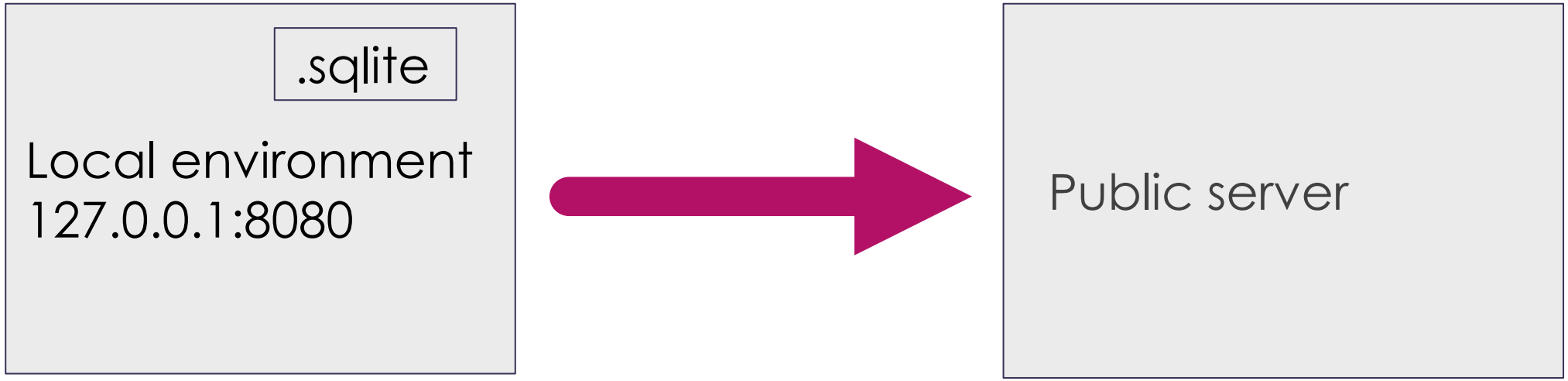
“Topic 4: Deployment”

Deploy

Finding a host to deploy the website, and what we need to do in order to get our site ready for production.



Hosting

Before host a website externally things that need to:

1. Make a few changes to your project settings.
2. Choose an environment for hosting the Django app.
3. Choose an environment for hosting any static files.
4. Set up a production-level infrastructure for serving your website.

What is a production environment?

The production environment is the environment provided by the server computer where you will run your website for external consumption. The environment includes:

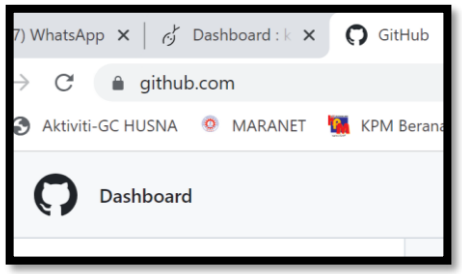
1. Computer hardware on which the website runs.
2. Operating system (e.g. Linux, Windows).
3. Programming language runtime and framework libraries on top of which your website is written.
4. Web server used to serve pages and other content (e.g. Nginx, Apache).
5. Application server that passes "dynamic" requests between your Django website and the webserver.
6. Databases on which your website is dependent.

Choosing a hosting provider

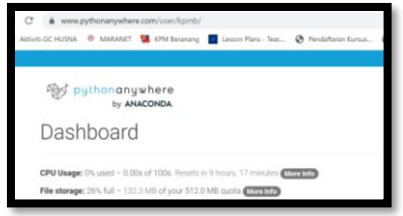
Things to be consider when choosing a host:

1. How busy your site is likely to be and the cost of data and computing resources required to meet that demand.
2. Level of support for scaling horizontally (adding more machines) and vertically (upgrading to more powerful machines) and the costs of doing so.
3. Where the supplier has data centres, and hence where access is likely to be fastest.
4. The host's historical uptime and downtime performance.
5. Tools provided for managing the site — are they easy to use and are they secure (e.g. SFTP vs FTP).
6. Inbuilt frameworks for monitoring your server.
7. Known limitations. Some hosts will deliberately block certain services (e.g. email). Others offer only a certain number of hours of "live time" in some price tiers, or only offer a small amount of storage.
8. Additional benefits. Some providers will offer free domain names and support for SSL certificates that you would otherwise have to pay for.
9. Whether the "free" tier you're relying on expires over time, and whether the cost of migrating to a more expensive tier means you would have been better off using some other service in the first place!

