## Task 2

Question:
## Task 2
Start the server-side applications (see "Running apps" below), then execute file `client.py`. You should see an output `Minimum Python version: ~=x.y` with some values of `x` and `y`.

Write at least two basic end-to-end tests that cover some aspects of client-server interaction.

Write at least two basic unit tests: one for testing the server app, one for testing the client app. Unit tests may cover any aspect of any part of code, small or big.

Answer: Please see the pdf document named **"Task 3"** for the instructions to execute the **client.py** file and run the **unit** and **E2E** (Integration) tests.

This pdf document named **"Task 2"** explains by giving you the summary regarding the changes are done to the existing code (**main.py and client.py**). For the unit and E2E tests, please see the repository shared with you.

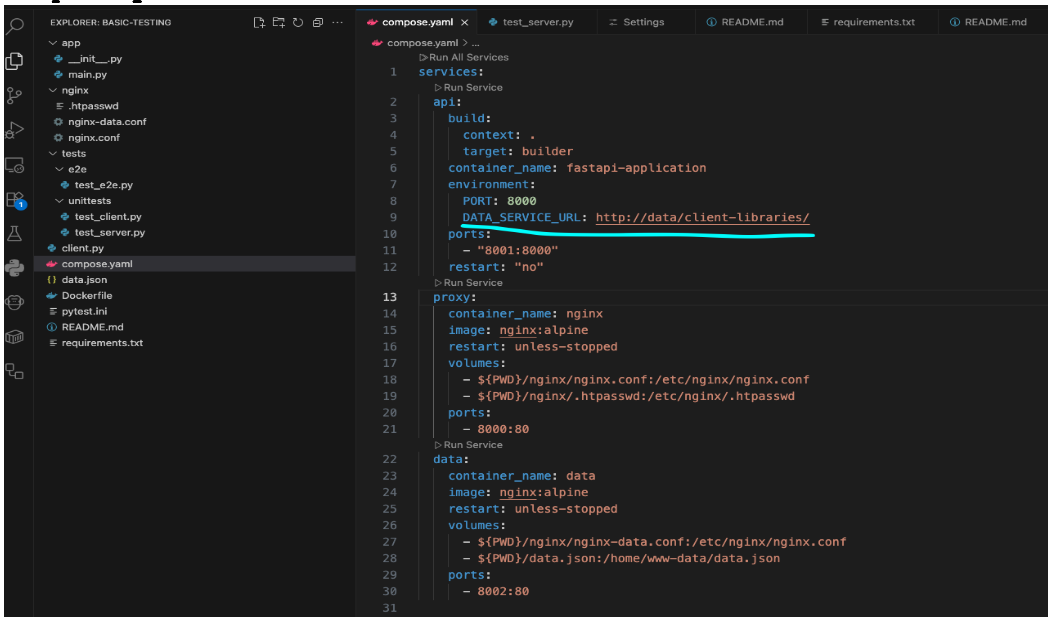
□ **In main.py** (FastAPI Server – http://localhost:8000/)

✓Added the environment variable in "compose.yaml" file and "main.py":

Inside (`compose.yaml file`), Added the environment variable:

```
environment:
  - DATA_SERVICE_URL=http://data/client-libraries/
```

