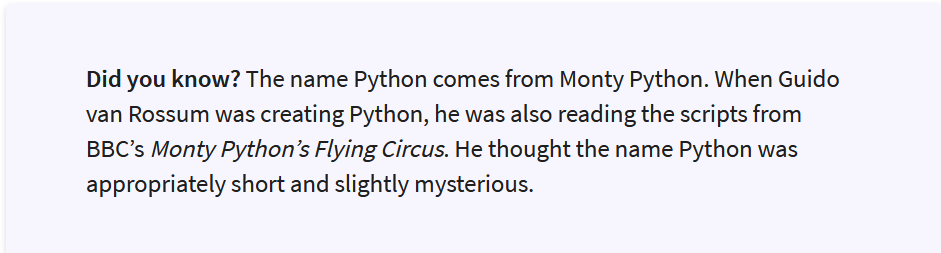
AI for the Arts and Humanities (A) Lab Book

Week 1-3: Getting Started

# Part 1: Getting started with Anaconda, Jupyter Notebook and Python

# Why Use Python?

Python is a highly popular programming language in software development. It finds extensive use in web development, artificial intelligence, and software engineering. Python is versatile, running on various platforms like Windows, Mac, Linux, and Raspberry Pi. Its syntax resembles English, often requiring fewer lines than languages like Java for the same task. You can quickly execute Python code to verify its functionality. This makes it ideal for a broad audience and user base wanting to learn interactively about machine learning and AI.



# Introducing Anaconda: an open source data science platform

Nowadays, many PCs and Macs come pre-installed with Python as part of their operating systems. However, in this course, we are going to use the Python distribution on the open source data science platform **Anaconda.** This will allow you to experiment with Python freely without disturbing the dependencies of your PC/Mac operating system.

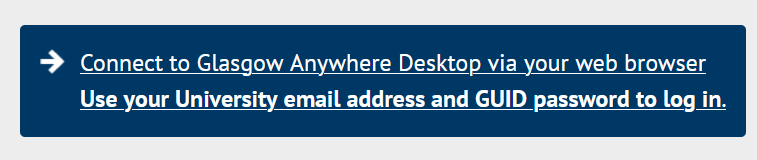
Anaconda also includes basic machine learning libraries and data visualisation libraries that we will be using in the course. In fact new packages have been added to **Anaconda** on a continuing basis over the years! This means you will be able to minimise the number of packages you have to install for upcoming exercises in the course. The **Anaconda** platform should already be installed on the **Glasgow Anywhere Desktop**.

# Getting onto Glasgow Anywhere Desktop

See the instructions for how to use the desktop at:

<https://www.gla.ac.uk/myglasgow/anywhere/desktop/>

Click on the link “Connect to Glasgow Anywhere Desktop via your web browser”. Visually it llooks like the following:



Log in with your University student email address and password at the above link. It will take you to a list of all remote desktops available to you. Click on the one that says “Student Desktop”. If you get any windows asking to you to allow printer, clipboard etc then just click “Allow”. You may need to provide your email address and password again. Use the **Google Chrome** browser if you can.

Glasgow Anywhere desktops work best on the Google Chrome browser – slower on Firefox and Safari and might have compatibility issues.

# If You Need/Want to Install Anaconda

If Anaconda is not installed on the Glasgow Anywhere Desktop or you would like to work on a computer/laptop of your own, you can install Anaconda yourself. Just follow the instructions for downloading the package at:

[https://www.anaconda.com/products/distribution](https://www.anaconda.com/products/distribution %20)



Make a note of where you save the package (e.g. Downloads folder). Navigate to the location of the package and double click the file. This will start a setup wizard. You should select the option to install it for “Just Me” and subsequently select the default options recommended by the wizard. The default download installs Python 3 which is the version we will be working with in this course.

# Opening the Anaconda-Navigator

A user friendly way to access everything Anaconda is through the Anaconda Navigator. To find the navigator, go to the **Start menu** of the computer, and type “anaconda navigator” into the search box to bring up a list of apps. Simply click on the option **Anaconda Navigator (anaconda 3**) to open the navigator (see Figure 1). If you get any messages about updating Anaconda, just click the “no, don’t show again” button. The app should be installed on the remote desktop, If the app is not listed or not installed, follow the instructions in the section “If You Need to Install Anaconda”.

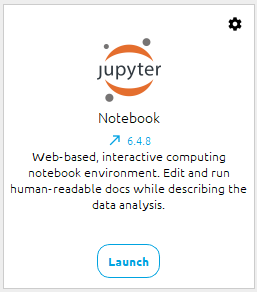
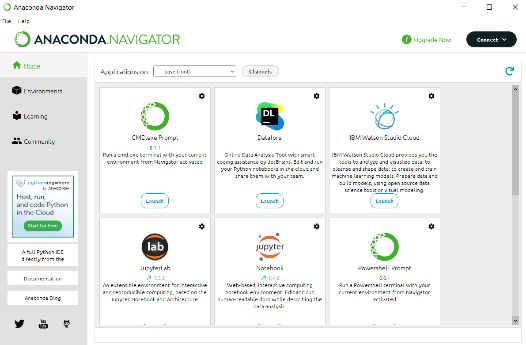




Figure 1. Screenshot of the Anaconda Navigator. The blue border box indicates where you can find the Jupyter Notebook.Icon

# Why Jupyter Notebook?

We will be using Jupyter Notebook to write and present Python and associated materials in your portfolio. The reason for using Jupytrer Notebook is because of its interactive features.

