

**PIXEL PROJECT**

**Prepared for**

StJohn Russel

VP Engineering, TooMuchWifi

**By**

Mpetha S. Motaung

Software Engineer, TooMuchWifi

14.12.2024

# Abstract

# Introduction

Before you get started here are the prerequisites:

* Understand virtual environments
* Understand package managers
* Understand cmd commands
* Understand Github Desktop, Github and Gitversion control

## Objectives (Initial Assumptions)

* Create Django project that can handle image uploads.
* The uploaded images have to be processed to extract center most pixel and hex value.
* Both uploaded image and hex value pair have to be viewable by the user (As a gallery of photos and their hex values).
* The uploaded images and their hex values must be displayed using Django’s template system.
* The web application must be able to handle errors for non-image file uploads and image processing fail cases.
* Minimum memory usage when extracting center pixels without loading the entire image (Save the url of the image as text in the database.)

## Requirements

## Architecture

### Overview

### Design Decisions

### APIs

endpoints, methods, parameters, and responses

### Workflows

# Setup & Project Configuration

Download and install Anaconda for package management and virtual environment. (used for sourcing, building, and deploying data science and AI initiatives)

\*You may use Anaconda or any other virtual environment platform (e.g. virtual env, pyenv, pipenv) but I prefer Anaconda\*

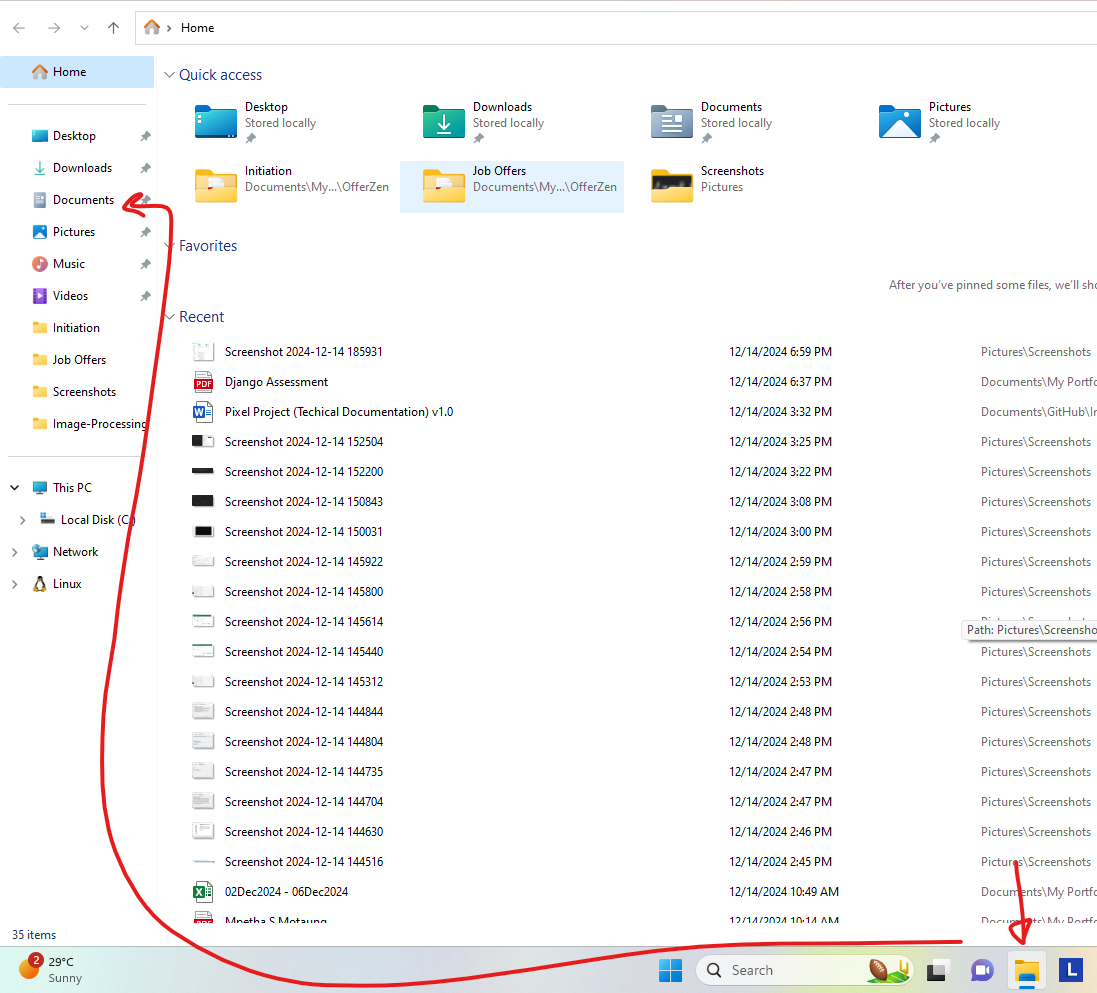
After Downloading <https://www.anaconda.com/download/success> and installing Anaconda. There are 2 methods of creating a virtual env

### Creating or choosing Project location

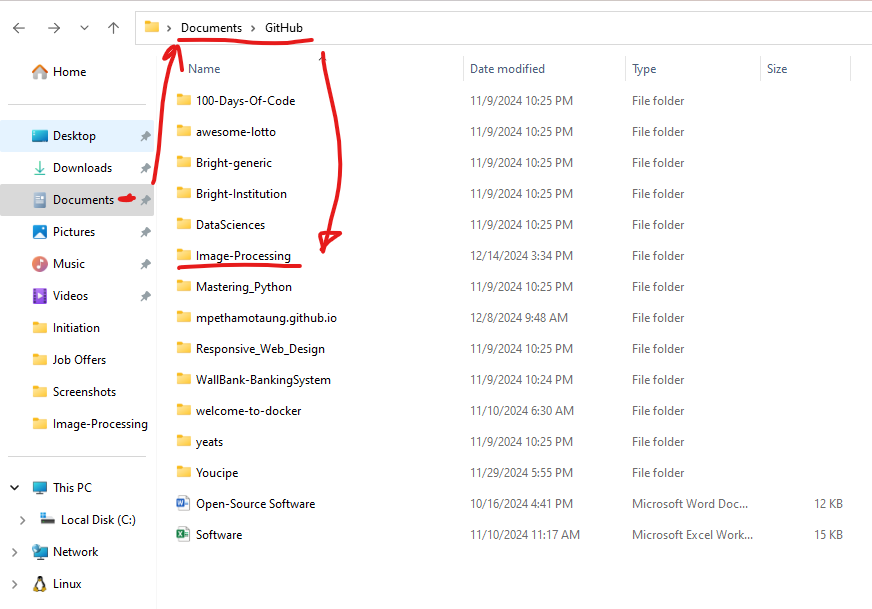
You can either create a folder using the Windows Graphic User Interface (GUI) or Command Line Interface (CLI). We must create a folder anywhere in file explorer but I suggest using Document/Github/(project name) to ensure that all your projects are organized. [[1](#_References)]

#### Method 1 (Windows GUI)

1. Open File Explorer (double left-click)



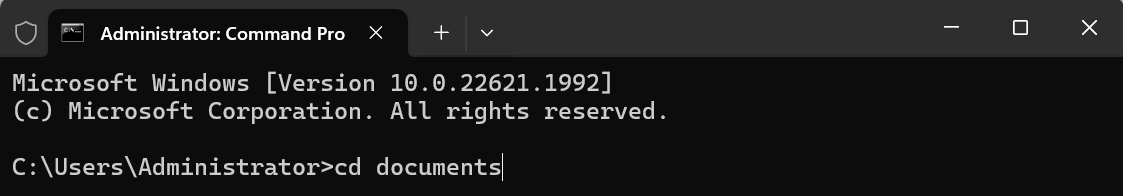
1. Navigate to documents, create a Github folder(if it doesn’t already exist). Within Github folder create project folder ‘Image Processing’ (Documents/Github/Image-Processing)



This is the folder that we will be executing the Django project from

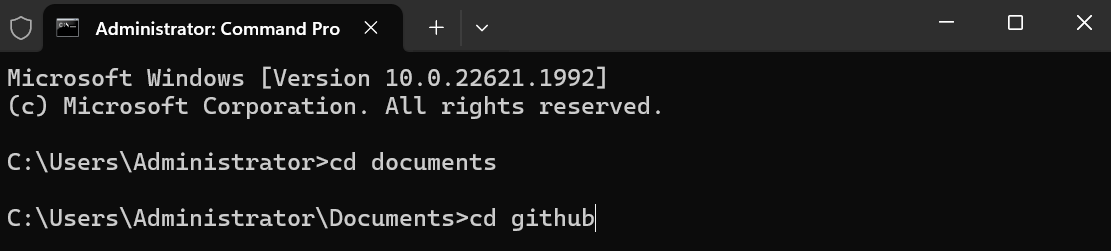
#### Method 2 (CMD)

1. Open CMD, then navigate to Documents/Github. Create the Github folder if it doesn’t already exist.



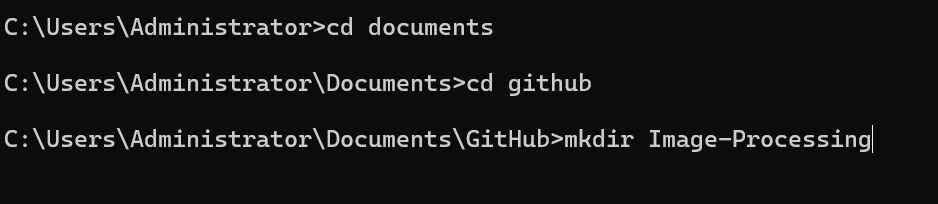
To navigate into documents cd documents, then hit enter

1. Navigate into github folder



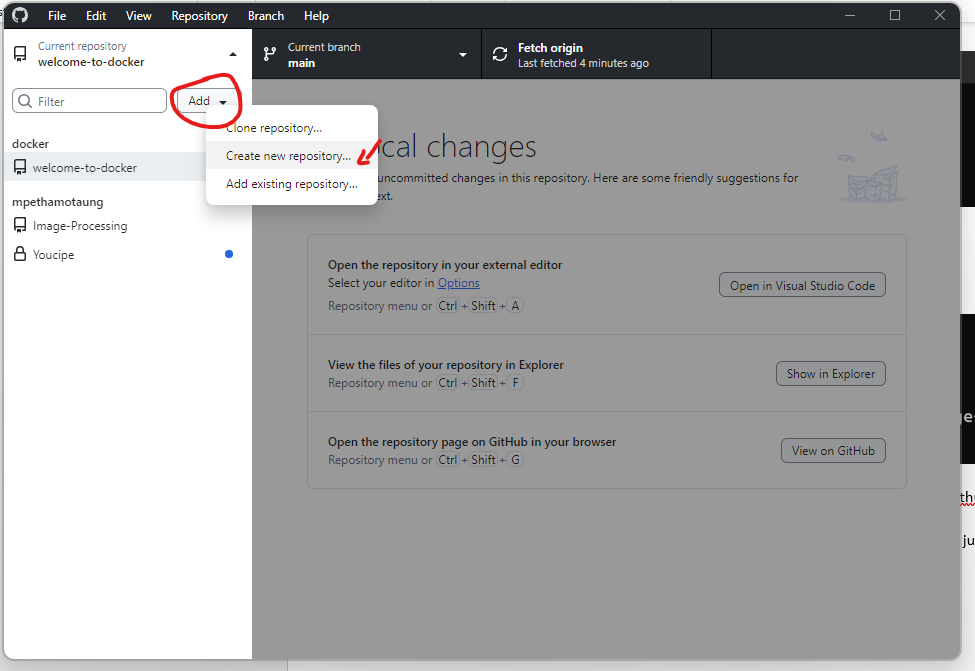
To navigate into Github folder in documents cd Github, then hit enter