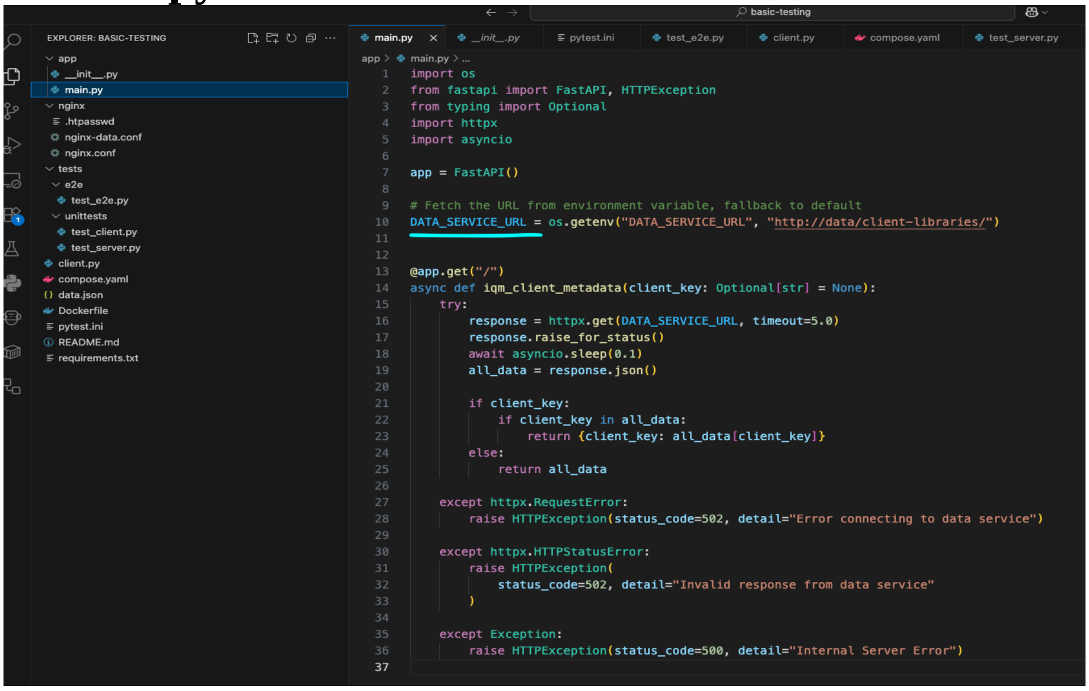
`compose.yaml file`



```yaml
services:
  api:
    build:
      context: .
      target: builder
    container_name: fastapi-application
    environment:
      PORT: 8000
      DATA_SERVICE_URL: http://data/client-libraries/
    ports:
      - "8001:8000"
    restart: "no"
  proxy:
    container_name: nginx
    image: nginx:alpine
    restart: unless-stopped
    volumes:
      - ${PWD}/nginx/nginx.conf:/etc/nginx/nginx.conf
      - ${PWD}/nginx/.htpasswd:/etc/nginx/.htpasswd
    ports:
      - 8000:80
  data:
    container_name: data
    image: nginx:alpine
    restart: unless-stopped
    volumes:
      - ${PWD}/nginx/nginx-data.conf:/etc/nginx/nginx.conf
      - ${PWD}/data.json:/home/www-data/data.json
    ports:
      - 8002:80

```

Now, “**main.py**” looks like this below:

✓main.py



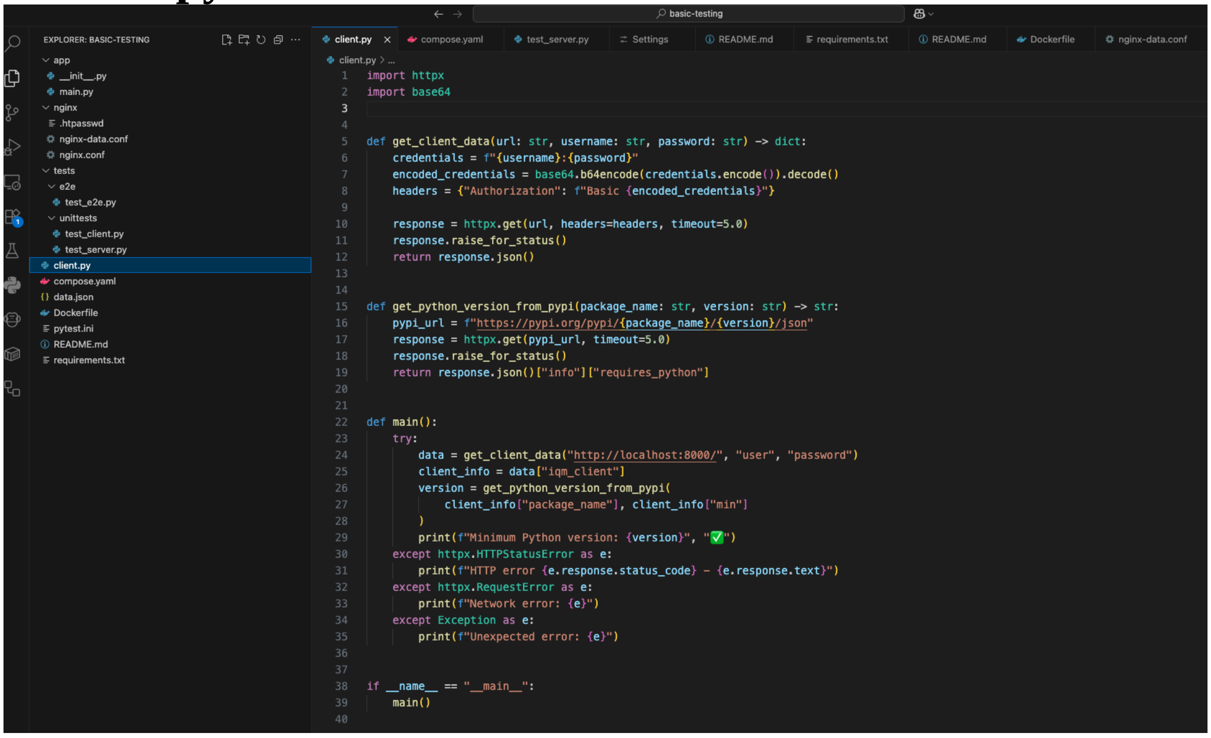
```python
import os
from fastapi import FastAPI, HTTPException
from typing import Optional
import httpx
import asyncio

app = FastAPI()

# Fetch the URL from environment variable, fallback to default
DATA_SERVICE_URL = os.getenv("DATA_SERVICE_URL", "http://data/client-libraries/")


@app.get("/")
async def iqm_client_metadata(client_key: Optional[str] = None):
    try:
        response = httpx.get(DATA_SERVICE_URL, timeout=5.0)
        response.raise_for_status()
        await asyncio.sleep(0.1)
        all_data = response.json()

        if client_key:
            if client_key in all_data:
                return {client_key: all_data[client_key]}
        else:
            return all_data

    except httpx.RequestError:
        raise HTTPException(status_code=502, detail="Error connecting to data service")

    except httpx.HTTPStatusError:
        raise HTTPException(
            status_code=502, detail="Invalid response from data service"
        )

    except Exception:
        raise HTTPException(status_code=500, detail="Internal Server Error")

```

✅Project Structure: (Please see the README.md file for project structure)

```
.
├── app
│   ├── __init__.py
│   └── main.py
│
├── client.py
├── compose.yaml
├── data.json
├── Dockerfile
├── pytest.ini
├── README.md
├── requirements.txt
│
├── nginx
│   ├── .htpasswd
│   ├── nginx.conf
│   └── nginx-data.conf
│
├── tests
│   ├── e2e.    --→ (E2E or Integration tests Folder)
│   │   └── test_e2e.py -→ (E2E or Integration tests)
│   └── unittests -→ (Unit tests folder)
│       ├── test_client.py -→ (Unit test for client.py)
│       └── test_server.py--→ (Unit test for FastAPI server (main.py))
│
├── __pycache__/
├── .pytest_cache/
├── .vscode/
```

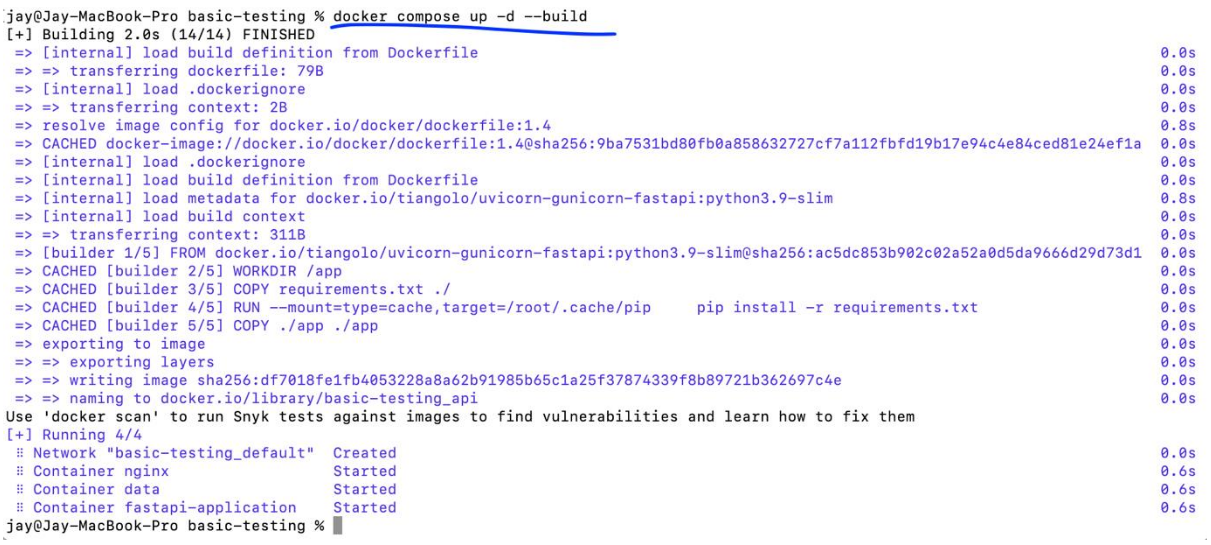
Now, “**client.py**” looks like this below:

✓client.py



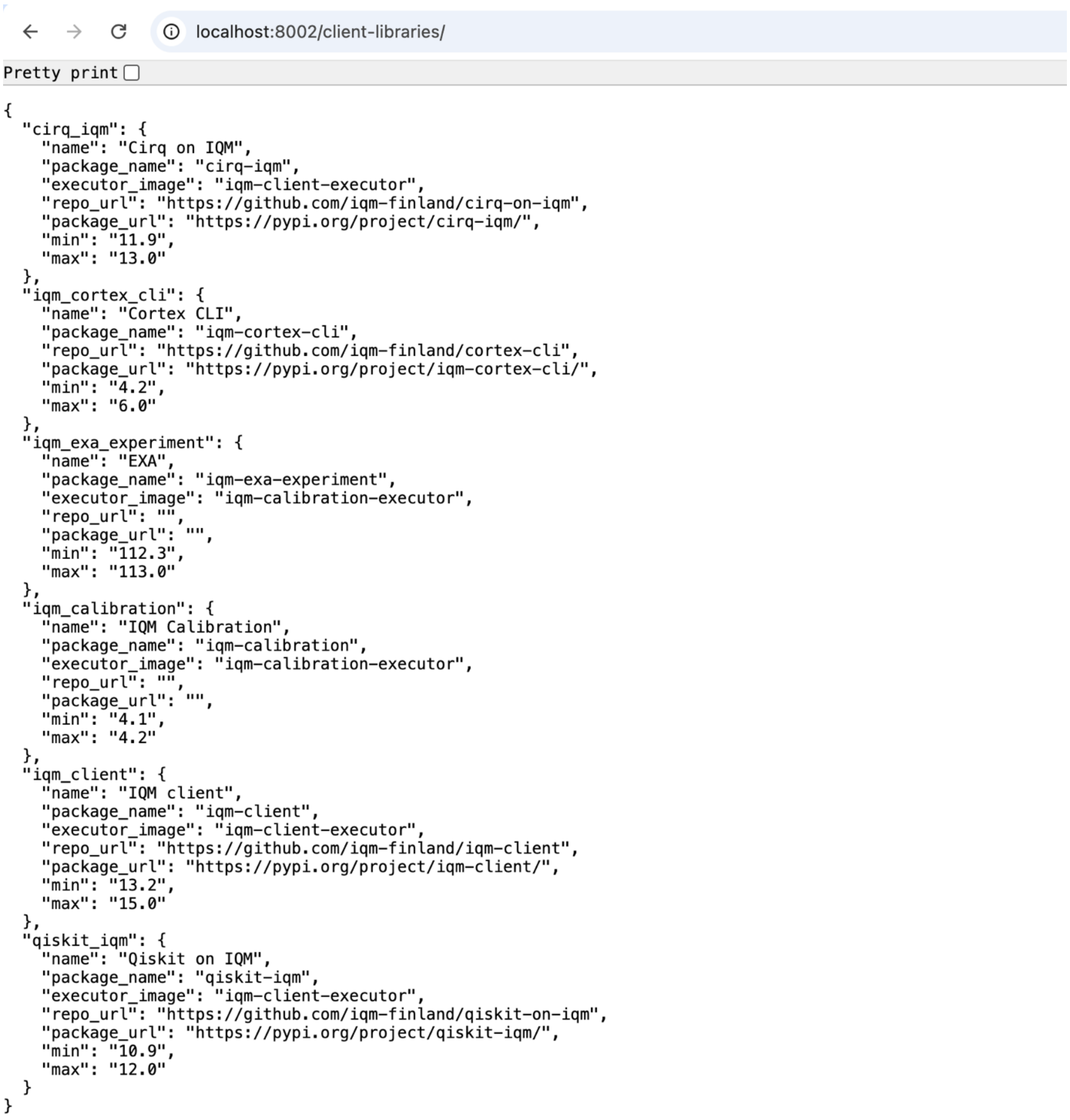
```python
import httpx
import base64


def get_client_data(url: str, username: str, password: str) -> dict:
    credentials = f"{username}:{password}"
    encoded_credentials = base64.b64encode(credentials.encode()).decode()
    headers = {"Authorization": f"Basic {encoded_credentials}"}

    response = httpx.get(url, headers=headers, timeout=5.0)
    response.raise_for_status()
    return response.json()


def get_python_version_from_pypi(package_name: str, version: str) -> str:
    pypi_url = f"https://pypi.org/pypi/{package_name}/{version}/json"
    response = httpx.get(pypi_url, timeout=5.0)
    response.raise_for_status()
    return response.json()["info"]["requires_python"]


def main():
    try:
        data = get_client_data("http://localhost:8000/", "user", "password")
        client_info = data["iqm_client"]
        version = get_python_version_from_pypi(
            client_info["package_name"], client_info["min"]
        )
        print(f"Minimum Python version: {version}", "✅")
    except httpx.HTTPStatusError as e:
        print(f"HTTP error {e.response.status_code} - {e.response.text}")
    except httpx.RequestError as e:
        print(f"Network error: {e}")
    except Exception as e:
        print(f"Unexpected error: {e}")


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

When you run your **client.py** file (Steps to run are mentioned in **“Task3”** Pdf document). Please make sure your **FastAPI server** is up and running and to do that follow below:

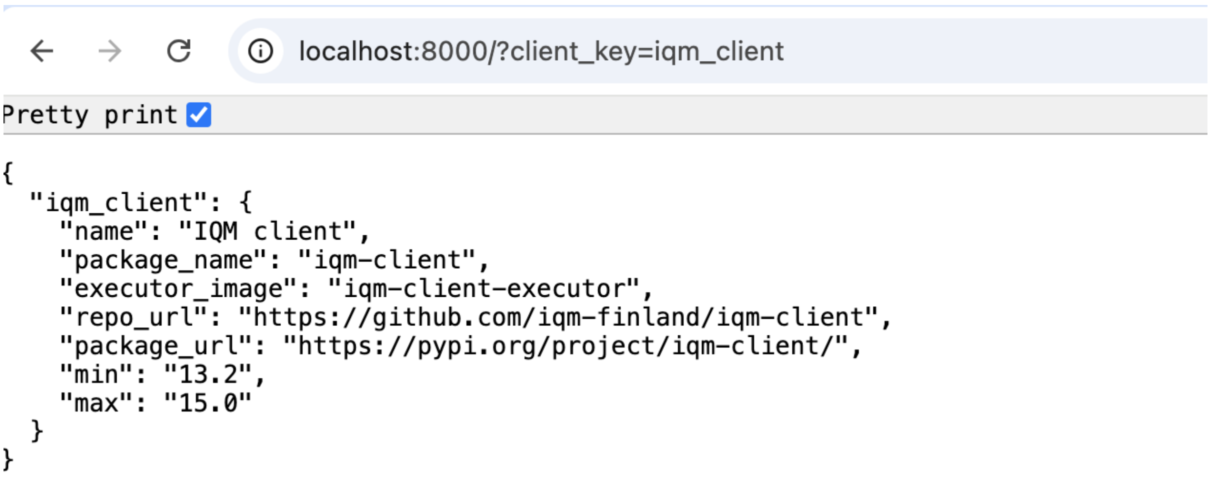