* For example, you can include multiple programs in the one notebook which can be run separately. This makes it easier to test and experiment with ideas before production, and, to collate different tasks in the one notebook, making it ideal as a **portfolio** that presents the results and discussions related to different tasks.
* Jupyter Notebook also allows markdown sections containing formatted text (that is, human readable sections) for contextualising the portfolio and tasks contained therein – for example, the motivation for the project, and/or other linked sources for other related work. This is different from in-line comments which is usually about the code itself and allows you to build a narrative to engage your audience with code.
* Further the Notebook allows you to embed multimedia content in your presentation both as part of the markdown and also as the output of code, helping you to showcase creatively your work to others in the arts and humanities.

# Jupyter Notebook Fundamentals

## Launching the Jupyter Notebook dashboard

Launch Jupyter Notebook by clicking on the “Launch” button in the Jupyter Notebook section displayed in the Anaconda Navigator (see the magnified tile in Figure 1). This will open up your web browsers and display the root folder of your Jupyter Notebook installation (Figure 2). This is a view of files and folders on the local computer you are working on (similar to Windows Explorer or Mac Finder) or, in the case of Glasgow Anywhere Desktop, files and folders on the remote desktop (e.g. Glasgow Anywhere).



Figure 2. Jupy Notebook root folder - this might look different for different people depending on the setup at installation of Anaconda. The dropdown menu for creating new files and folders is highlighted with a blue border box.

## Task 1-1: How to Create Your Project Folder with Jupyter Notebook

It is recommended that you create a new folder for the course so as to keep the course portfolio (i.e. notebook) and files for the notebook in one place. You can call this what you like as long as it makes sense, but in the example here we will call it **2023\_AI\_Arts\_A**. Starting the folder name with 2022 allows you to easily identify the year your project took place. Although it is tempting to abbreviate the course name to something like **AAHA**, this will make it harder to remember later what the acronym was for and even more ambiguous for others who look at your folder (e.g. if you are working in a team and/or need to share code with others). **Thinking about how best you might name your folder** is a transferable skill for portfolio management and an assessment criteria in this course. **Always ask yourself whether it is an informative name - would it make sense when you and/or others look at it later?**

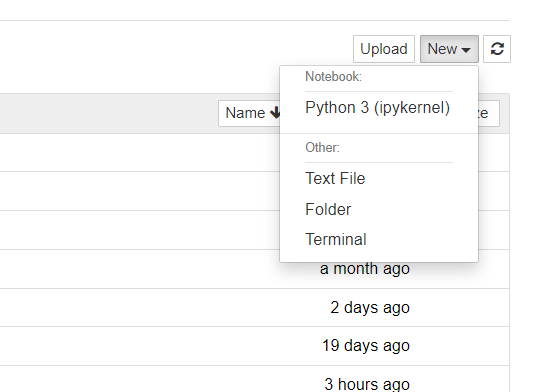


Figure 3. Button for creating new folders. The "folder" item is highlighted with a blue border box. Note that there are also options for creating a Python 3 notebook, a text file and opening a terminal.

* First navigate to the location where you want to create the folder. I recommend the folder named **One Drive – University of Glasgow**. This is your space on the University One Drive – it is backed up regularly and is the largest storage available to most people in the University. To enter the folder, simply click on the name.
* On the top righthand side of the browser, you can find a dropdown menu labelled “New” (see the bolded blue border box in Figure 2). If you click this menu you will see an option that says “folder” (see the part highlighted with a bolded blue border box in Figure 3). Select “folder”. This will create a new folder called “Untitled Folder”.
* To change the name of the folder, tick the box next to the folder name (Figure 4) and then click the **Rename** button (highlighted with a bolded blue border box in Figure 5). A window will pop up for your to input your new folder name.
* Go ahead and type in a name (keep it meaningful) and press **Enter**.

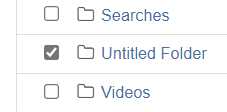


Figure 4. Ticking the box to choose the "Untitled Folder".

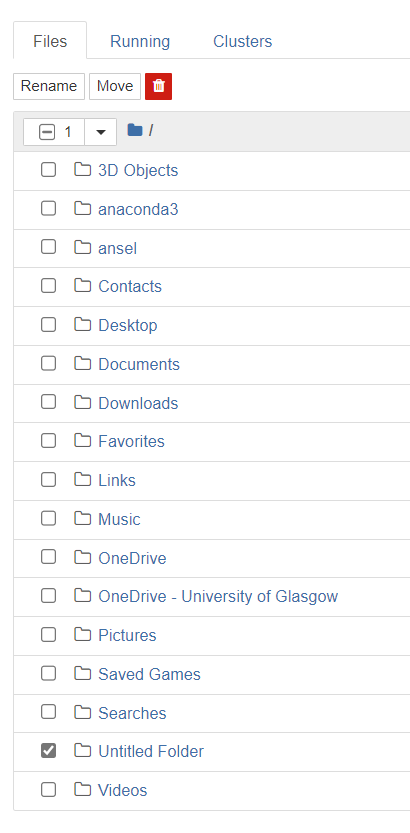


Figure 5. Rename button highlighted with a blue bold border box.

