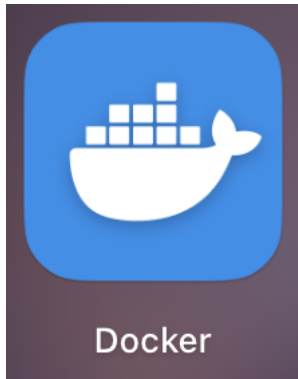


Docker Practical -1

1] <https://www.docker.com/products/docker-desktop/> - download docker desktop for windows/Mac/linux (accordingly)

2] open app.

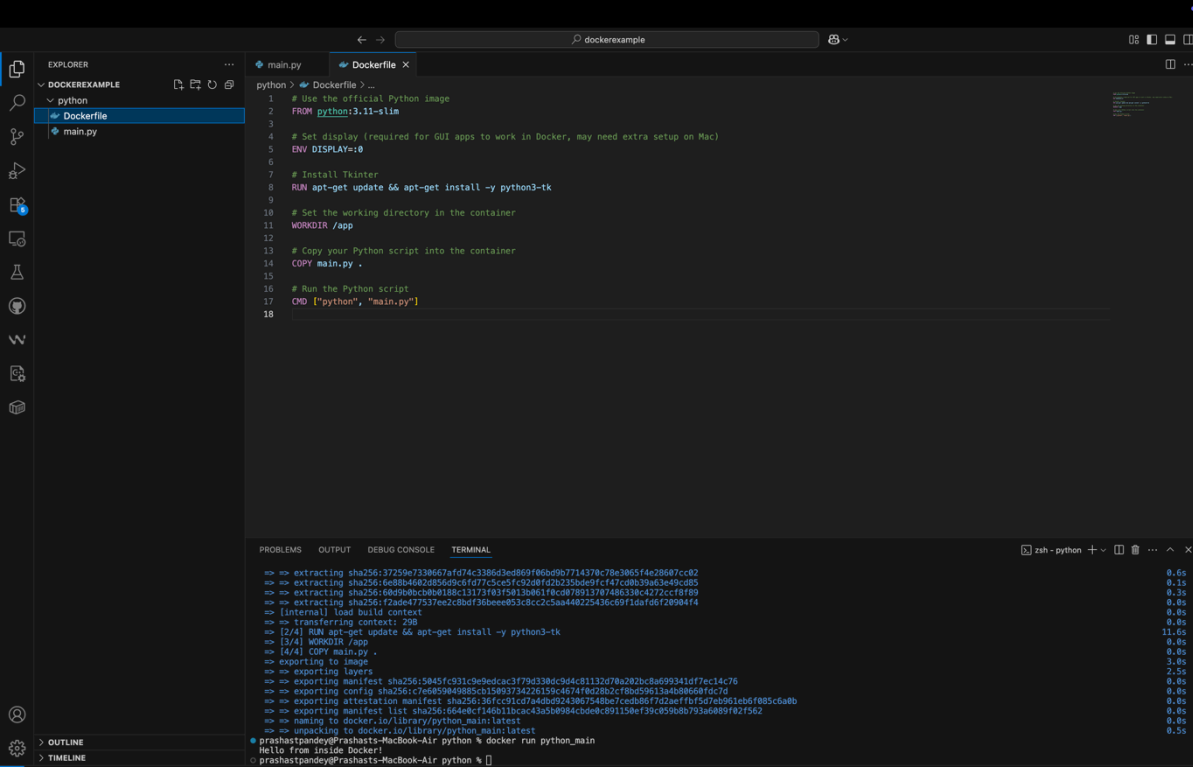


3] Make an example new python file/node application (in VSCode-alongwith Docker and python extension enabling).

```
python > main.py
1 print('Hello from inside Docker!')
2 # This is a simple Python script that will be run inside a Docker container.
3 # It prints a message to the console.
```

```
==> extracting sha256:37259e738667afd74c3386d3ed869f06dd9b7714378c78e3865f4e28687cc82 0.6s
==> extracting sha256:6e88b46828659c6f4d77c5c5f928f62b35bde9cf47cd0b3983e49cd85 0.1s
==> extracting sha256:6e89b0c2b0b188c1317183f913b0e1f4cd878913747486338c372ccf8f99 0.3s
==> extracting sha256:f2a0e477537ee2c8bdf38bee853c8c2c5a440225436c69f1da7d6f20904f4 0.8s
==> [internal] Load build context
==> Transferring context: 298 0.8s
==> [2/4] RUN apt-get update && apt-get install -y python3-tk 11.6s
==> [3/4] WORKDIR /app 0.8s
==> [4/4] COPY main.py . 3.8s
==> exporting to image 2.5s
==> exporting layers 0.8s
==> exporting manifest sha256:5845fc931c9e0edac3779d338dc9d4c81132070a202bc8d699341d7fec14c76 0.8s
==> exporting config sha256:c7a6e994988c015083724226159c4074f628b2cf8b055812e4b0668f0c76 0.8s
==> exporting attestation manifest sha256:36fcc91cd7a4dbd9243807548be7cedb86f7d2aeffbf5d7e961eb6f885c6a0b 0.8s
==> exporting manifest list sha256:664e4cf146b11bc43a5b0984cb0e8c891150ef39c859b0b793a6889f027562 0.8s
==> naming to docker.io/library/python_main:latest 0.8s
==> unpacking to docker.io/library/python_main:latest 0.5s
prashastpandey@Prashast-MacBook-Air python % docker run python_main
Hello from inside Docker!
prashastpandey@Prashast-MacBook-Air python %
```

