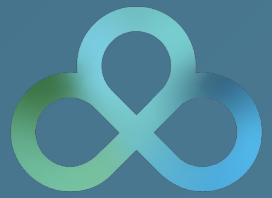


EDB
Postgres for the AI Generation

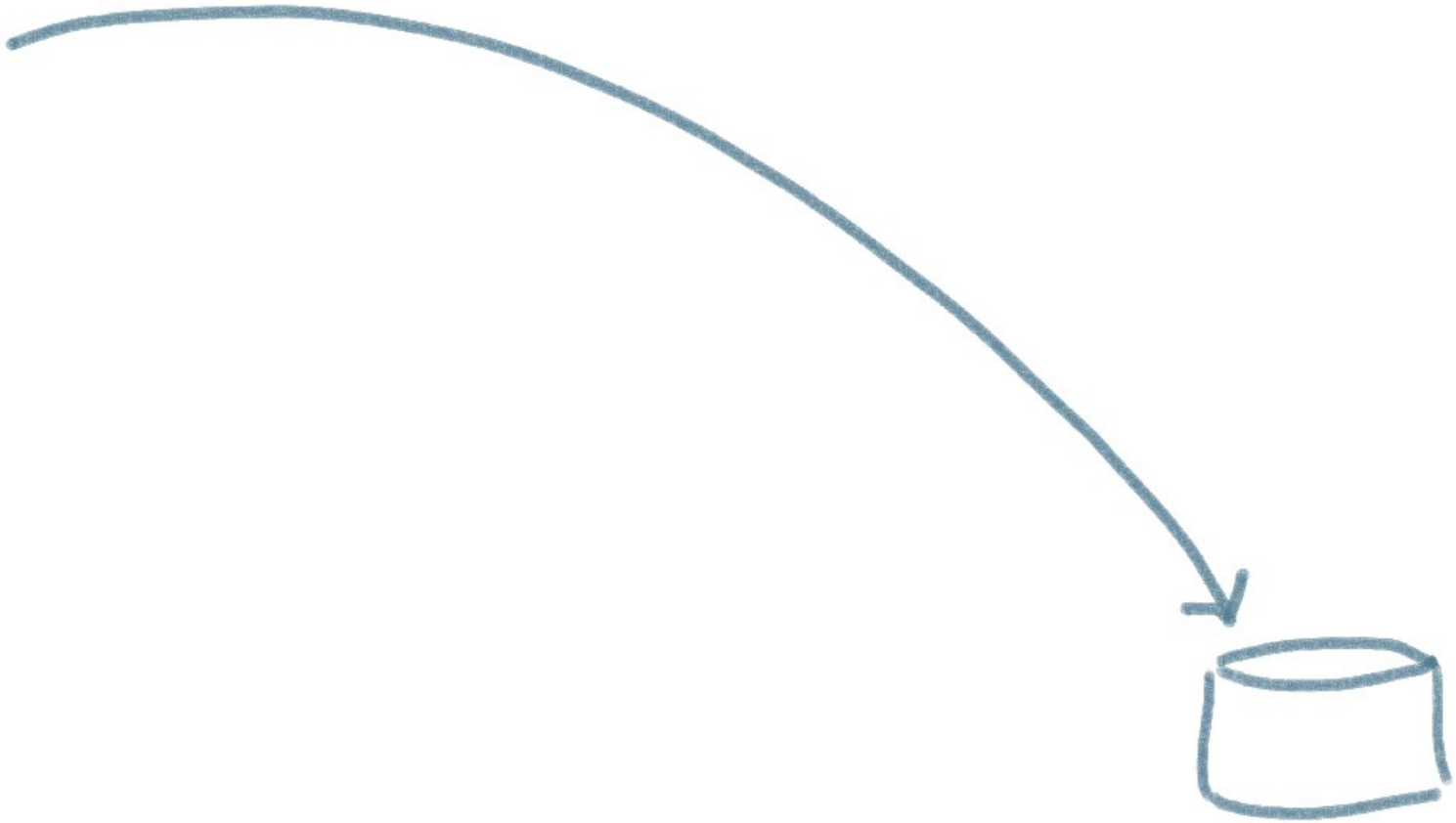
PostgreSQL as a Vector database

Gülçin Yildirim Jelinek
Staff Engineer

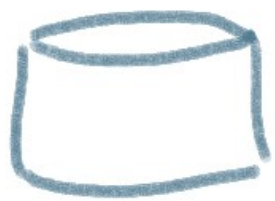
Boriss Mejías
Solutions Architect

4th of June, 2024

The Muffin Chihuahua Challenge

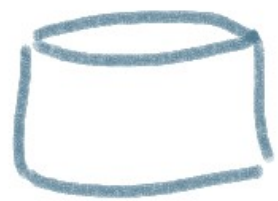


[0.42, 0.1, -0,7, ..., 0.3]



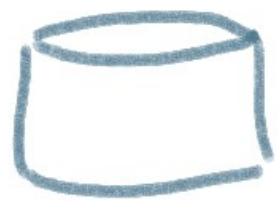
Embedding Model

[0.42, 0.1, -0,7, ..., 0.3]



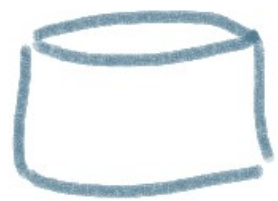
Embedding Model

[0.42, 0.1, -0,7, ..., 0.3]



Embedding Model

[0.42, 0.1, -0.7, ..., 0.3]



Embedding Model

Embedding Model

Embedding Model

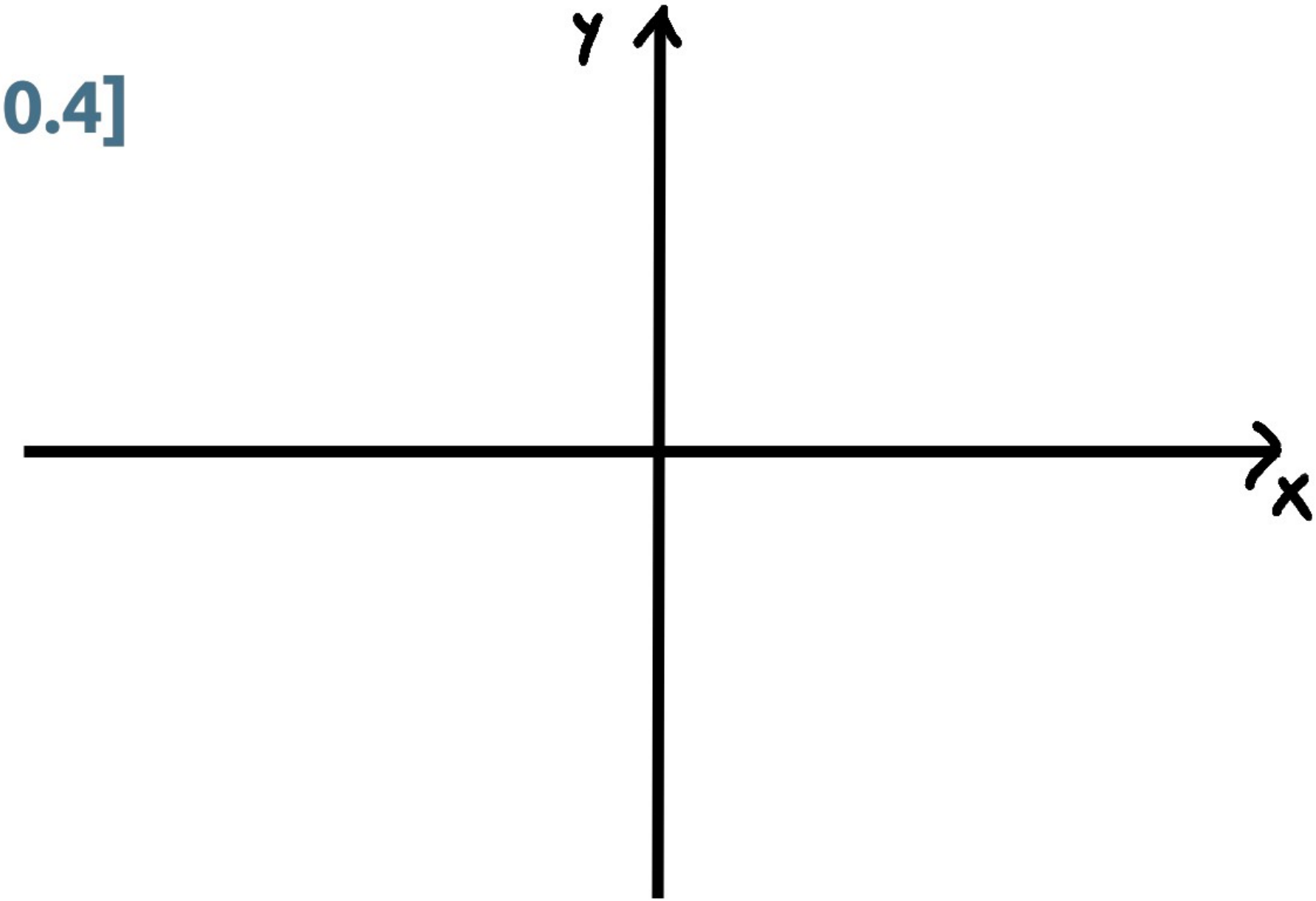
[0.42, 0.1, -0.7, ..., 0.3]

[0.17, -0.73, ..., 0.56]

