AI for the Arts & Humanities (A) Lab Book

Week 4 – More Anaconda and GitHub

# Part 1: Setting Up Your Own Anaconda Environment

## Task 1: README! Before you start …

Always read the README section – when working with code, applications and software you use in your machine learning projects

So far, you have learned to start a Python notebook and to write some code, markup text etc. You’ve also created a GitHub account and repository. Take the following three steps to get ready for the tasks in the exercise.

Do these three steps every time you attempt the exercises.

### Step 1: First, open Glasgow Anywhere Remote Desktop. Use the Student Desktop.

You can use your own machine but it can take more time to set up just so for your course work. The remote desktop, in contrast, has almost every package.

### Step 2: Go through the standard approach to opening a notebook. Start Anaconda Navigator. Launch Jupyter Notebook. Navigate to your course project folder. Create a notebook or open up your preferred existing notebook.

I suggest not opening a Jupyter Notebook directly – always go through Anaconda Navigator. This allows you to clearly see which version notebook you are opening. The correct version leads to the availability of necessary Python libraries.

### Step 3: Open a web browser (recommend Google Chrome in incognito mode). Navigate to GitHub and log in. Navigate to your repository for the course AI for the Arts and Humanities (A).

Getting used to how GitHub is organised will help you manage your work. By logging in on a regular basis, you will commit your knowledge to muscle memory.

## Task 2: Copy the URL of your GitHub repository from the address bar. Share your GitHub repository information on the Moodle at the link “[AI for the Arts & Humanities (A): Share your GitHub Repository URL](https://forms.office.com/e/LSK58WQt8k)”.

Do this now. Don’t wait until later or you might forget. If we don’t have this URL and information, we cannot assess your work!

## Task 3: Creating a New Anaconda Environment

In this part of the lab exercises, we will learn how to create a new environment for Anaconda.

* This is useful where you do not have administrative access to make changes to the base root environment of Anaconda (which is usually the case in University labs). For example, you might want to install some new Python libraries but you will not have permission.
* It also helps manage your environment, for example, to keep packages only installed when relevant to a selected AI project and to keep things running efficiently.
* The separate environment also helps determine what packages and/or libraries are necessary for your project code.

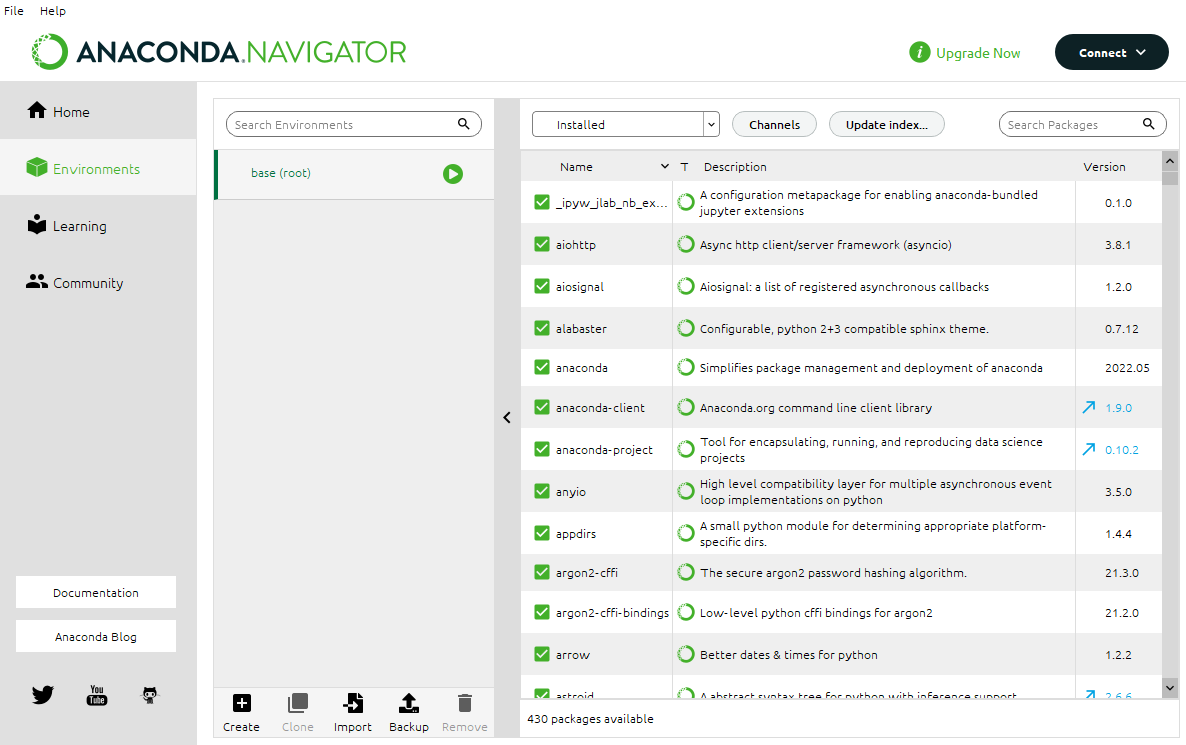
The instructions below assume that you are using a computer (e.g. Glasgow Anywhere) which already has Anaconda installed. If not, go to the link for [Anaconda](https://www.anaconda.com/products/distribution) and follow the instructions for installing the application.