## Task 1-2: Creating a Python Notebook in Your Folder

To enter the folder, simply click on the folder name. To create a new working Python 3 notebook inside the folder, go to the dropdown menu “New” again, and, this time, choose **Python 3 (ipykernel)**. This will open up a Python notebook (Figure 6). You will see that the file is currently untitled (see the bolded blue border box in Figure 6).

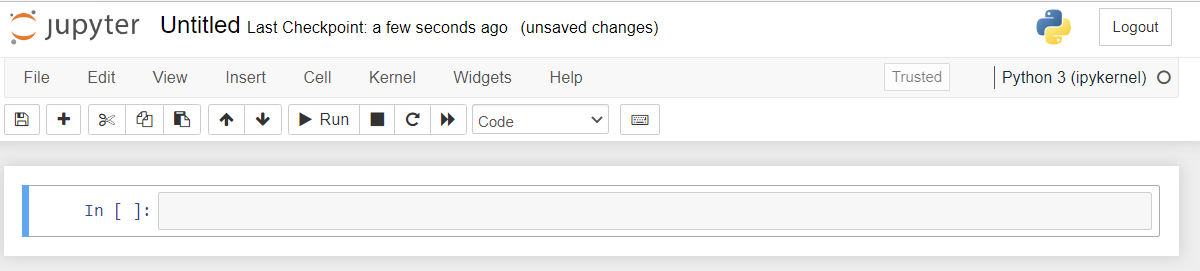


Figure 6. Python Notebook. The title of the file is highlighted with a blue border box.

To rename it, just click on “Untitled”. You will get a popup window (see Figure 7) for typing in a new file name. Change your file name to something informative – for example, *“{Your\_ GUID}*\_AILab\_Week\_*{Number}*” - by replacing *{Your\_GUID}* with your own GUID, and *{Number}* with the actual number of the week.Now click **Rename**. The file will automatically be assigned with an extension .ipynb.

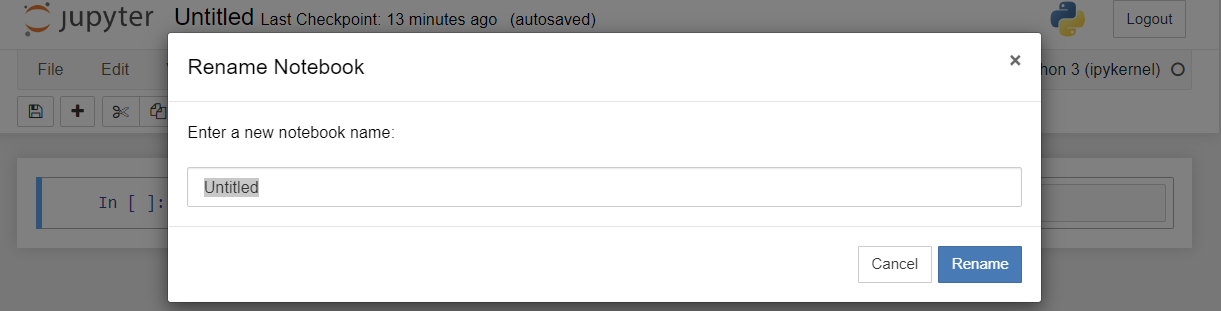


Figure 7. Window that appears when you click on the filename - in our case "Untitled" will be clicked on to change it to "portfolio".

## What is a Cell?

Each of the executable boxes in the notebook are called a cell (see Figure 8 – cell highlighted in blue bold border box).

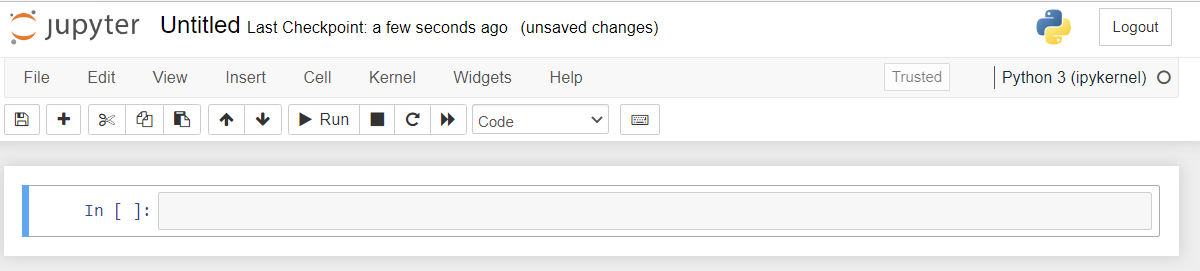


Figure 8. Python notebook. Cell are highlighted with bold blue border box.

Cells are where you write your code or write your textual markdown sections (that is, sections for formatted human readable content). By default, when a new cell is created, the type is set for writing code. You will shortly see that you can change the cell type to accommodate markdown. To run the commands in a cell, you just need to click on the desired cell and click the run button.

If you like keyboard shortcuts, you can use Shift + Enter (press Shift and Enter at the same time). Note that when you run the commands in a selected cell, another cell is created automatically below that cell unless another cell already exists.

