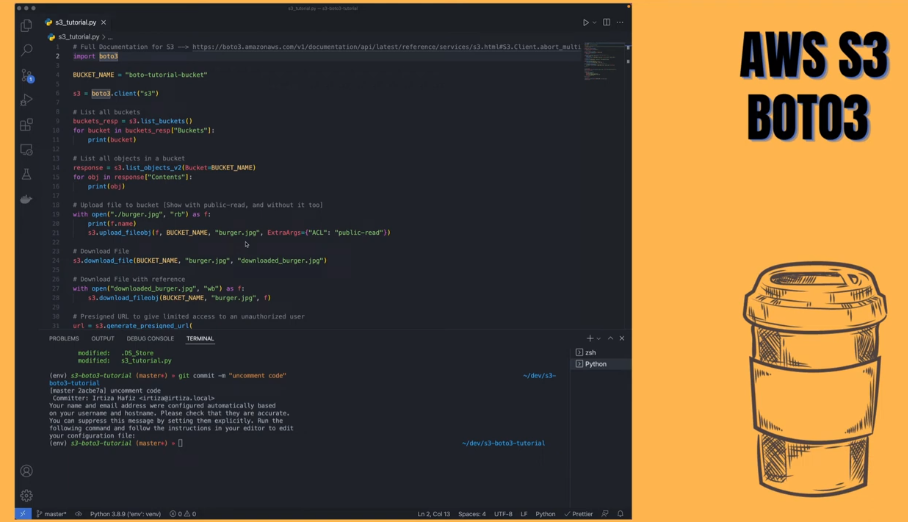
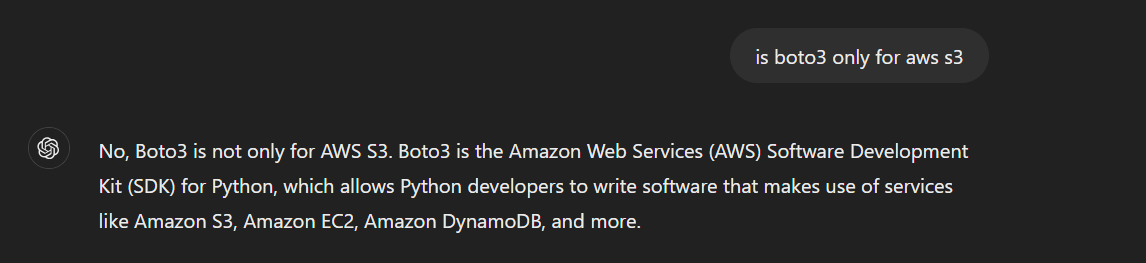
**Boto3**



(How to use Python to talk to AWS)

In here the object name inside a bucket 🡪 is called a key.

First let’s import boto3. (This one might have to be installed using pip first) 🡪 ‘pip install boto3’



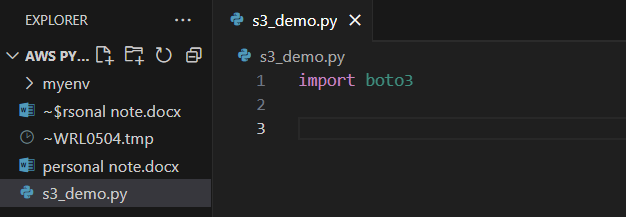
Yeah, I had to install it through pip. (unlike os, time, smtp),

Let’s go with a virtual environment to install packages (it is easier with installing python packages). Created the venv in the same project folder.

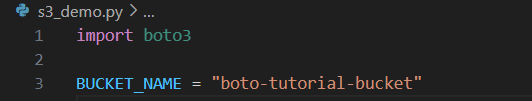
And then run all the commands through that virtual environment (like we did with Flask),

So now,

First let’s import boto3,

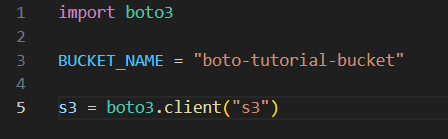


Let’s give the bucket a name,



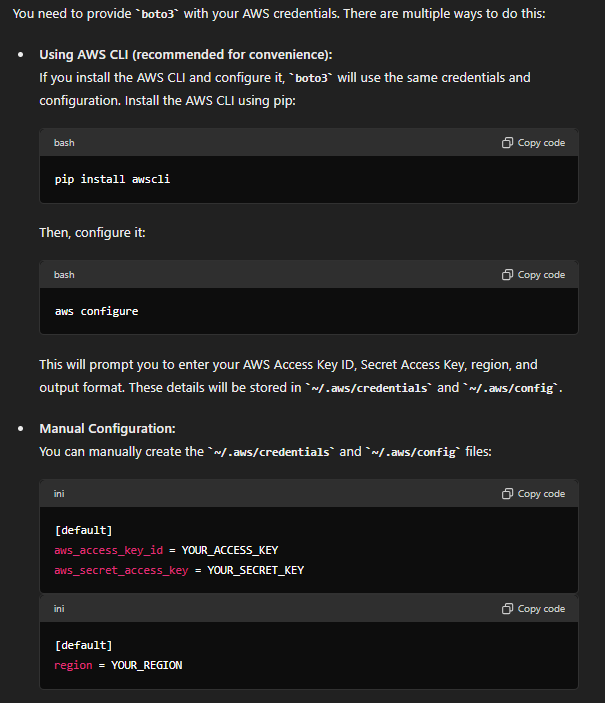
And then, the first thing you need is a reference to S3. Means you are going to need a S3 client.

And for every operation you are going to be using that S3 client,

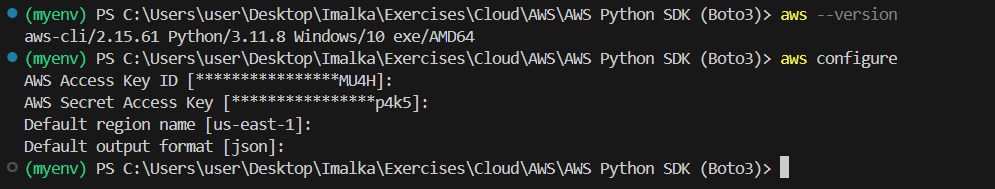


(Initializing a S3 session)

And most importantly,

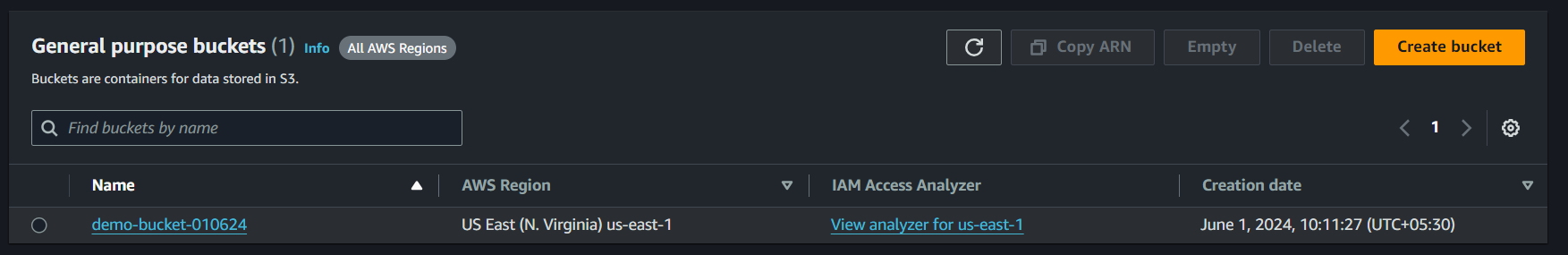


So the easiest way is if you have installed aws CLI and have configured it, boto3 will use the same configurations, so you don’t have to set it up again,