### Step 1: Open Anaconda Navigator if you have not already done it above

Search for the Anaconda Navigator on your computer. On Windows, you can do this using the search box in the Start menu. Type “anaconda navigator” to bring up a list of Apps and Documents that match your keywords. There should be something similar for your own system. **Select the item “Anaconda Navigator” in the search results, to launch the application.**

### Step 2: List the Existing Anaconda Environments

To see the list of environments available to you. Go to the main Anaconda Navigator window and click Environments in the lefthand side column (depicted below for reference). Usually there will only be a base or root environment, as in the example image.



### Step 3: Create a New Environment

To create a new environment:

* Click on create, or plus sign +, at the bottom (of the section displaying the list of environments).
* A window will appear for your to type in a name for your new environment. Call it whatever you like but name it something sensible. Remember what it is called because you will need to make sure it is running when you do your work for the course, and before starting your Jupyter Notebook. You will also need the name for activating your environment in Anaconda Prompt (one of the tasks in this lab).
* Make sure you have chosen Python 3.9 or a higher version (you select this on a dropdown list in the dialog box).
* Once you are happy, click Create.

**If you are prompted to update your Anaconda Navigator, do not update**!

## Task 4: Setting Up Your New Environment as the Default for Anaconda

I recommend you save your new environment as your default environment. That way Anaconda will start in your new environment, so that you do not need to change this after starting the Navigator.

To do this:

1. Go to the File menu and choose Preferences from the dropdown menu. You can also use the keyboard shortcut Ctrl + p (if you are on Windows).
2. A window will open with a list of settings. One of these will be a dropdown menu for the default conda environment - choose your new environment from the list and click Apply to save the settings.
3. Now, to test that it worked, quit and restart Anaconda Navigator. If you navigate to the Environments section again you will see the environment you created selected as the current active environment (highlighted in green).

## Task 5: Install a Package

In this course so far you have been importing several Python libraries. In real life these are not already installed for you. When you get the error message “no module found”, you need to install it yourself. Once you have started Anaconda Navigator and activate your environment. Just click on the environment name in the list. If it is your default environment, it should already be activated and highlighted in green. Carry out the following three steps to install packages/libraries (also known as modules) and beef up your machine learning environment.

### Step 1: List Packages and/or Libraries Available

Select your newly created environment by clicking on the name. Near the top lefthand side of Environments view, there should be a dropdown menu which allows you to choose what is shown in your window:

* Installed (default)
* Not installed
* Updatable
* Selected
* All.

Choose All. This allows you to see all the packages/libraries available for your environment. Those already installed will have a tick in a green filled box next to it. Those not installed will have an empty box next to the package name.

### Step 2: Search and Install Libraries

You can search for any package name in the search box - available in the top right area. There may be many packages containing the word you search for. Scroll down the list until you find the **exact package** **name** you want to install and tick the box. A green downward arrow will appear. Click Apply at the bottom righthand side.

Anaconda will search for all the other packages necessary for your package (these are called **dependencies**). This can take some time as it is trying to sort out any conflicts etc. Once it has finished searching, it will display the packages in a popup window, waiting for you to click apply. If you are sure you want to install then click Apply.

This will start the installation of your requested packages and their associated files.

**The process may take a while, so be patient**! When the packages are installed, the box on the left of a package name will turn green and the box will be ticked.