# Your First Markdown Cell

To write styled textual content (e.g. introductory section with headings on why you are interested in AI) you need to change the cell type to markdown. This is done by clicking on the “Cell” menu and choosing the option “Cell Type” and sub-option “Markdown”. You can also directly choose this from the pulldown menu (bolded blue box in Figure 9).

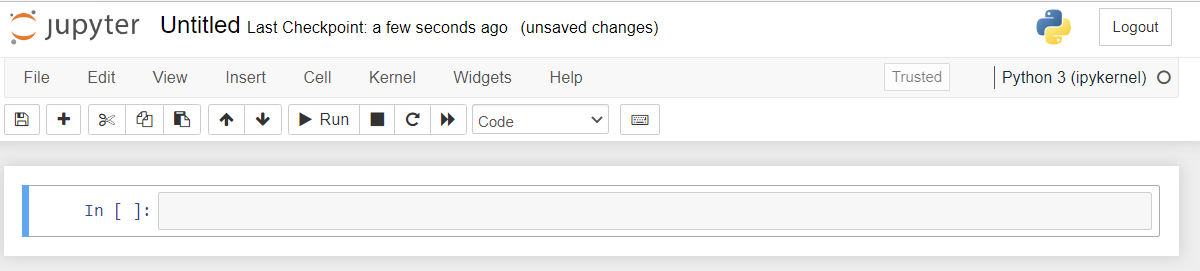


Figure 9. Pull down menu to change the cell type – highlighted using bolded blue border box.

* You should already have one cell available to you in your new notebook. Change that cell type to Markdown.

### Get Started with Your Markdown cell

* **Headings:** Type a hash tag # followed by your heading (see Figure 10). Depending on how many hashtags you use, the heading style will be different – more hashtags there are, the smaller the font size.
* **Paragraphs:** You can also create plain paragraph content by removing all hashtags.



Figure 10. Creating a heading and some associated content in a Markdown Cell.

* Run the cell to get the final result of the styled text (Figure 11).

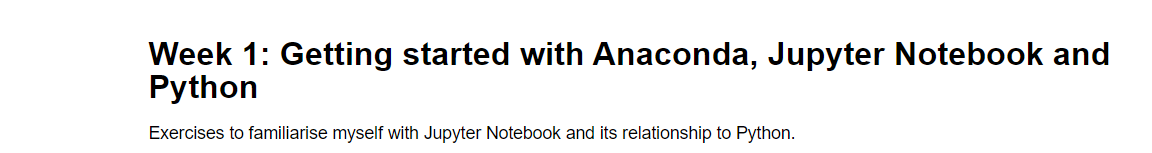


Figure 11. Markdown result after you run the cell.

* To modify or edit, double click on the Markdown Cell.

### How to Style Your Markdown Cell

There many other ways to style your markdown content. In fact, if you are familiar with Hyper Text Markup Language (HTML), a lot of that will work for this also. For an overview of Markdown syntax native to Jupyter Notebook, you might like to review the cheat sheet at:

<https://www.ibm.com/docs/en/watson-studio-local/1.2.3?topic=notebooks-markdown-jupyter-cheatsheet>

## Task 1-3: Create a styled markdown section to add some context to your portfolio

* First add the heading “Week 1: Getting started with Anaconda, Jupyter Notebook and Python”.
* Create three subsections and, in each of the sections, respectively, write a couple of sentences on:
  1. Why you chose to join this course – for, motivation, vision, aspiration?
  2. Prior experience, if any, you have with AI and/or Python, and,
  3. What you expect to learn from the course (aim for 3-5 bullet points)

# Creating and deleting Cells

You may already have noticed that cells can appear automatically below the cell you are running.

* You can also insert new cells manually using the “Insert” menu to insert cells above or insert cells below your current position. This allows you, for example, to add cells you may have forgotten to add in the first instance, and/or split content in a cell into two cells. You can also use the button with the **+** symbol.
* Likewise, to delete any existing cells, click on the cell you want to delete and go to the Edit menu and select “delete cells”. You can also just “cut the cell” by using the button with the **scissors**

# Your First Coding Cell: Hello World

## Task 1-4: Create a code cell with a very simple Python code

* Following your markdown cell, create a new cell (if it is not already created) and make sure it is a cell for writing code. In the cell, simply type:

print ("Hello, World!")

* Now run the cell (review the section “What is a Cell?” if you do not remember how to do this.

This code has a simple function– to printmessages out onto the screen. Printing is a very common command in programming. What do you get when you run the code? Compare the result with the members of your peer group to see if you got the same result – **comparing output** is one of the many ways to engage with the accuracy of your code. The text **“Hello, World!”** above is called a **string** in Python – you can spot these by the fact that they are surrounded by quotation marks**.**

Strings are always wrapped in quotation marks.

A string iswhat is called a **data type** in programming terminology. It **is** distinct from something like a **number**. It is one of the most frequent data types used in programming. We will encounter more data types later.