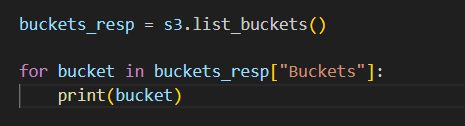
(the aws CLI configurations that we did is also recognized here, if not go ahead and install and configure these first)

First thing we are going to look at is how can you list all the buckets,



(A bucket that I already have)

For that,

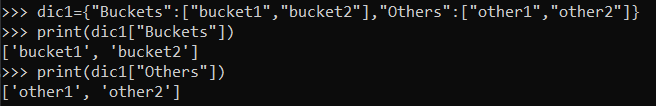


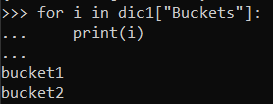
The reason why you have to go with,

buckets\_resp[“Buckets”]

Look at this example,

It can be assumed that the buckets\_response is like a dictionary(or in JSON format),





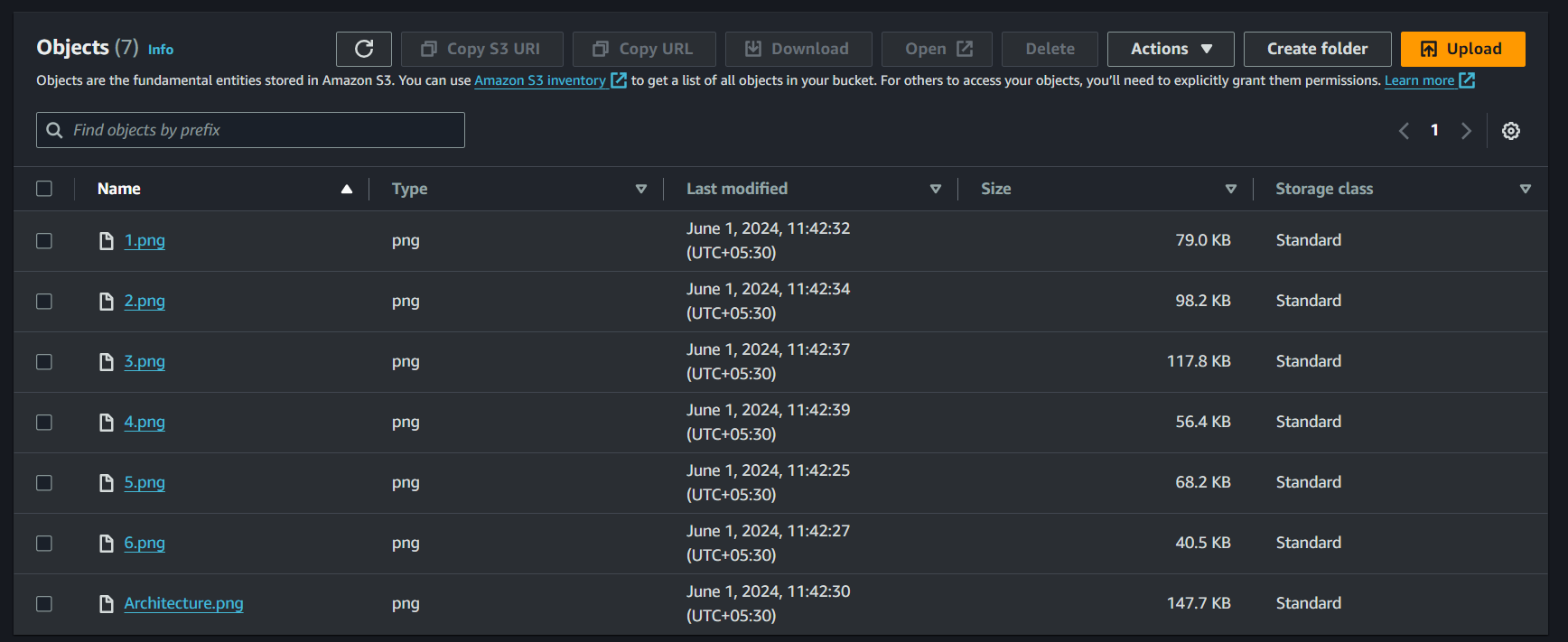
(So, we can only get the buckets listed out)