4] Create a new file named 'Dockerfile' inside the same folder or working directory.



The screenshot shows a Visual Studio Code editor with a project named 'DOCKEREXAMPLE'. The Explorer sidebar on the left shows a file structure with 'python' as a subdirectory, containing 'Dockerfile' and 'main.py'. The main editor window displays the content of 'Dockerfile', which is a Dockerfile for a Python application. The Dockerfile includes instructions to use the official Python image, set the working directory to /app, copy the main.py file, and run the Python script. The bottom panel shows the 'TERMINAL' output, which displays the Docker build process, including the extraction of the Python image, the installation of dependencies, and the successful execution of the Python script, resulting in the output 'Hello from inside Docker!'.

```
python > Dockerfile x
1 # Use the official Python image
2 FROM python:3.11-slim
3
4 # Set display (required for GUI apps to work in Docker, may need extra setup on Mac)
5 ENV DISPLAY=:0
6
7 # Install Tkinter
8 RUN apt-get update && apt-get install -y python3-tk
9
10 # Set the working directory in the container
11 WORKDIR /app
12
13 # Copy your Python script into the container
14 COPY main.py .
15
16 # Run the Python script
17 CMD ["python", "main.py"]
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
prashastpandey@Prashasto-MacBook-Air python % docker run python_main
Hello from inside Docker!
```

Dockerfile CODE:

Use the official Python image

FROM python:3.11-slim

Set display (required for GUI apps to work in Docker, may need extra setup on Mac)

ENV DISPLAY=:0

Install Tkinter

RUN apt-get update && apt-get install -y python3-tk

Set the working directory in the container

WORKDIR /app

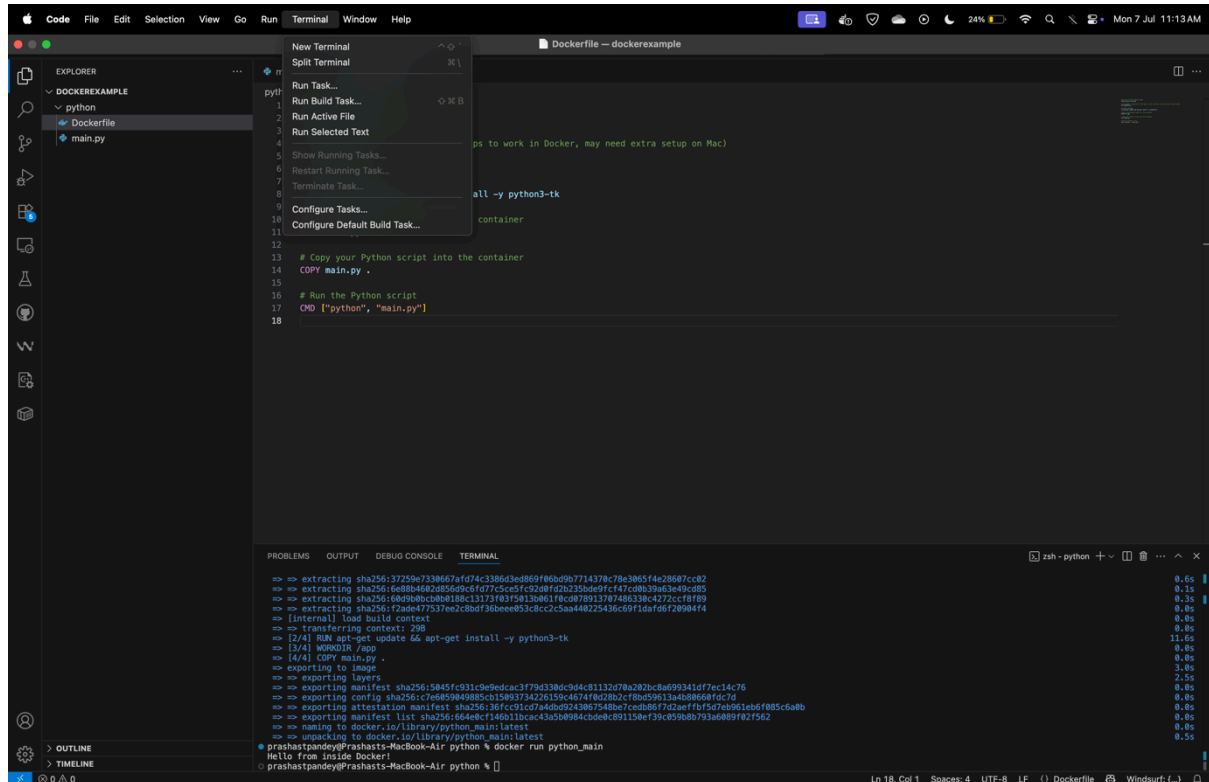
Copy your Python script into the container