## Using Variables

You can also use a *variable* (say, call it **message)** to store a *value* – in this case, the *string* “Hello World!” - so that, if you should decide to change your message, you need only change what is stored in the *variable* once, and do not need to change for every instance where the string is used. You assign your value (in this case, “Hello, World!”) to the variable **message** using an equal sign (=). Then use the command: **print (message).**



## Task 1-5: Use the Code Above to Answer the Following Questions

* Try changing the value of **message** to print out your own greeting.
* What happens when you print **message + message**? What is your output?
* What happens when you print **message\*3**? What is your output?
* What happens when you print **message** [0]? What is your output? Why? What if you change 0 to a different number?
* Do you think **message** is a good variable name? If you should change it, you need to change the line with the **print** function. How would you change it to get it to print out the value of the newly named variable? **Note:** the code will remember the value of the previous variable **message** – so don’t be fooled into thinking it works without changing the **print** commend.

In this course, we are only going to engage with the very basics of Python, just enough to help you understand some of machine learning concepts – i.e. get you started **reading code rather than writing code**. If you are serious to learn more, there are a number of great online resources to check out, depending on your level:

* [Kaggle, Intro to Programming](https://www.kaggle.com/learn/intro-to-programming)
* [Kaggle, Python Course](https://www.kaggle.com/learn/python)
* [Non-Programmer’s Tutorial for Python 3](https://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3)
* [Learn Python](https://www.learnpython.org/)

**Homework:** you might like to experiment some more with the concept of **strings** and other data types at Kaggle (<https://www.kaggle.com/code/alexisbcook/data-types>) and at the [Non-Programmer’s Tutorial for Python 3 page](https://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3/Who_Goes_There%3F).

# Displaying a YouTube Video in Jupyter Notebook

To finish off today’s exercise with Jupyter Notebook, we will learn how to display an embedded YouTube video in Jupyter Notebook as output of code. This is one of the ways you can include multimedia with your code, potentially, to make it more engaging for your audience. The codes provided in this exercise are displayed as an image so you will not be able to copy and paste the lines – that’s right, you need to type it in!

## Task 1-6: A first look at importing library and packages

Create another cell in your notebook. As a first step, you need to get the library/package/module that has the command for displaying videos. The library in this case is called **IPython.display**. To get the library you use the key word **from** followed by the name of the library (in our case IPython.display) and then, on the same line, **import** followed by the name of the specific package you want to import (Figure 12). When you use an asterisk after keyword **import**, then this means that you want all the commands/functions available.



Figure 12. How to import all the commands from the library IPython.display on Jupyter Notebook.

The **import** statement above will make everything in the library **IPython.display** available for you to use in your code. Importing everything is an overkill for our purposes here, but we introduce it so that you get a feel for how it is done. In this course, we will be importing a number of libraries, models etc. So it is important to get used to this process.

If you should get an importing error, or if you are not sure it did the job, compare with your peers to see if it worked for them and compare the code. Remember, Python is case sensitive,

## Task 1-7: Find and Display a YouTube Video

* Now use your web browser to search YouTube for a video for which the licence permits you to display it for private/educational purposes. In the example below I have chosen a video of the cartoon character “Bugs Bunny” screaming. Normally, it would be better to choose something related to your message in the previous task. Perhaps a video of hello in a selected language? Feel free to be creative!
* Now simply use the following command to display it in your notebook. The ID in the quotation code below – “05PKG\_pWsVY” - is the ID of the Bugs Bunny video. You will need to locate the ID for your own video – this is usually included in the URL. Go ahead and embed your chosen YouTube video by using the syntax **YouTubeVideo(*video\_ID*),** replacing ***video\_ID*** with that provided on YouTube**.**

Visual demonstration of how the YouTubeVidoe command is used in Python.

You can combine this command and the import command in one cell to run together or have separate cells for each command – the choice is yours. **When you are first starting out with coding, separating it, can help you to understand where the first error is if your code does not work.**

Once you have run the commands, **discuss it with your peer group**.

* If someone is having trouble help each other out.
* See what other people have chosen for their video.

There are also commands in the library to embed other media such as images and audio. Check out the examples at:

<https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html>

# Understanding a Program: Parsing and Reading

So, You’ve learned about importing libraries, variables, and printing. The following program takes as input a site URL, and a date, and searches the Internet Archive for a version closest to the date you specified and displays it in your browser:

**import webbrowser**

**import requests**

**print("Shall we hunt down an old website?")**

**site = input("Type a website URL: ")**

**era = input("Type year, month, and date, e.g., 20150613: ")**

**url = "http://archive.org/wayback/available?url=%s&timestamp=%s" % (site, era)**

**response = requests.get(url)**

**data = response.json()**

**try:**

**old\_site = data["archived\_snapshots"]["closest"]["url"]**

**print("Found this copy: ", old\_site)**

**print("It should appear in your browser.")**

**webbrowser.open(old\_site)**

**except:**

**print("Sorry, could not find the site.")**

**Do not expect to understand everything now – it is a big program, if you are only starting out with Python!** However, you should be able to identify how many libraries were imported, how many variables were used, how many strings appear in the programme, and how many times something was printed? What was printed?

## Optional Task 1-8: Run the code below

* Create a code cell and type this code into your Python notebook. **The indents are important and part of the python syntax!** Use 4 spaces per indentation level. This is what the [PEP 8 Style Guide for Python](https://pep8.org/) recommends.
* Once you have typed it into your cell, run it to see what it does.

## Optional Task 1-9: Annotate each line with a comment

* Python comments are preceded by a hashtag #. These are lines of the code that will be ignored when you run it. Use comments to annotate the code in your notebook to say whether it is printing something, assigning a value to a variable, and/or importing a library.