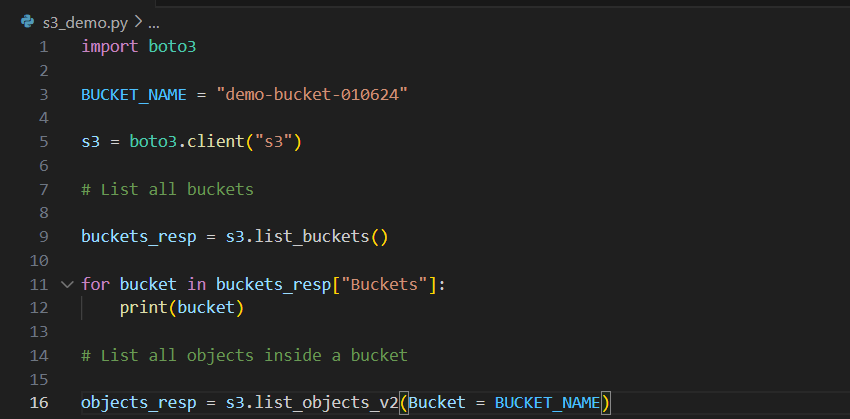
So, if we execute the above code,

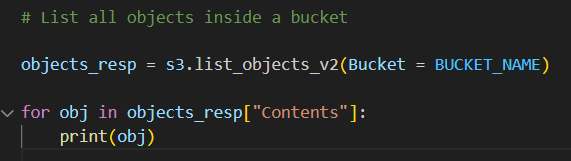


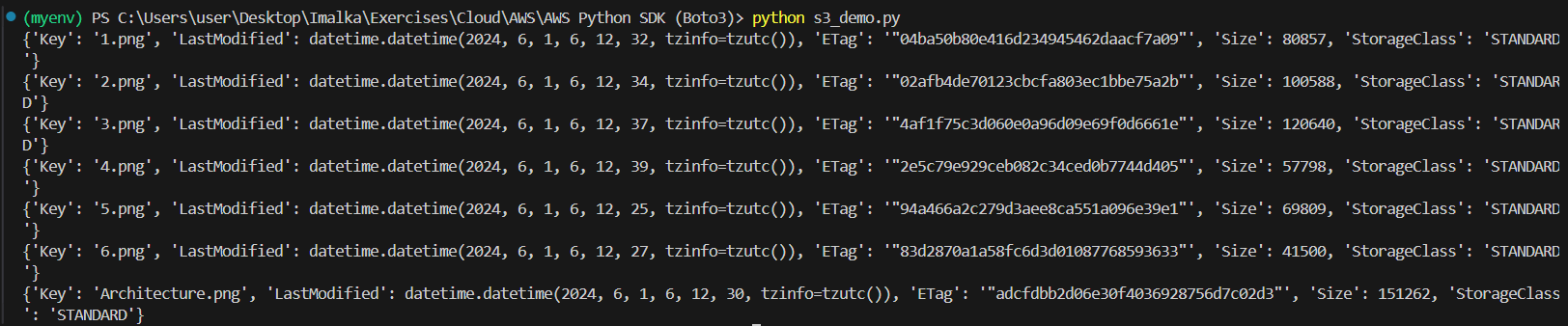
(it works)

Now, let’s see how to list all the objects inside a bucket.





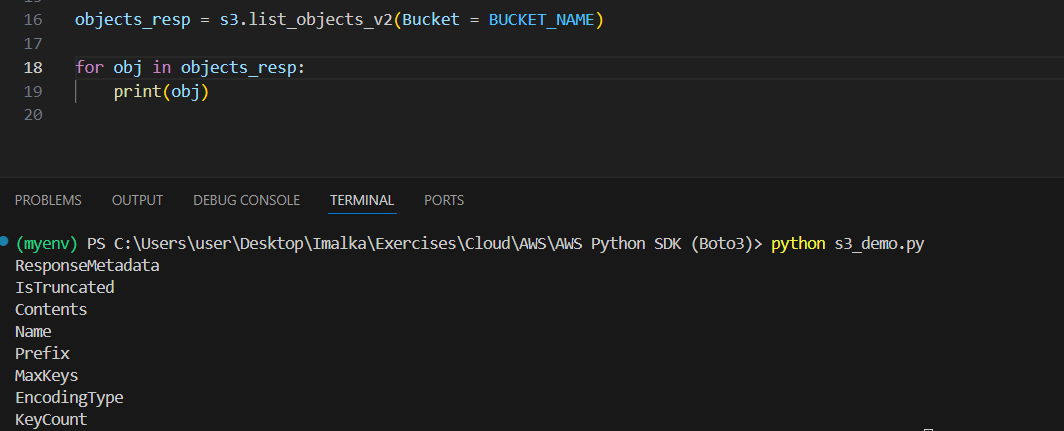




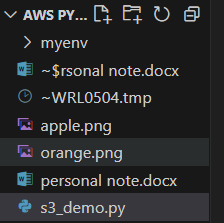
(As you can see it lists all the objects that is inside the bucket. These are dictionary objects as you can see. And also, these are metadata)

Additional 🡪

You can find the keys of dictionary/json:

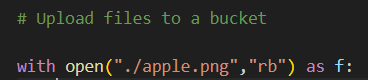


Now, let’s see how to upload a file,

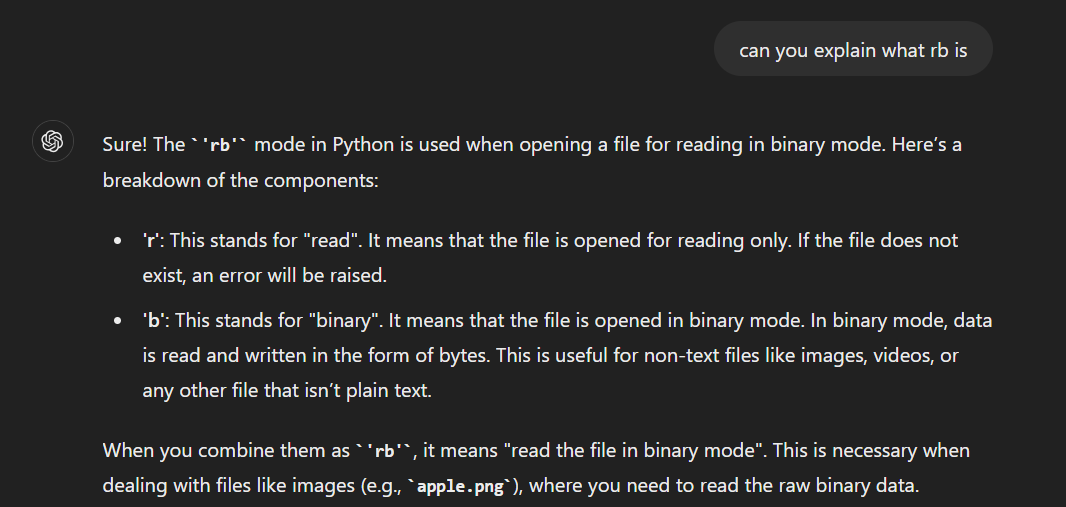


(Need to upload these two files)

First,

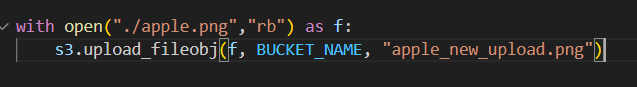


Open the file you want to upload, and open it in ‘rb’ mode,



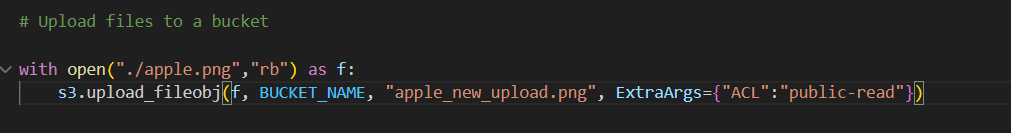
And for the file we have used an alias called ‘f’,

And then,



Key of the object/new name if the object inside the S3 bucket

You can also give other extra arguments as well,



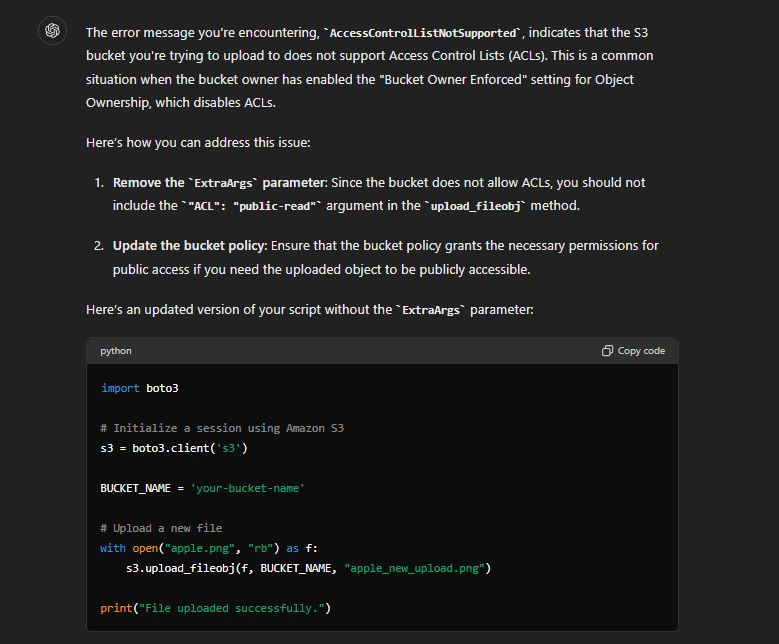
The extra argument is a dictionary, where you give it a key, value pair.