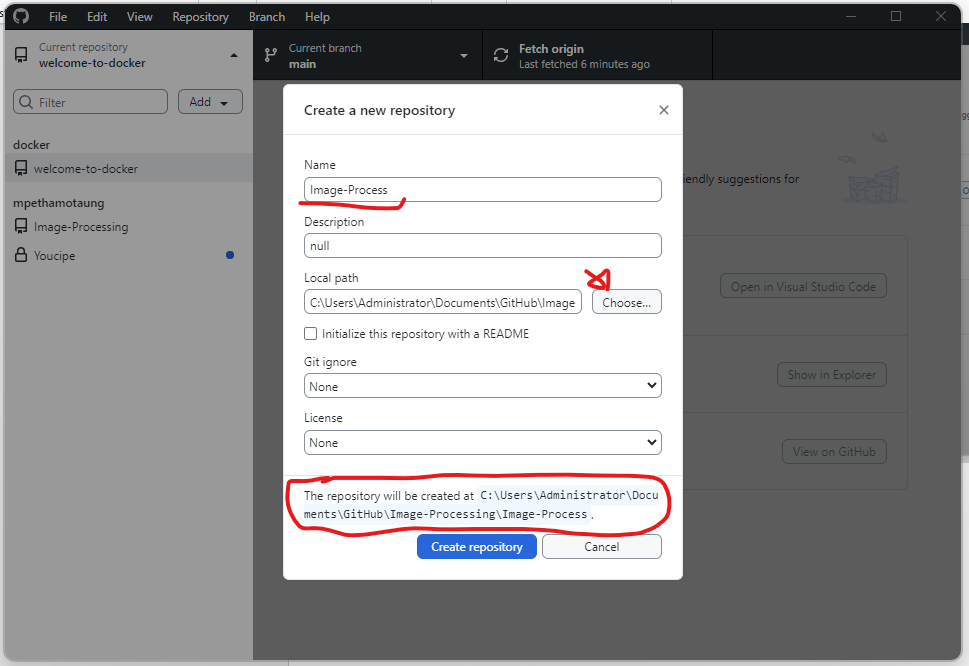
1. Create project folder for our Django project



This will create a folder for our Django project but it is still not connected to Github or any repository on Github

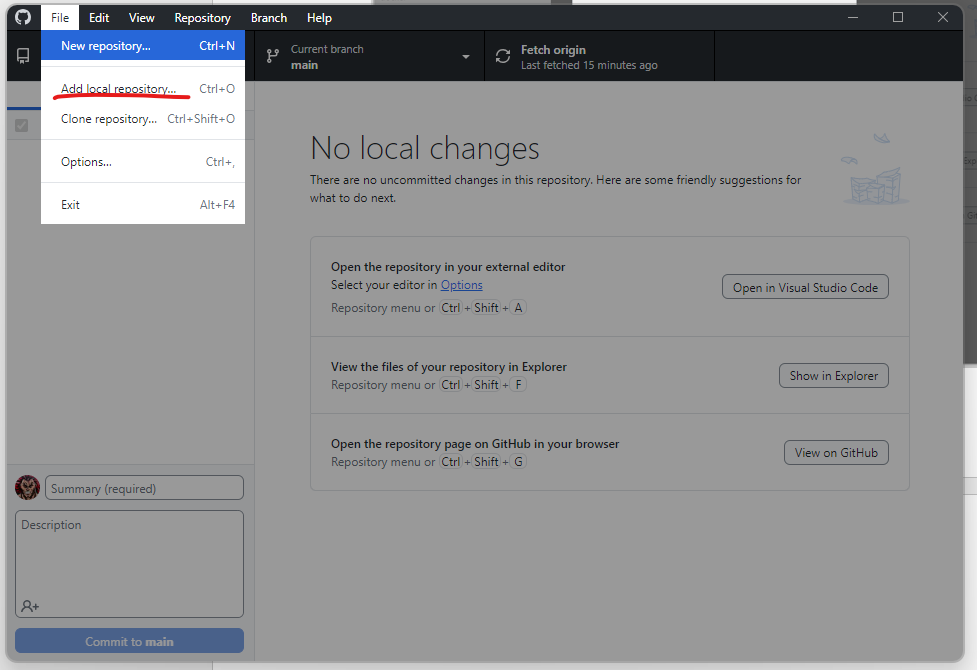
1. Next open Github desktop, create a new repository and link it to the folder we just created



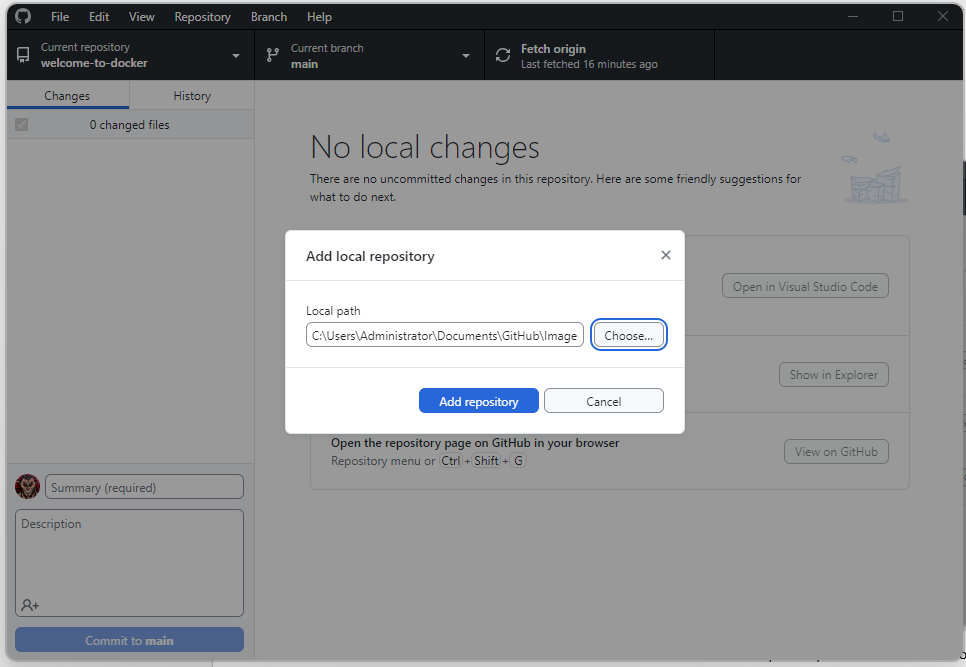


OR

Open Github Desktop, Go to File> Add local repository



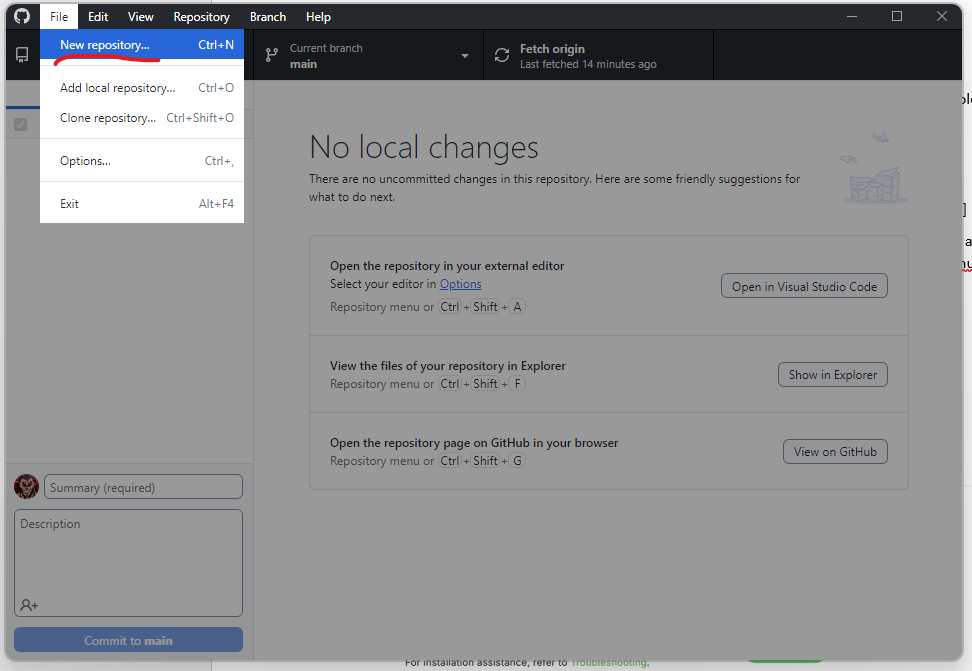
Point Github to the local repository/folder that you want to add to Github Desktop. Then publish to Github.



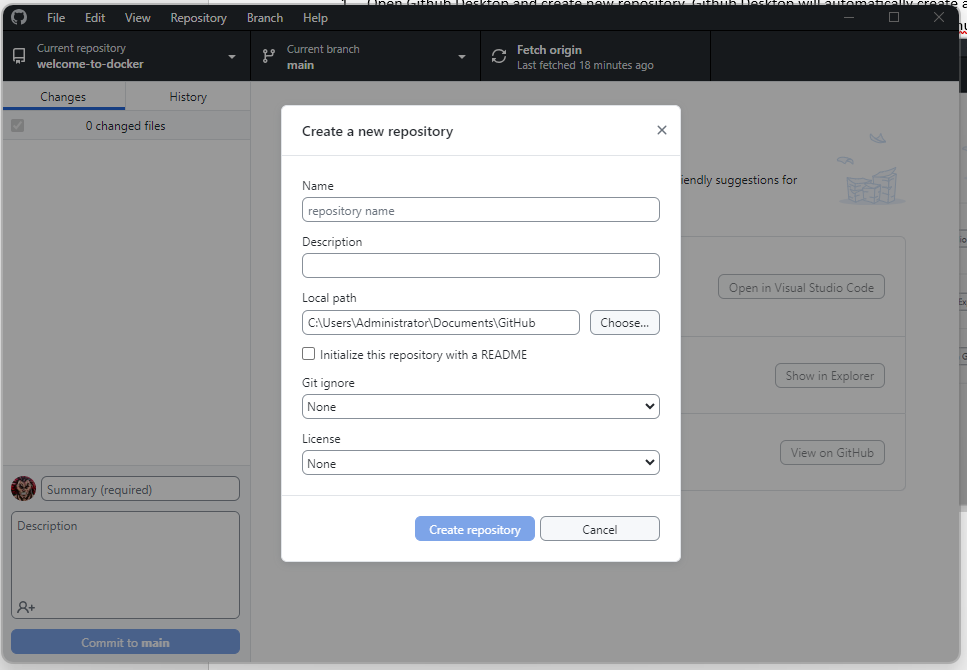
#### Alternatively (To save time)

\*Note\* You must have Github Desktop installed for you to proceed with this alternative method [[2](#_References)]

1. Open Github Desktop and create new repository. Github Desktop will automatically create a project folder on your local machine that will be used through and IDE to interact with Github.