**Hint:** there are only three lines that do not fall into any of these categories. Can you which these are and what they are meant to do?

# Closing Down Your Notebook and Anaconda

Well done if you made it up to this point!

To close down your notebook:

* Go to the **File** menu and select “Save and Checkpoint”. It should be autosaved anyway, but best to be sure.
* Remember to close the Anaconda Navigator window (confirm “Yes” to close it down).
* Don’t forget to log out from the computer if it is not your own computer – likewise log out of the Glasgow Anywhere Desktop. If you do not log out, things could be running in the background for sometime. **This is bad for security, for the protection of your data, and for energy conservation!**

# Summary

So far, you learned:

* Why we use Python, Anaconda, and Jupyter Notebook
* How to open Anaconda navigator, and open Jupyter Notebook
* How to create a folder, name a folder, open a Python notebook, and, rename a notebook.
* You should have:
* Written your first markdown cell for this course and styled it.
* Written your first code for the course: Hello World! - using print function without and with the use of a variable.
* Experimented with operators like + and \* on strings.
* Displayed your chosen YouTube video – practiced importing libraries
* Explored (optionally) a bigger program to test how much you have learned about reading Python code so far.

# Comment: looking forward …

It is all very well to create a single self-contained notebook. However, once you start embedding standalone images, working with many notebooks, and/or testing on your own data, the situation will easily become unmanageable. Keeping files together in the correct folder structure and keeping track of different file versions will become challenge. Next week we will discuss the options for handling this.

# PART 2: Getting Started with GitHub

In this week's lab you will be setting up your own repository on a **web server** for your Portfolio. In this course, we will use **GitHub** for this purpose. GitHub boasts more than 100 million developers under its wing, helping them to manage their files. You to gain knowledge and experience managing files across your own storage (your computer and/or University OneDrive) and the GitHub repository. It is first step in engaging with a popular professional production environment – a transferable skill for this course and valued by the broader professional developer communities that collaborate with a multidisciplinary team. GitHub is a great way for you to showcase your projects, portfolios, or personal blogs. Here's a step-by-step guide to help you get started.

# **Getting Started with GitHub repositories**

#### Task 2-1: Create a GitHub Account

If you don't have one already, sign up for a GitHub account at https://github.com/.

* GitHub is a third-party platform and your repository and your website will be publicly accessible.
* Without your repository URL, it is not likely that someone will identify it as yours, especially if you do not use your name.
* However, anything you put up on the repository could potentially be viewed by anybody on the web. Please do not put anything personal, confidential, defamatory, or private on GitHub.
* If you have concerns about creating an account on GitHub then contact the course convenor to discuss your options.

## Task 2-2. Create a Repository

1. Log in to your GitHub account.
2. Click the '+' icon in the top right corner of the GitHub homepage and select "New Repository."
3. Fill in the repository name as follows: <username>.github.io. Replace <username> with your GitHub username. This naming convention is required for GitHub Pages to work.
4. Choose the repository visibility to be Public – for a free account this is the only option for using GitHub Pages. Please keep your website clean and free of personal information.
5. Check the box that says "Initialize this repository with a README" to create an initial README file for your repository.
6. Click the "Create repository" button.
7. Try to keep the folder structure of your GitHub repository the same as that of your University space for AI project files. This will make your life much easier when you upload your files. Also, you are likely to include some links in your notebook that might point to other files.

## **Task 2-3: Upload your notebook from Part 1 to your GitHub Repository**

1. Go to your GitHub repository's main page.
2. Click on the "Add file" button and select "Upload files."
3. Drag and drop your website files (your notebook, any linked multimedia files like audio, video, image – as the ones in Part 3 of this lab sheet) into the file upload area. Keep your local file structure by creating corresponding folders on GitHub. If you should have trouble dragging and dropping, you can also click on “chouse your files” in the same upload area. This will allow you to navigate to your directory and upload selected files.
4. Scroll down and click the "Commit changes" button.

## Optional Task 2-4: Setting up Git on your computer

### What is Git?

You may have heard of something called **Git**. Git is **not the same** thing as **GitHub**. Git is a software that your install on your computer to help you:

* Manage the versions of your project files on your computer.
* Link up the version of files on a remote server so that it is in sync with the correct version of your project files on your computer.

Whenever we make changes to files on the computer, we often tend to create another version of the work – a file with a new name. You may have already noticed that sometimes this can become unmanageable (e.g. not knowing which the latest version), especially where you are carrying out a project with many files such as a website. Git keeps track of changes in your entire project so that you can roll back to selected restore points that you have created along the way with “Commit”.

Your files on GitHub are tracked and records versions for each time you "commit changes" (Step 4 of Task 4). You can revert your projects to these versions if desired. However,

* This is not synched to the changes you make on your own local computer or storage.
* And versions on your computer are not tracked, leaving you to manually keep track.

To resolve these challenges, [Git](https://git-scm.com/) can come in handy. The Glasgow Anywhere desktops should have Git installed. Otherwise you can install it on your own computer by following the link in the last sentence.

### How it Works

The principle is simple:

1. You designate a folder as your local “repository”.
2. You define which files you want Git to track – usually all the files in your “repository”.
3. You work on your local computer, executing commit for versions you want remembered.
4. You push it to the remote “repository” to keep it synched.

