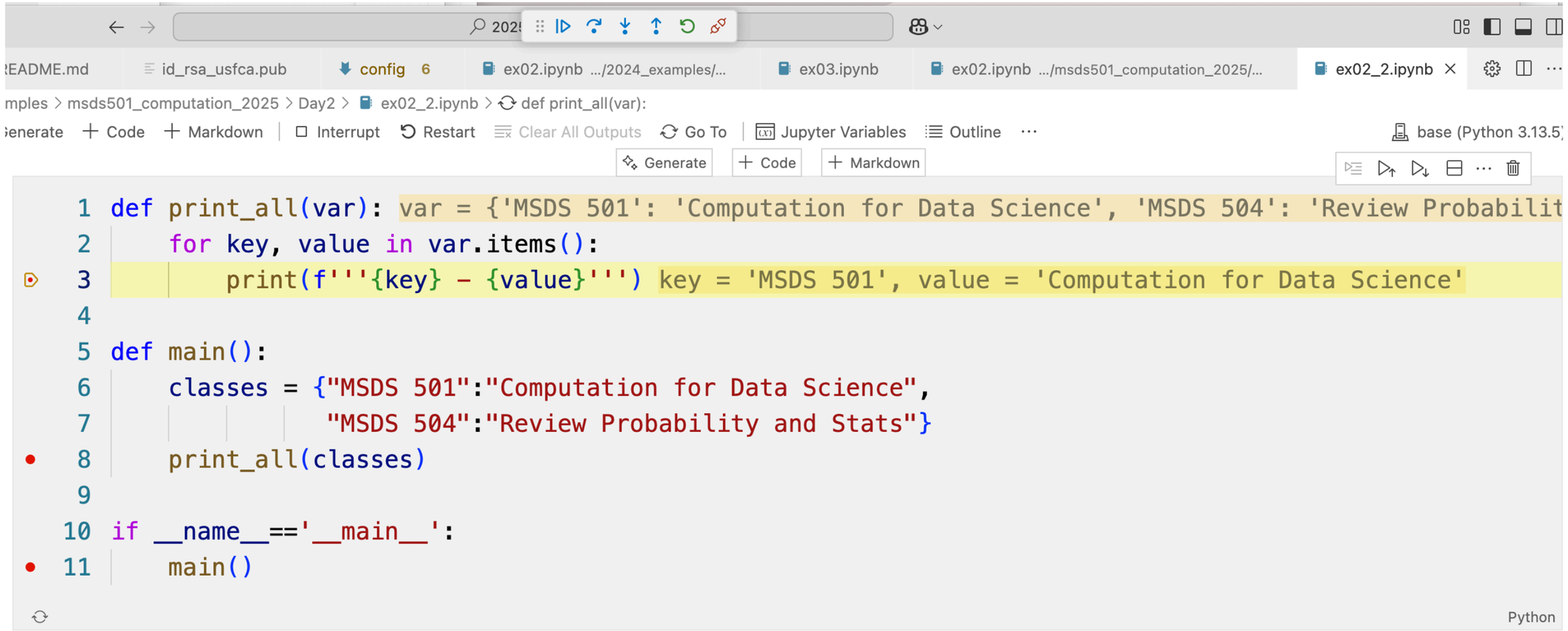
3. It will show a debugger menu.

- **Continue** : Continue program's execution until it encounters a breakpoint or the program finishes
- **Step Over** : If the current line contains a function call, it executes the entire function as a single step
- **Step Into** : If the line contains a function call, it will enter the function's code and stop at the first line.
- **Step Out** : If the debugger is currently inside a function, this will execute the remaining code in that function and then return to the calling function.
- **Restart** : Restart the debugger.
- **Disconnect** : Finish the debugger.

https://code.visualstudio.com/docs/datascience/jupyter-notebooks
https://code.visualstudio.com/docs/python/debugging