```
jay@Jay-MacBook-Pro basic-testing % docker compose up -d --build
[+] Building 2.0s (14/14) FINISHED
 => [internal] load build definition from Dockerfile                                                   0.0s
 => => transferring dockerfile: 79B                                                                    0.0s
 => [internal] load .dockerignore                                                                      0.0s
 => => transferring context: 2B                                                                        0.0s
 => resolve image config for docker.io/docker/dockerfile:1.4                                           0.8s
 => CACHED docker-image://docker.io/docker/dockerfile:1.4@sha256:9ba7531bd80fb0a858632727cf7a112fbfd19b17e94c4e84ced81e24ef1a  0.0s
 => [internal] load .dockerignore                                                                      0.0s
 => [internal] load build definition from Dockerfile                                                   0.0s
 => [internal] load metadata for docker.io/tiangolo/uvicorn-gunicorn-fastapi:python3.9-slim            0.8s
 => [internal] load build context                                                                      0.0s
 => => transferring context: 311B                                                                      0.0s
 => [builder 1/5] FROM docker.io/tiangolo/uvicorn-gunicorn-fastapi:python3.9-slim@sha256:ac5dc853b902c02a52a0d5da9666d29d73d1  0.0s
 => CACHED [builder 2/5] WORKDIR /app                                                                  0.0s
 => CACHED [builder 3/5] COPY requirements.txt ./                                                      0.0s
 => CACHED [builder 4/5] RUN --mount=type=cache,target=/root/.cache/pip     pip install -r requirements.txt  0.0s
 => CACHED [builder 5/5] COPY ./app ./app                                                              0.0s
 => exporting to image                                                                                 0.0s
 => => exporting layers                                                                                0.0s
 => => writing image sha256:df7018fe1fb4053228a8a62b91985b65c1a25f37874339f8b89721b362697c4e           0.0s
 => => naming to docker.io/library/basic-testing_api                                                   0.0s
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 4/4
 ⠿ Network "basic-testing_default"  Created                                                            0.0s
 ⠿ Container nginx                  Started                                                            0.6s
 ⠿ Container data                   Started                                                            0.6s
 ⠿ Container fastapi-application    Started                                                            0.6s
jay@Jay-MacBook-Pro basic-testing % 
```

You should see the **localhost** opened on browsers like below:



```
{
  "cirq_iqm": {
    "name": "Cirq on IQM",
    "package_name": "cirq-iqm",
    "executor_image": "iqm-client-executor",
    "repo_url": "https://github.com/iqm-finland/cirq-on-iqm",
    "package_url": "https://pypi.org/project/cirq-iqm/",
    "min": "11.9",
    "max": "13.0"
  },
  "iqm_cortex_cli": {
    "name": "Cortex CLI",
    "package_name": "iqm-cortex-cli",
    "repo_url": "https://github.com/iqm-finland/cortex-cli",
    "package_url": "https://pypi.org/project/iqm-cortex-cli/",
    "min": "4.2",
    "max": "6.0"
  },
  "iqm_exa_experiment": {
    "name": "EXA",
    "package_name": "iqm-exa-experiment",
    "executor_image": "iqm-calibration-executor",
    "repo_url": "",
    "package_url": "",
    "min": "112.3",
    "max": "113.0"
  },
  "iqm_calibration": {
    "name": "IQM Calibration",
    "package_name": "iqm-calibration",
    "executor_image": "iqm-calibration-executor",
    "repo_url": "",
    "package_url": "",
    "min": "4.1",
    "max": "4.2"
  },
  "iqm_client": {
    "name": "IQM client",
    "package_name": "iqm-client",
    "executor_image": "iqm-client-executor",
    "repo_url": "https://github.com/iqm-finland/iqm-client",
    "package_url": "https://pypi.org/project/iqm-client/",
    "min": "13.2",
    "max": "15.0"
  },
  "qiskit_iqm": {
    "name": "Qiskit on IQM",
    "package_name": "qiskit-iqm",
    "executor_image": "iqm-client-executor",
    "repo_url": "https://github.com/iqm-finland/qiskit-on-iqm",
    "package_url": "https://pypi.org/project/qiskit-iqm/",
    "min": "10.9",
    "max": "12.0"
  }
}
```



```
{
  "iqm_client": {
    "name": "IQM client",
    "package_name": "iqm-client",
    "executor_image": "iqm-client-executor",
    "repo_url": "https://github.com/iqm-finland/iqm-client",
    "package_url": "https://pypi.org/project/iqm-client/",
    "min": "13.2",
    "max": "15.0"
  }
}
```