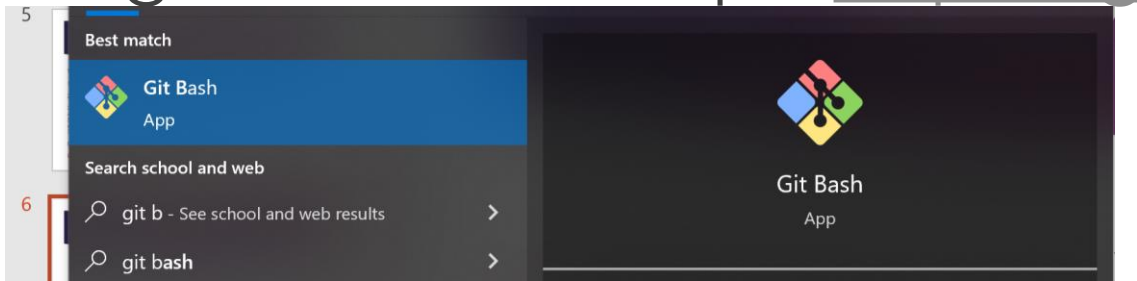
THINGS TO DO



1. Register github account at https://github.com/
2. Register pythonanywhere account at https://www.pythonanywhere.com/



1. Install git bash from https: https://git-scm.com/download/



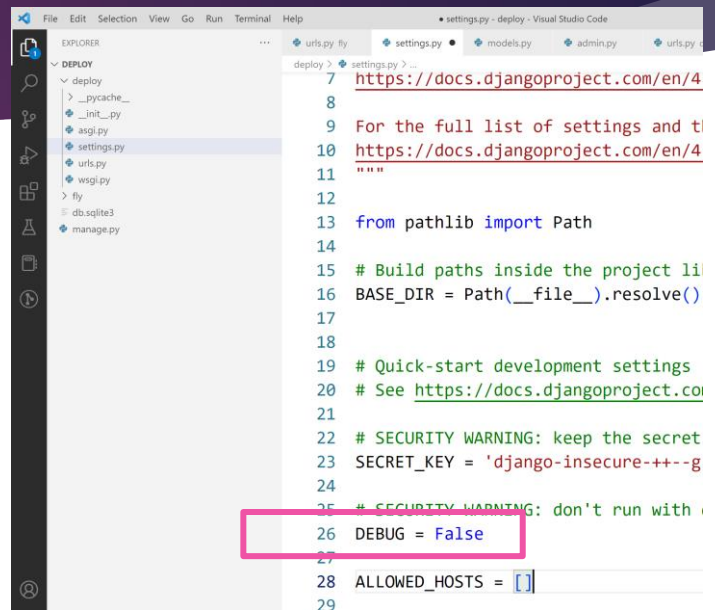
APPLICATION SETUP

In order to publish Django application, project need to be set up the appropriate environment and dependencies, and also understand how it is launched. There is information in a number of text files:

1. runtime.txt: states the programming language and version to use.
2. requirements.txt: lists the Python dependencies needed for your site, including Django.
3. Procfile: A list of processes to be executed to start the web application. For Django this will usually be the Gunicorn web application server (with a .wsgi script).
4. wsgi.py: WSGI configuration to call our Django application in the Heroku environment.

DEPLOYMENT

Step 1 : Modify setting in Django (setting.py)



- DEBUG
  a) This should be set as False in production (DEBUG = False).
  b) This stops the sensitive/confidential debug trace and variable information from being displayed.
  c) Debug allow user to see error. So is important to be false so the user can't see the error

- SECRET_KEY
  a) This is a large random value used for CSRF protection, etc.
  b) It is important that the key used in production is not in source control or accessible outside the production server.
  c) The Django documents suggest that this might best be loaded from an environment variable or read from a server-only file.
  d) Important : Each application use different secret key

Step 1 : Deployment setting in Django (setting.py)

1. Open KPMB project in visual code
2. Go to setting
   a) Add header os

```
from pathlib import Path
import os
```

   a) Change DEBUG

```
DEBUG = os.environ.get('DJANGO_DEBUG', '') != 'False'
```

   b) Change SECRET_KEY os.environ.get('DJANGO_SECRET_KEY', ')

```
SECRET_KEY = os.environ.get('DJANGO_SECRET_KEY', 'cg#p$g+j9tax!#a3cup@1$8obt2_+&k3q+pmu)5%asj6yjpkag')
```

Get a secret key from https://miniwebtool.com/django-secret-key-generator

Step 2 : CREATE AND INSTALL FILE

2. Go to visual code of the project (KPMB)
   - create new file Procfile

Django project name

```
Procfile
web: gunicorn kpmb.wsgi --log-file -
```

Type web(got 1 whitespace) : (got 1whitespace) gunicorn kpmb.wsgi - -log-file -

3. Go to command prompt to install gunicorn
   - Type pip3 install gunicorn

```
C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>pip3 install gunicorn
Requirement already satisfied: gunicorn in c:\users\sk216988\appdata\local\programs\python\python310\lib\site-packages (
20.1.0)
Requirement already satisfied: setuptools>=3.0 in c:\users\sk216988\appdata\local\programs\python\python310\lib\site-pac
kages (from gunicorn) (63.2.0)

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>
```

Step 3 : Installation for database

1. Go to command line
2. Type pip3 install dj-database-url
   - dj-database-url is used to extract the Django database configuration from an environment variable.

3. Go visual code update setting

```
from pathlib import Path
import os
import dj_database_url
```

```
C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>pip3
Collecting dj-database-url
  Downloading dj_database_url-1.0.0-py3-none-any.whl (6.6 kB)
Requirement already satisfied: Django>3.2 in c:\users\sk216988
 (from dj-database-url) (4.0.5)
Requirement already satisfied: asgiref<4,>=3.4.1 in c:\users\s
ackages (from Django>3.2->dj-database-url) (3.5.2)
Requirement already satisfied: sqlparse>=0.2.2 in c:\users\sk2
kages (from Django>3.2->dj-database-url) (0.4.2)
Requirement already satisfied: tzdata in c:\users\sk216988\app
om Django>3.2->dj-database-url) (2022.1)
Installing collected packages: dj-database-url
Successfully installed dj-database-url-1.0.0

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: python.exe -m pip install --upgrade p

C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>
```

```
# Update database configuration from $DATABASE_URL.
db_from_env = dj_database_url.config(conn_max_age=500)
DATABASES['default'].update(db_from_env)
```

→ Code that show that the database will be from pythonanywhere

Step 3 : Installation for database

4. Go to command line
5. Type pip3 install psycopg2-binary
   - Django needs psycopg2 to work with Postgres databases.
   - Install it locally so that it becomes part of our requirements for Heroku to set up on the remote server:

```
C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>pip3 install psycopg2-binary
Collecting psycopg2-binary
  Downloading psycopg2_binary-2.9.5-cp310-cp310-win_amd64.whl (1.2 MB)
     ---------------------------------------- 1.2/1.2 MB 527.2 kB/s eta 0:00:00
Installing collected packages: psycopg2-binary
Successfully installed psycopg2-binary-2.9.5

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>_
```

Step 4 : Serving static files in production

1. Go to visual code setting
2. Replace STATIC_URL with

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/s

# The absolute path to the directory where collectstatic will collect static files
STATIC_ROOT = BASE_DIR / 'staticfiles'

# The URL to use when referring to static files (where they will be served from)
STATIC_URL = '/static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Step 5 : Serving static files in production