Debugging

VSCode - Use a debugger

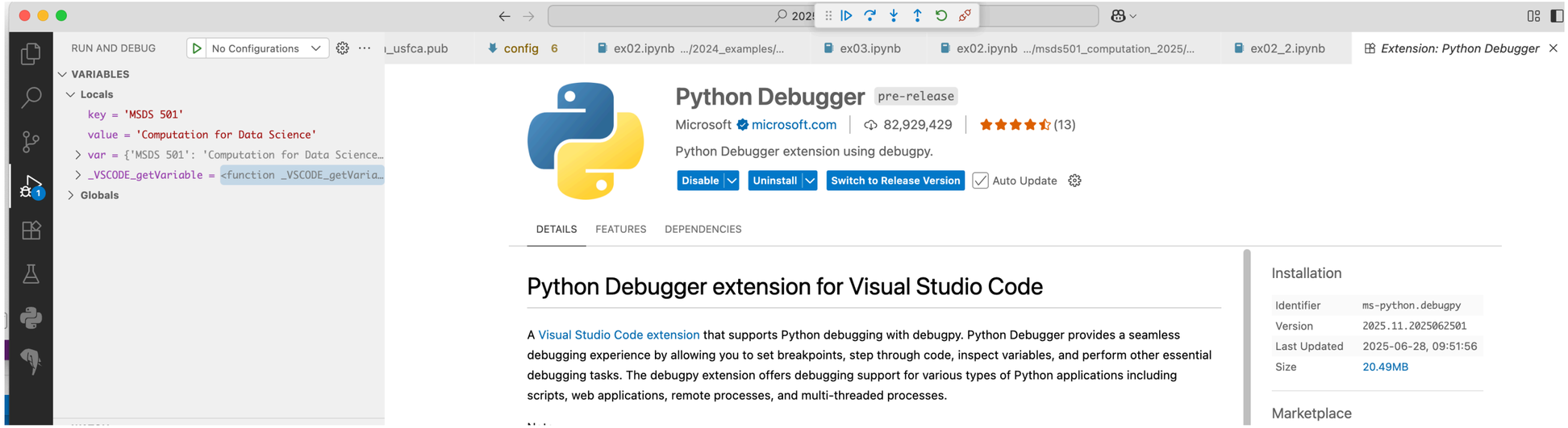


https://code.visualstudio.com/docs/datascience/jupyter-notebooks
https://code.visualstudio.com/docs/python/debugging

Debugging

VSCode

- For .py file follow https://code.visualstudio.com/docs/python/debugging to install the Python Debugger Extension and configure.



https://code.visualstudio.com/docs/datascience/jupyter-notebooks
https://code.visualstudio.com/docs/python/debugging

Example 2

Using print statement, track your variables and values.

```
(base) ML-ITS-901885:Day2 dwoodbridge$ python ex02.py
before print_all()
===============
0
MSDS 501 - Computation for Analytics
===============
1
MSDS 504 - Review Probability and Stats
after print_all()
```

```
(base) ML-ITS-901885:Day2 dwoodbridge$ python ex02.py  > ex02_output.txt
```

Contents

Error/Exception Handling

For some cases, Python will catch serious conditions and throws Exceptions, while most errors cannot be detected until runtime.

- Some exception types - See which line causes this issue.
  - **SyntaxError** : Fix wrong syntax
  - **IndentationError** : Fix incorrect indentation
  - **TabError**: Fix inconsistent uses of tabs and spaces.
  - **TypeError**: an operation or function is applied to an object of inappropriate type
  - **ArithmeticError** : arithmetic errors including ZeroDivisions
  - **ModuleNotFoundError** : importing a library that is not installed. Need to fix with conda install
  - **NameError** : Raised when a local or global name is not found. Make sure the name is defined

Best way to fix it??

https://docs.python.org/3/library/exceptions.html

Example

```
import pytorch
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Input In [2], in <cell line: 1>()
----> 1 import pytorch

ModuleNotFoundError: No module named 'pytorch'
```

Example

Google

ModuleNotFoundError: No module named 'pytorch'

All Images Videos News Shopping More Tools

About 650,000 results (0.40 seconds)

https://stackoverflow.com › questions › no-module-na...

No module named "Torch" - python - Stack Overflow

Feb 23, 2019 · 20 answers

Try to install **PyTorch** using pip: First create a Conda environment using: conda create -n env_pytorch python=3.6.

"**no module named** torch". But installed **pytorch** 1.3.0 with ...	Feb 7, 2020
ModuleNotFoundError: **No module named** 'torch'	Nov 27, 2019
ImportError: **No module named** torch - Stack Overflow	Aug 5, 2020
Conda - **ModuleNotFoundError**: **No module named** 'torch'	Mar 10, 2021

More results from stackoverflow.com

Error/Exception Handling

You can handle exceptions using `try, except` statements.

- Syntax

```
try:
    code_that_may_cause_exceptions
except exception_error_type_1:
    error_handling_code_1
except exception_error_type_2:
    error_handling_code_2
```

Example 3

The following codes have errors.

- inputs : supposed to have only integers, but may contain strings
- When exception happens, print an message indicating the index number of wrong data types considering the error types.

[1]:
```
inputs = [1, 2, 3, 4, "9", "10"]
```

[2]:
```
print(sum(inputs))
```

Contents

Checklist

- [ ] Installed git, and created a GitHub account
- [ ] Installed VSCode (or your preferred IDE)
- [ ] Get familiar with git commands
- [ ] Fork the class git repo to yours and clone that to your machine
- [ ] Be able to debug codes on VSCode (or using print statements)
- [ ] Be able to detect and resolve errors in Python

Day 2 - Comments (What you liked/disliked so far? What should I do for you?)

0

Nobody has responded yet.

Hang tight! Responses are coming in.

References

Git : https://git-scm.com/book/en/

GitHub : https://docs.github.com/en/

Jupyter: https://www.python.org/

Jupyter Lab : https://jupyterlab.readthedocs.io/en/stable/

Python : https://docs.python.org/3/library/exceptions.html