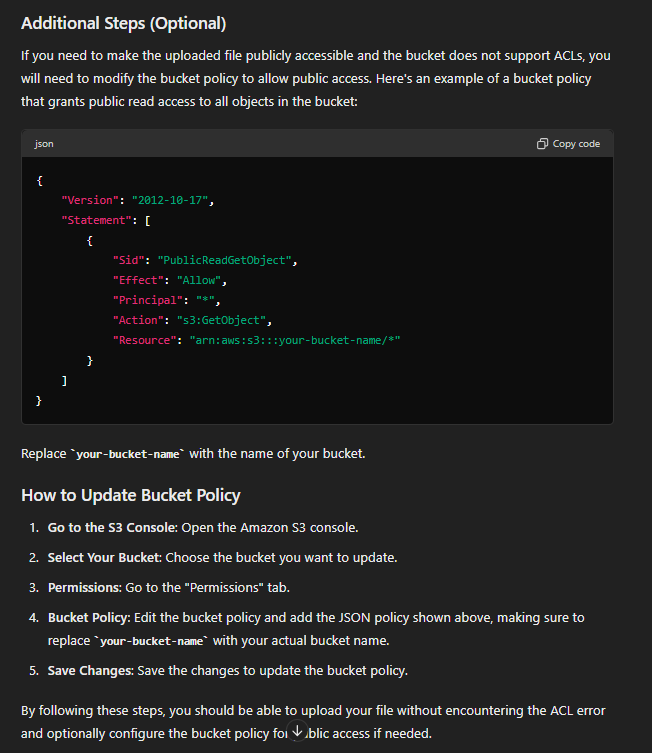
What this does is 🡪 When it uploads the file it sets the files access control list to public read so that everyone in the public internet can read this file. Of course, usually you don’t want this, but for testing purposes right now, we are going to use public read so that we can verify that the file has been uploaded correctly.

How ever we are going to look at a better way to do this, without running into security risks.

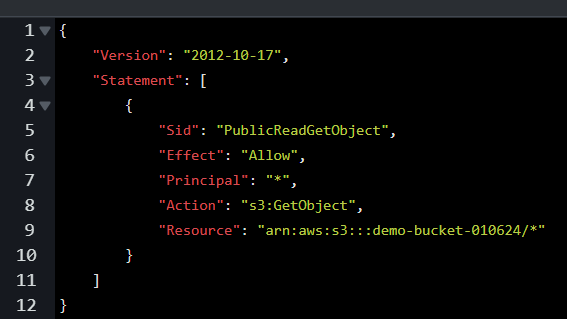
Now, let’s go ahead and run this,

But when running this I encountered an error saying,

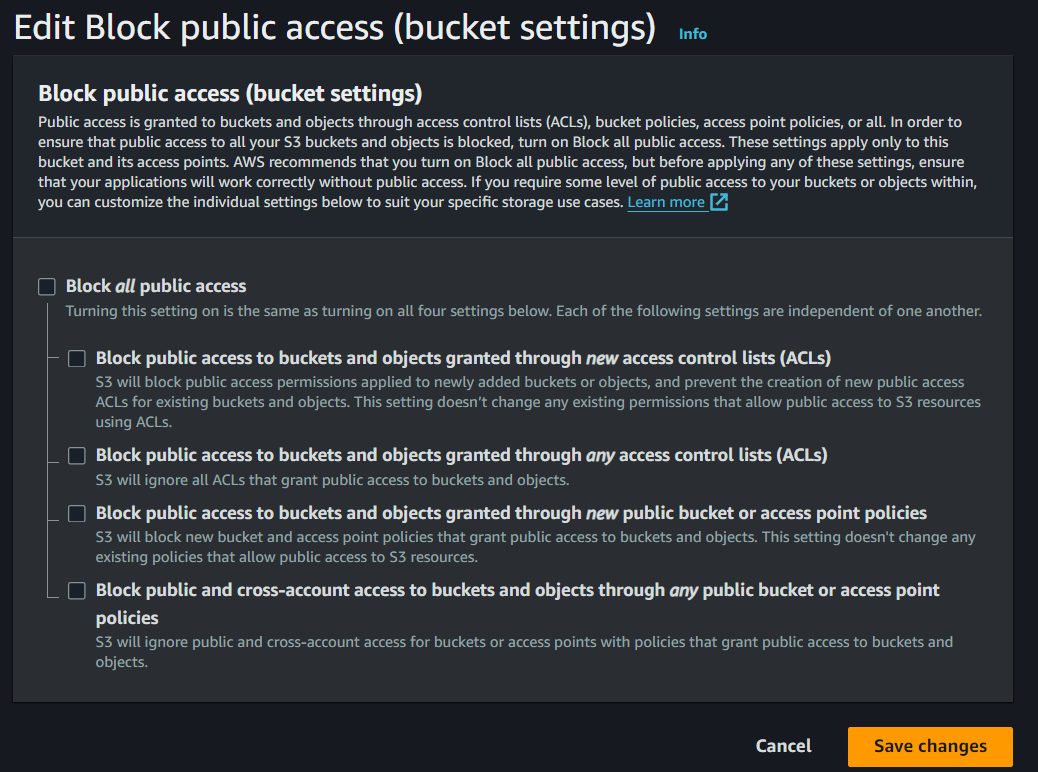




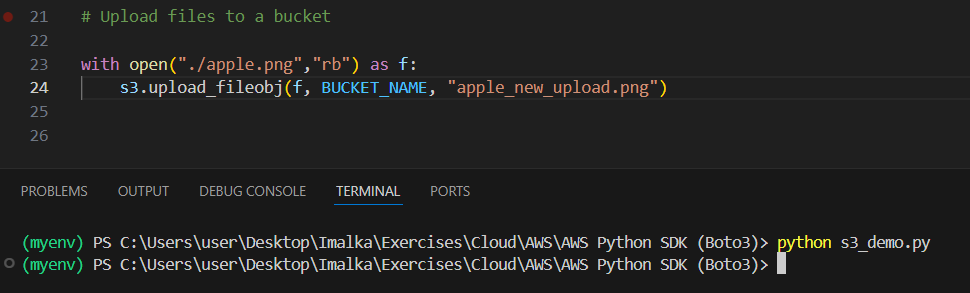
So, let’s remove the extra argument and upload it, but let’s update the policy of the s3 bucket using the console,



