
# Smart Cities Hackathon
# Tier 3 Challenge 9

# Tier 3: Challenge 9

**Tier 3  Challenge 9: Python Docker and Scripting Challenge**

This challenge will see you set up your environment for Python and create a Python script to retrieve the latest reading for a sensor of your choosing.

If you want to run the Python locally you can jump to Step 4. Otherwise, if you want to build as a Docker image, you can follow the following steps
https://code.visualstudio.com/docs/languages/python

https://code.visualstudio.com/docs/python/python-tutorial

Step 1 - if you have already installed docker, skip to Step 2

Go to the official Docker website: https://www.docker.com/products/docker-desktop
Download Docker Desktop for your operating system (Windows, macOS, or Linux).
Follow the installation instructions provided for your operating system, making sure to install docker app repository - this contains the programme engine required to run commands.

Open a terminal or command prompt and start up the docker engine. You can ask your AI Assistant how to do this for your operating system, as well as how to configure your system to start docker on boot.

Verify installation & start of docker engine;

```
sudo docker run hello-world
```

Step 2

Create a new folder for your Tier 3 challenges and a subfolder for this challenge.

Step 3

Create a **Dockerfile** and add the following elements.

```
# Use an official Python runtime as a parent image
FROM python:3.10-slim

# Set the working directory in the container
WORKDIR /app

# Copy the requirements file into the container at /app
COPY requirements.txt /app/

# Install any needed dependencies specified in requirements.txt
```

```
RUN pip install --no-cache-dir -r requirements.txt

# Copy all files from the host's current into the container's /app folder
COPY . /app/
```

Step 4

Next, create a **requirements.txt file** and add the following contents.

```
matplotlib
requests
```

Step 5

Create a file called hello.py and add the following code.

```
# hello.py

def main():
   print("Hello, world!")

if __name__ == "__main__":
   main()
```

Step 6

Create a file called **compose.yml** and add the following contents.

```
services:
 python-app:
   build:
     context: .
     dockerfile: Dockerfile
   ports:
     - "80:80"
   volumes:
     - .:/app
```

Step 7

Open a command prompt and navigate to the challenge directory
Run the following command:

```
sudo docker compose run python-app python hello.py
```

This will build the Docker file initially. You should see the following output:

**Hello, world!**

Step 8

Log into https://core.aql.com/login and create a bearer token if you haven't created one from an earlier challenge. Copy this as you will need it for one of the following steps.

Step 9

From the sensors list, choose a series of sensor IDs - we will need these IDs for the coding element. IDs are garnered from the sensor webpage URL, similar to device IDs.

Step 10

Within your directory, create a new Python file called challenge9.py.

Step 11

Enter the code below, then replace the token and IDs with the chosen values from the earlier steps.

```
import requests

# Constants
CORE_URL = 'https://api.core.aql.com/v1/'
TOKEN = 'YOUR_TOKEN_ID'
SENSOR_IDS = [
   "REPLACE_WITH_SENSOR_ID",
   "REPLACE_WITH_SENSOR_ID"
]
```

Step 12

Let's add the loop to handle each sensor.

```
# Get data from each station
for sensor_id in SENSOR_IDS:
```

```
    print("Getting data from", sensor_id)
```

Run the challenge using the docker command to check that it is working:

```
sudo docker compose run python-app python challenge9.py
```

You should see the sensor IDs echoed to the console output.

Step 13

Add a definition to query https://api.core.aql.com/doc/#api-Sensor-Sensor_data/latest with the bearer token created earlier.

```
def get_reading(sensor_id):
    url = f"{CORE_URL}sensors/{sensor_id}/sensor-data/latest"
    headers = {'Authorization': f'Bearer {TOKEN}'}
    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.HTTPError as e:
    print(e)
```

Add the following updates to the initial loop:

```
reading = get_reading(sensor_id);
print(reading)
```

Run the code using:

```
sudo docker compose run python-app python challenge9.py
```

The latest readings should be displayed to the screen.

You have now completed this challenge. You could extend this challenge by calling https://api.core.aql.com/doc/#api-Sensor-Sensor_data/aggregate and returning a range of data and then plotting this to a web output.