**Remember, in these exercises, good things come to those who wait!**

### Step 3: Useful packages to install

The following packages are packages commonly used across numerous data science projects including machine learning.

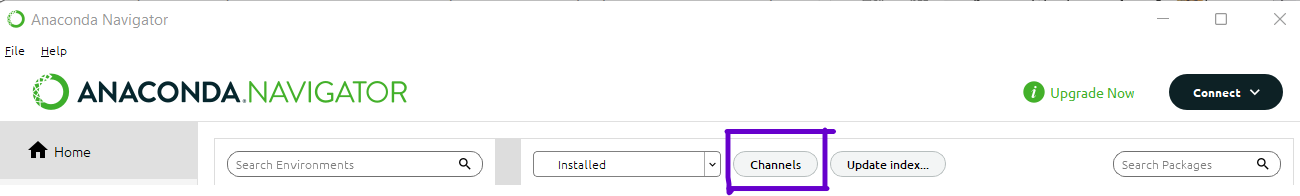
* notebook (this is jupyter notebook – **try to ensure it is 6.5.2**. This is the version that the exercises are written for. Versions are listed in the right most column of the window.
* matplotlib (basic data visualisation library)
* scikit-learn (sklearn, machine learning models, and practice data)
* pandas (data handling and basic statistics)
* seaborn (more sophisticated data visualisations)
* numpy(manipulating maths and arrays)

**You can also use anaconda prompt for installing packages and more but we will learn about that a bit later**.

## Task 6: Adding Channels: Widening the Search for Packages[¶](http://localhost:8889/notebooks/OneDrive%20-%20University%20of%20Glasgow/2022_AI_for_the_Arts/2022_AI_for_the_Arts_B/Labs/Week1/setting_up_your_environment%20.ipynb#1.3-Adding-a-Channel-for-Installing-Packages)

The libraries and packages you can install are initially collected by Anaconda Navigator from the default channel located at <https://repo.anaconda.com/pkgs>. Sometimes it is necessary to get packages from additional channels.

One popular channel is conda-forge. To search this channel, it is necessary to add the channel to Anaconda Navigator so that it knows where to look for packages. To add the channel, click on the button named Channels at the top of the Navigator (highlighted with a purple border box in the image below). This will open a new window. click the button Add ... on the top right hand corner. This will allow you to type in the name of the channel you want to add. Go ahead and type in the name conda-forge and press return (enter on some machines) key. And then click update channels.



You should now be able to see conda-forge listed as one of your channels, whenever you click the button Channels.

Packages openai (of the DALLE fame) and pipreqs (for automatically creating a file listing your environment packages - requirements.txt) are packages available through conda-forge. Now that you have the new channel available, install the following packages following the steps of [Section 1.2](http://localhost:8889/notebooks/OneDrive%20-%20University%20of%20Glasgow/2022_AI_for_the_Arts/2022_AI_for_the_Arts_B/Labs/Week1/setting_up_your_environment%20.ipynb#1.2-Installing-a-Package):

* openai - for setting up api to access gpt-3, codex, and DALL.E
* pipreqs - for automatically creating requirements.txt files for your projects
* tensorflow - for neural network model building and operations. Include the numpy library.
* tensorflow-datsets - library of datsets ready to use with tensorflow
* opencv - for computer vision and machine learning software library

Some of these may take some time - be patient! Depending on your storage allowance and permissions, it may not be possible to install all of these on your network space at the University and/or Glasgow Anywhere Remote Desktop. If so, discuss with the lab tutor and lecturers.

# Part 2: Working with the Anaconda Prompt

Sometimes you might have other commands you want to run in your environment which are **not available on Anaconda Navigator**. For example, you could be running some commands to automatically create the requirements.txt file. The package pipreqs (installed on your Anaconda environment in the last task) allows you to create this automatically. The Anaconda Prompt is useful for this and we will see how this is done.

Note that, anything you might do on Anaconda Navigato, you can do on Anaconda Prompt, but not the other way around. If you like command line better than Anaconda Navigator, you might keep this [conda cheat sheet](https://docs.conda.io/projects/conda/en/latest/user-guide/cheatsheet.html) handy.

## Task 1. Activating Your Anaconda Environment in Anaconda Prompt

To run commands in your specific Anaconda environment, you need to use the Anaconda prompt and use conda activate to start the environment.