### To Get Started

If you want to explore the option of using Git for your project, there is an [excellent tutorial on Youtube](https://youtube.com/playlist?list=PL4cUxeGkcC9goXbgTDQ0n_4TBzOO0ocPR&feature=shared) which I highly recommend.

**The use of Git is not a requirement of AI for the Arts and Humanities (A).**

# Part 3: Exploring Multimedia and Scikit-Learn Datasets

In this part of your lab book, you will explore multimedia using a number of python libraries. Last week we used the library **IPython.display** to display a YouTube video in your Jupyter notebook. We will use the same library to display images and sounds as multimedia. If you have not done the exercise from Week 1, do that first!

We will also work with the python library **matplotlib** – one of the most popular data visualization tools used with python. We will browse datasets available within the machine learning library **scikit-Learn** (a.k.a. **sklearn** – this is what it is called when you import the library from Python).

## Task 3.1: Using IPython.display to display images and audio

**Step 1.** Download the files **picture1.jpg**, **audio1.mid** and **audio2.ogg** linked on the Moodle [Lab Resources](https://moodle.gla.ac.uk/course/view.php?id=39566#section-3) section. Make sure you save it in the folder you created last week for the course work.

**Step 2.** Open Jupyter Notebook like you did last week (first open Anaconda then launch Jupyter Notebook from the Anaconda navigator). From the Jupyter Notebook dashboard, navigate to the folder you created last week. Assuming you downloaded the file to the correct area in Step 1, you should see the files you downloaded listed in the folder.

**Step 3.** Open the Python Notebook you were working on last week (this should also be in the folder you created last week!) and create a new markdown cell. For example, name it in a way that it is clear that the section is for Week 2 exercises (e.g. “Week 2. Exploring Data in Multiple Ways”). You should know how to do this if you did the exercises in Week 1.

**Step 4.** In Week 1, we imported everything from the library **IPython.display**. But only used the function YouTubeVideo from the library – this time import only the function **Image** from the library by replacing the asterisk with the function name. This would look something like:

from IPython.display import Image

Remember: **this command is case sensitive**. Click run and then run the following as well:

Image (“picture1.jpg”)

The quotation marks are used because file names are data type string (review Week 1 exercise). What do you see?

**Step 5.** Now import the function Audio and use it to run similar commands, one each for files **audio1.mid** and **audio2.ogg**. What do you see?

If you did it correctly, you will see something like:

A white rectangular object with a black line

Description automatically generated

Press the play button. Do they both play? If not, which plays, and why do you think this is the case?

**Step 6.** Write a short text (using markdown cells) to reflect on the results.

The sound file audio2.ogg is owned by Artoffuge Mehmet Okonsar. Remember to add the attribution and include a comment in the cell (in a cell of type code, you do this by prefixing text with a hash tag) to flag the following licensing information about **audio2.ogg:**

This file is licensed under the [Creative Commons](https://en.wikipedia.org/wiki/en:Creative_Commons) [Attribution-Share Alike 3.0 Unported](https://creativecommons.org/licenses/by-sa/3.0/deed.en) license.

You are free:

* **to share** – to copy, distribute and transmit the work
* **to remix** – to adapt the work

Under the following conditions:

* **attribution** – You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
* **share alike** – If you remix, transform, or build upon the material, you must distribute your contributions under the [same or compatible license](https://creativecommons.org/share-your-work/licensing-considerations/compatible-licenses) as the original.

The original ogg file was found at the url:

<https://en.wikipedia.org/wiki/File:GoldbergVariations_MehmetOkonsar-1of3_Var1to10.ogg>

## Task 3.2: Using the **matplotlib** library to look at picture as numerical data

You might like to create a subsection for this exercise in your Python notebook using a markdown cell. As mentioned above, the library matplotlib is a very popular tool for visualizing and working with data. It could help to enhance the presentation of your code. The pyplot function is what we want from matplotlib.

**Step 1**. So go ahead, you know how to do this. Import pyplot from the library matplotlib. If you are lost, talk to your peer group members to get hints.

**Step 2.** Last week you used the variable **message** to save the string “Hello World!” and then print it out on the screen. We will use variables **test\_picture** to save the content of the files **picture1.jpg** andview the content both as an image and as the data underlying the image. To do this, type the commands in the following code (included here as an image) into a Notebook cell and run it.

from matplotlib import pyplot
test_picture = pyplot.imread("picture1.jpg")
print("Numpy array of the image is: ", test_picture)
pyplot.imshow(test_picture)

The code assigns the file content to the variable **test\_picture** – the content is read in by the function **pyplot.imread**. The **print** function (recall from Week 1) is used to display the numerical data on screen. And, **pyplot.imshow** is used to present the actual picture. The numerical data will be presented as a data structure known as an array. More specifically, in this case, it is a **numpy** array – **numpy** is another library in python for manipulating numerical data.

**Step 3.** You can directly manipulate values in the array to see how that affects the image. For example, you can assign **test\_picture\_filtered = 2\*test\_picture/3**. This doubles the values in the array and then divides the values by three. Now run the command **pyplot.imshow(test\_picture\_filtered).** What do you see displayed? What is going on here? Discuss your thoughts with your class mates and note your thought and discussions down in your notebook.