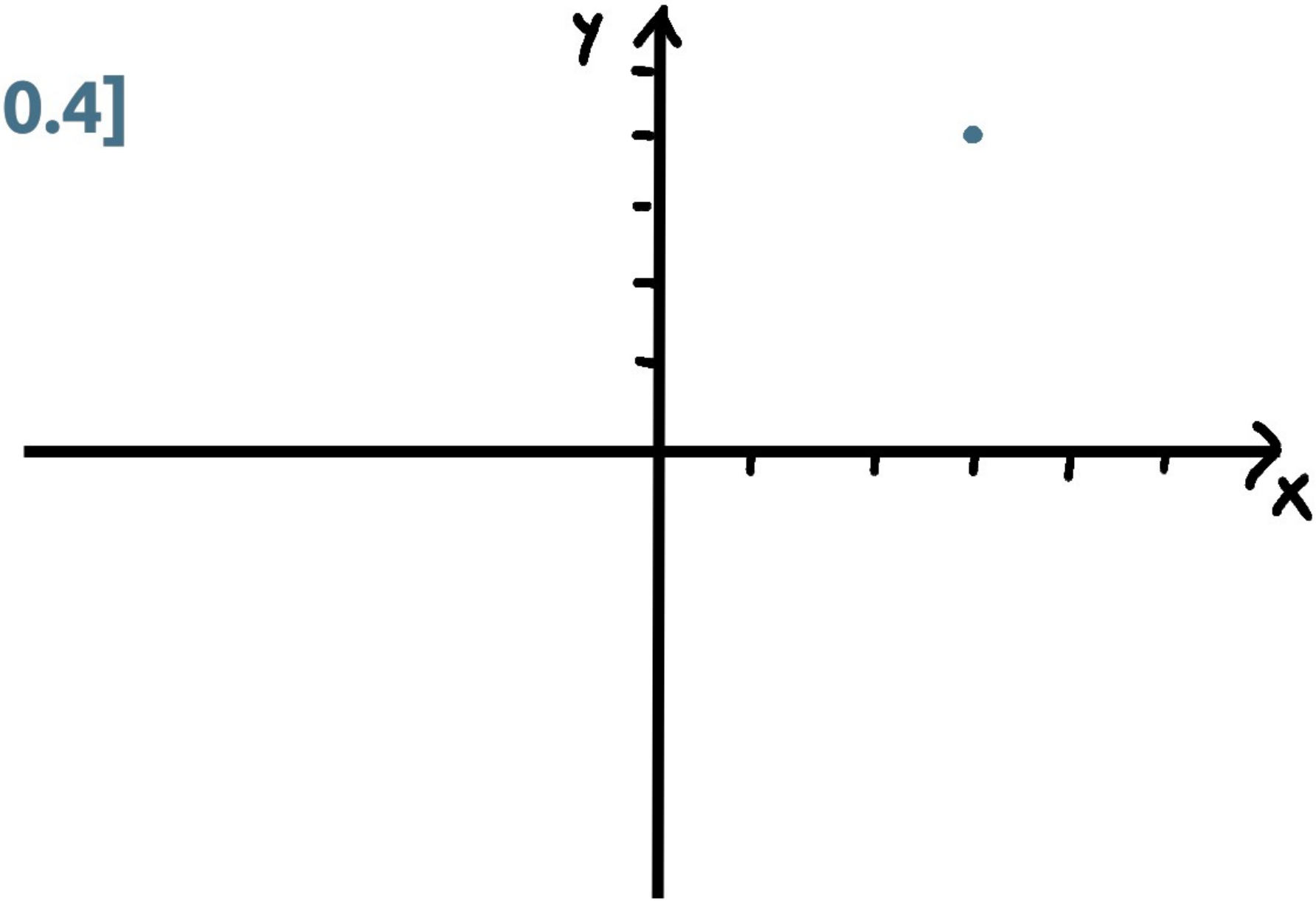
[-0.2, ..., 0.6]

[0.3, 0.4]

[0.3, 0.4]

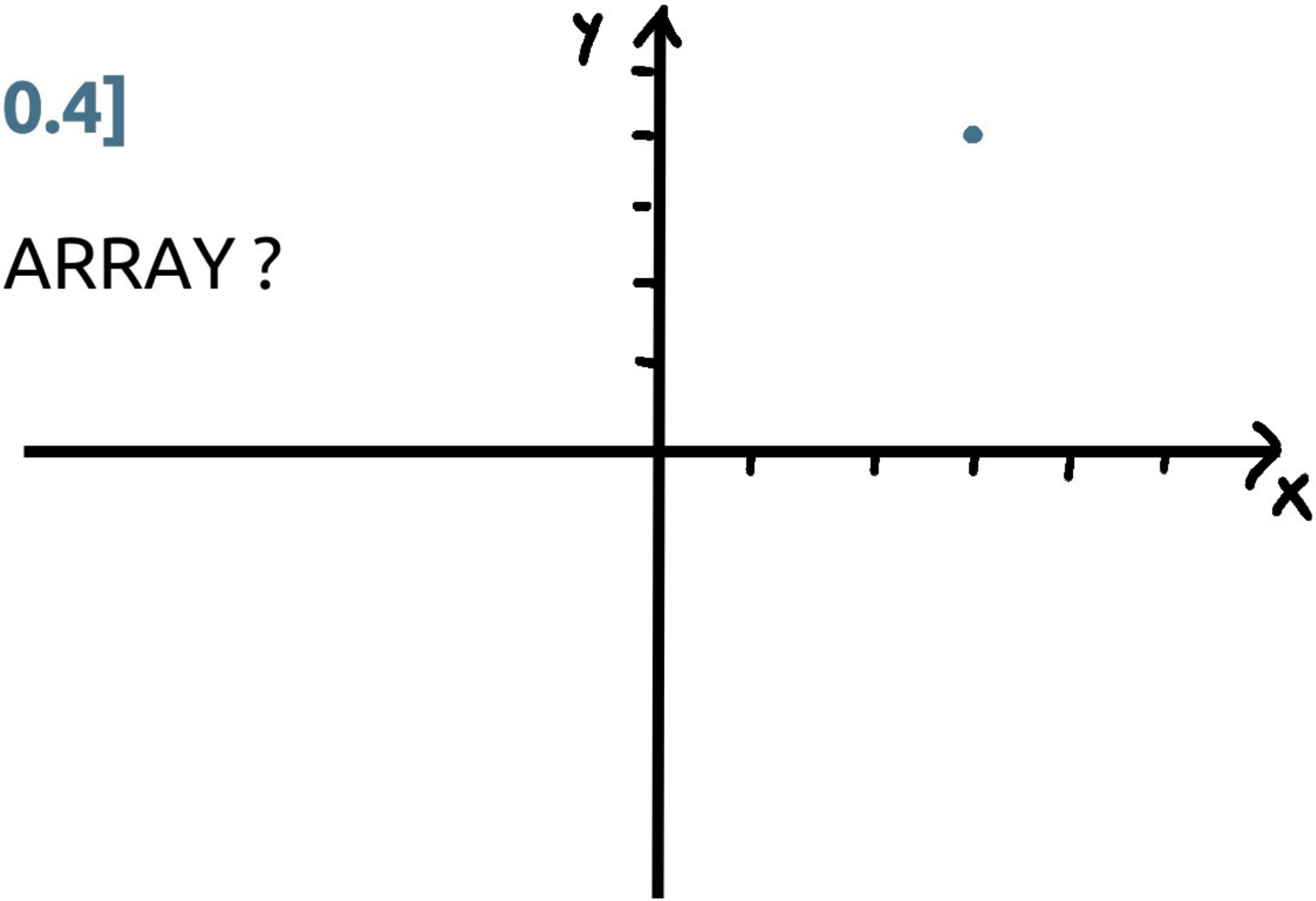


[0.3, 0.4]

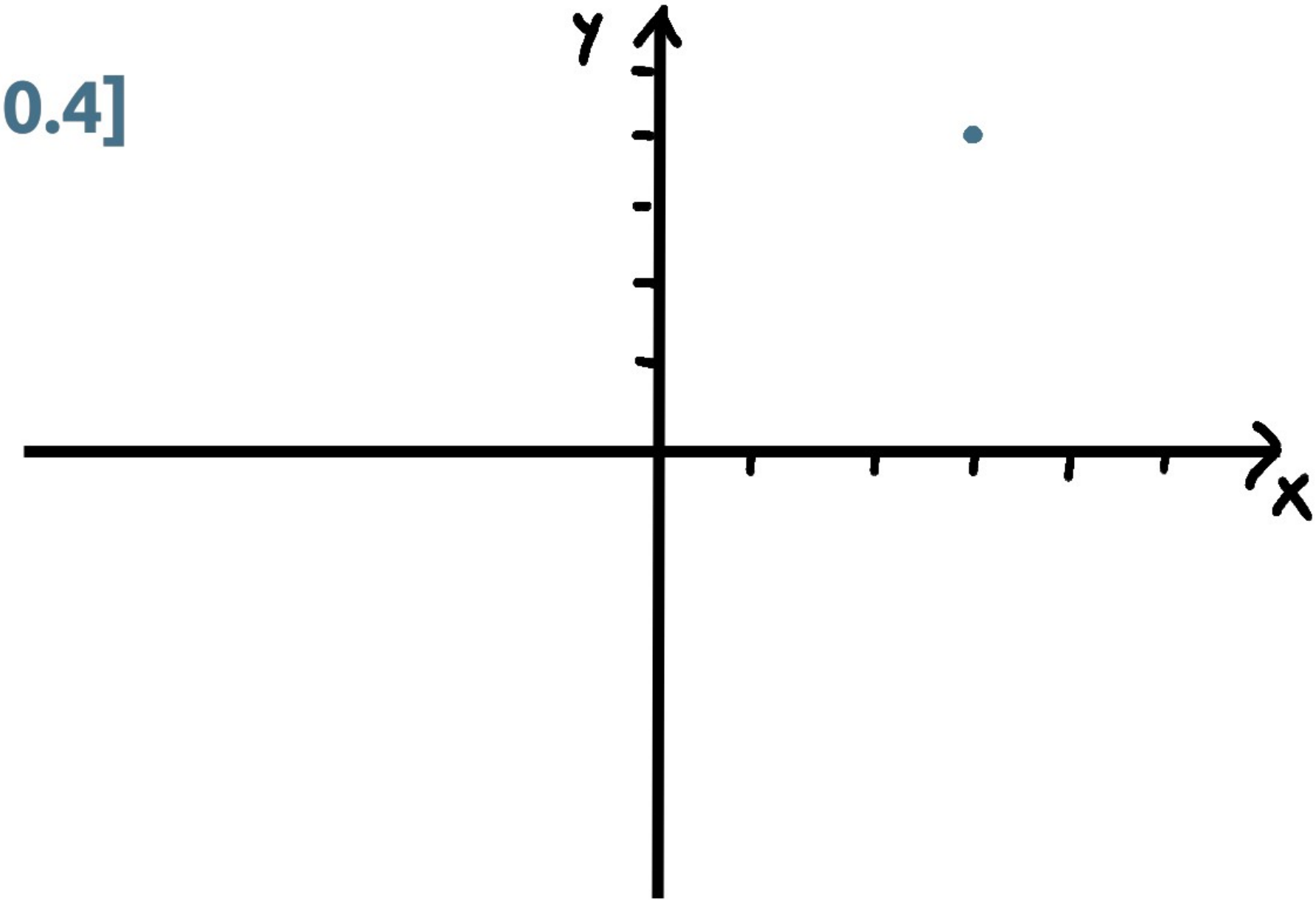


[0.3, 0.4]