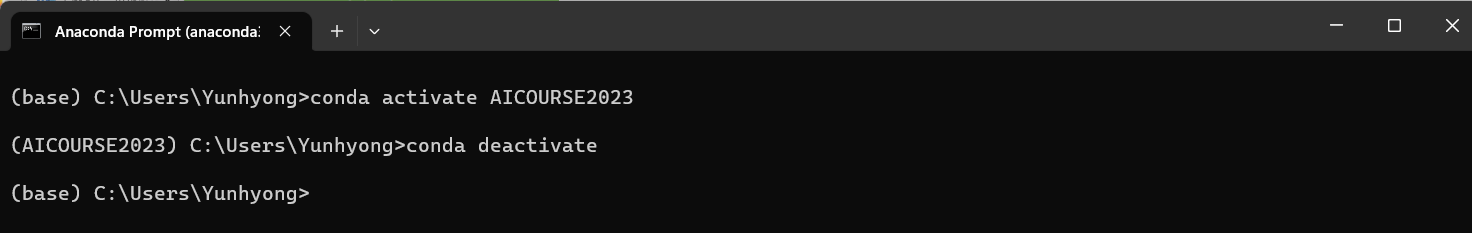
You can search for and start the Anaconda prompt on your computer the same way you looked for and started the Anaconda Navigator - by using the Start menu or equivalent search facility of your system. In MacOs and Linux the Prompt is called terminal.

To activate your project environment, simply use the command:  conda activate *[your environment name]*, where you replace *[your environment name]* with the appropriate name. The command in action is shown below where the environment name is called AICOURSE2023:



## Task 2: Deactivating Your Anaconda Environment to return to the base environment

To deactivate the environment just activated, use the command conda deactivate which will return it to the base root environment.



## Task 3: Generating requirements.txt - a List of Packages Required for Your Project

### Step 1: README!

If you want to share your project with other people, it is important to also communicate all the packages that you need to run your code. Not all packages/libraries are explicitly mentioned in your code. So, you need a way to document these in another way to inform those using your code, what libraries/packages are required to run your code.

The package pipreqs can help you automatically generate this file. There are other tools/commands like pip freeze and conda list -e, but these list all packages in your environment regardless of whether it is used in your project. These are also often tied to **how you installed the packages** - if you installed it with Anaconda Navigator, this may not be that pip-friendly and/or conda-friendly resulting in missing packages in the generated file. The command pipreqs, on the other hand, looks directly at the packages you imported in your code and lists them, avoiding these two problems.

The command pipreqs is best run in the Anaconda Prompt with a project folder containing files relevant to your project. So far in the course AI for the Arts (A), we have worked with Python 3 Jupyter notebooks. For pipreqs, these need to be exported as Python files.

In this section we will learn how to export Jupter notebooks to download it as a python file as well as other formats such as static HTML files. Then we will generate the requirements.txt file.

### Step 2: Export your portfolio notebooks to a Python file (with extension .py)

Start Anaconda Navigator and your portfolio notebook from AI for the the Arts and Humanities (A). Click on the File menu and select Download as -> Python (.py). Depending on your browser setup, the file will be saved in your browser's designated folder for downloads, or you will be able to choose a location. The file name will be automatically generated from your notebook name so choose a good notebook name - something short and without white space characters! Use underscores or hyphens instead.

**How to Export to a static HTML file with extension .html**

The same method as above can be used to create a **static HTML file** by choosing HTML (.html) instead of Python (.py). The static HTML file will be **required as part of your final submission for AI for the Arts and Humanities (A)**. We will talk about this again a wee bit later.

More generally, take a look at all the **different formats** available to you in that area of the File menu.

### Step 3: Copy the path to your new folder

Open Windows Explorer or similar file folder window for your operating system and navigate to the folder containing the python file you just saved above. Right-click on the folder and select Copy as path. This will save the abosolute path to this folder in your computer buffer. Keep this information safe (perhaps paste the path information temporarily into a notepad++ file or similar text editor).

### Step 4: Navigate to the folder in Anaconda Prompt

Open a Anaconda Prompt (or Terminal as per instructions Task 6). Type the command: cd followed by a space, and then paste the path to the folder you just copied, by right clicking on the space after cd. Press the return key (or enter key). This will change your directory to the folder with your python file.

Now type the command pipreqs and press return (or enter). The pipreqs command will look for Python files in the folder, and in the python file you created, and should create a file requirements.txt in the same folder. Check the contents of the file using a text editor (e.g. notepad++) to see if it listed any packages you used in your notebook.