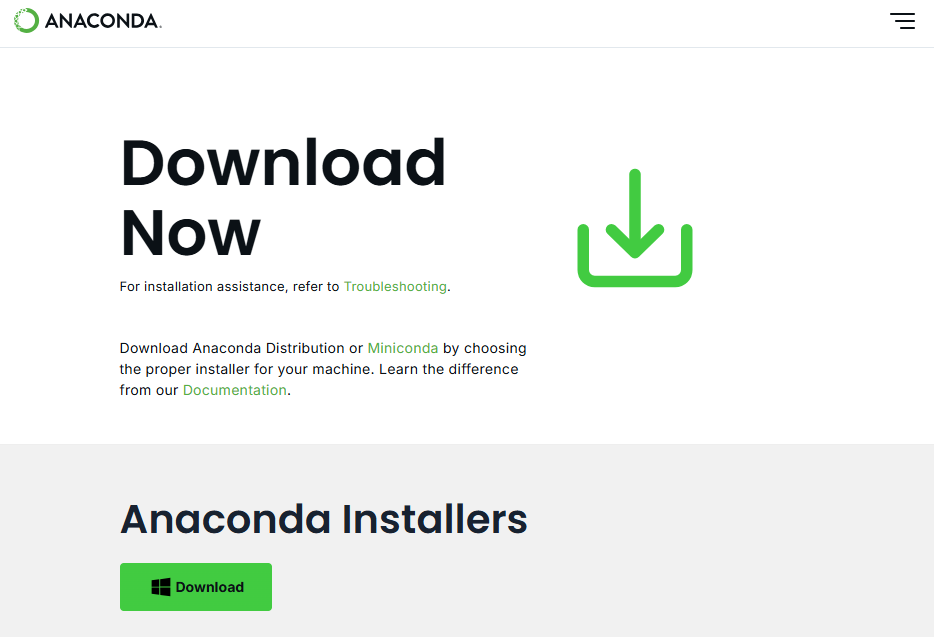
1. Give your repository a name, this will also automatically be the name of the folder that Github will create in your chosen path. Read up on Github Desktop, how to create and publish a repository [[3](#_References)]



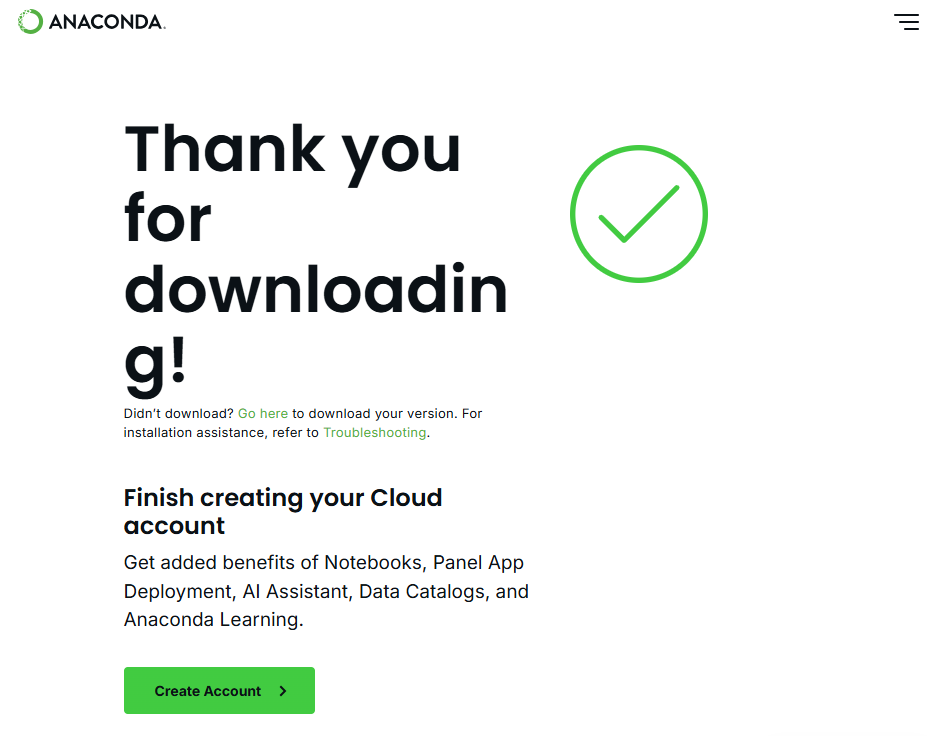
Now that our folder has been created and we have created a new repository and linked it to our folder we can move to the next step

### Downloading and Installing Anaconda

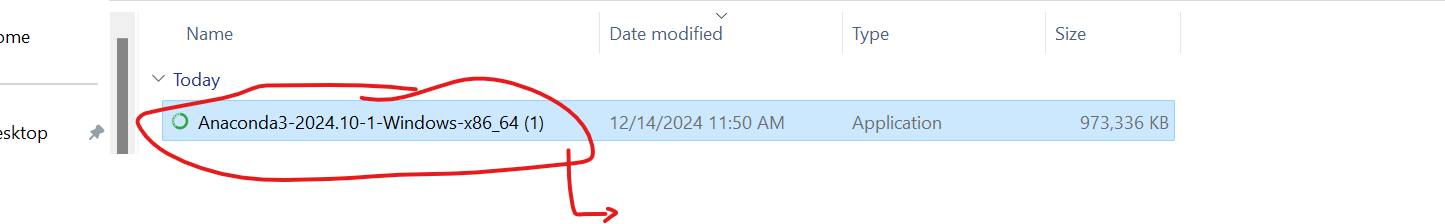
1. Click on the Download button

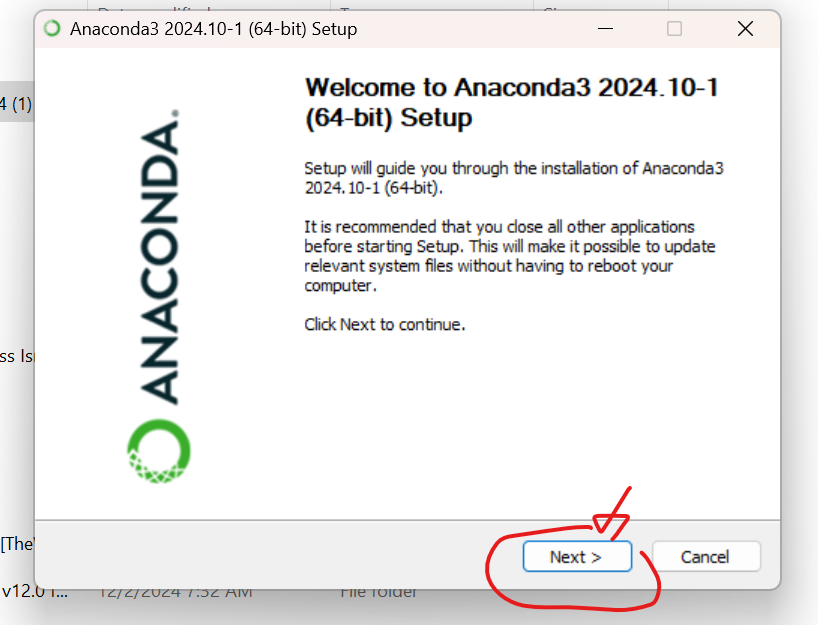


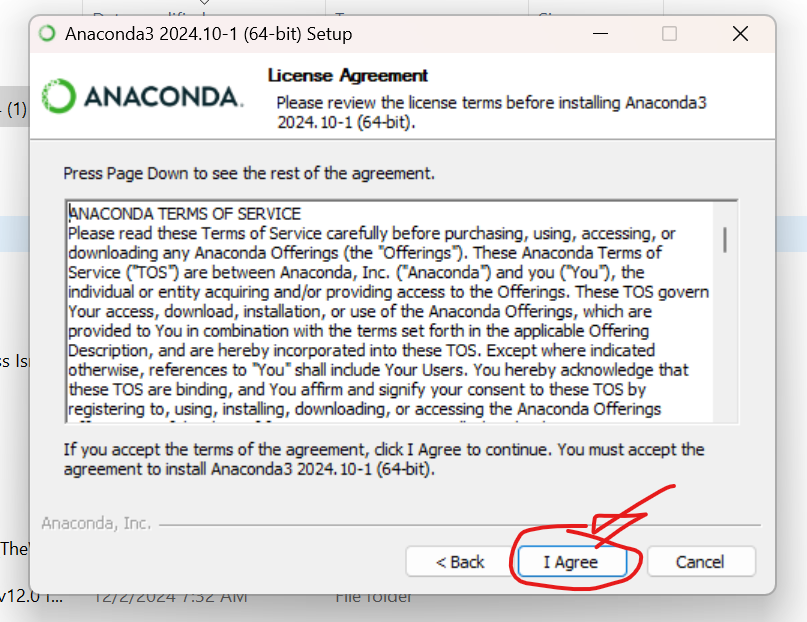
1. You’ll be redirected to this page and your download should start

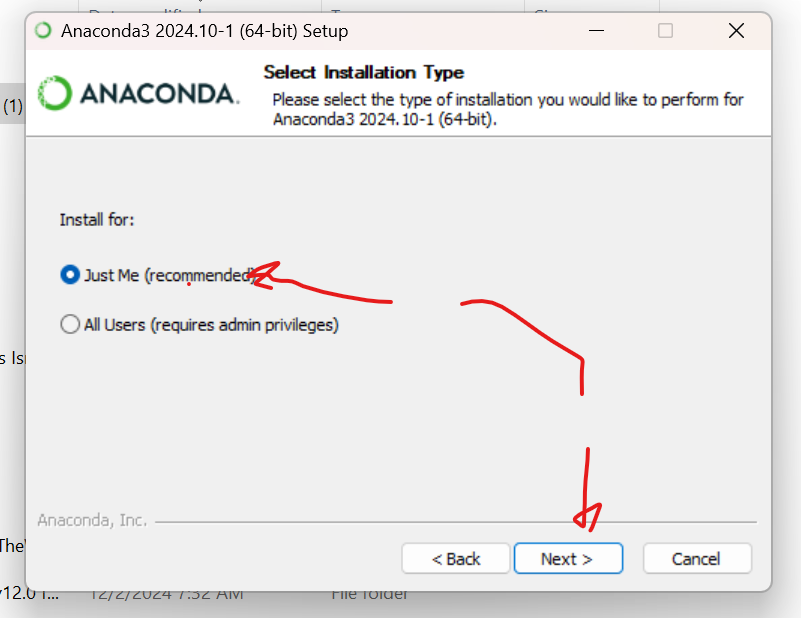


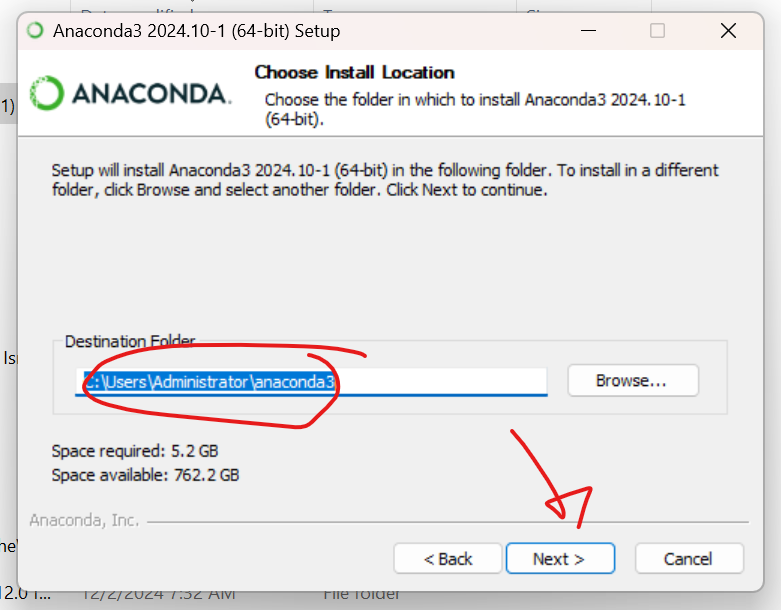
1. Once the file has finished downloading (/downloads), double click on the file and install Anaconda