1. Go to commad prompt Install whitenoise
   - Whitenoise for management of static file

```
C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>pip3 install whitenoise
Collecting whitenoise
  Downloading whitenoise-6.2.0-py3-none-any.whl (19 kB)
Installing collected packages: whitenoise
Successfully installed whitenoise-6.2.0

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>
```

2. Go to visual code – setting add withnose in MiDDLEWARE

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
]
```

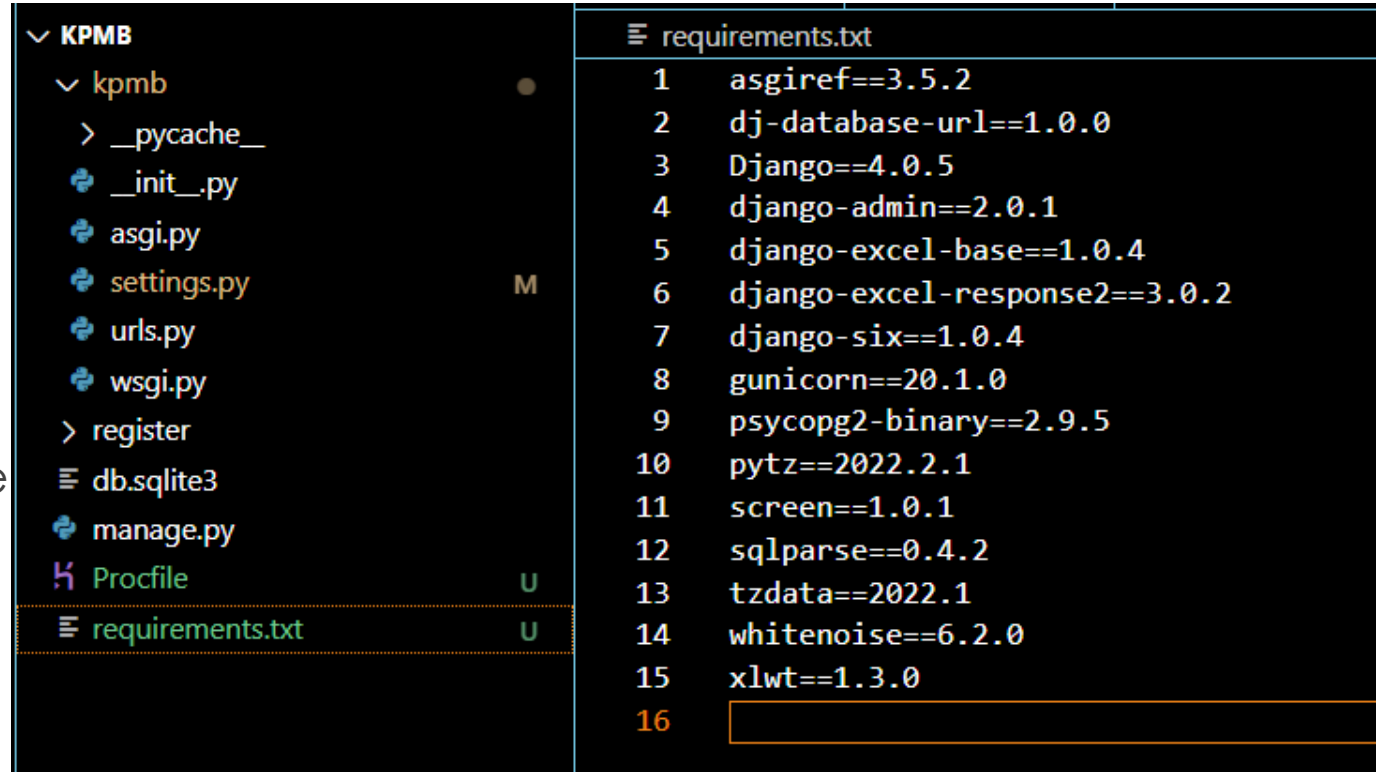
Step 6 : Requirement.txt

1. The Python requirements of your web application must be stored in a file **requirements.txt** in the root of our repository. Heroku will then install these automatically when it rebuilds your environment.

2. Go to command prompt
   - Type pip3 freeze > requirements.txt