COPY DockerDemo.py .

Run the Python script

CMD ["python", "DockerDemo.py"]

5] Then open a new terminal.(by pressing keys **ctrl+~**)



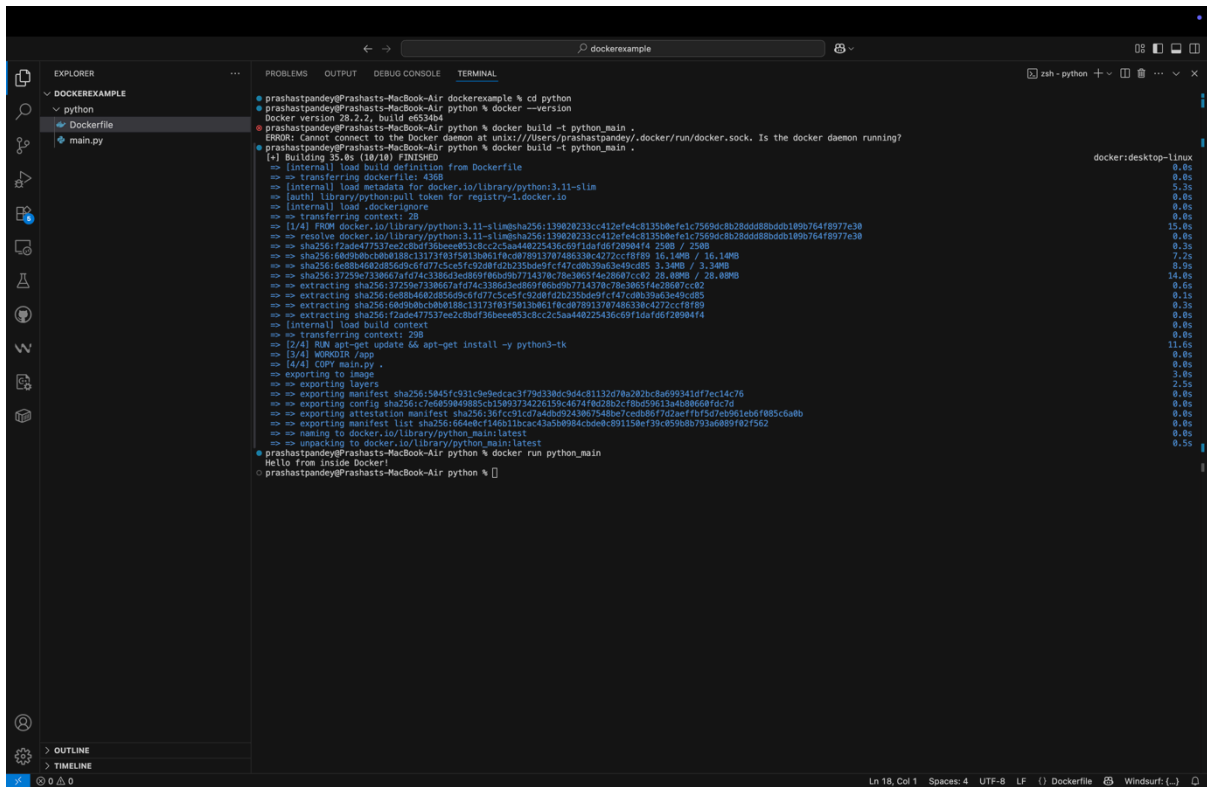
The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project named 'DOCKEREXAMPLE' with files 'python', 'Dockerfile', and 'main.py'. The Dockerfile is open in the editor, showing the following content:

```
1 # Copy your Python script into the container
2 COPY main.py .
3
4 # Run the Python script
5 CMD ["python", "main.py"]
```