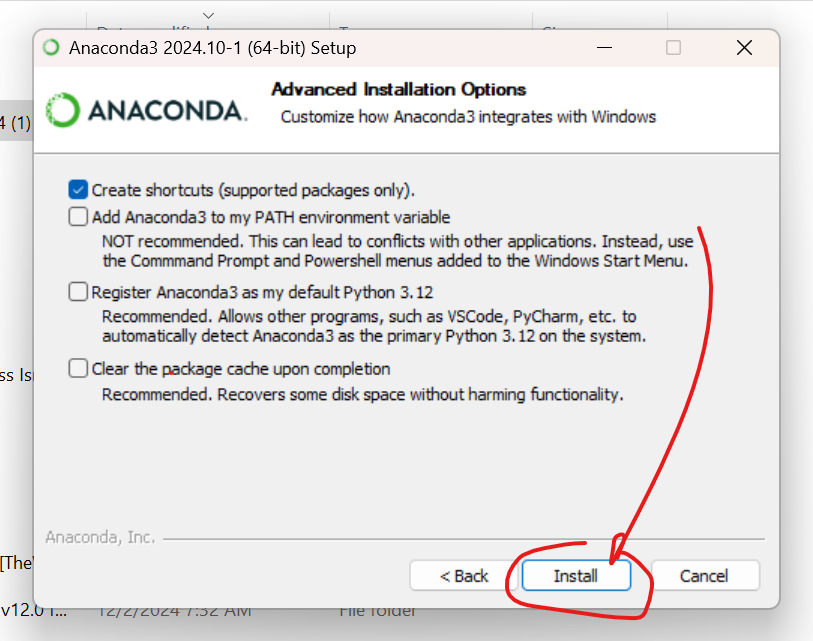






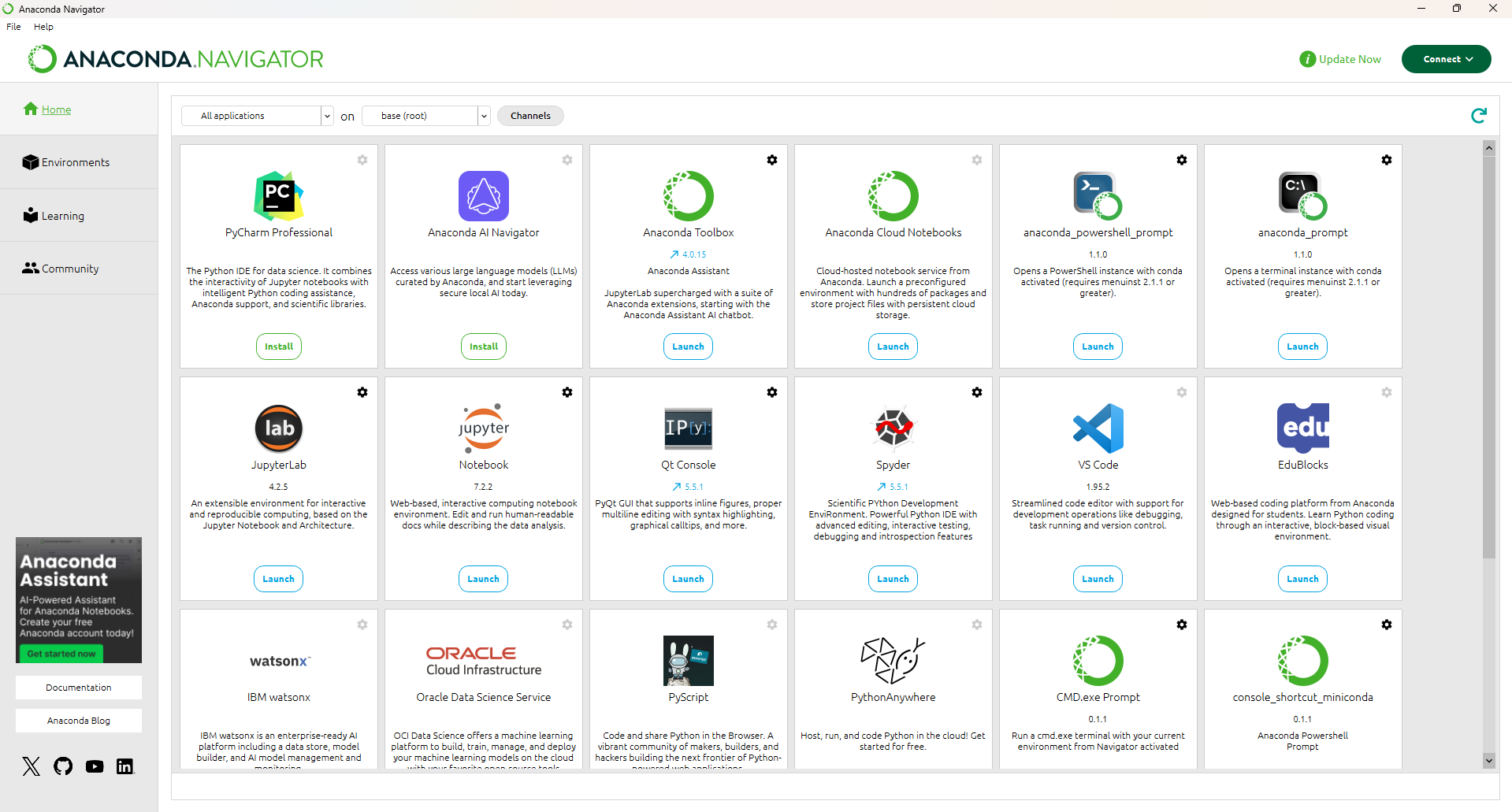






1. Now that you have downloaded and installed Anaconda. Let’s launch it for the first time

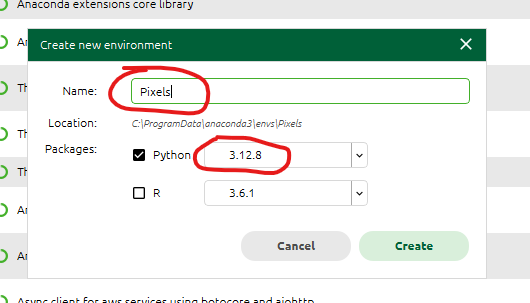
\*If you have never used Anaconda, I suggest you watch a few short tutorials (YouTube or on the Anaconda website). It’s quite user friendly\*



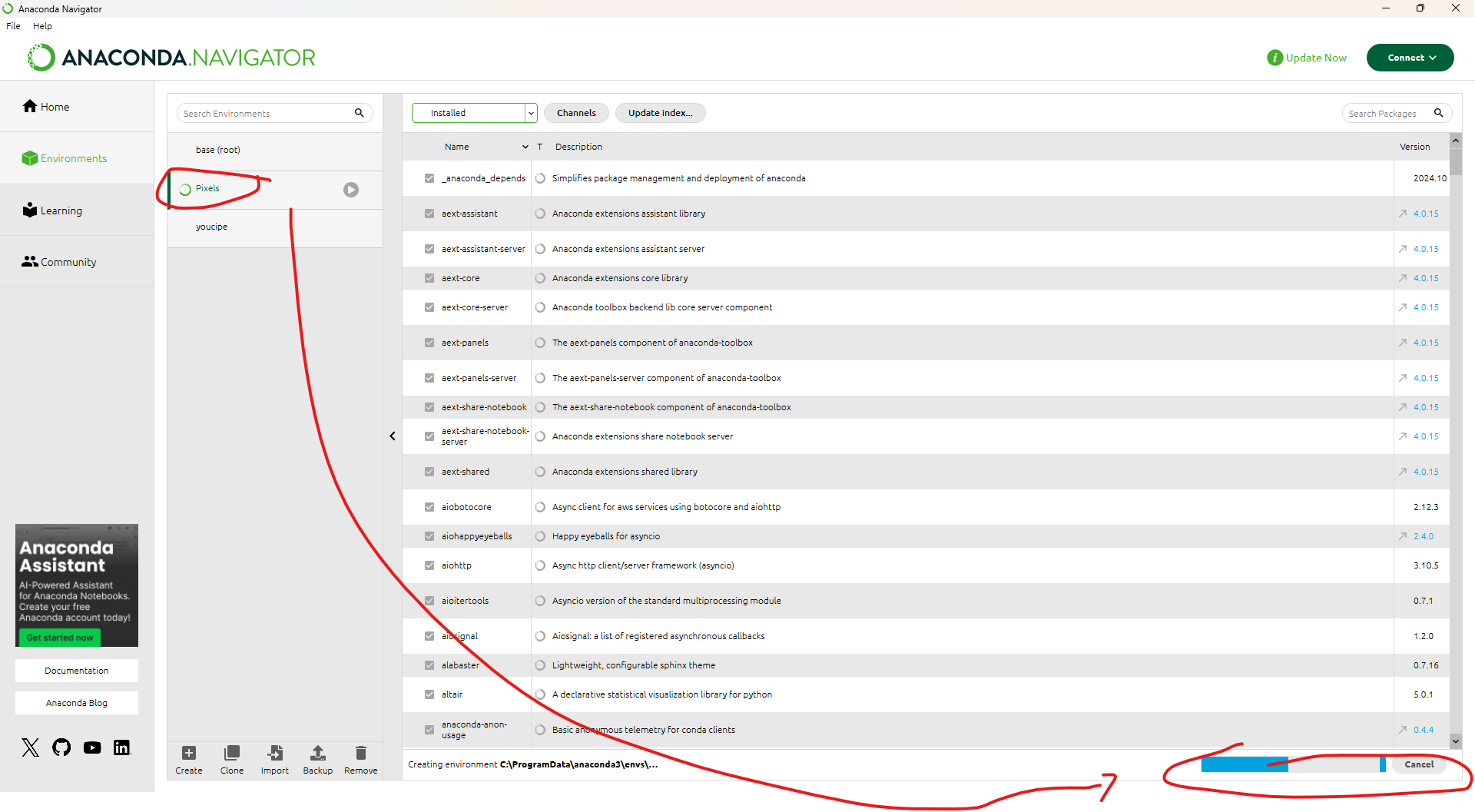
### Creating a virtual environment - Method 1 (GUI)