```
oject\project wad 4\kpmb>pip3 freeze > requirements.txt

oject\project wad 4\kpmb>
```

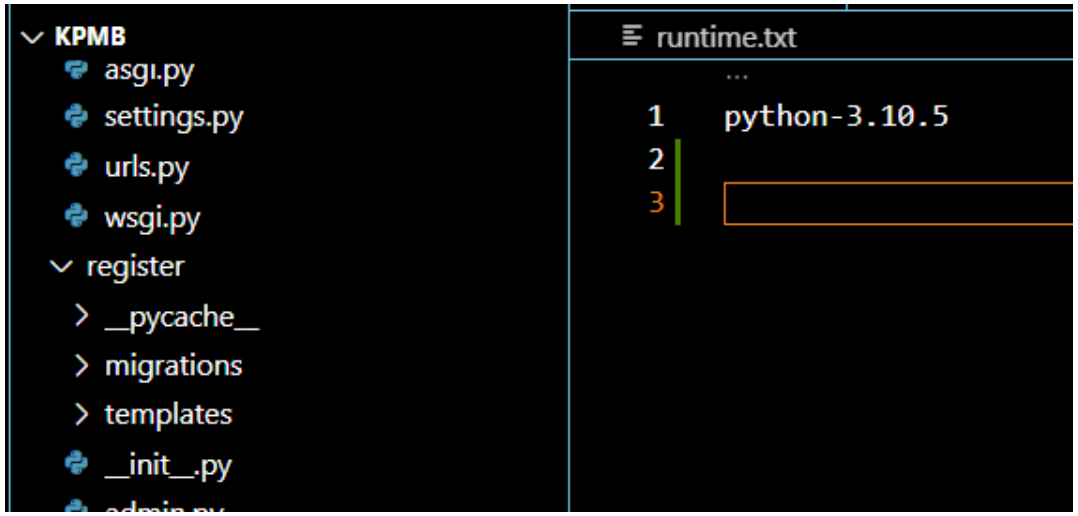
3. Result – it will generate requirements.txt file



Step 7 : runtime.txt

1. The runtime.txt file, if defined, tells Heroku which version of Python to use.
2. Create the file in KPMB name runtime.txt:
3. Edit runtime.txt
   - Type your python version

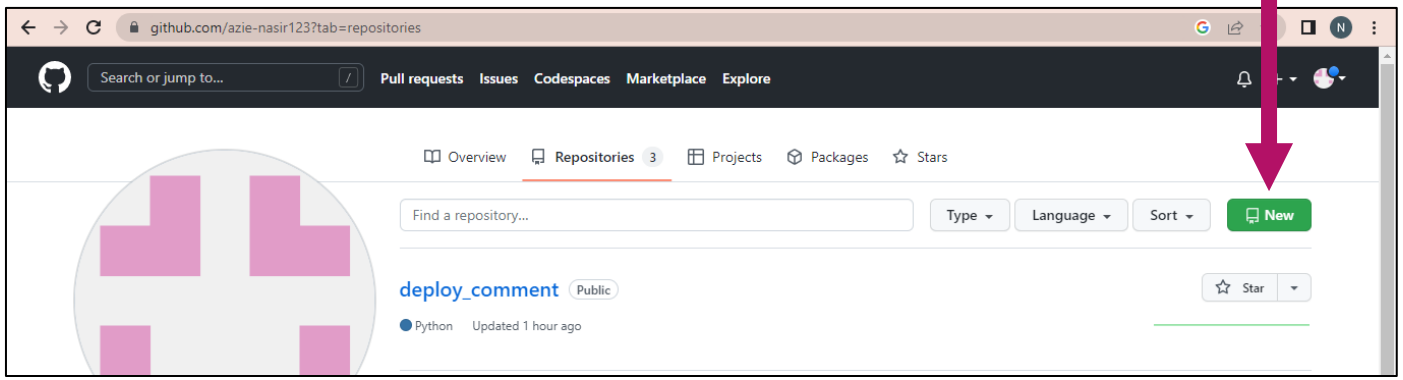
```
C:\Users\SK216988\Desktop\Wad project\project wad 4\kpmb>python --version
Python 3.10.5
```

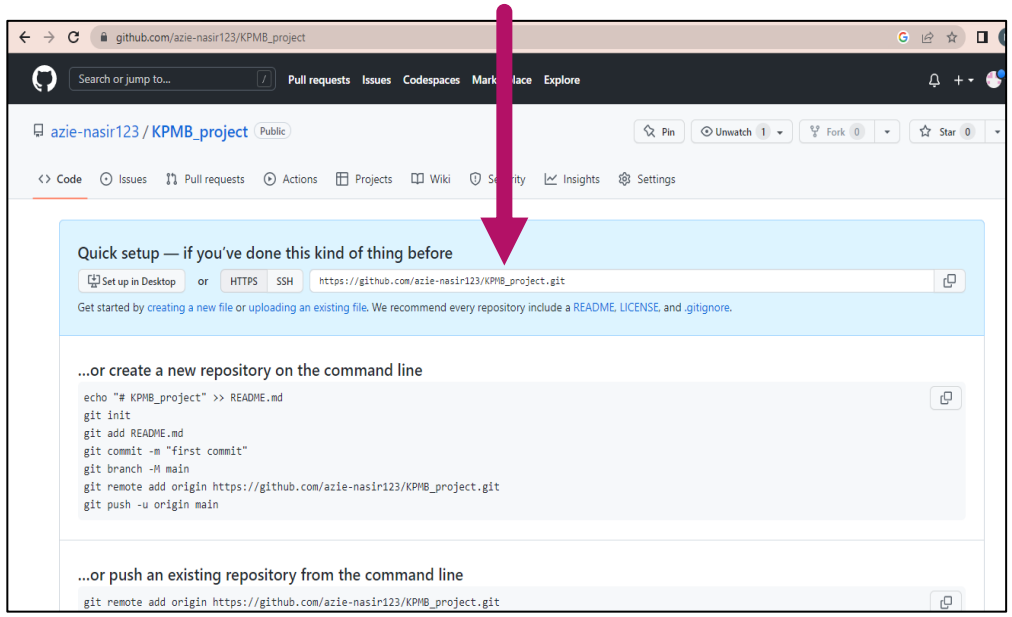


UPLOAD TO GITHUB

UPLOAD TO GITHUB

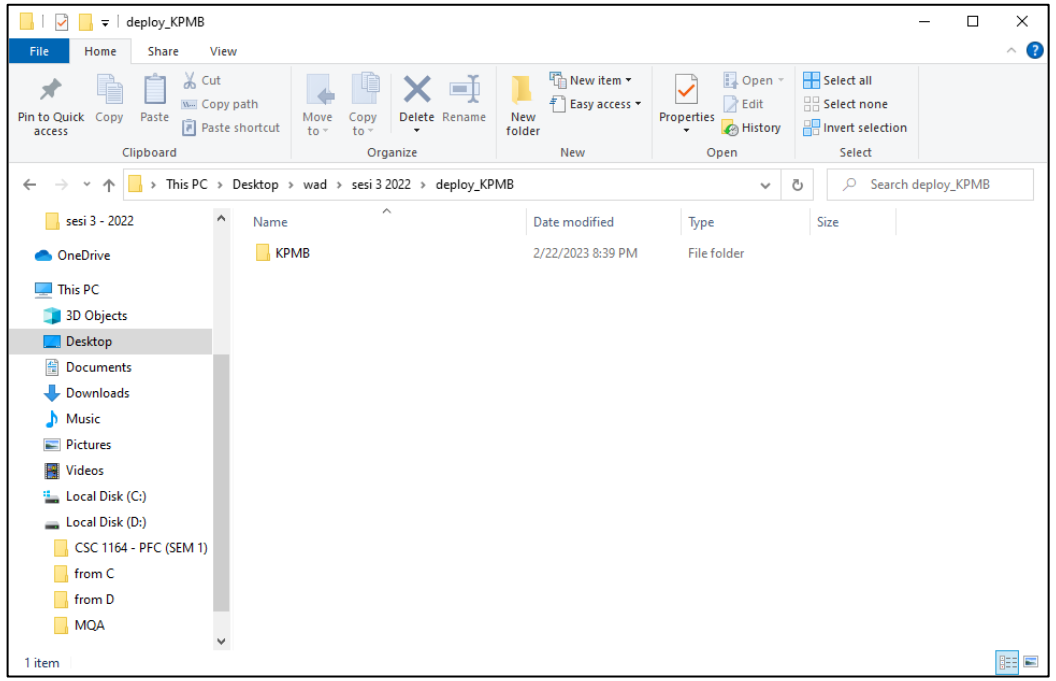
1. Go to your github account
   - Go to Repository
   - Click button NEW
   - Set Repository name = KPMB_project
   - Click create repository

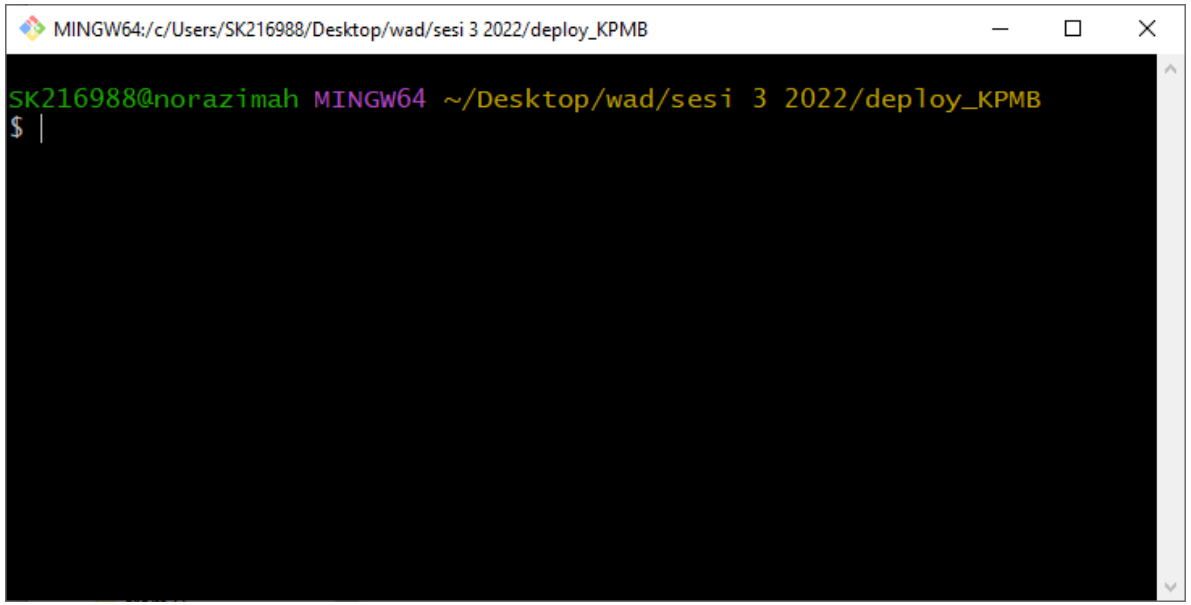




UPLOAD TO GITHUB