The Run and Debug menu is open, showing options like 'New Terminal', 'Split Terminal', 'Run Task...', 'Run Active File', 'Run Selected Text', 'Show Running Tasks...', 'Restart Running Task...', 'Terminate Task...', 'Configure Tasks...', and 'Configure Default Build Task...'. The Terminal panel at the bottom shows the output of the Docker build process, including the extraction of the Docker image and the execution of the Python script inside the container.

```
prashastpandey@Prashasts-MacBook-Air python % docker run python_main
Hello from inside Docker!
```

6] Follow terminal commands to execute the code in container. (NOTE: make sure your docker app is open)



```
prashastpandey@Prashasts-MacBook-Air:~/dockerexample$ cd python
prashastpandey@Prashasts-MacBook-Air:~/python$ docker --version
Docker version 20.10.17, build 100e703
prashastpandey@Prashasts-MacBook-Air:~/python$ docker build -t python_main .
ERROR: Cannot connect to the Docker daemon at unix:///Users/prashastpandey/.docker/run/docker.sock. Is the docker daemon running?
prashastpandey@Prashasts-MacBook-Air:~/python$ docker build -t python_main .
[+] Building 35.0s (18/18) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load metadata for docker.io/library/python:3.11-slim
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load dockerignore
=> [internal] load build definition from Dockerfile
=> [1/4] FROM docker.io/library/python:3.11-slim@sha256:139020233cc412efec8135b0efc7569dc8b28add880db189b764f8977e39
=> resolve docker.io/library/python:3.11-slim@sha256:139020233cc412efec8135b0efc7569dc8b28add880db189b764f8977e39
=> sha256:f2ade477537ec2c8d8f36e0e053c8cc2c5a44022543c60f1da6d720984f4 25MB / 25MB
=> sha256:6e88b4682085d9c9f77c5c5f9c9208f42b235bde9fcf47c0b39a63e49cd85 3.34MB / 3.34MB
=> sha256:6e88b4682085d9c9f77c5c5f9c9208f42b235bde9fcf47c0b39a63e49cd85 3.34MB / 3.34MB
=> sha256:37259e7338667af4c3386d3e0869f06b09b7714370c78e3065f4e20807cc82 28.80MB / 28.80MB
=> extracting sha256:37259e7338667af4c3386d3e0869f06b09b7714370c78e3065f4e20807cc82
=> extracting sha256:6e88b4682085d9c9f77c5c5f9c9208f42b235bde9fcf47c0b39a63e49cd85
=> extracting sha256:6e88b4682085d9c9f77c5c5f9c9208f42b235bde9fcf47c0b39a63e49cd85
=> extracting sha256:f2ade477537ec2c8d8f36e0e053c8cc2c5a44022543c60f1da6d720984f4
=> [internal] load build context
=> Transferring context: 2MB
=> [2/4] RUN apt-get update && apt-get install -y python3-tk
=> [3/4] WORKDIR /app
=> [4/4] COPY main.py .
=> exporting to image
=> exporting layers
=> exporting manifest sha256:5845fc931c9e0edc3f794338dc9d4c81132070a202bcb8e99341d7fec14c76
=> exporting config sha256:c7e4e890a9088c01509374226155c4074f82b2cf80d50613a408609f0c76
=> exporting attestation manifest sha256:36fcc91c07a4db09243087548be7cedb087702aefbf75d7e961eb6f085c6a0b
=> exporting manifest list sha256:6648cf146b11bcac43a5b094cbde0c891150ef39c0508b793a6089f827562
=> naming to docker.io/library/python_main:latest
=> unpacking to docker.io/library/python_main:latest
Hello from inside Docker!
prashastpandey@Prashasts-MacBook-Air:~/python$ docker run python_main
```

COMMANDS:

cd python: to change into your working directory

docker --version: to see your docker version on your system.

docker build -t <virtual Directory name> .

: build image of your code. NOTE: -t is tagging the build with the 'python_main' name and '.' Symbolises your current working directory I think

docker run <virtual Directory Name>: finally run the container using the tagged name