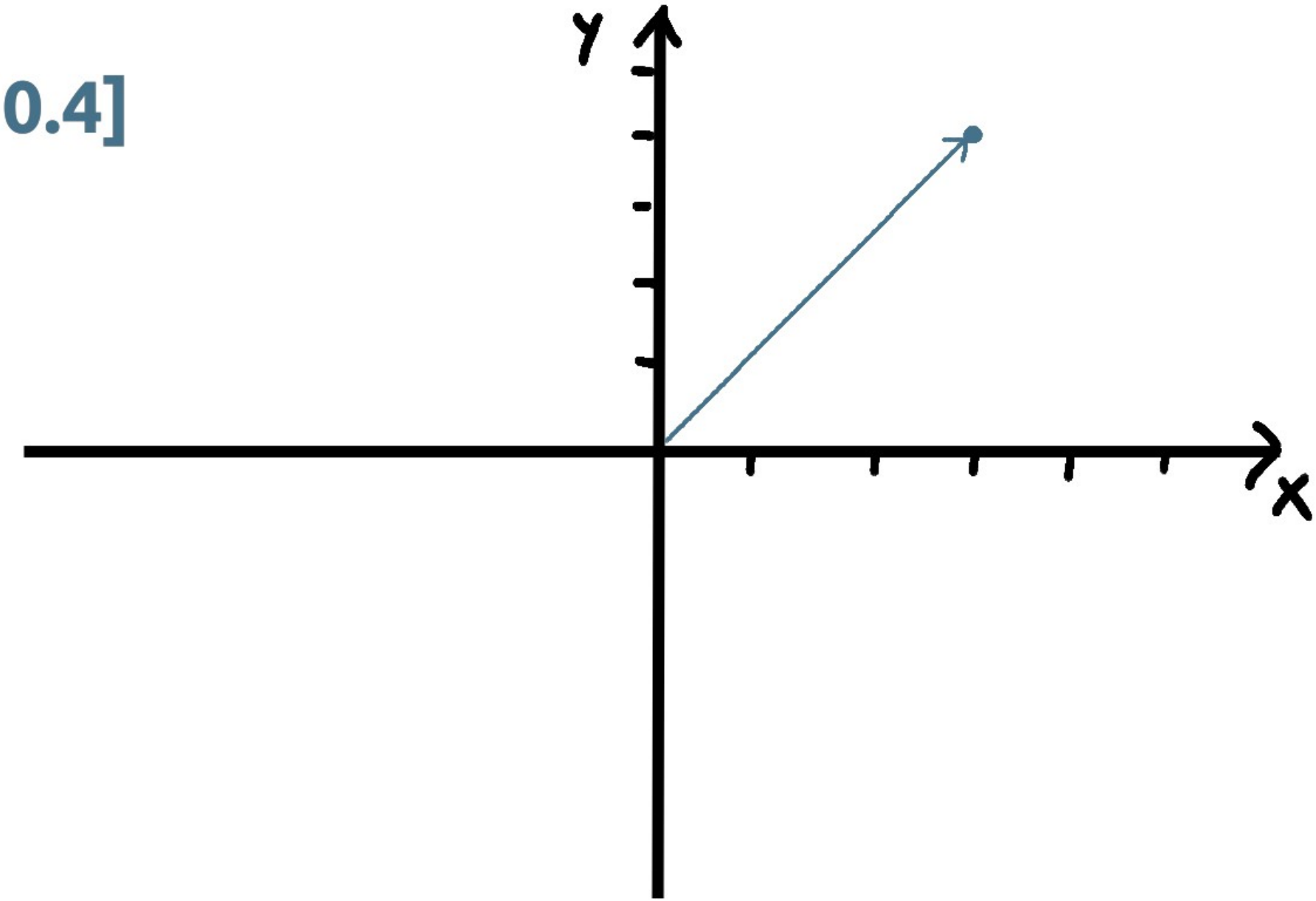
ARRAY ?



[0.3, 0.4]

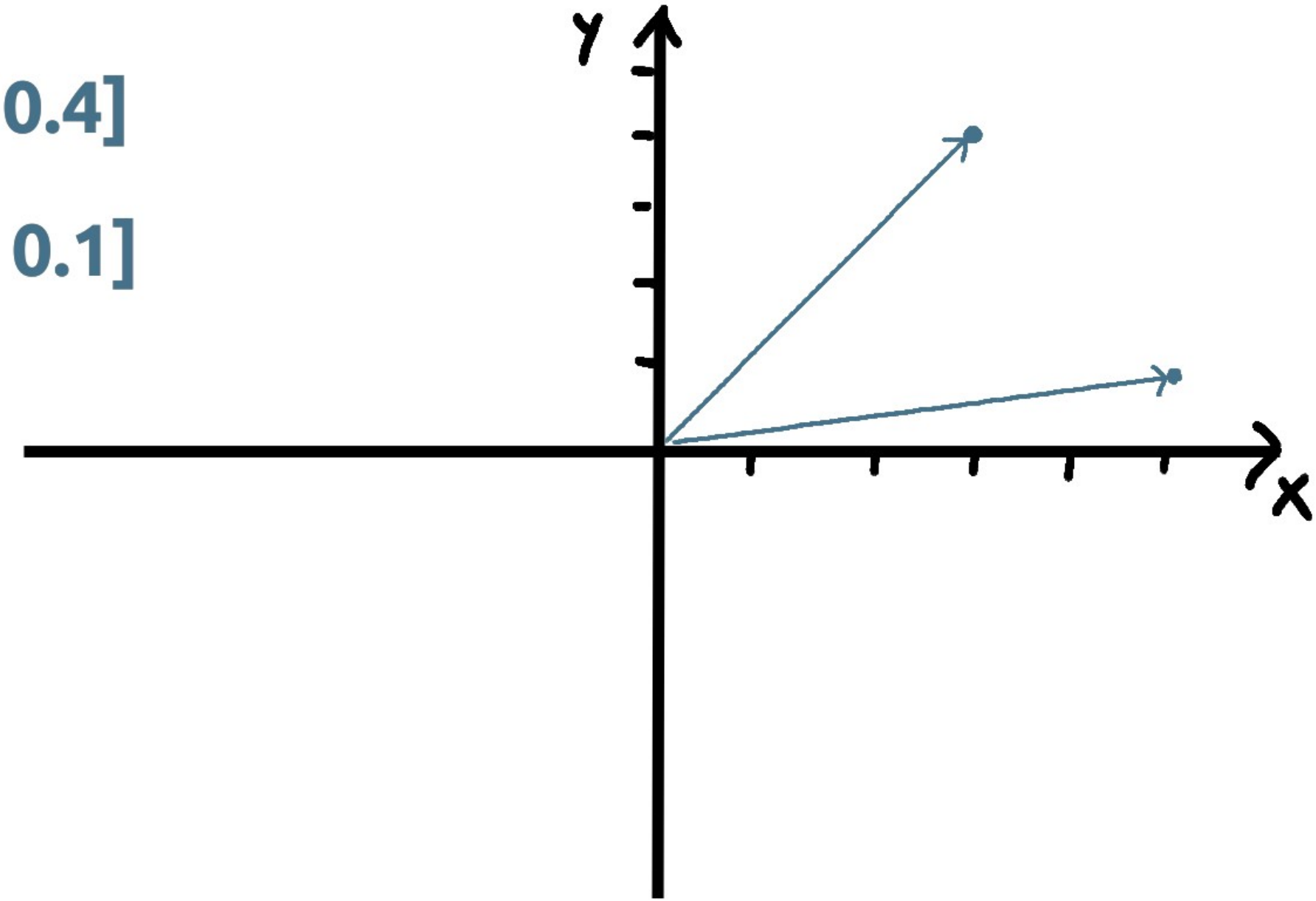


[0.3, 0.4]



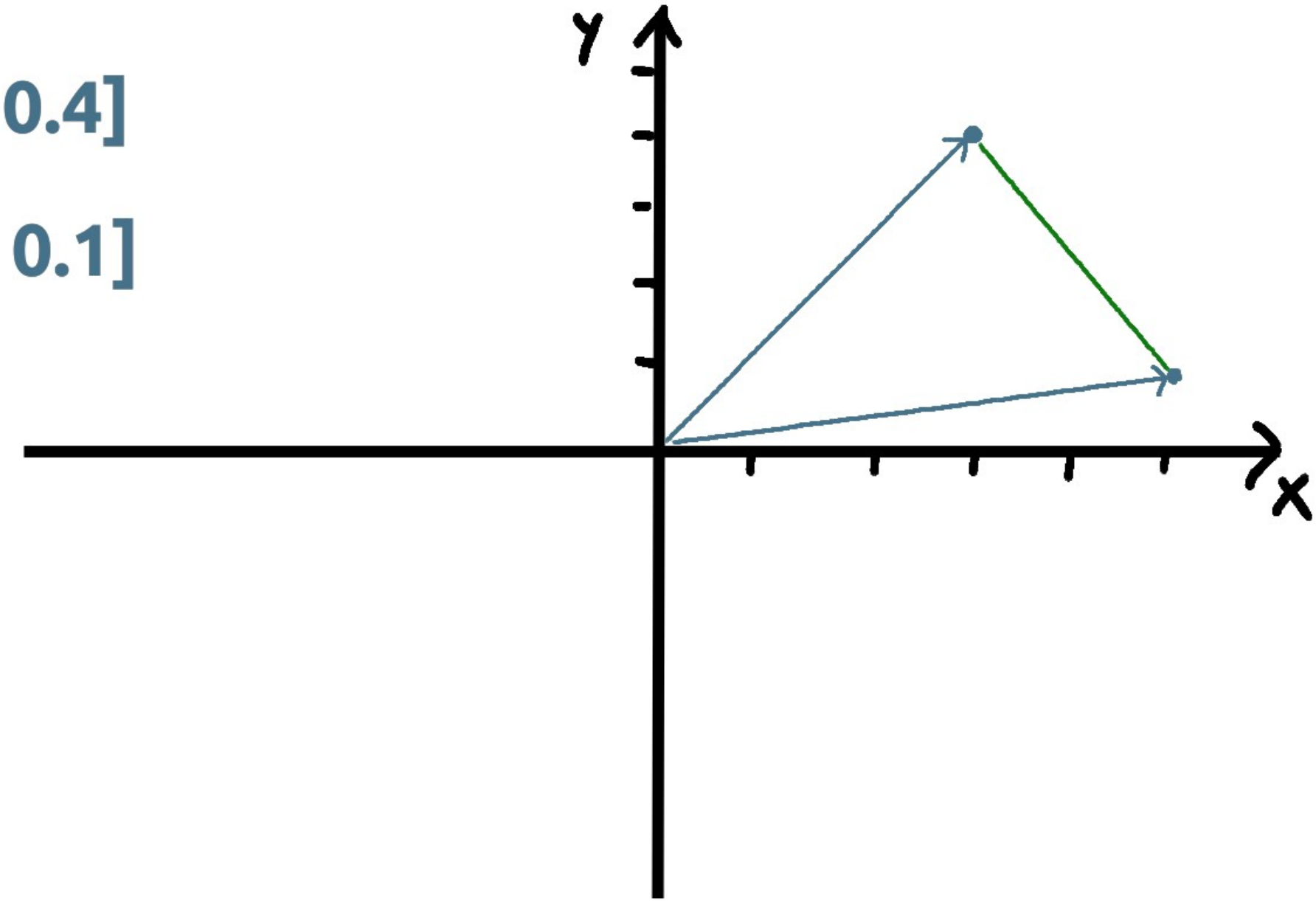
[0.3, 0.4]