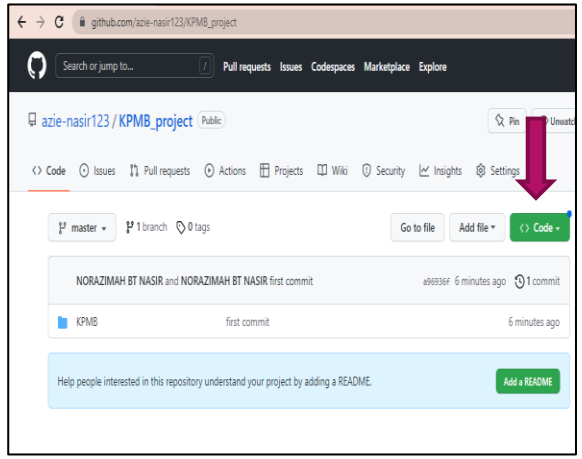
2. Go to your KPMB project folder
   - Right click open Git Bash





UPLOAD TO GITHUB

3. In Git Bash do the following steps: (this is the step of uploading your project to github, make sure you sign in/log in through pop window appear from git bash. This pop up window will connect the git bash to www.github.com

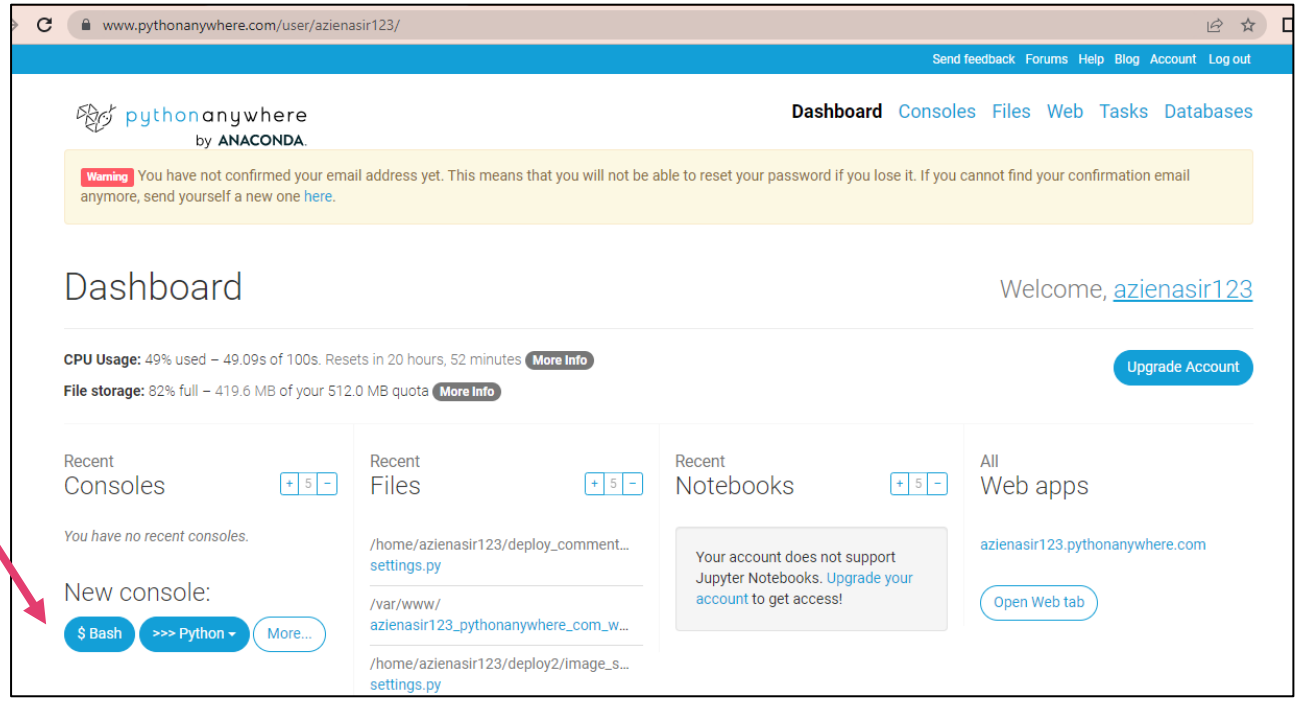


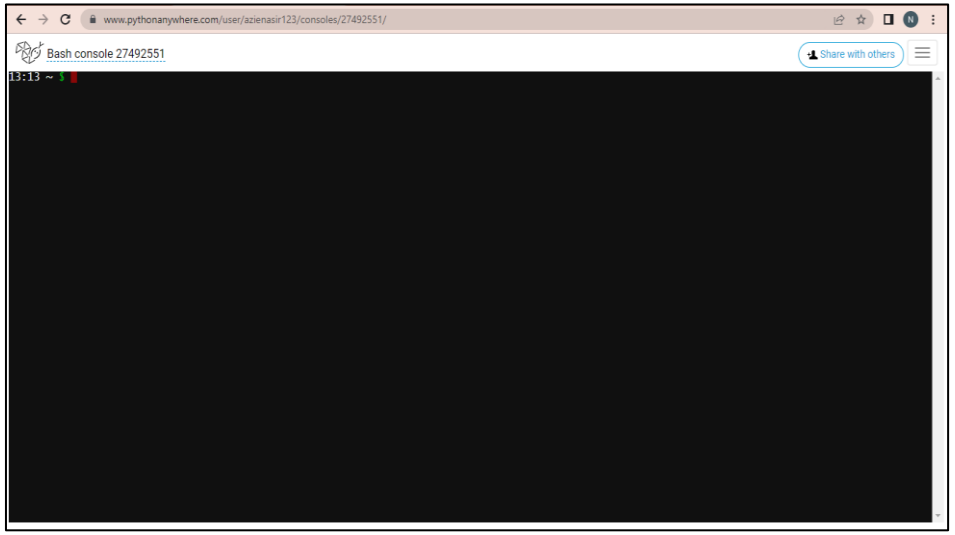
   1. Type git init then enter
   2. Type git add . , then enter
   3. Type git commit -m "first commit" then enter
   4. Copy git remove from the githup repository to store all your file
      - git remote add origin https://github.com/azie-nasir123/KPMB_project.git
   5. The push all file to the repository
      - git push -u origin master
   6. Refresh your github page

UPLOAD TO pythonanywhere