And also make sure to remove the block all public access option from the console,

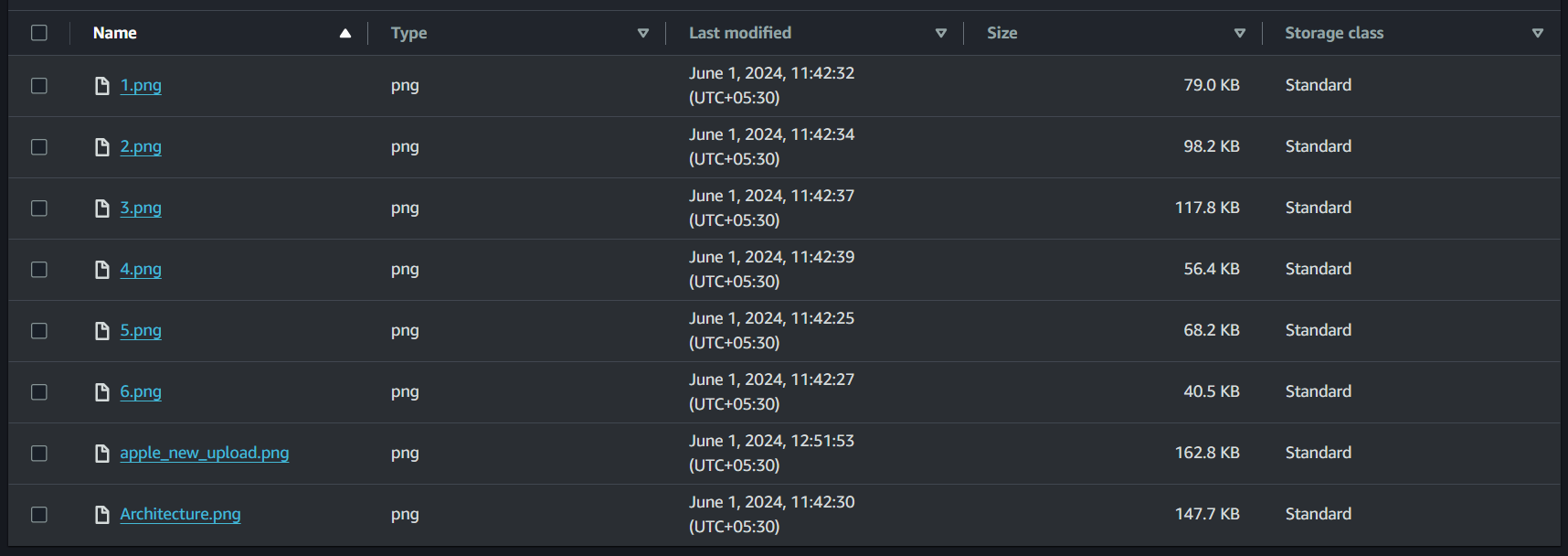


Now let’s run this code,



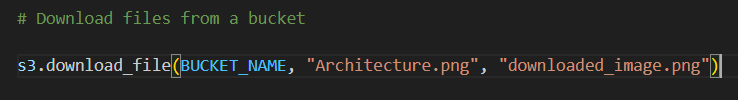
It will take some time.

After it is done, go ahead and check inside the bucket,



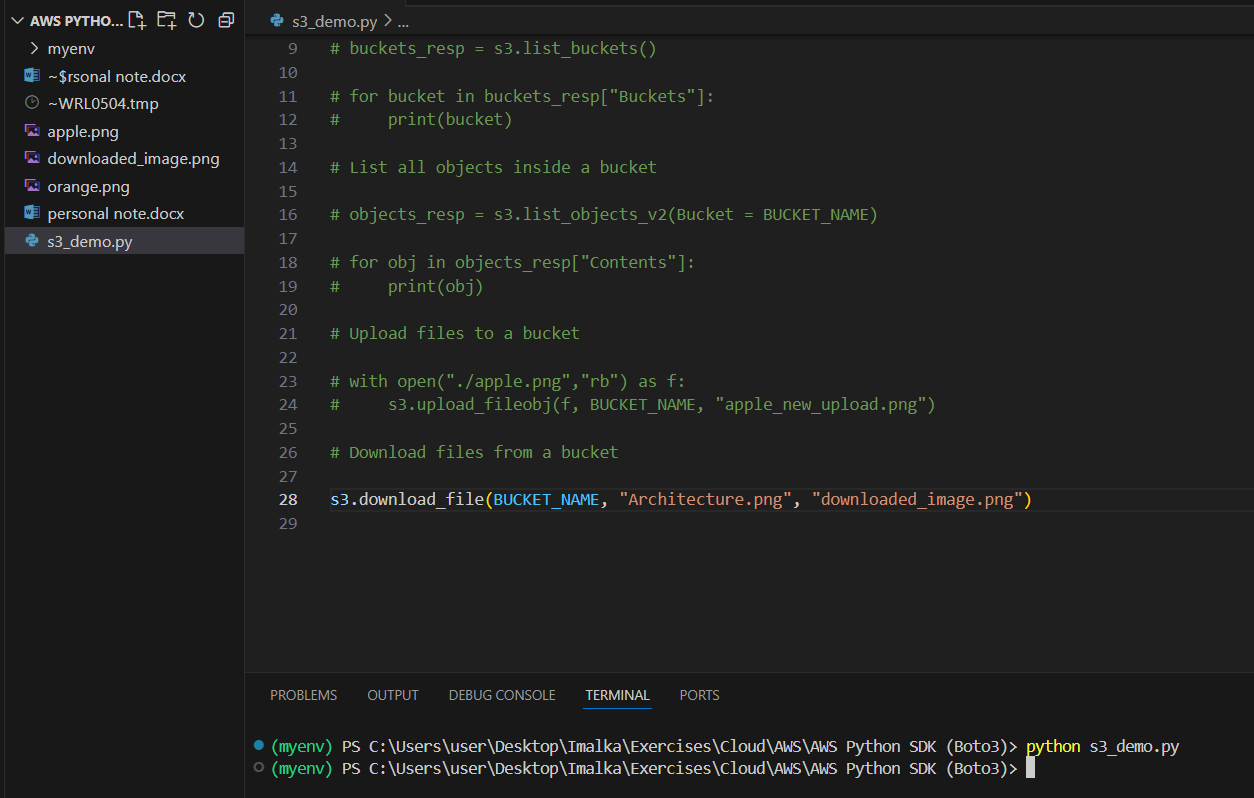
(The file has been uploaded)

The next thing we are going to look at is, how to download a file,

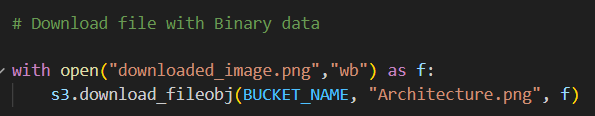


Name of the downloaded file has to be.