1. Open Anaconda
2. On the sidebar on the left click>environments
3. Then click create

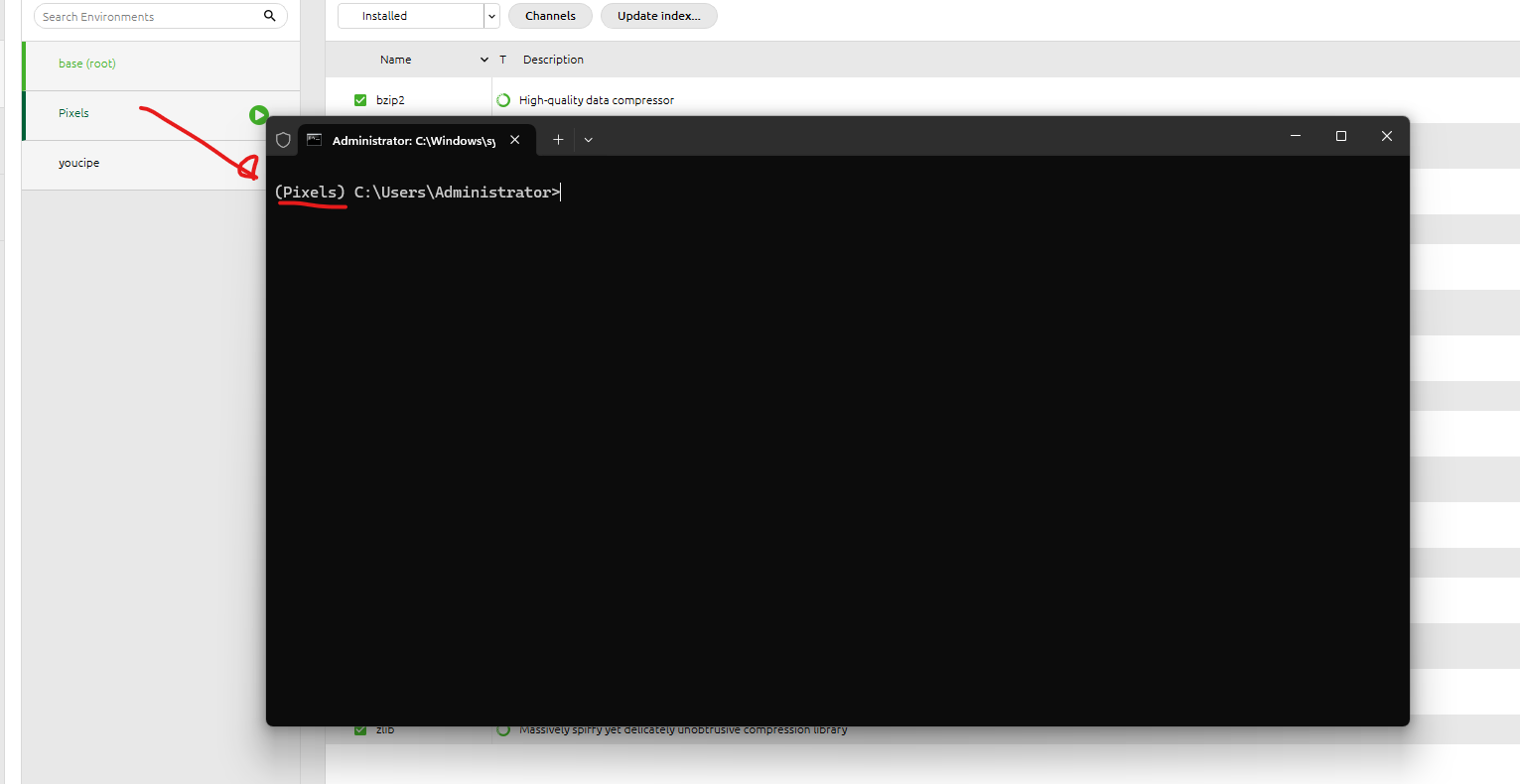
* Name your environment according to the project that you’re working on so that you can remember it. In this case I will name this one ‘Pixels’



1. Anaconda will now create a virtual environment with the python version and the name that you chose. It should take 2-5 minutes

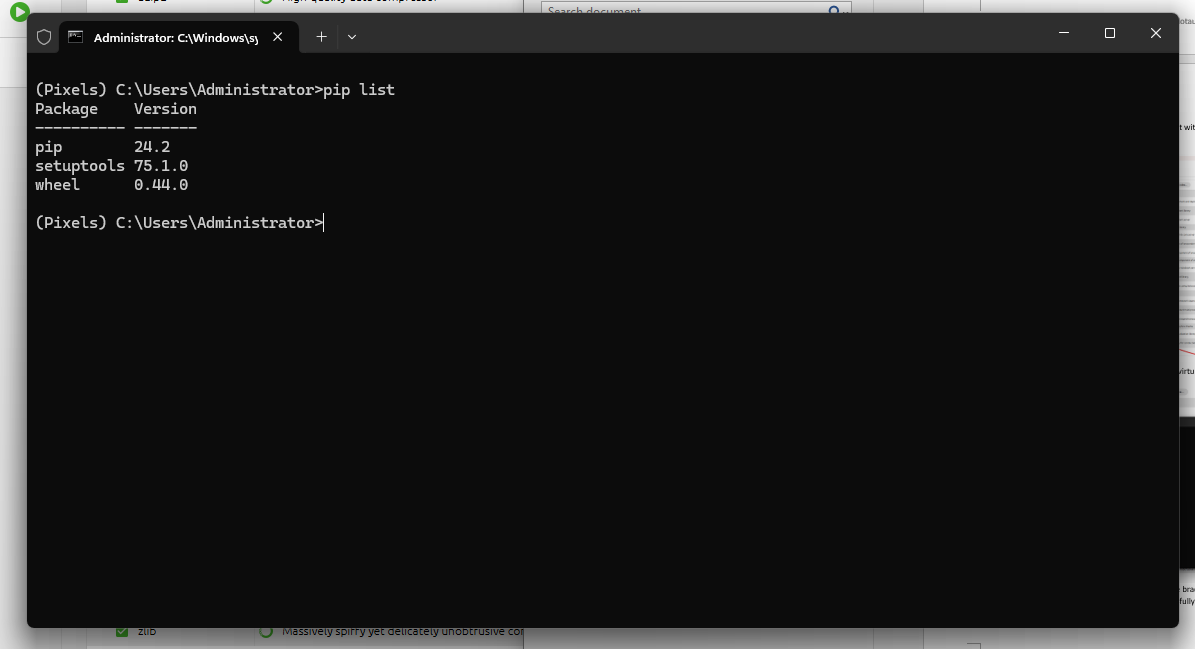


1. Next you can hit the play button to check your virtual environment. If it is created successfully, you should see the following:



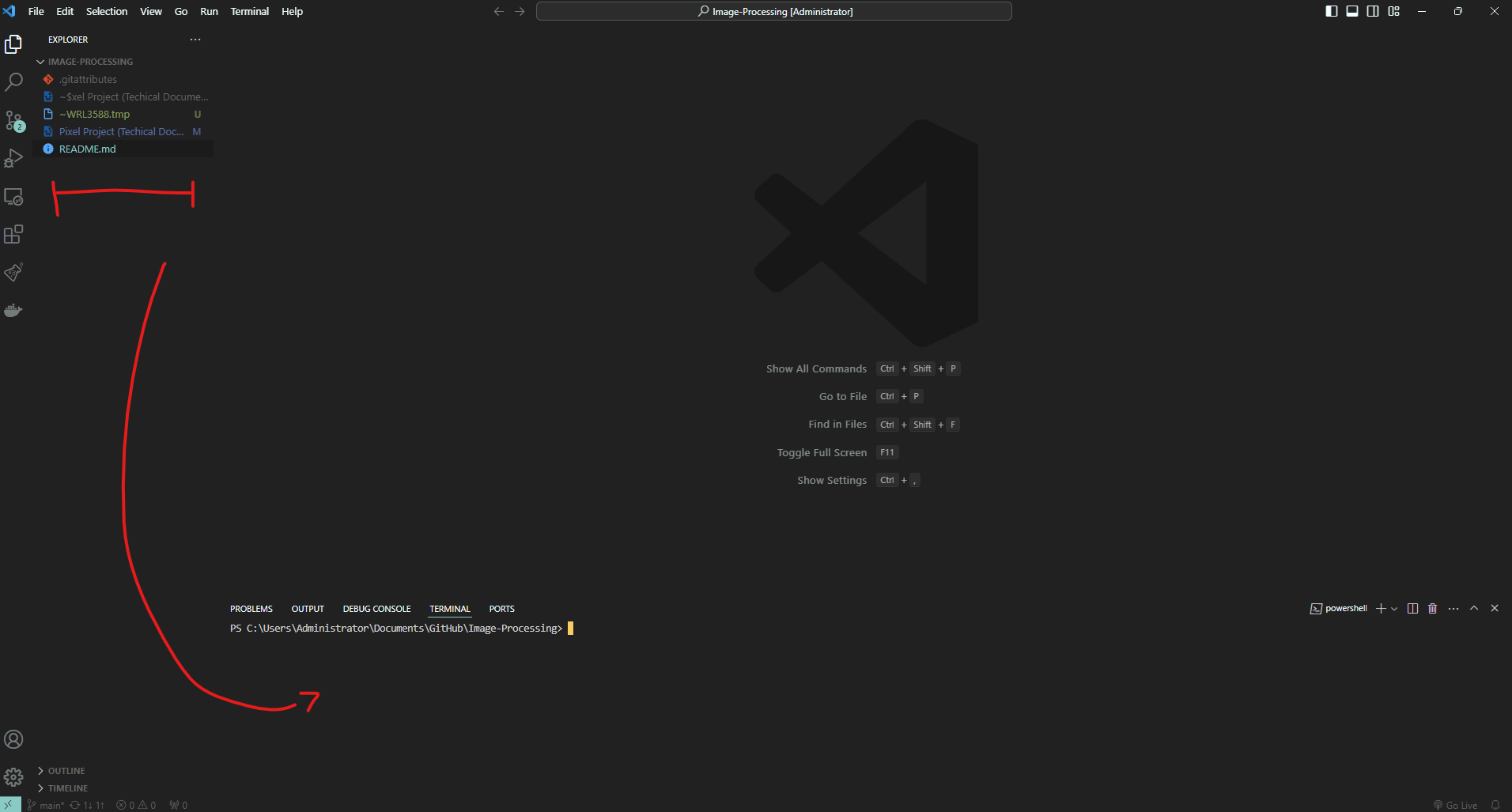
The name of our virtual environment inside the brackets confirm that our virtual environment was created successfully and is running successfully.

1. Type pip list in the terminal to see all the installed packages in the Pixels virtual environment. Since we do not have any additionally installed. We will only see the default packages that are built in(default) with every environment creation using pip

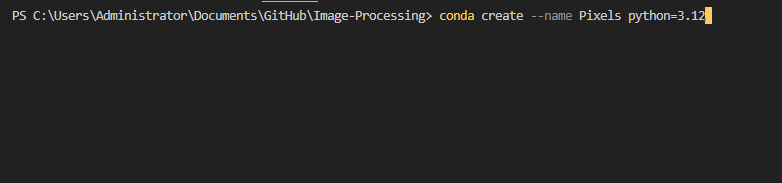


### Creating a virtual environment - Method 2 (CMD)

1. You can follow along with CommandPromt(CMD) or open the terminal window in an IDE of your choice. In this case, I’m going to use my preferred IDE, which is VSCODE