UPLOAD TO PHYTONANYWHERE

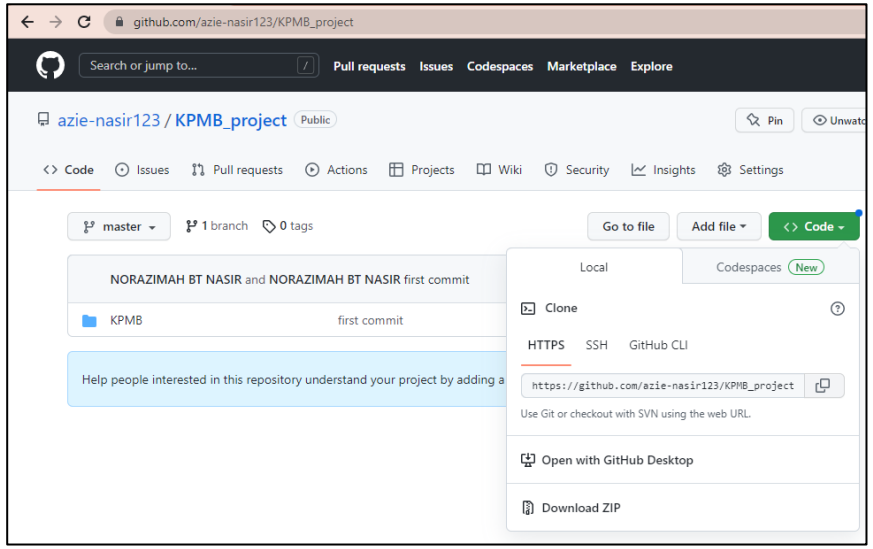
1. Go to your pythonanywhere account
   - On Dashboard
   - Click Bash in new console

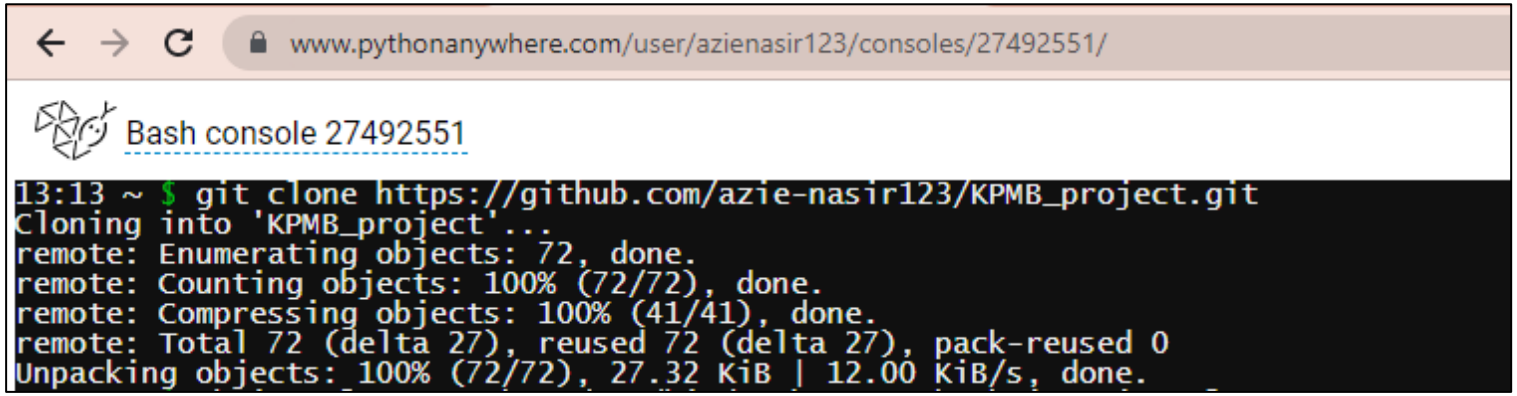




UPLOAD TO PHYTONANYWHERE

2. In Bash Console do the following steps:
   1. Copy address of your project in git hub
   2. Type git clone past the address
      - git clone https://github.com/azie-nasir123/KPMB_project.git





UPLOAD TO

2. In Batch Consoledo the following steps:
   3. Create a virtualenv
      - Type

mkvirtualenv - -python=/usr/bin/python3.10 kpmbsite-virtualenv

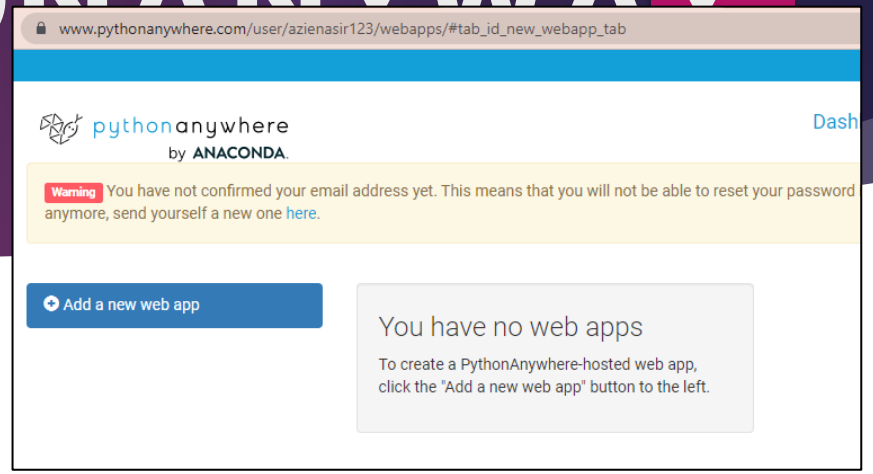
      - Type pip install django
      - Type ls (LS small caps)
      - Go to your folder (cd) until you see the file for manage.py

www.pythonanywhere.com/user/azienasir123/consoles/27493436/

Bash console 27493436