## Task 3-3: Exploring scikit-learn (a.k.a sklearn)

Again, you might like to create a markdown cell to create a separate section for the exercises in this section.

In this section, we will be using the library **scikit-learn**. We will try out machine learning models with some small datasets a little later in this course. At this point, the exercises are just meant to get you started with exploring the tools therein.

**Step 1-1.** So go ahead **–** import the package **datasets** from the library **sklearn**. You should be gettingused to importing libraries by now. 😊

**Step 1-2.** You can list what is in the library in number of ways but one way you can do this is by typing **dir(datasets)**. This command works for other libraries as well. Run this command now to browse the list of datasets accessible through **sklearn**. There are two types: ones that you fetch remotely from somewhere else, and those that you load from the **sklearn** library itself. We are going to explore the latter. Look through the names starting with the keyword **load.** Choose a couple of datasets and note down their names to explore further.

**Step 1-3.** In a markdown cell explain why you chose these datasets!

**Step2.** Load the dataset and assign it to a variable (you should know what this means by now!). Remember to give it a meaningful name. In the example below we have loaded the dataset for wine to a variable named **wine\_data** (change the variable name to be appropriate for the datasets you have chosen in **Step 1-2)**. Note that you need the empty round brackets at the end when importing datasets.

**wine_data = datasets.load_wine()**

**Step 3-1**. Once you have loaded a couple of datasets, explore the description of the dataset by running the command:

wine\_data.DESCR

Remember to replace **wine\_data** here with whatever variable you used for your chosen dataset! You may have noticed that the description is rather hard to read because it is unformatted. By using the **print** command, you can prettify the output:

print(wine\_data.DESCR)

This command’s output starts out with something like the following:

.. \_wine\_dataset:

Wine recognition dataset

------------------------

\*\*Data Set Characteristics:\*\*

:Number of Instances: 178

:Number of Attributes: 13 numeric, predictive attributes and the class

:Attribute Information:

- Alcohol

- Malic acid

- Ash

- Alcalinity of ash

- Magnesium

- Total phenols

- Flavanoids

- Nonflavanoid phenols

- Proanthocyanins

- Color intensity

- Hue

- OD280/OD315 of diluted wines

- Proline

This immediately tells you a few things about the data – how big it is and how many attributes each item has.

**Step 3-2.** You can also explore the attributes of the wine recorded in the dataset by running the command **wine\_data.feature\_names**. How many features does your dataset have?

**Step 3-3.** Run the command **wine\_data.target\_names**. What is this dataset aiming to learn? Can you tell? Do this exercise for a number of datasets to get used to **sklearn.datasets**.

Where do “feature\_names”, “target\_names” etc are coming from? These are known as keys. You can run the command wine\_data.keys() to know what keys are available to you.

## Task 3-4: Basic Data Exploration with Python library Pandas

Exploring datasets directly with **sklearn** can be a bit limiting and not very user friendly. It would be nice to see the data in a well laid out tabulated form. In this section we will help you do this by using the library **pandas**. The pandas library is one of the most popular methods for carrying out data analysis and manipulation.

**Step 1.** Import the **pandas** library. Now simply run the following commands.

from sklearn import datasets
import pandas

wine_data = datasets.load_wine()

wine_dataframe = pandas.DataFrame(data=wine_data['data'], columns = wine_data['feature_names'])

**What is going on here?** - this code gets the **wine\_data** from **sklearn** as before then converts it into something called a pandas dataframe and saves it in the variable **wine\_dataframe**. To explore this dataframe further, run the commands **wine\_dataframe.head()** and **wine\_dataframe.describe().** Examine the results. What do you think these commands do? Feel free to discuss with others in your group and jot it down in your notebook using markdown.

We will use more of the library pandas later in the course.

## Task 3-5: Thinking about data bias (home work)

Read the article by [Prabhakar Krishnamurthy](https://towardsdatascience.com/survey-d4f168791e57)  and listen to the [podcast interview with Melanie Mitchell](https://www.human-current.com/episode-077-exploring-artificial-intelligence-with-melanie-mitchell) (starts 2 minutes and 30 seconds or so into the recording) on social concerns regarding AI, including issues of data bias. How would you explore your datasets to assess bias? What would you look for? Jot your thoughts down in a markdown cell in your notebook ready for discussion. Arrange a meeting with your peer group to consolidate your thoughts. What is agreed and what opinions differ? Your group allocation can be found on the Moodle participants list for the course.

## Summary

In this lab we look at number of useful libraries. WE revisited the **IPython.display** library to see how to display images and audio. We saw how to use matplotlib to access the numerical data underlying an image to manipulate the image. We looked at some datasets in **scikit-learn** (a.k.a **sklearn**) and learned hoe to access them using pandas for a more user friendly presentation.

The exercise demonstrates how pictures and audio relate to data that computers can understand. The exercise introduces you to some key tools in data science and machine learning (e.g. **matplotlib** and **sklearn**) and highlights that there are number of ways to look at the same data.

In the coming weeks we will expand on what have learned further to hone our skills in navigating libraries and datasets in the context of machine learning algorithms.

## Before you wrap it up

Once you have completed this exercise, if you have time, review the video on **Jupyter Notebook** on Moodle. This video goes over some of the things we have learned in the first two weeks, but also adds a few extra tips that could help you make sense of **Jupyter Notebook**.