If you already have a requirements.txt file, it will not overwrite the existing file unless you use pipreqs --force instead of pipreqs - a safety feature so that you do not overwrite the existing requirements.txt by mistake.

# Further things to think about

The Anaconda environment you are building in your project is called a vitual environment. The requirements.txt is meant to help others create the same environment to ensure that your code runs for them. By listing all the necessary packages and their versions, you can inform others what needs installed in their own environment.

This, however, does not account for variations coming from different operating systems. To address this some people use [Docker Desktop](https://www.docker.com/products/docker-desktop/) - a container for applications to make them portable. We will not be going into details about this in this course.

If you are interested in this topic, there is a detailed explanation about this in the course core text "Generative Deep Learning".

# Part 3: More on Working with GitHub

## Task 1: README!

By this time in the course you should have created your own GitHub account. You will also have shared your GitHub repository URL – if not, no worries! That task is repeated in this part of the exercise because i**t is super important**, both for your assessment and for a consistent workflow. In addition, you will learn some of the basics in managing your files on GitHub. Your will explore how to organise your repository folder structure and how to access different versions (marked by your **commit history**).

Unless you are using Git (in addition to GitHub), the versions on your computer will not be tracked. If you want to learn how to use Git, Check out the [Git & GitHub Tutorial for Beginners](https://www.youtube.com/playlist?list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR) brought to you by Net Ninja. If you would like to use Git but are not comfortable with command line tasks then the [Git, GitHub & GitHub Desktop Tutorial](https://www.youtube.com/watch?v=8Dd7KRpKeaE) by Coder Coder can also be useful.

Remember: using **Git is not a requirement** **for the course but using GitHub is!**

Carry out the three steps to prepare yourself for the exercises – detailed in **Part 1, Task 1**!

## Task 2: Share Your GitHub Repository URL with Your Course Convenor

Submit your GitHub repository information at the course Moodle link - [AI for the Arts and Humanities (A): Share Your GitHub Repository URL](https://moodle.gla.ac.uk/mod/url/view.php?id=4075746). Follow the Instructions at the link. If you carried out the exercises in **Part 1, Task 2**, you should have done this already. **If not, then do it now!**

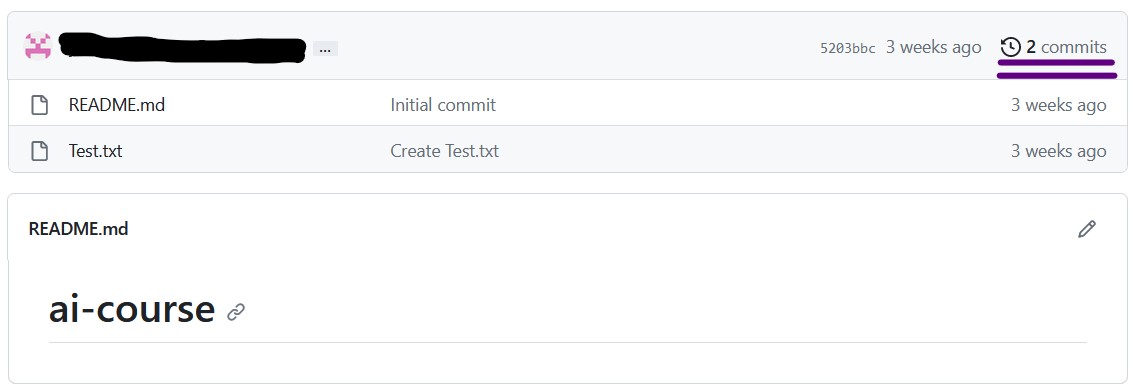
## Task 3: Log into your GitHub Repository and Navigate to your Repository

Depending on where from and how you log in, GitHub might ask you to submit a code sent to your email, so be ready to check your email for this. Once you log in, you will have to find your repository from the list on the left hand side – look for the name of your repository which you chose last week.