```
14:16 ~ $ git clone https://github.com/azie-nasir123/KPMB_project.git
Cloning into 'KPMB_project'...
remote: Enumerating objects: 72, done.
remote: Counting objects: 100% (72/72), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 72 (delta 27), reused 72 (delta 27), pack-reused 0
Unpacking objects: 100% (72/72), 27.32 KiB | 21.00 KiB/s, done.
14:16 ~ $ mkvirtualenv --python=/usr/bin/python3.10 myKPMB-virtualenv
created virtual environment CPython3.10.5.final.0-64 in 6867ms
  creator CPython3Posix(dest=/home/azienasir123/.virtualenvs/myKPMB-virtualenv, clear=Fa
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=co
    added seed packages: pip==22.3.1, setuptools==66.1.1, wheel==0.38.4
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActi
virtualenvwrapper.user_scripts creating /home/azienasir123/.virtualenvs/myKPMB-virtualer
virtualenvwrapper.user_scripts creating /home/azienasir123/.virtualenvs/myKPMB-virtualer
virtualenvwrapper.user_scripts creating /home/azienasir123/.virtualenvs/myKPMB-virtualer
virtualenvwrapper.user_scripts creating /home/azienasir123/.virtualenvs/myKPMB-virtualer
virtualenvwrapper.user_scripts creating /home/azienasir123/.virtualenvs/myKPMB-virtualer
(myKPMB-virtualenv) 14:17 ~ $ pip install django
Looking in links: /usr/share/pip-wheels
Collecting django
  Using cached Django-4.1.7-py3-none-any.whl (8.1 MB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.3-py3-none-any.whl (42 kB)
Collecting asgiref<4,>=3.5.2
  Using cached asgiref-3.6.0-py3-none-any.whl (23 kB)
Installing collected packages: sqlparse, asgiref, django
Successfully installed asgiref-3.6.0 django-4.1.7 sqlparse-0.4.3
(myKPMB-virtualenv) 14:19 ~ $ ls
KPMB_project  README.txt
(myKPMB-virtualenv) 14:19 ~ $ cd KPMB_project
(myKPMB-virtualenv) 14:20 ~/KPMB_project (master)$ ls
KPMB
(myKPMB-virtualenv) 14:20 ~/KPMB_project (master)$ cd KPMB
(myKPMB-virtualenv) 14:20 ~/KPMB_project/KPMB (master)$ ls
KPMB  Procfile  Registration  db.sqlite3  manage.py  requirements.txt  runtime.txt
(myKPMB-virtualenv) 14:20 ~/KPMB_project/KPMB (master)$
```

UPLOAD TO PHYTONANYWAY

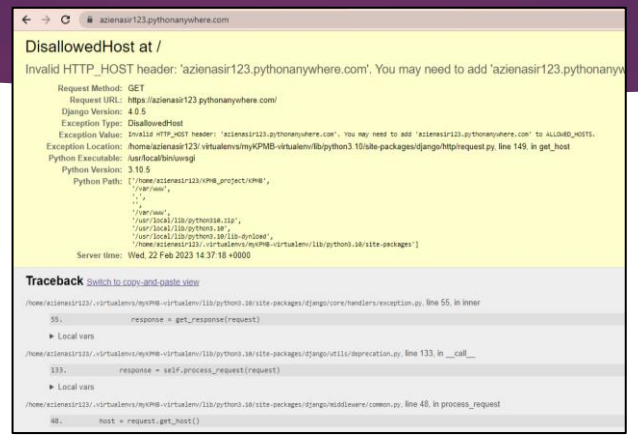


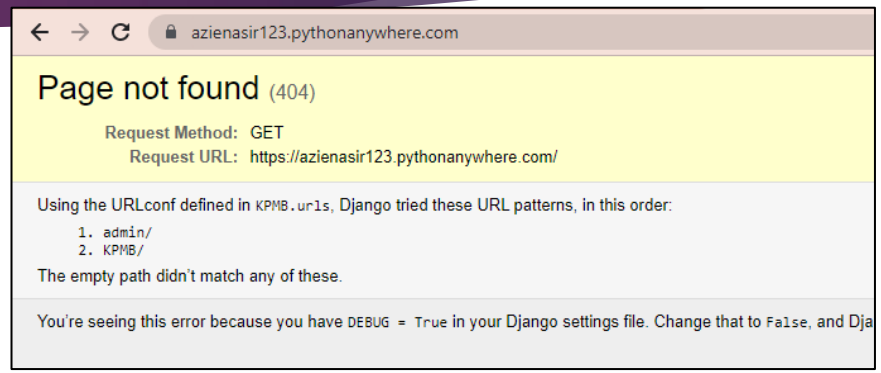
3. Go to WEB to set up Web app and WSGI file
   1. Add new web app
   2. Go to Virtualenv and enter path. Copy your virtualenv in bash (mykpmb-virtualenv) and past in path
   3. Go WSGI configuration file click the link and delete all code
   4. Paste code – edit path and DJANGO_SETTING the save reload

4. Go to Bash
   - Type pip install -r requirements.txt
   - Type python manage.py createsuperuser
   - Type python manage.py migrate