[0.5, 0.1]



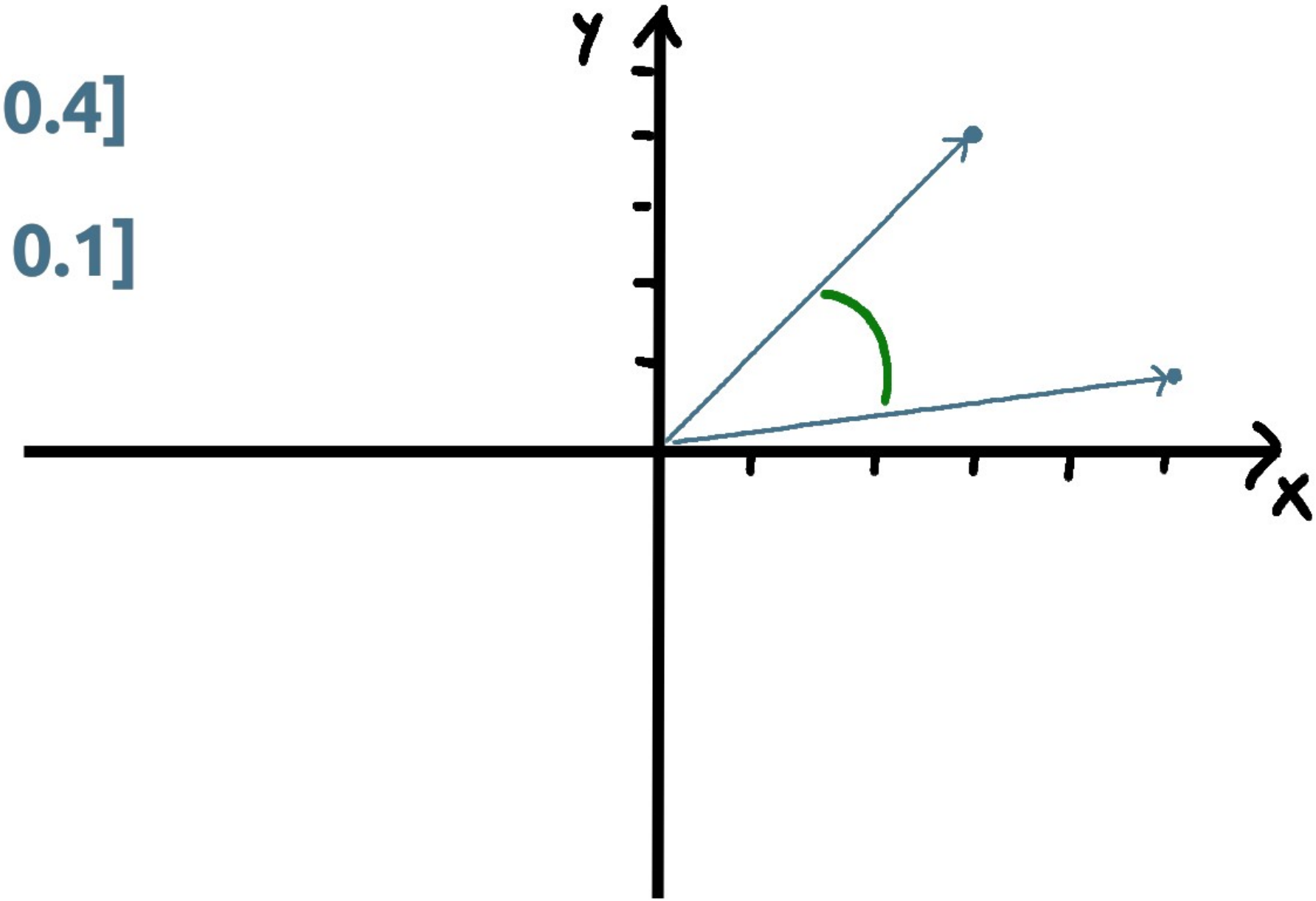
[0.3, 0.4]

[0.5, 0.1]



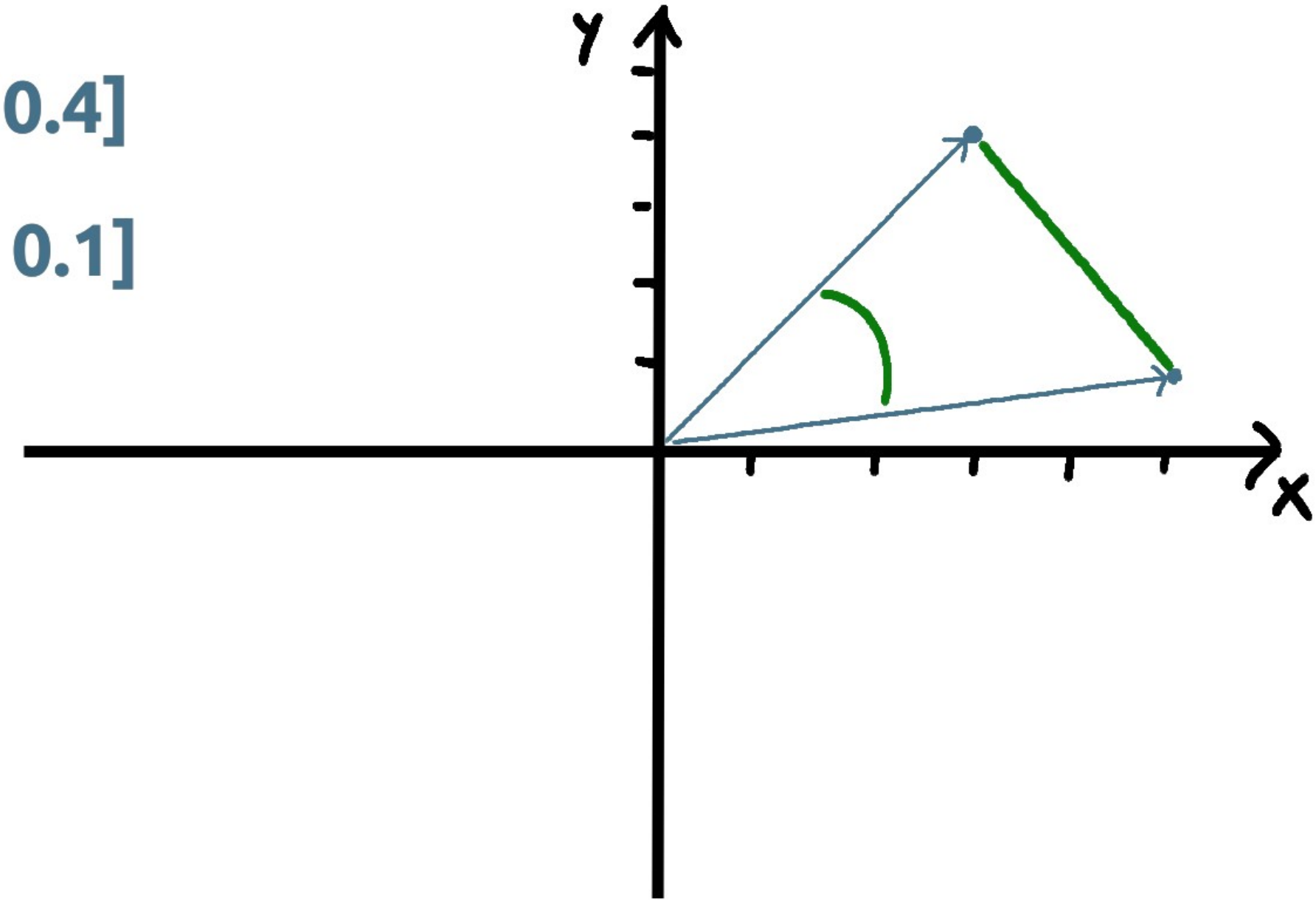
[0.3, 0.4]

[0.5, 0.1]



[0.3, 0.4]