Let’s run this code,



Now, let’s see how to download a file with Binary data,



With this way you directly have reference to what you have downloaded.

It is going to download the burger.jpg file from the bucket, and put it in that f variable so that we have access to the binary data of the object and then you can do some image manipulation or you can pass it back to the back end as binary data or you can just save the file like, we previously did.

(File is downloaded and it is put into that ‘f’ variable)

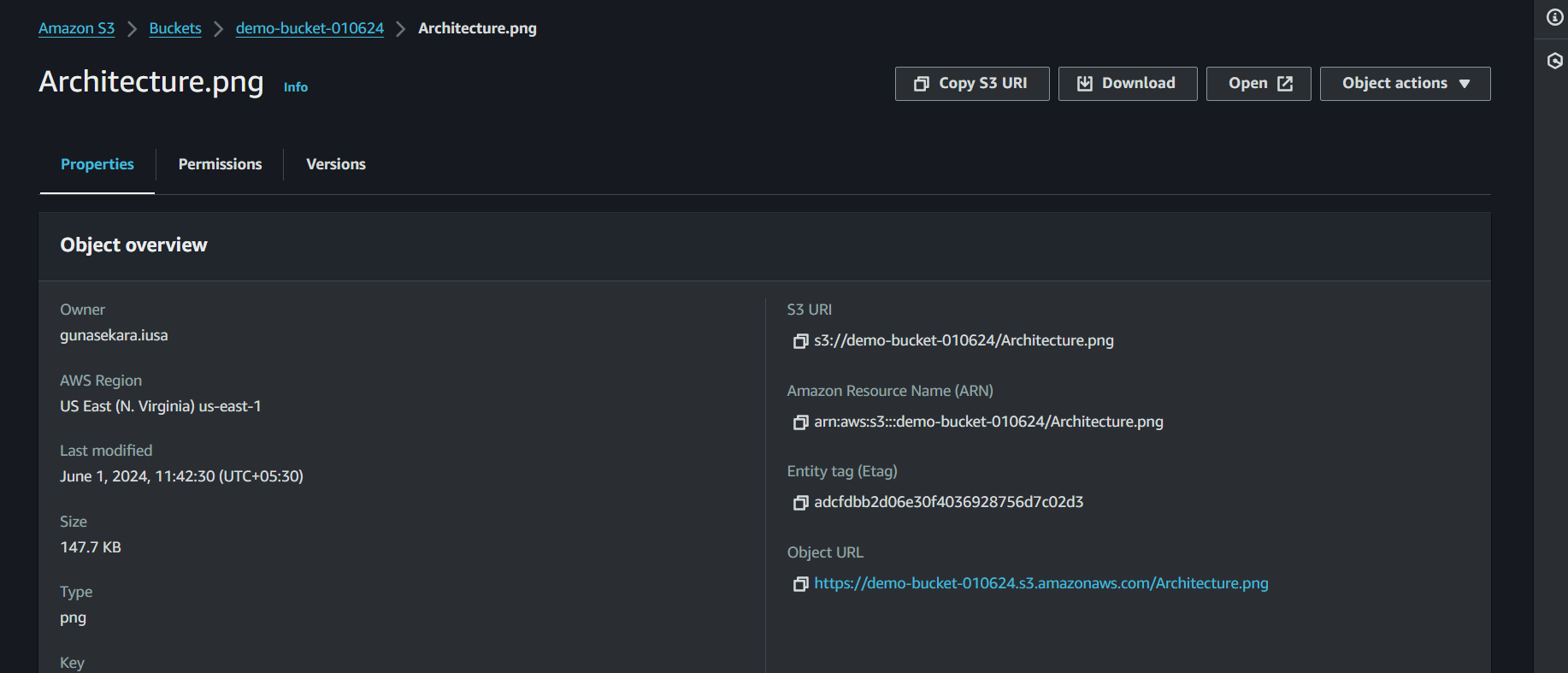
Let’s run the code,



In the same way the file is also downloaded like in the previous method, but now you also have ‘f’ as a reference, so you can do anything with it.

Now we’ll look at pre-signed URL,

Pre-signed URLs are used to 🡪 give limited access to an unauthorized user.



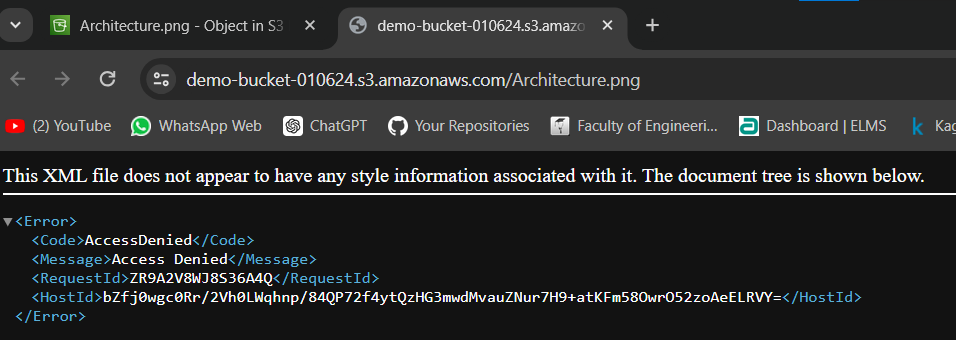
(As you can remember by default the bucket is blocking all public access so when you click at the object URL/public URL, you get access denied. But when you click at the ‘open’ button we can temporarily access it. So, what happens when you click the ‘open’ button is that it, provides us with a pre signed URL)

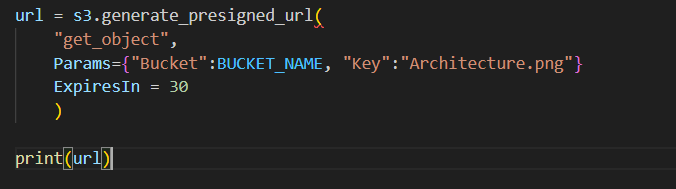
So, instead what you want is, you want only your back end to have access to that file. And when the back end is giving the URL to the front end, instead of giving the URL, it’s going to give the pre-signed URL. So, the backend is signing that URL and giving it to the front end, telling the front end you have ‘x’ number of seconds or minutes to access this data. After the given time, you will not have access to it.

This is a very easy way to have private resources in S3. But at the same time give a limited amount of time, where other people can have access to it.