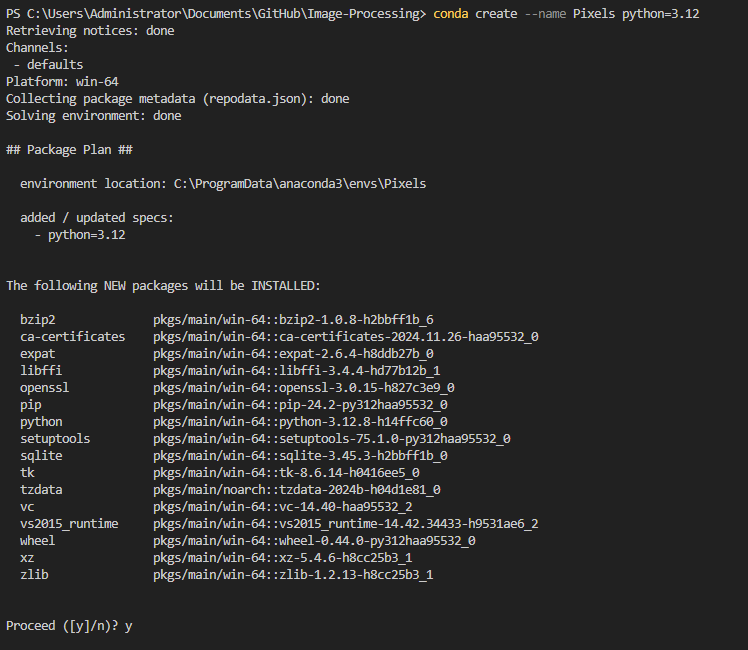


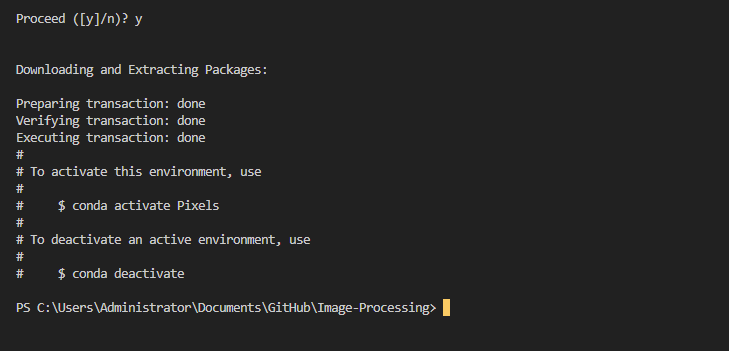
1. Navigate to the terminal and type the following command conda create –name (name of your environment) python=3.12



You can choose any name for your environment but to keep things simple and aligned, go with Pixels. Additionally, you can choose any python version to install on your virtual environment (i.e, python=3.8, python=2.7). Knowing this will be vital so that you can test your software in different versions of python to ensure consistency and to test for issues.

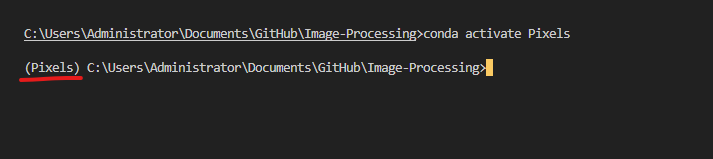
1. Conda will now create a virtual environment





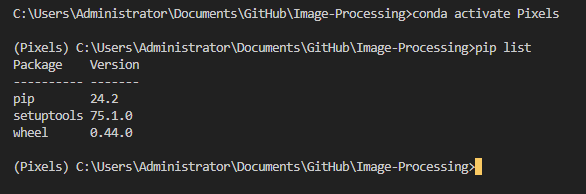
Choose ‘y’ to proceed to install the suggested packages in your environment.

1. Now that we have created our virtual environment let’s try to activate it and test if it has been successfully created. Run the following command conda activate Pixels



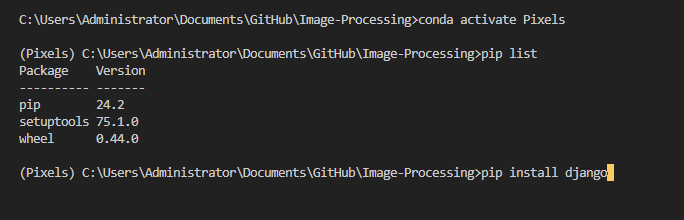
If you followed all the steps correctly, you should see (Pixels) which is showing that your virtual environment has been created successfully.

1. When you run the command pip list, you should see the default installed packages when a virtual env is initiated through Anaconda



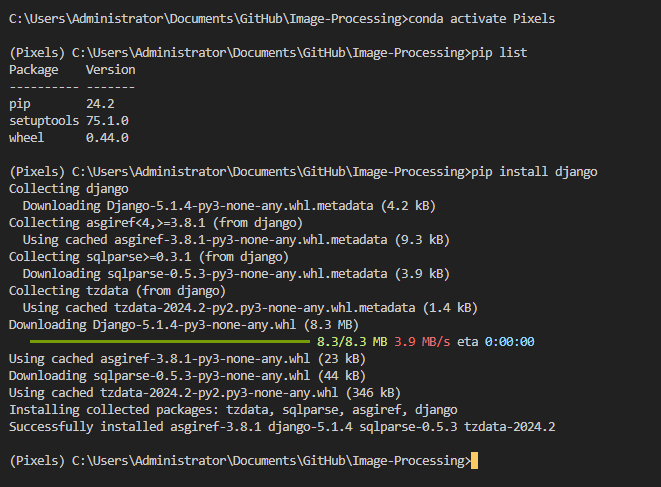
### Install Django

Now let’s install Django into our virtual environment. Run the following command pip install Django (\*note\*: if you do not specify the Django version, pip will install the latest Django version)



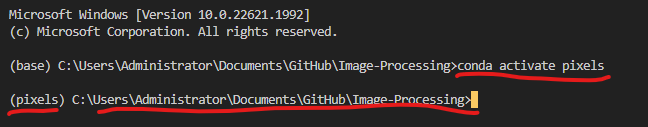
\*Ensure that when working on your project, that your virtual environment is active before running any commands or trying to install packages. Remember we do not want to modify our system packages thus we created a virtual environment. \*

If the above command runs successfully, you should see the following



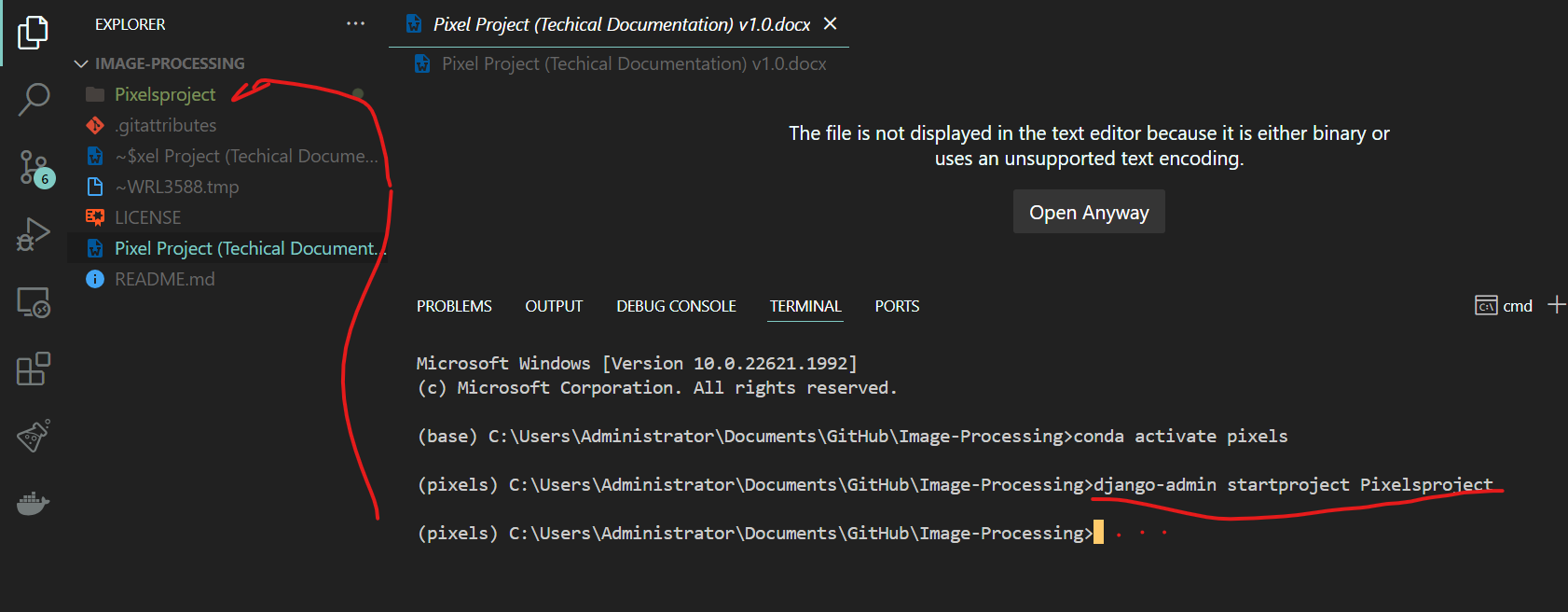
### Create Django Project

Now before we dive into it, let’s make sure that you’re in the correct directory. In my case I’m in Visual Studio code CMD terminal



#### Create Django project [[4](#_References)]

Run command django-admin startproject (myprojectname)



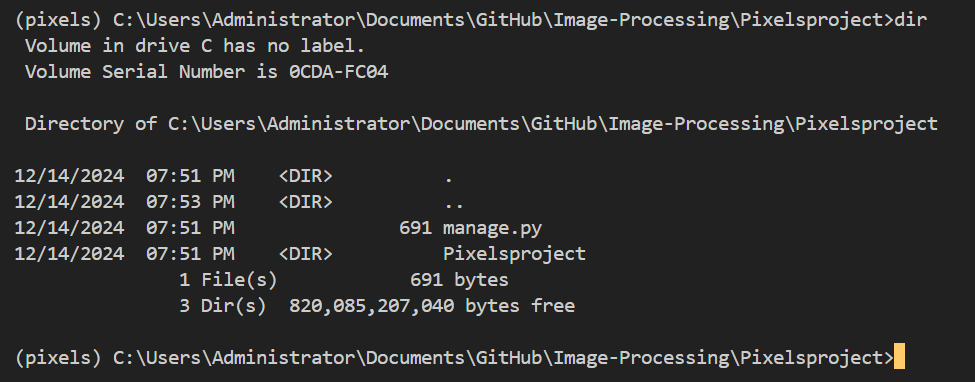
After you run the command a project folder will be created with the name that you selected when you ran the command django-admin startproject (myprojectname)

#### Now let’s create an app to handle the processing of our images. Navigate into the Pixelsproject



#### In the pixelsproject path, run the following command to create Django app

\*Before you run the following command, check if you are in the correct directory by running this command first dir

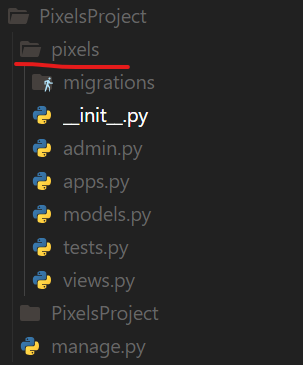


This is our main Django project folder with the manage.py and Pixelsproject folder

#### After ensuring that you’re in the correct path. Run the following command python manage.py startapp pixels to create our pixels app [[5](#_References)]