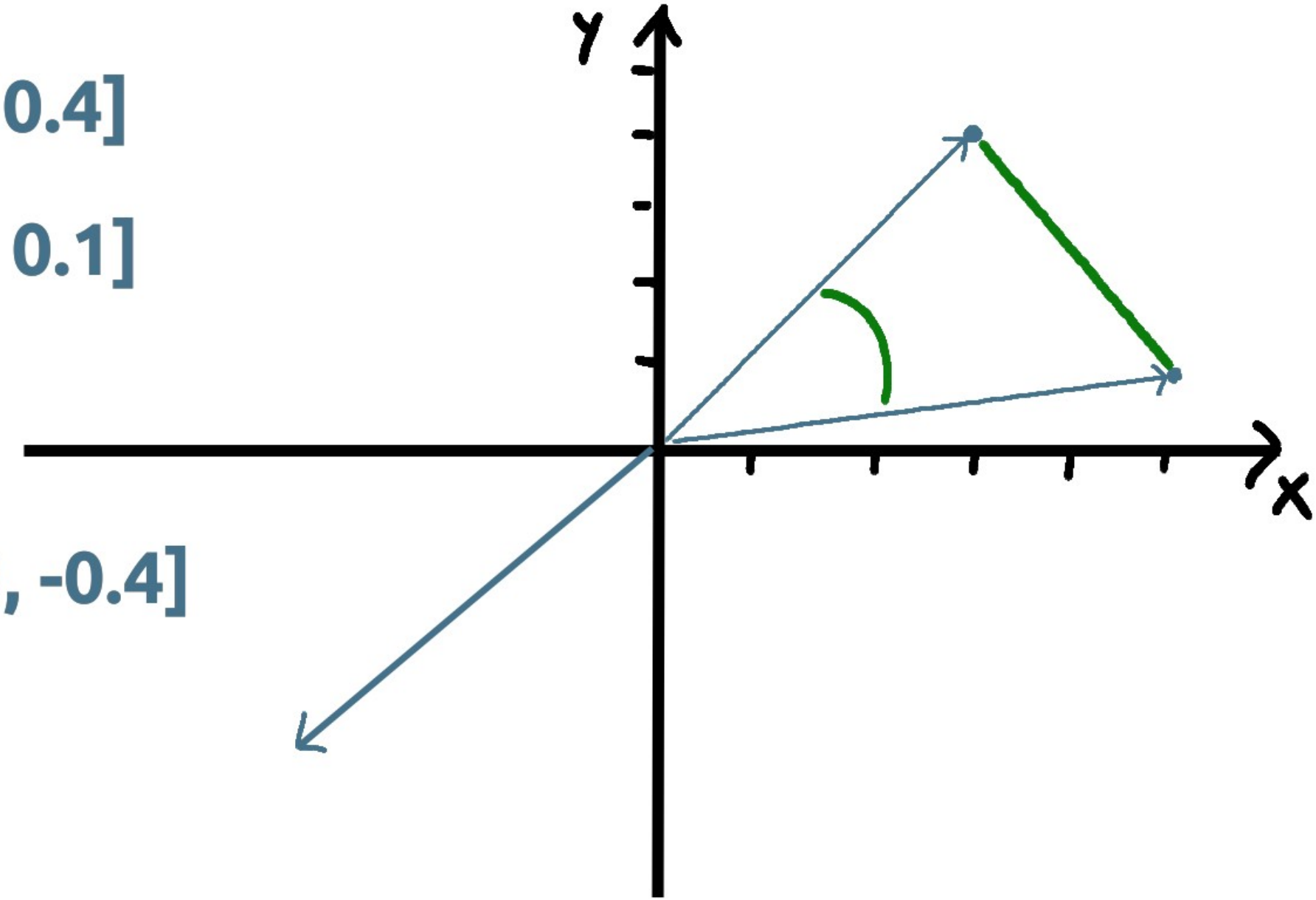
[0.5, 0.1]



[0.3, 0.4]

[0.5, 0.1]

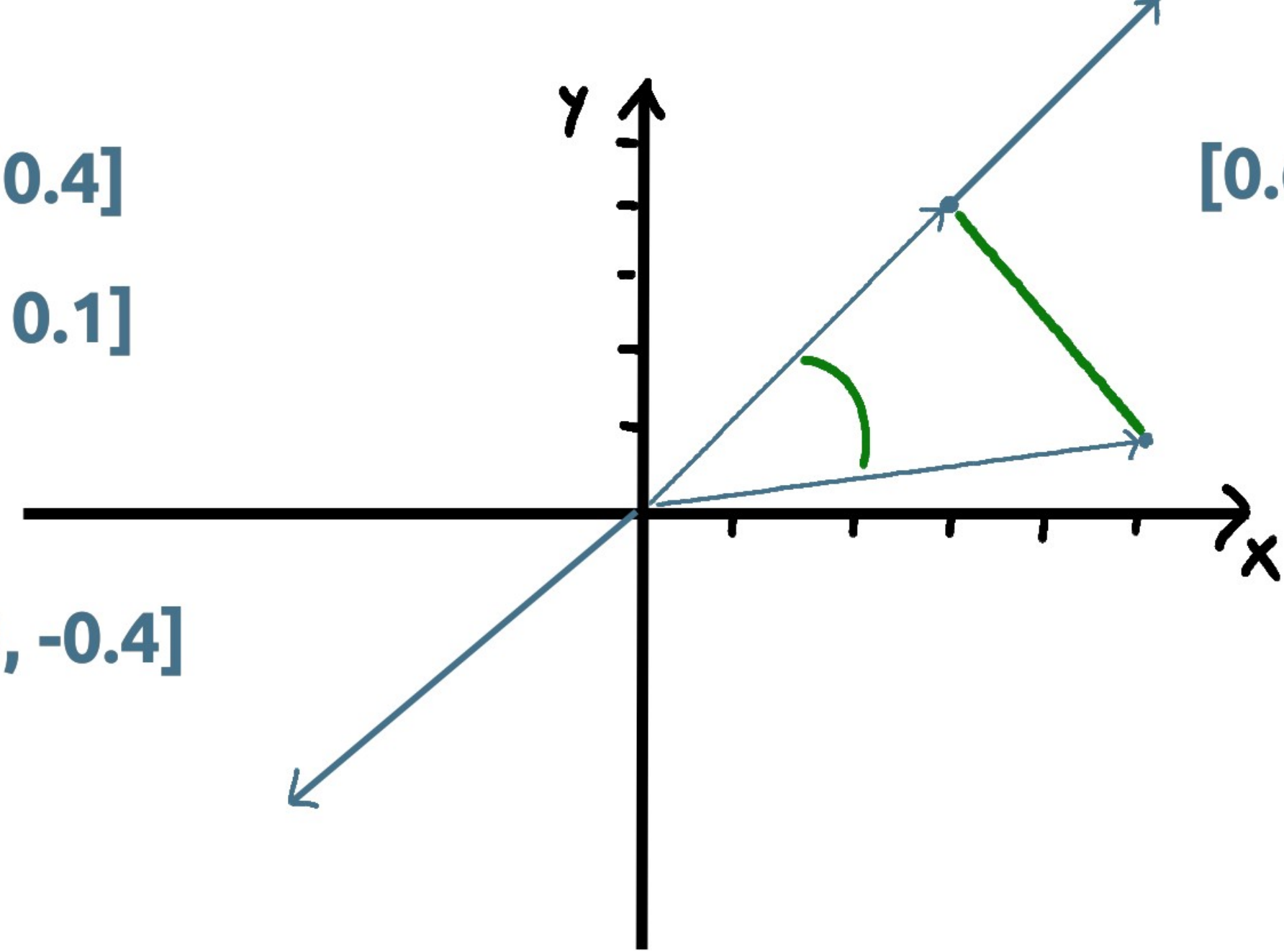
[-0.3, -0.4]



[0.3, 0.4]

[0.5, 0.1]

[-0.3, -0.4]



[0.6, 0.8]

pgvector

Vectors in your PostgreSQL database (pgvector)

- Specialized vector data type
- Indexing
- Search capabilities
- Distance operators

pgvector is an extension

```
CREATE EXTENSION vector;
```

vector is a data type

vector is a data type

```
CREATE SCHEMA p2d2;
CREATE TABLE p2d2.pictures (
    id          BIGSERIAL   PRIMARY KEY
  , filename    text        NOT NULL
  , embedding vector(768) NOT NULL
);
```

Storage

```
p2d2.pictures
    id          BIGSERIAL    PLAIN
    filename    text         EXTENDED
    embedding   vector(768)  EXTERNAL
```

Go for PLAIN

```
ALTER TABLE p2d2.pictures
ALTER COLUMN embedding
SET STORAGE PLAIN;
```

Storage

```
p2d2.pictures
    id          BIGSERIAL    PLAIN
    filename    text         EXTENDED
    embedding   vector(768)  PLAIN
```

Lab – Part I

Each Participant

Ip address: https://42.0.0.101 Self-Signed Cert

Credentials: user42 - secrettowel

Database: p2d2db42

```
psql -d p2d2db42
CREATE EXTENSION vector;
\dx
```

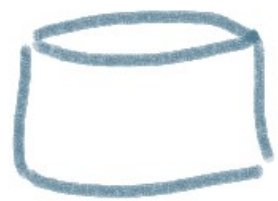
CREATE TABLE

```
psql -d p2d2db42

CREATE SCHEMA p2d2;

CREATE TABLE p2d2.pictures (
    id        BIGSERIAL   PRIMARY KEY
  , filename  text        NOT NULL
  , embedding vector(768) NOT NULL

);

ALTER TABLE p2d2.pictures
ALTER COLUMN embedding SET STORAGE PLAIN;
```

Lab – Part II

Embedding Model

[0.42, 0.1, -0,7, ..., 0.3]

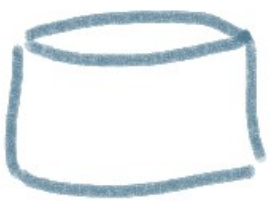


Embedding Model

[0.42, 0.1, -0,7, ..., 0.3]