Let’s see how to do this with code,

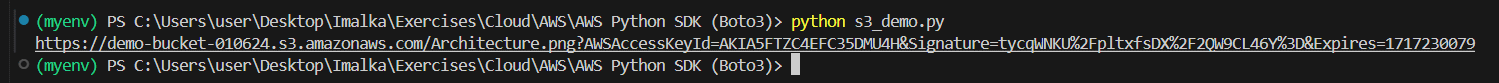
First let’s again enable the block public access and remove the policy that we add, so that we have the default state where the public URL’s access is denied,

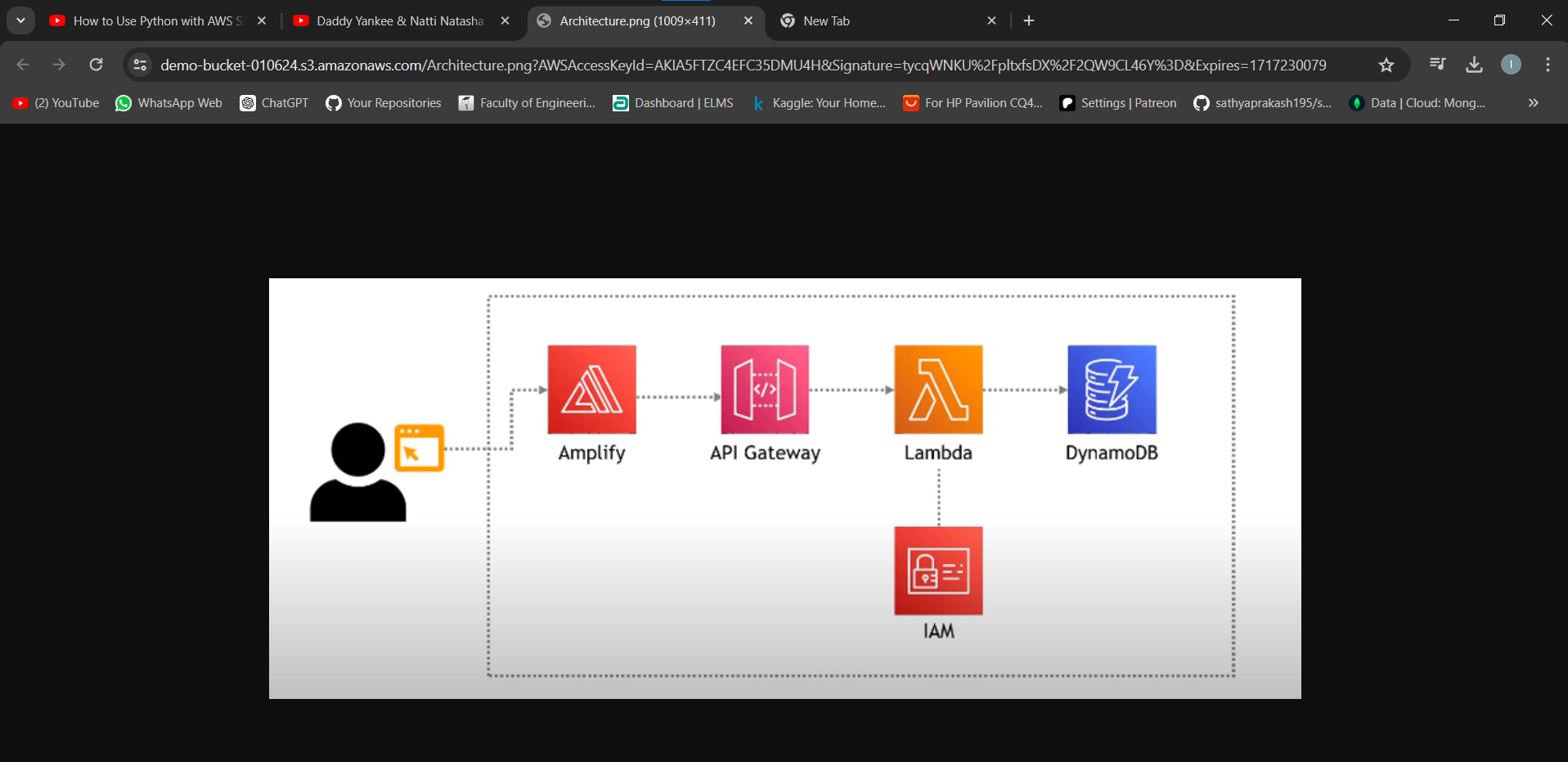




(URL expires in 30 seconds)

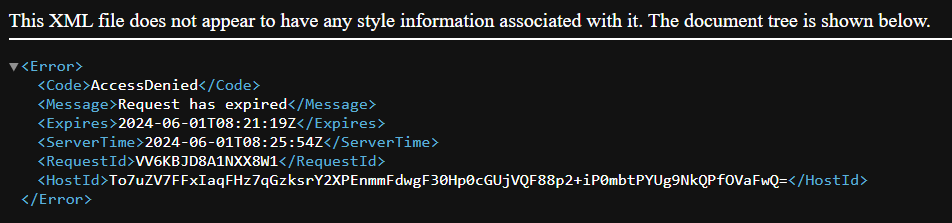
And let’s run this code,



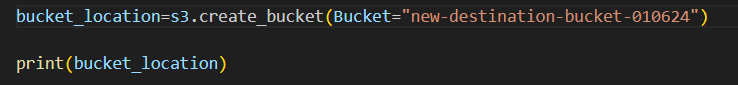


(It will be valid for 30 seconds)