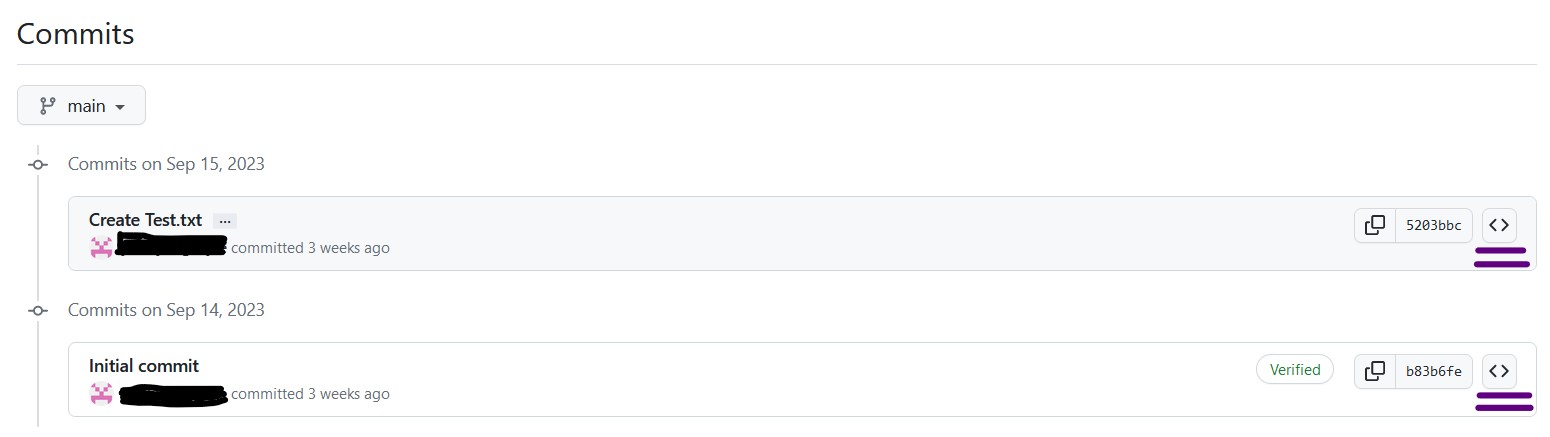
## Task 4: Explore How to View and/or Download Different Versions of Your Repository

Every time you upload new files or new versions of files to GitHub and commit the changes, you create a new version of your repository. GitHub gives access to the versions associated to each committed change.

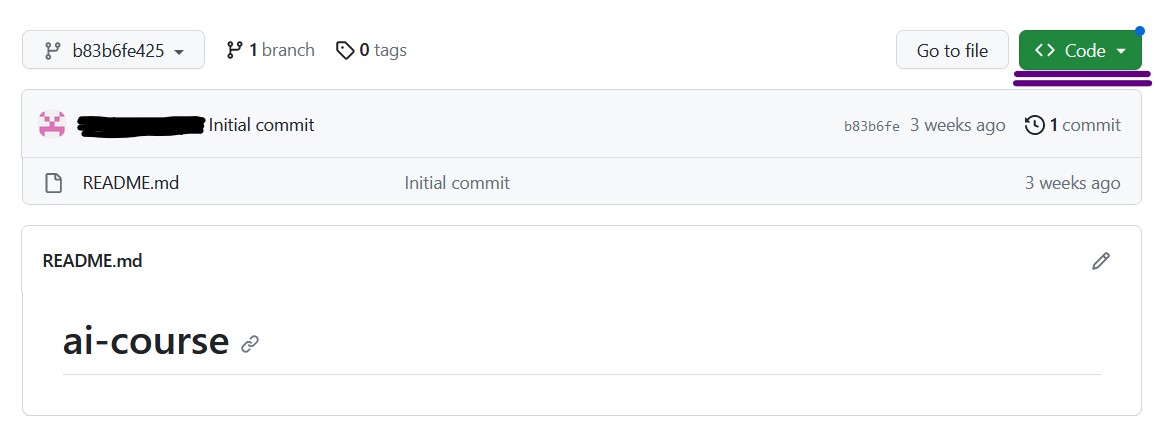
Your repository usually looks like the following screenshot.