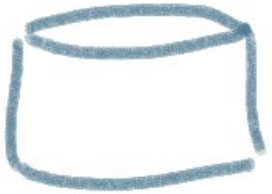
Embedding Model

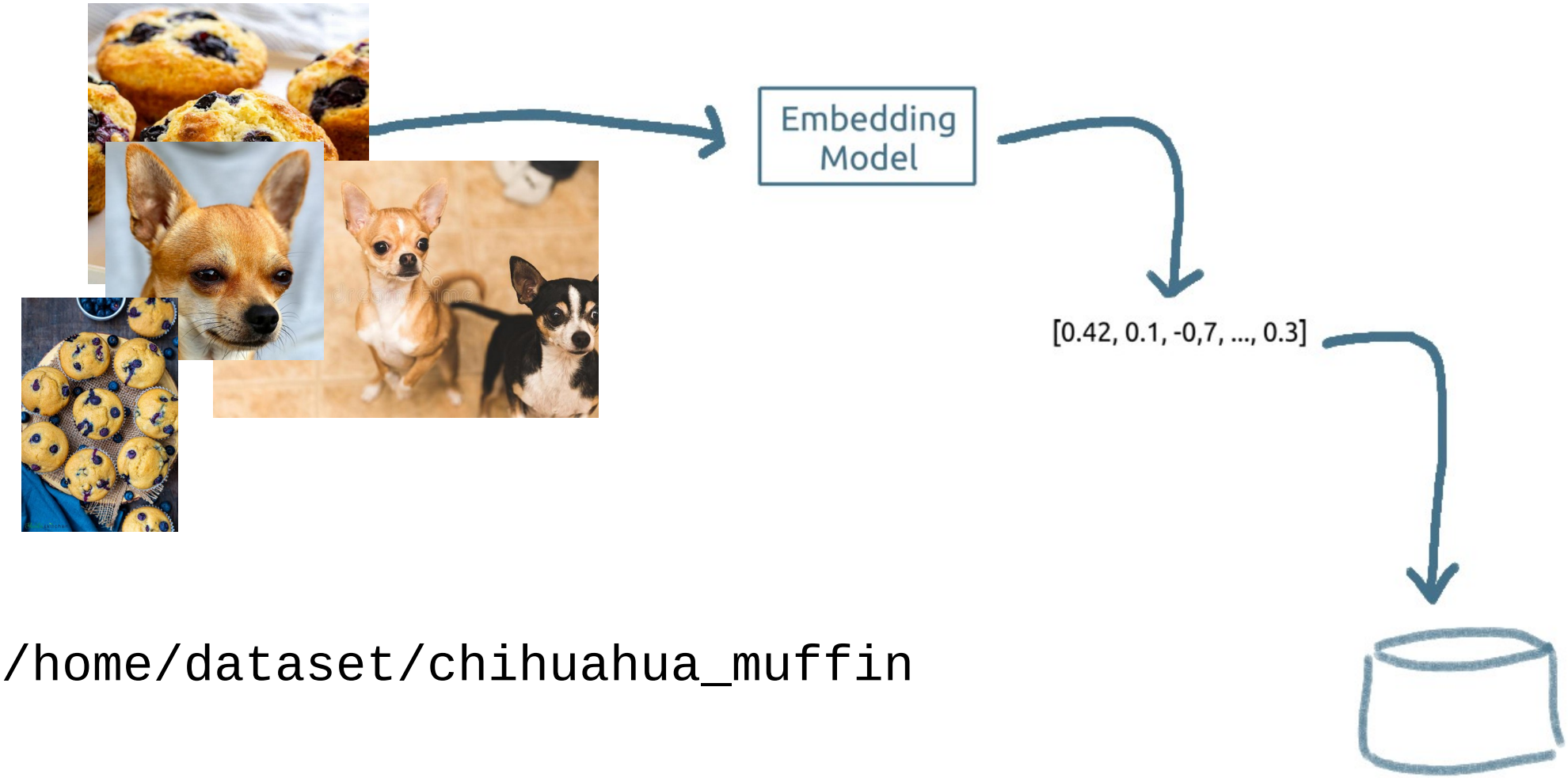
[0.42, 0.1, -0,7, ..., 0.3]



Embedding Model

[0.42, 0.1, -0,7, ..., 0.3]



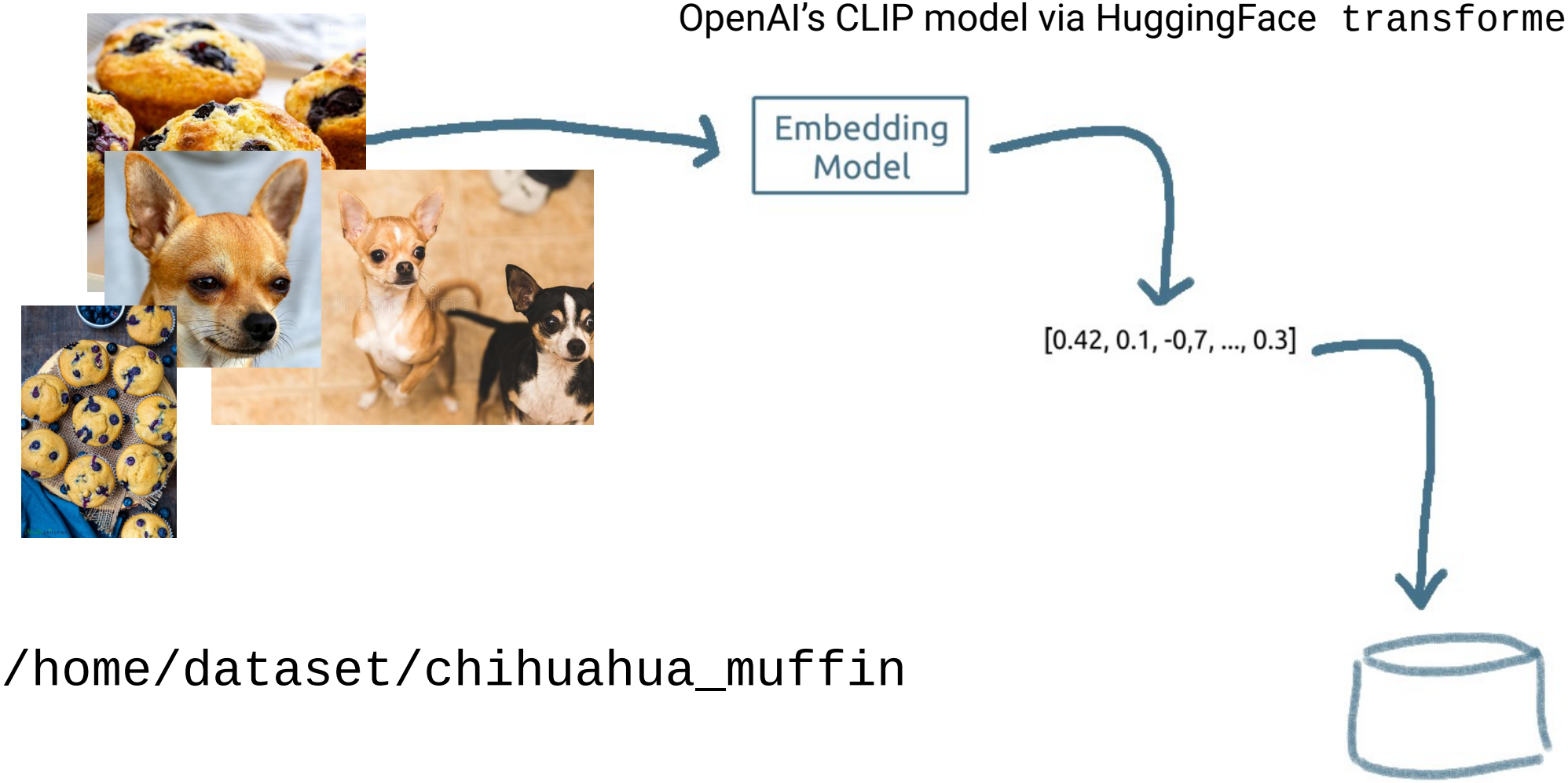


/home/dataset/chihuahua_muffin

https://www.kaggle.com/datasets/samuelcortinhas/muffin-vs-chihuahua-image-classification/

python3 with imgbeddings

OpenAI's CLIP model via HuggingFace `transformers`



`/home/dataset/chihuahua_muffin`

`https://www.kaggle.com/datasets/samuelcortinhas/muffin-vs-chihuahua-image-classification/`

Load Embeddings

Check /home/user42/.load_embeddings

load_embeddings.py

Load Embeddings

Check /home/user42/.load_embeddings

load_embeddings.py

```
ibed = imgbeddings()
ibed.to_embeddings(img)
```

Load Embeddings

Check /home/user42/.load_embeddings

load_embeddings.py

```
ibed = imgbeddings()
ibed.to_embeddings(img)
```

```
psycopg2.connect(…)
conn.
```

Load Embeddings

Check /home/user42/.load_embeddings

load_embeddings.py

```
ibed = imgbeddings()
ibed.to_embeddings(img)
```

```
psycopg2.connect(…)
conn.commit()
```

Load Embeddings

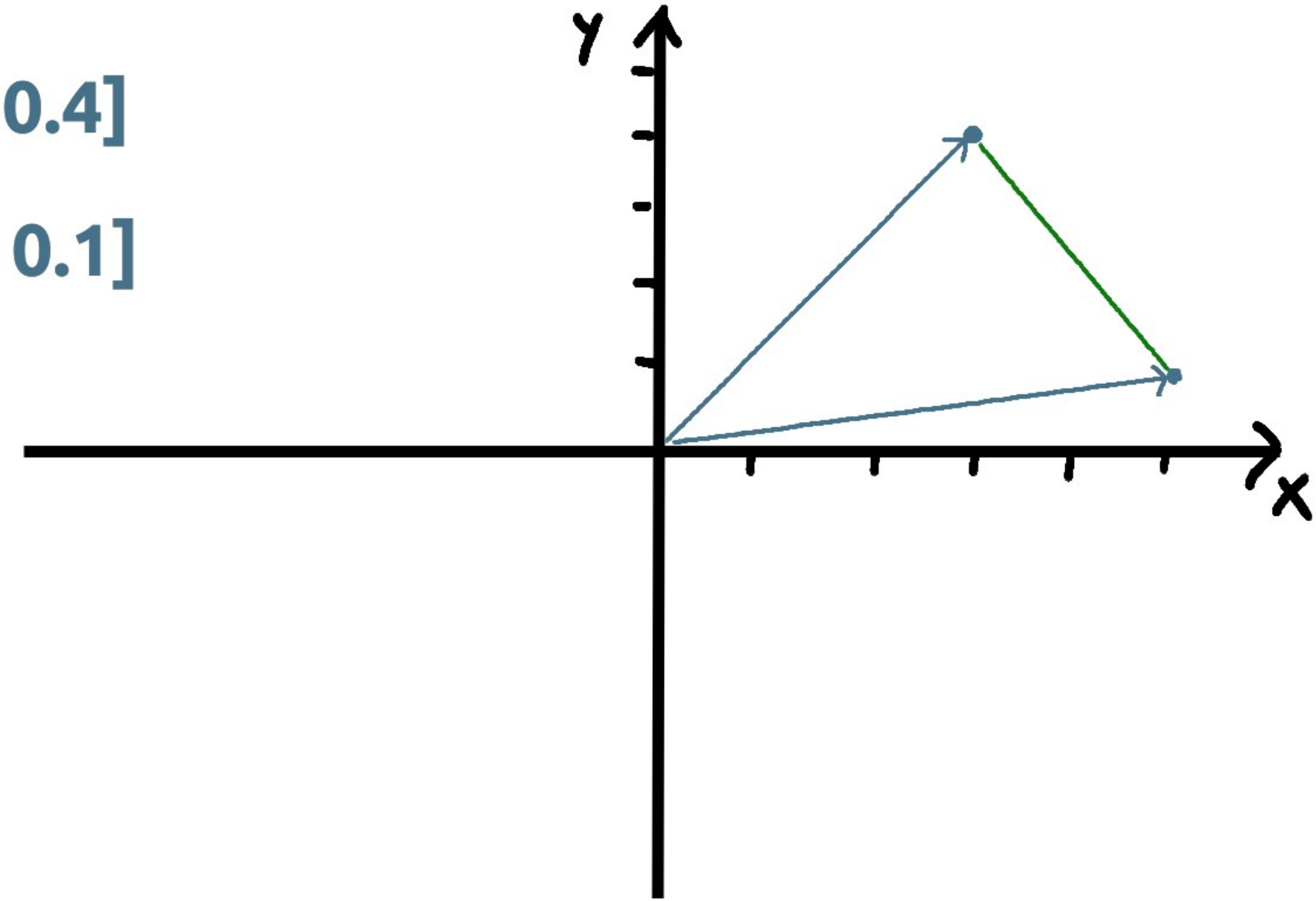
```
./load_embeddings.py                                    \
      /home/dataset/chihuahua_muffin/chihuahua \
      /var/run/postgresql 5432 p2d2db42 user42


./load_embeddings.py                                    \
      /home/dataset/chihuahua_muffin/muffin    \
      /var/run/postgresql 5432 p2d2db42 user42
```