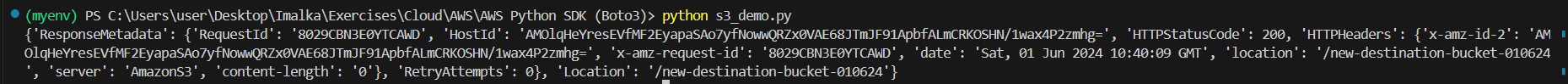
After it expires,



Now, let’s see how to create a new bucket,

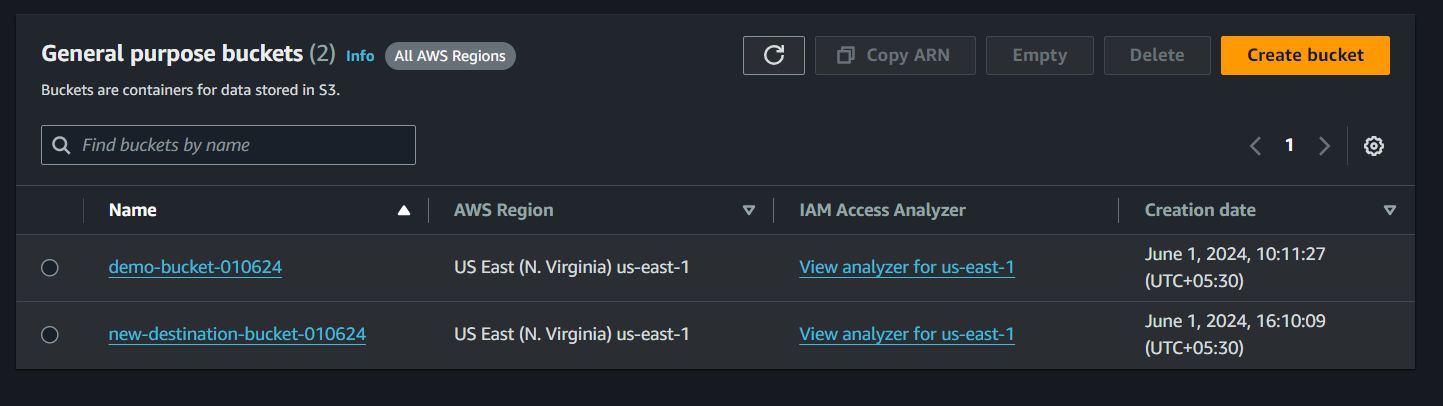


If you run this code,



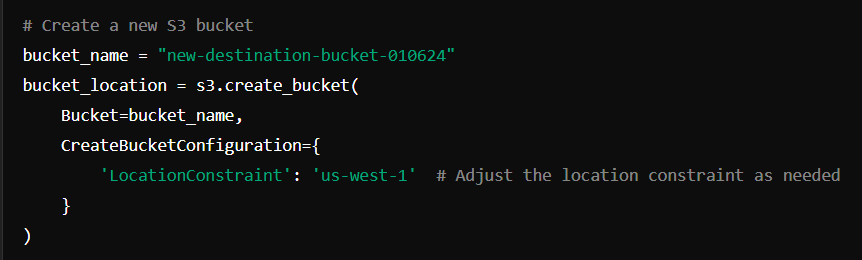
(we get the meta data)

If we check this,

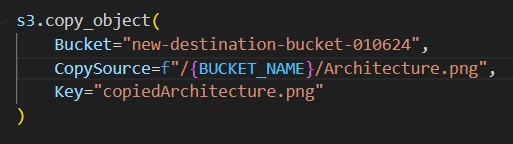


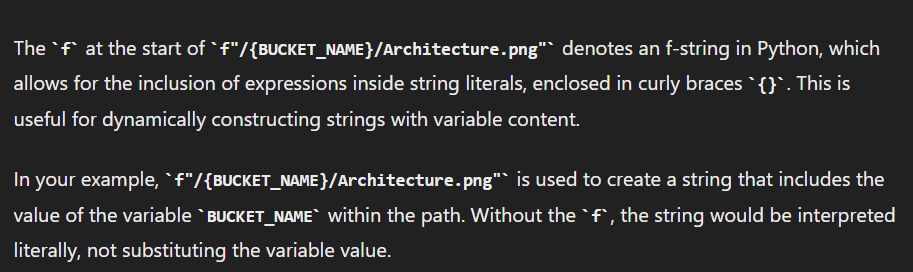
(A new bucket is created)

If you want you can specify the location too when creating the bucket,

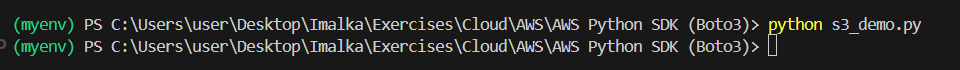


Let’s see how to copy objects within buckets,

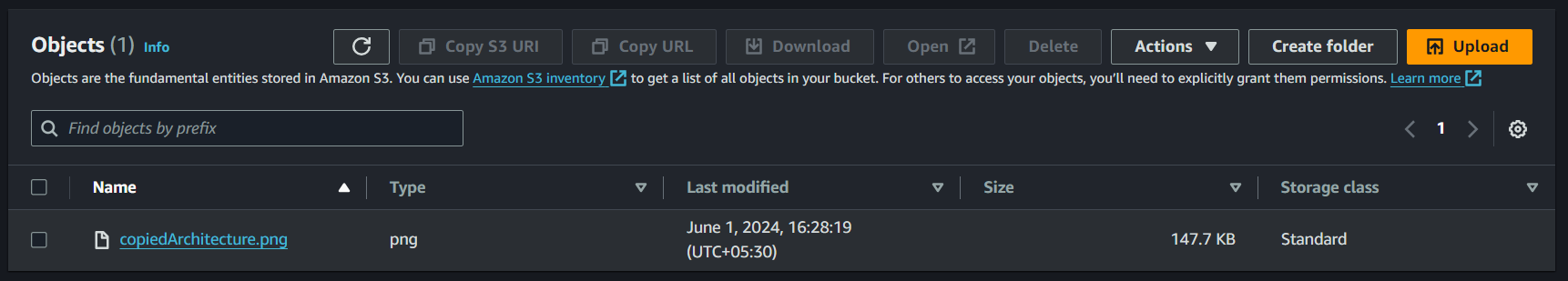




If we run the above code,

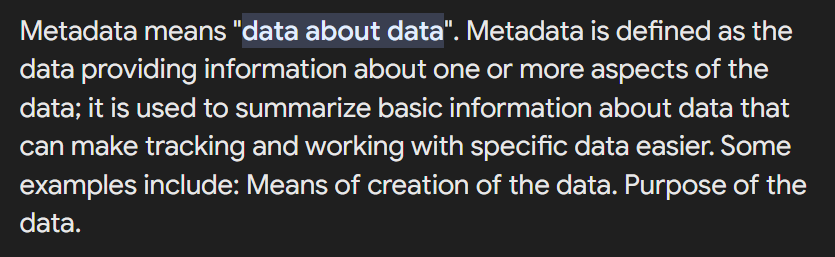


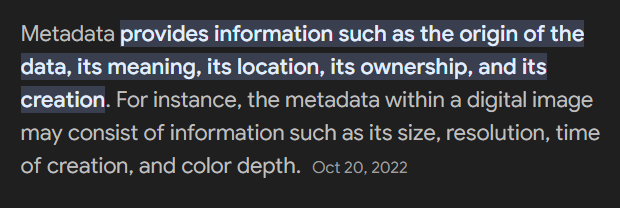
Let’s check,

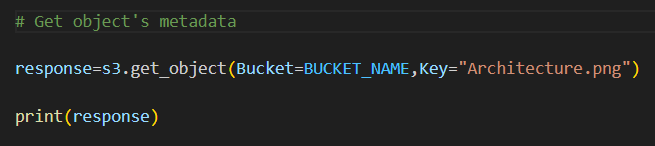


(It is copied)

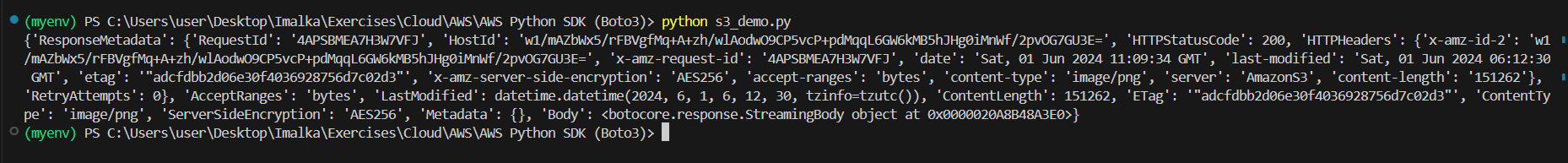
Now, let’s see how to get details/metadata about an object,



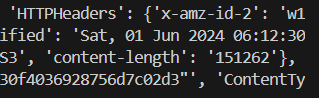




Let’s run this code,



(meta data of the object)



(size of the file is 151,262 bytes)