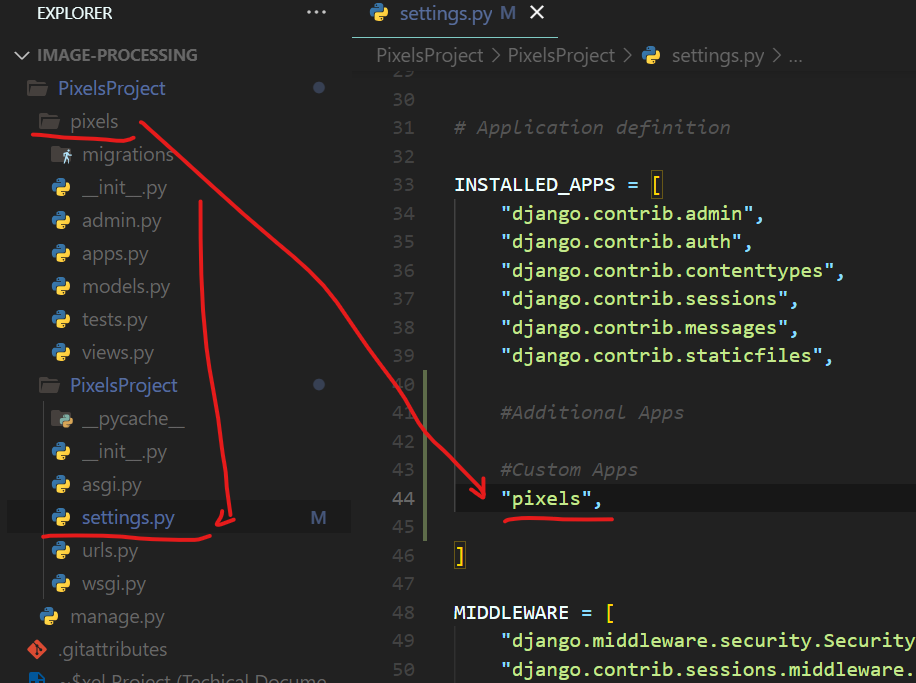
You should now see a folder named pixels



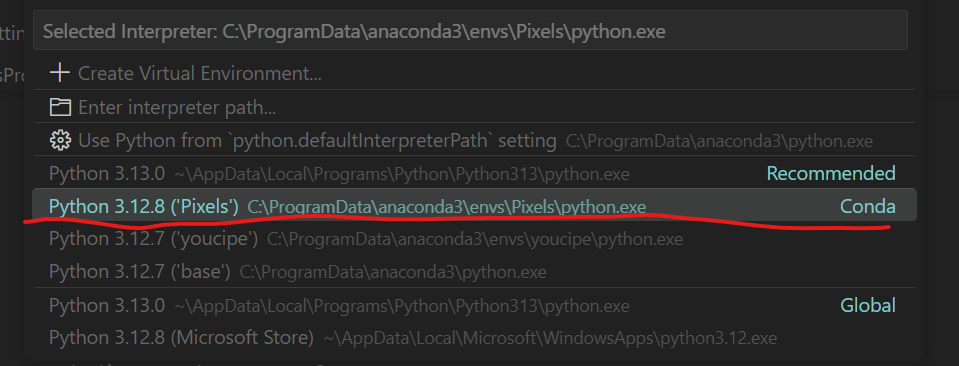
#### Registering pixels app in settings.py of main PixelsProject

Navigate to PixelsProject>Settings.Py

In Settings.py under installed apps, add pixels app

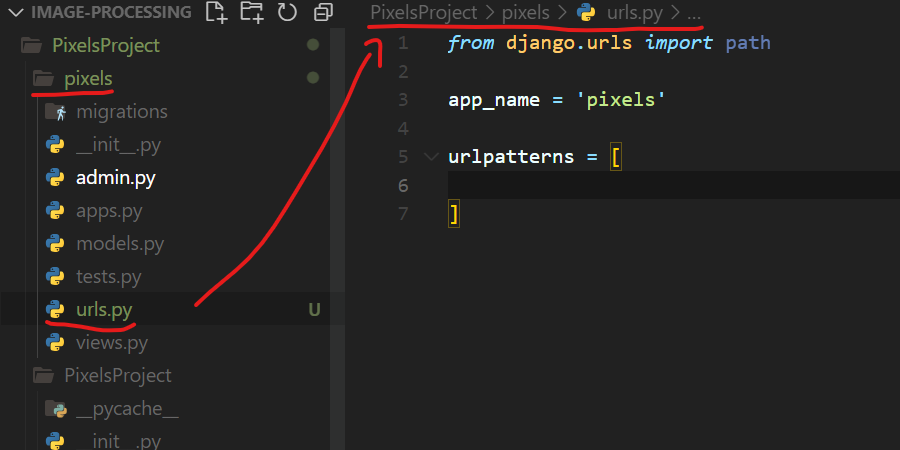


\*Note\* in your IDE make sure that you have selected your virtual environment to avoid errors

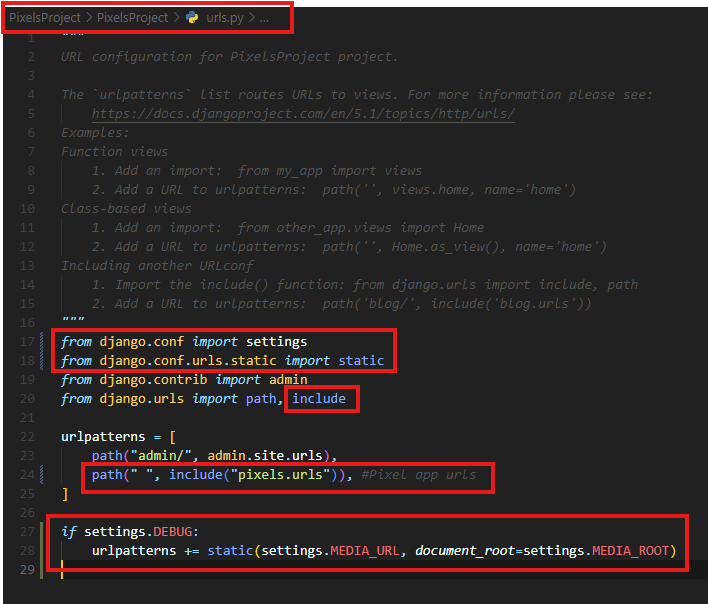


#### Create urls.py file for pixels and link to main urls [[4](#_References)]

Create urls.py file for app pixels’ urls



Link app pixels urls to main urls.py of Pixelsproject []

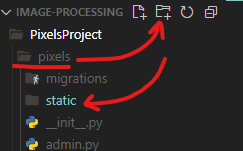


### Create & Configure static folder and files

This is to serve additional files such as images, javascript, or css [ ]

1. Let’s create a static folder in our app {{{{{{{ conda install anaconda-navigator

**Option 1 (create through IDE GUI)**



**Option 2 (create through CLI)**

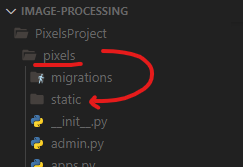
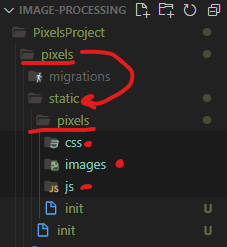
* Navigate to pixels app



* Run the command mkdir static



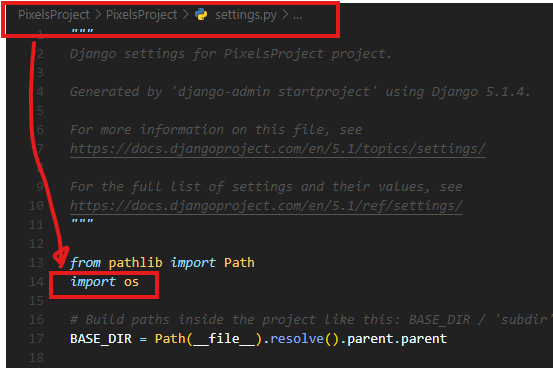
* You should now see a new folder in the pixels app named ‘static’

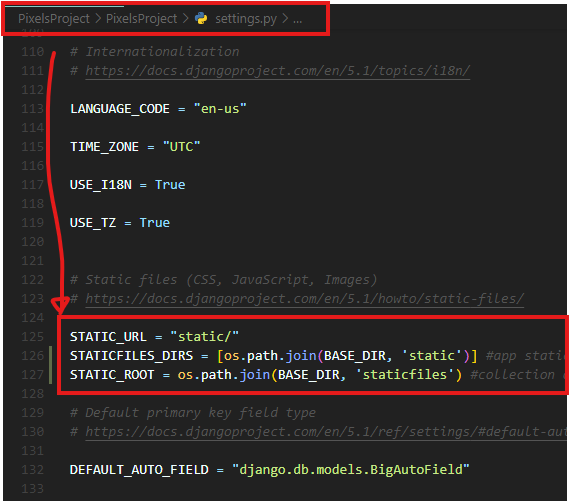
\*Best practice is also to create static sub folders for future usage\*

1. Navigate to pixelsproject/pixelsproject/settings.py []

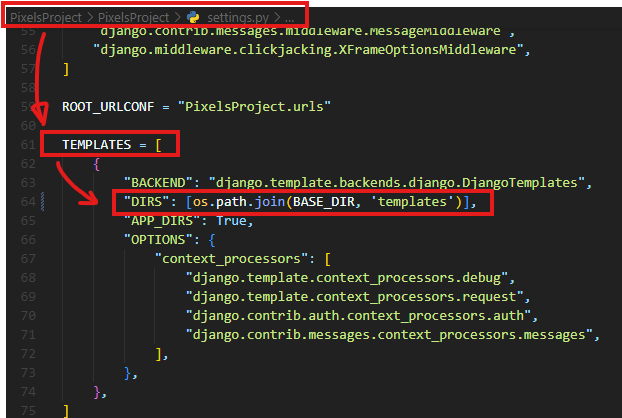
First, we want to import os [] for directory handling and processing



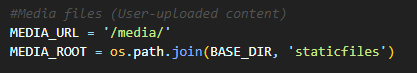
Add directory paths in the Static files section of settings.py [ ]



Configure templates in settings.py

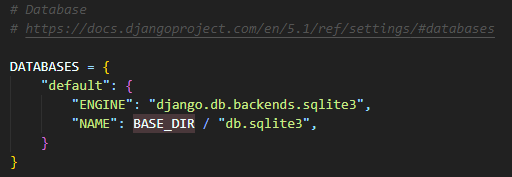


Configure Media (managing files) []



### Configuring database

In our case since we are using SQLite3 the native Django database. We do not need to configure the database. Leave as is:



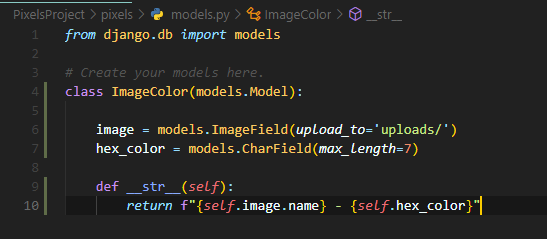
If working with a different database such as Postgres. This is how you would configure it:

1. Create a blank database on Postgres (pgadmin UI for simplicity) [ ]
2. note the database credentials and modify the db settings of settings.py as follows:



# Model Creation

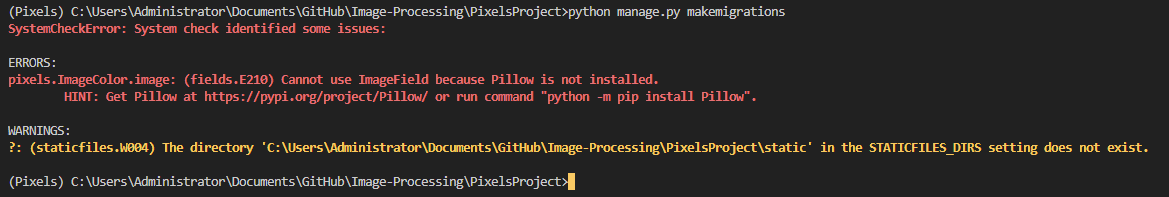
## Open models.py of pixels app (pixelsproject/pixels/models.py) and define a class with Imagefield and charfield



Next, go to the CLI to make migrations. After creating a model in Django, before running our server (the server can still run without making migrations but you will get a makemigrations error) we have to makemigrations to create a migrations file that contains code for the tabled schema of our model. This is done through the Django ORM. It creates a table according to the schema defined in the migration file.