Lab – Part III

querying

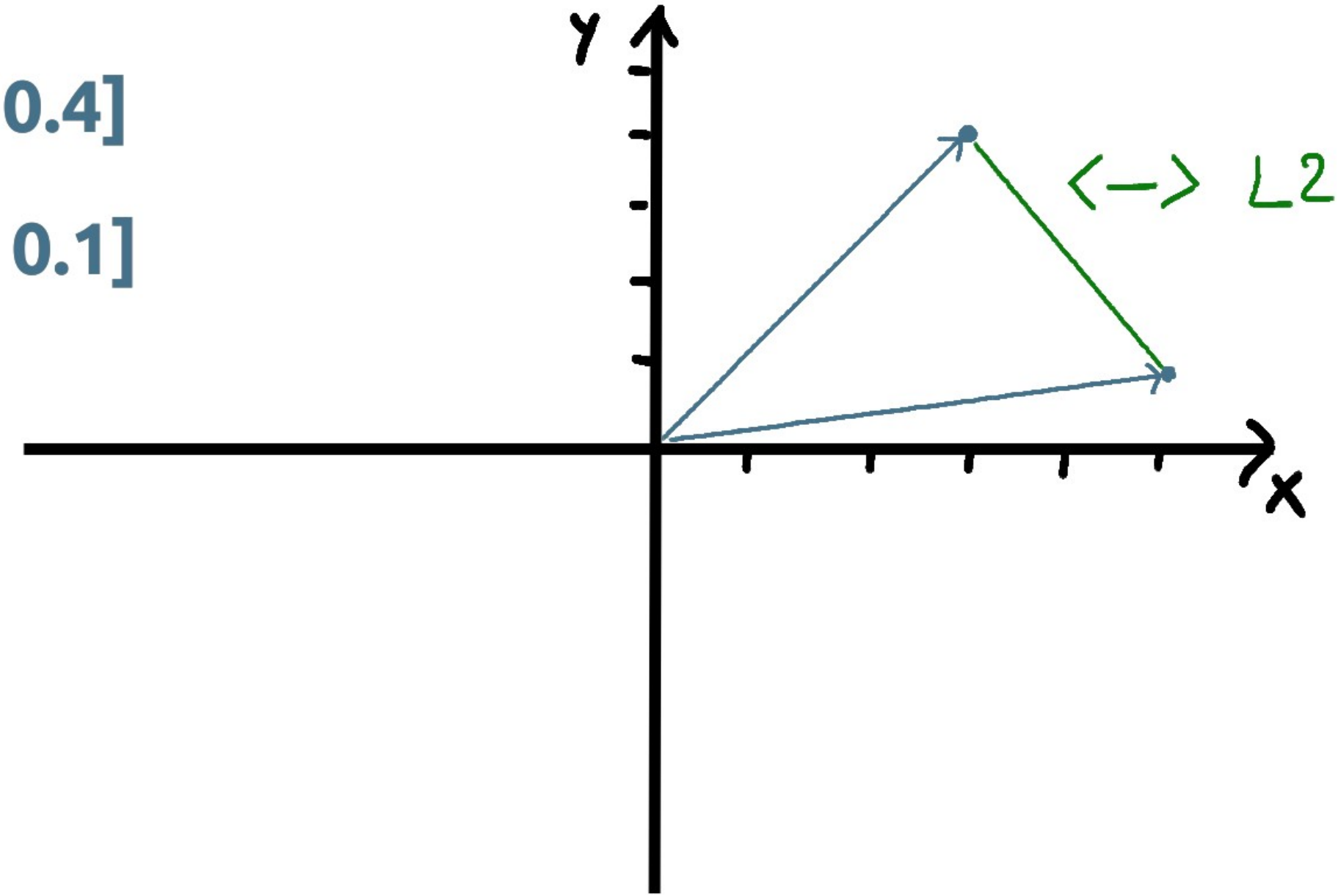
[0.3, 0.4]

[0.5, 0.1]



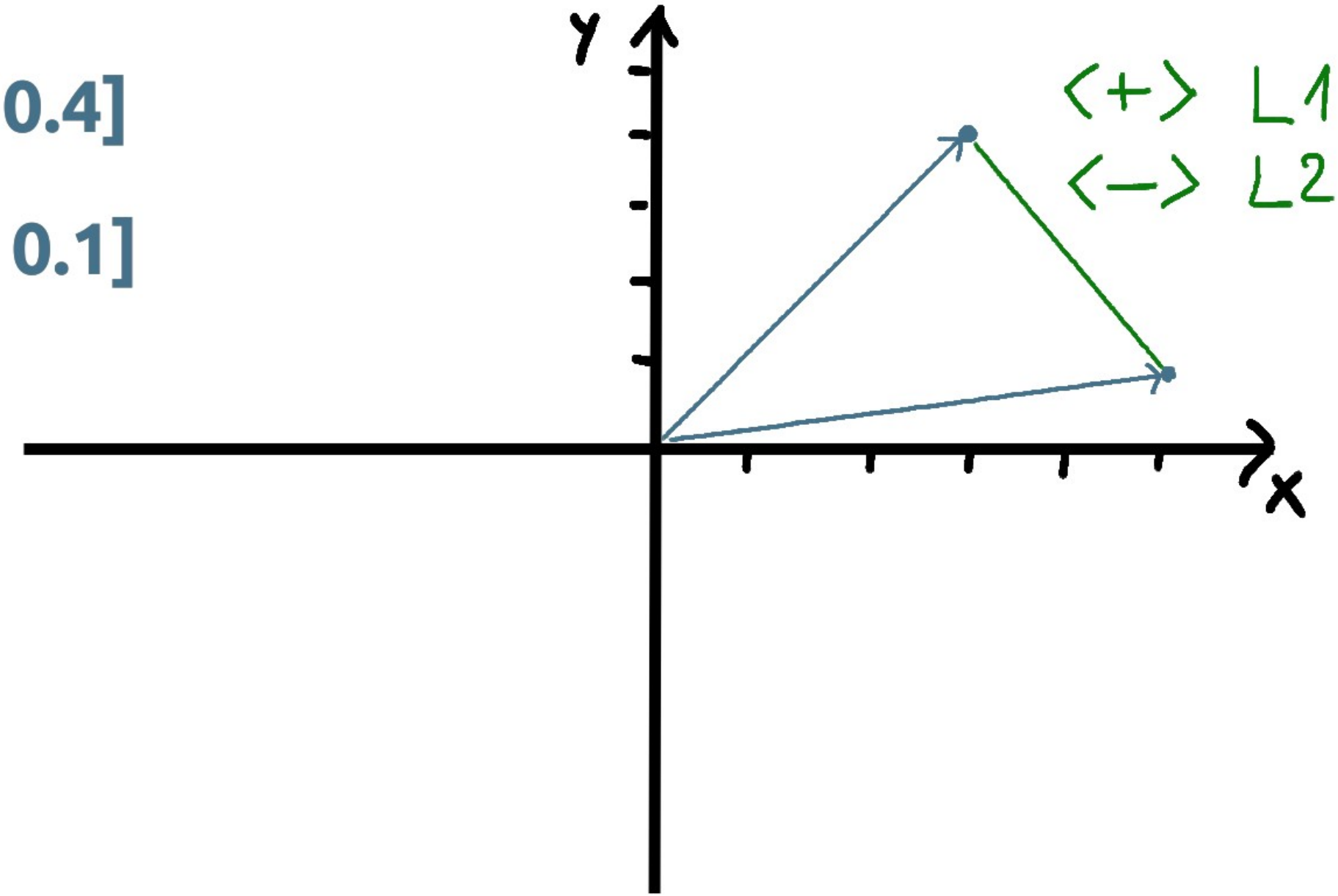
[0.3, 0.4]

[0.5, 0.1]