```
# +++++++++++ DJANGO +++++++++++
# To use your own Django app use code like this:
import os
import sys

# assuming your Django settings file is at '/home/myusername/mysite/mysite/settings.py'
path = '/home/myusername/mysite'
if path not in sys.path:
    sys.path.insert(0, path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

## Uncomment the lines below depending on your Django version
###### then, for Django >=1.5:
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
###### or, for older Django <=1.4
#import django.core.handlers.wsgi
#application = django.core.handlers.wsgi.WSGIHandler()
```

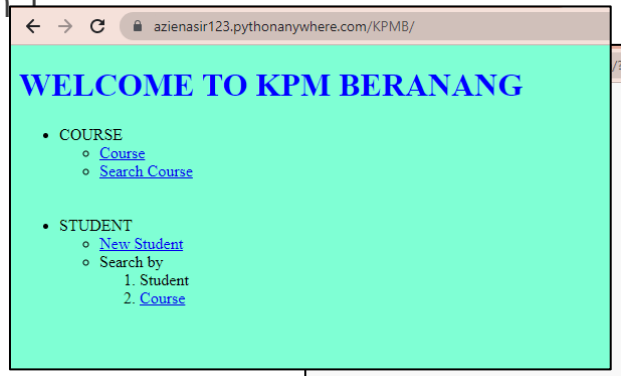
UPLOAD TO PHYTONANYWAY

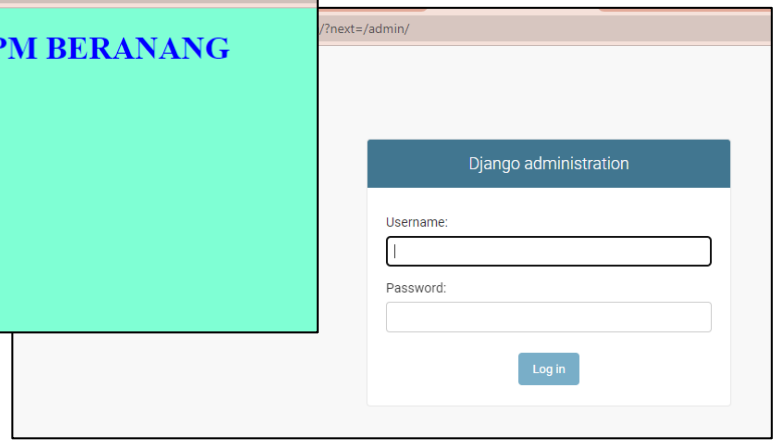




5. Go to WEB
   - Reload and click configure

6. Go to File
   - Find setting – edit ALLOWED_HOSTS = []
   - ALLOWED_HOSTS = ['azienasir123.pythonanywhere.com']
   - Save and reload





7. Type /KPMB/ on the address then enter to access website
8. Type /admin/ on the address then enter to access database
9. Now your website is published to internet and you can access anyway

DEMO VIDEO IS ATTACHED AS EXTERNAL FILE