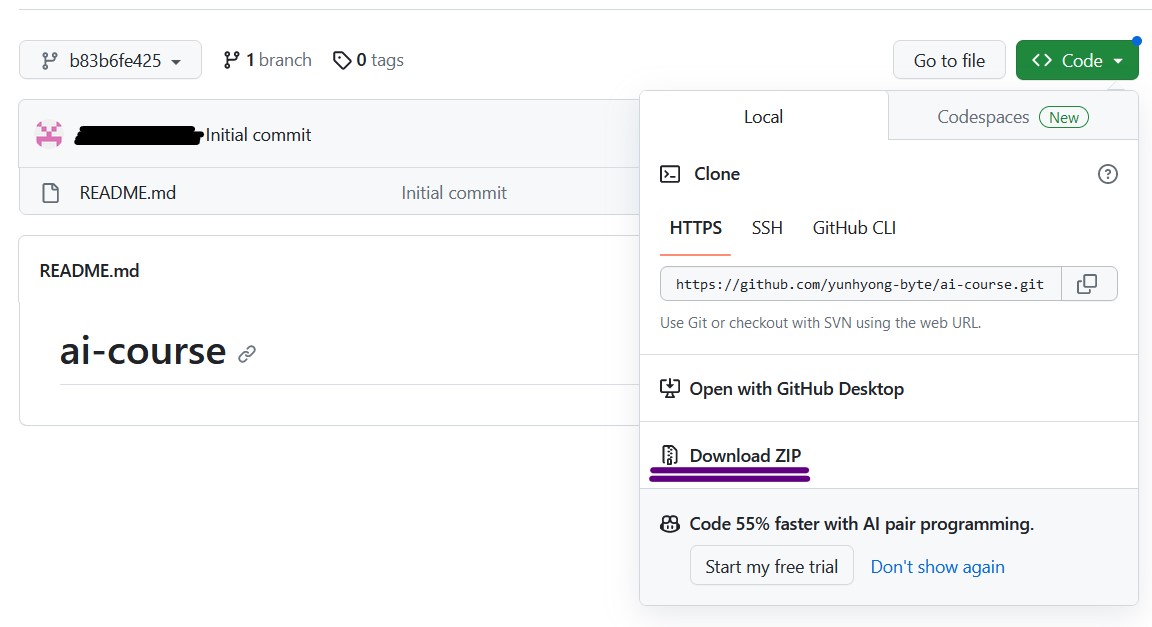
In the top right hand corner you can see a clock-like icon with the number of commits you have made, for example “2 commits” (in the image this is marked with two purple underlines). Click on that text/icon. This will take you to a list of all the commits (screenshot below).



For each item, at the end of the line, there is a symbol consisting of pointy brackets “<>”. Click on this to view that version of the repository. This will show you screen with the items in that version of the repository (screenshot below).



To download the version you are looking at click on the button that says Code (underlined with two purple lines in the screenshot). This is bring up a dropdown list – select “Download Zip” (underlined with two purple lines in the screenshot below).



Each commit has a hash value associated to it. This is displayed in the top left part of the window. See below screenshot for an example underlined with two purple lines.

A screenshot of a computer

Description automatically generated

If you want to make changes to this version as an experiment, you can click on the displayed hash value to create a branch. **This will create a separate branch that you can make changes to without changing the main branch which already has a later commit!**

# Part 4: Practice the Workflow for Uploading Material into the Your GitHub Repository

In Part 1, of this document, you learned how to export your notebook to an HTML file. Say you amended this notebook or created new ones you would like added to your Repository. The following steps represent the workflow that you should employ for this.

1. Work with notebooks in your local environment.
2. Run all the cells in your notebooks. Create a static HTML for your notebooks, emulating the instructions in **Part 1, Task 8, Step 2**.
3. Export notebooks as **Python files** ( files with extension .py - you learned this as well!).
4. **Generate a requirements.txt file.**
5. **Clear the output** from your cells in the notebook (select Cell 🡪 All Output 🡪 Clear) – this will keep your code but remove the output. We do this because data and media loaded into the notebook can cause the notebooks to become more than 25MB is size – GitHub does not allow files of size more than this.
6. Upload **everything inside** your course project folder (subfolders and files) - drop the whole folder content into your GitHub Repository.
7. Commit your changes.

Check your GitHub repository to make sure everything you expected is on your repository.

# Summary

In this document, you learned about:

* Setting up your own Anaconda Environment
* Working with Anaconda Prompt to:
  + Export your notebooks to Python files with .py extension
  + Generate requirements.txt using pireqs
  + Export your notebooks as static HTML
* How to manage content on your GitHub repository
* Practicing the workflow for uploading content to the GitHub

In the next set of exercises, we will dive into machine learning algorithm. **While you wait, be sure to read Chapter 2 of “Hans-on Machine Learning with Scikit-learn, Keras, and Tensorflow” and work on your literature review!**