1. Run command python manage.py makemigrations

Encountered errors

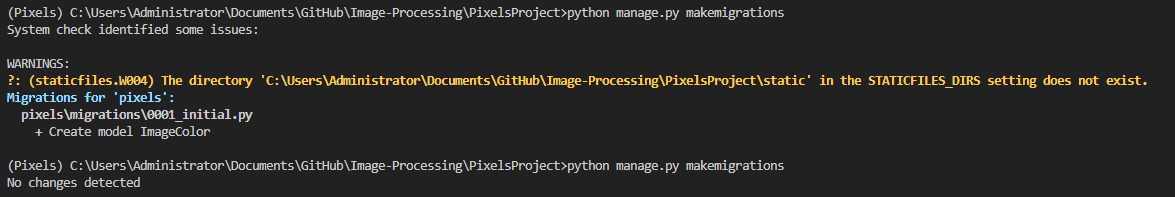


Static file settings incorrect and pillow not installed (Pillow package is a prerequisite for using ImageField) []

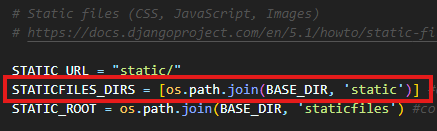
1. Run command pip install pillow



1. Run command manage.py makemigrations to initialize our database

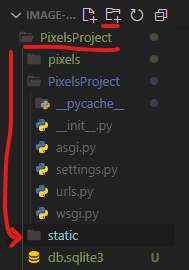


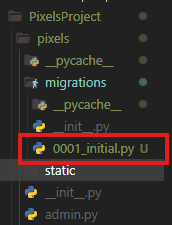
Because we have configured settings for global static(Pixelproject/PixelProject/Settings.py). We need to create a global static folder



In our PixelsProject root folder, let’s create static folder to resolve this issue:

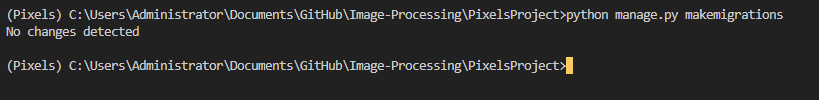
?: (staticfiles.W004) The directory 'C:\Users\Administrator\Documents\GitHub\Image-Processing\PixelsProject\static' in the STATICFILES\_DIRS setting does not exist.



1. Now that we have installed pillow and created a global static folder. Let’s run migrations again. Because we have already ran migrations. We now have a migrations file:  
   

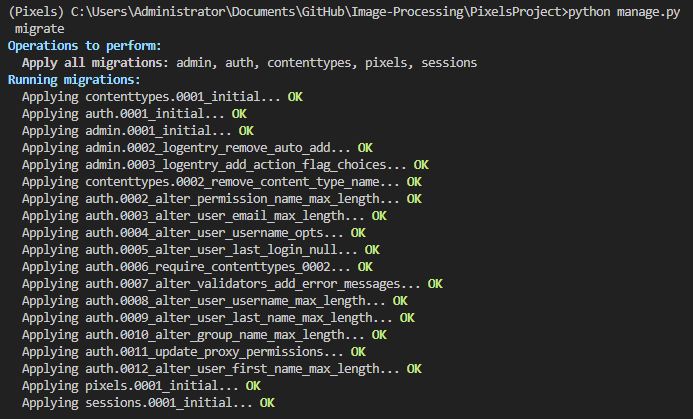
This is the initial migrations file

So now when we run migrations. We will see the following:



This is because we have already ran migrations to our database but have not made any modifications to our model class.

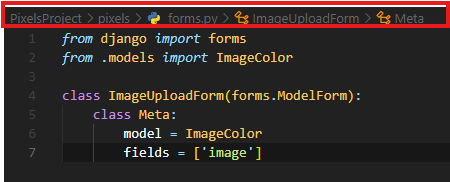
1. Run python manage.py migrate to apply changes to our database



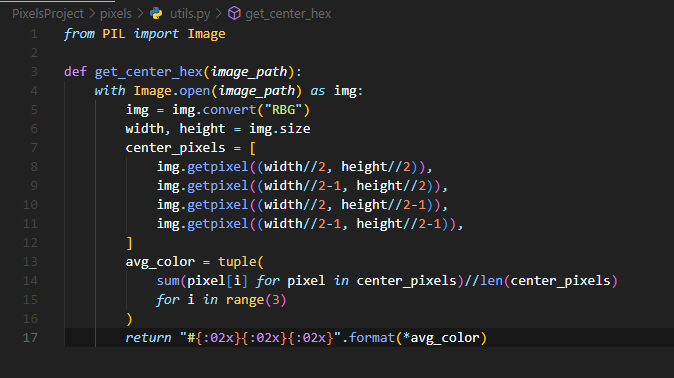
# Image Upload & Processing

1. Let’s create a form.py for handling user image uploads [] []

Create forms.py file in Pixels app (PixelsProject/pixels/forms.py)

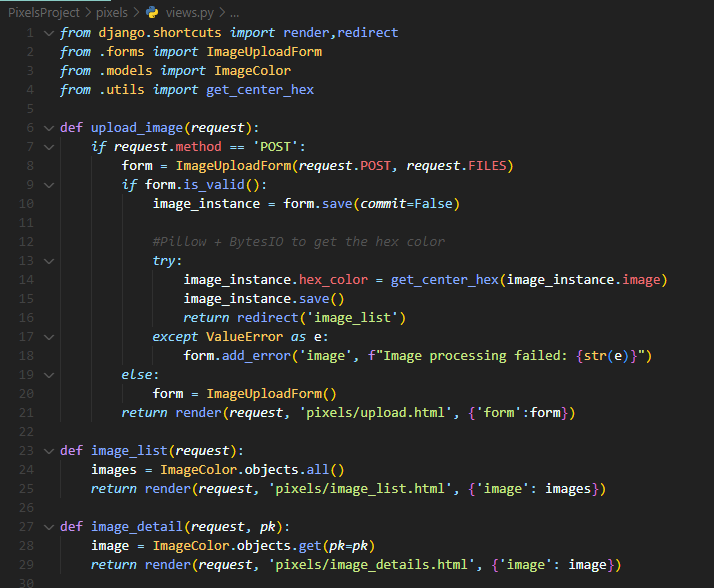


1. Now let’s create utility for processing images through Pillow

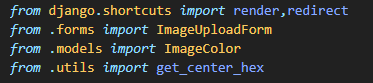


# Views Creation

## Overview

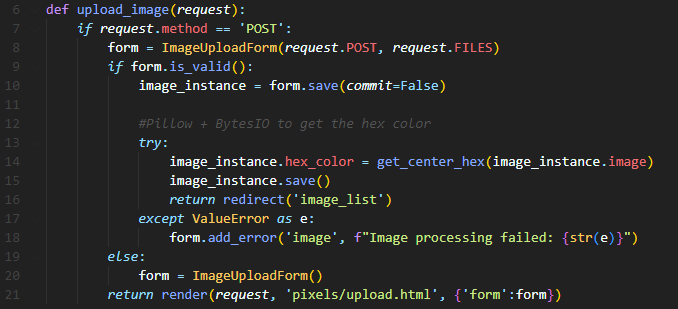


1. First, we have to import necessary dependencies and the created py files (forms, models, utils)

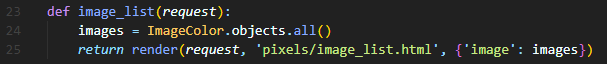


1. Secondly we need functions to handle image uploads, image list, and image detail

**Upload image function**



**List image function**

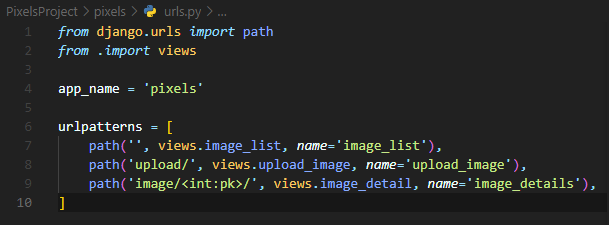
****

**Image detail function**

****

# URLS creation(app)

## Overview



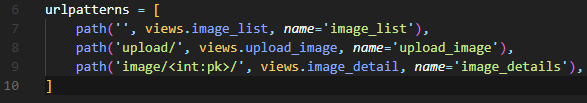
1. Firstly, let’s import the necessary dependencies and views



1. Define app\_namespace so that app urls are not added to global namespace. If app\_name = ‘yourappname’ is not defined it will lead to url conflicts at a later stage. [ ]



1. Then we define urls patterns to route the functions that we created in our views



# API Testing (Postman)

## Test case 1: POST request

### Request:

### Response:

## Test case 2: GET request

### Request:

### Response

## Test case 3: PUT request

### Request:

### Response:

## Test case 4: DEL request

### Request:

### Response:

# Bonus Points

## Frontend Enhancement

Create global static folder, which will contain our project layout and base html template

Serving static files []

Serving files uploaded by user[]

## Error Handling

## Efficiency

# Evaluation Criteria

## Functionality

Does the application work as describe without error?

### Test Functionality

Since we only have a few tests. We will be using the default test.py file created upon project initiation. However for larger web applications, you will have to use a test package []

## Code Quality

## Efficiency

## Error Handling

## Bonus

# Architecture

# API

# Conclusion

# Appendix A

## References

[1] Creating directories through CLI

<https://www.ibm.com/docs/en/aix/7.1?topic=directories-creating-mkdir-command>

[2] Installing Github Desktop

<https://docs.github.com/en/desktop/installing-and-authenticating-to-github-desktop/installing-github-desktop>

[3] Creating a repository using Github Desktop

<https://docs.github.com/en/desktop/overview/creating-your-first-repository-using-github-desktop>

[4] Create Django project

<https://docs.djangoproject.com/en/5.1/intro/tutorial01/>

[5] Django Admin and Manage.py

<https://docs.djangoproject.com/en/5.1/ref/django-admin/>

[] Django Settings

*https://docs.djangoproject.com/en/5.1/topics/settings/*

<https://docs.djangoproject.com/en/5.1/ref/settings/>

[] OS Module  
<https://www.geeksforgeeks.org/os-module-python-examples/>

[] Create Database through pgAdmin

<https://www.pgadmin.org/docs/pgadmin4/8.14/database_dialog.html>

[] How to deliver images (Django Forum)

https://forum.djangoproject.com/t/whats-the-best-way-to-deliver-images/27525

[] Writing and Running Tests in Django

<https://docs.djangoproject.com/en/5.1/topics/testing/overview/>

[] Colors on an Image (Pillow)

<https://www.geeksforgeeks.org/python-pillow-colors-on-an-image/>

[] Why declare app\_name in app URLs

https://stackoverflow.com/questions/61254816/what-is-the-purpose-of-app-name-in-urls-py-in-django

[ ] How to create a conda virtual environment (CLI)

https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

[ ] Create Static folder and Media Folder (static is responsible for housing css, js, html and images). Media folder is responsible for housing media uploaded by the user

https://docs.djangoproject.com/en/5.1/howto/static-files/

[ ] Image Field

<https://docs.djangoproject.com/en/5.1/ref/forms/fields/>

[ ] Image Upload Form

https://docs.djangoproject.com/en/5.1/ref/forms/api/

[ ] Bind Uploaded files to form

<https://docs.djangoproject.com/en/5.1/ref/forms/api/#binding-uploaded-files>

[ ] Pillow (needed for Image Field usage)

<https://pypi.org/project/pillow/>

# Appendix B

## Figures