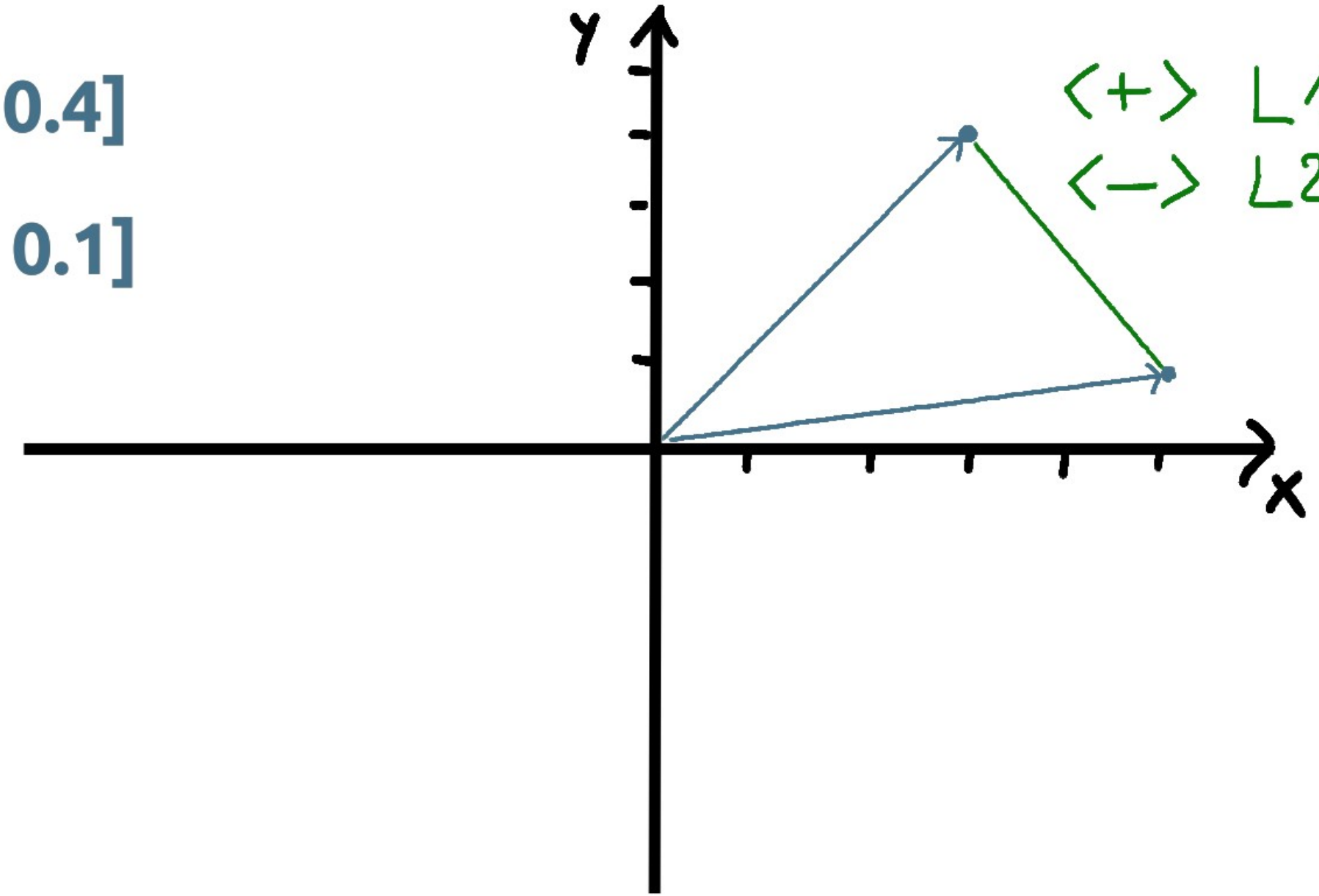
[0.3, 0.4]

[0.5, 0.1]



[0.3, 0.4]

[0.5, 0.1]

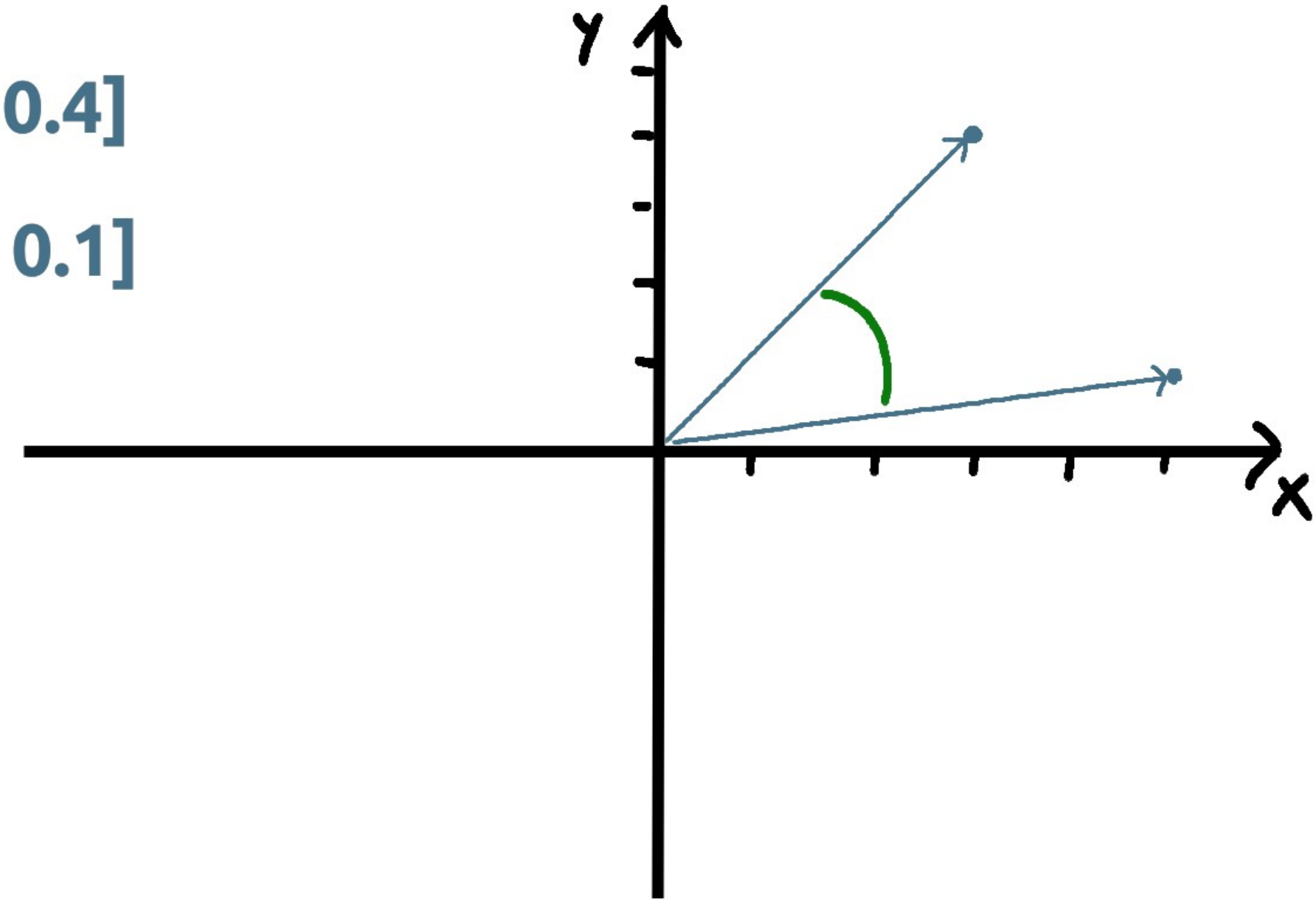


<+> L1 $\sum |x_i|$

<−> L2 $\sqrt{\sum |x_i|^2}$

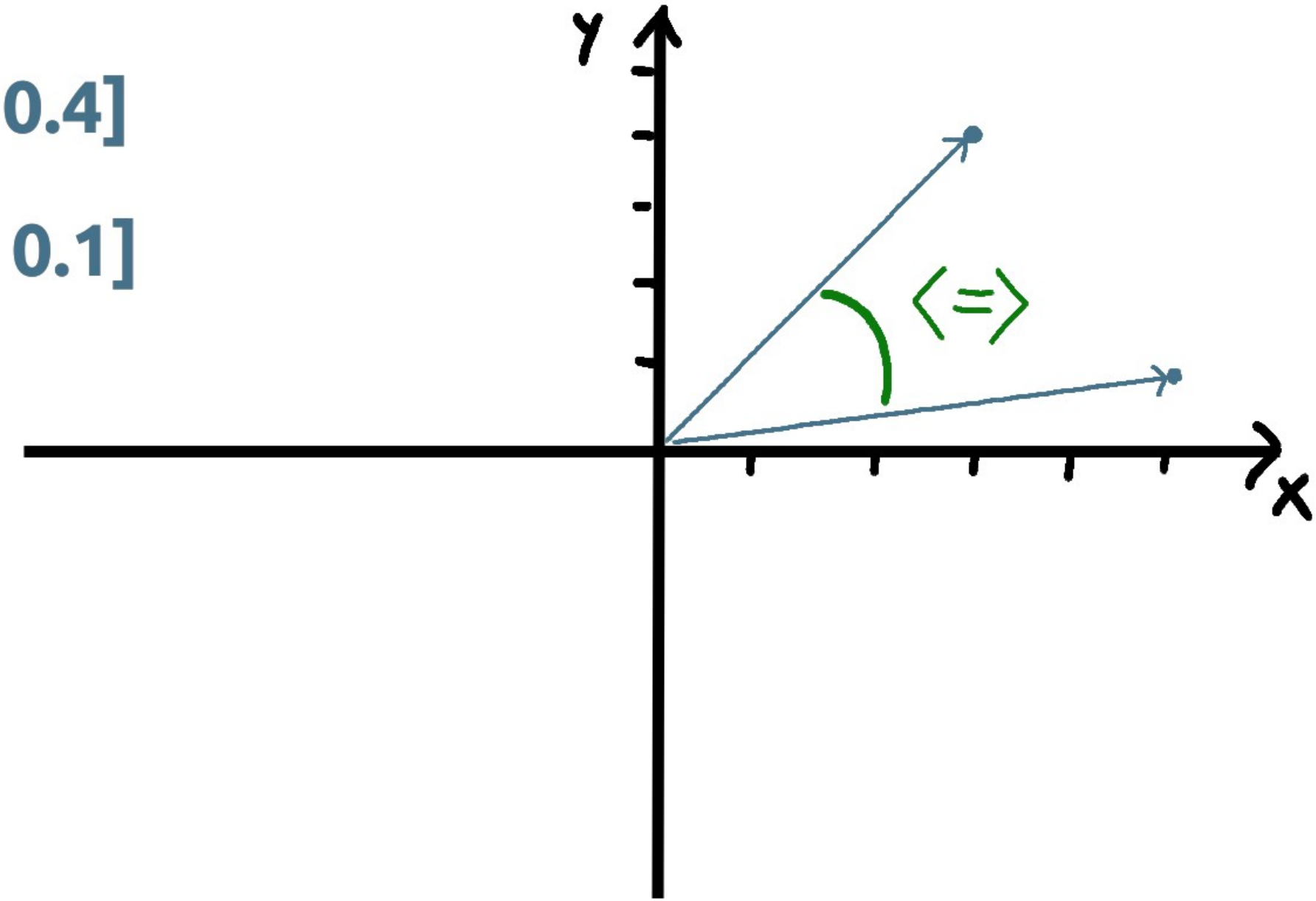
[0.3, 0.4]

[0.5, 0.1]



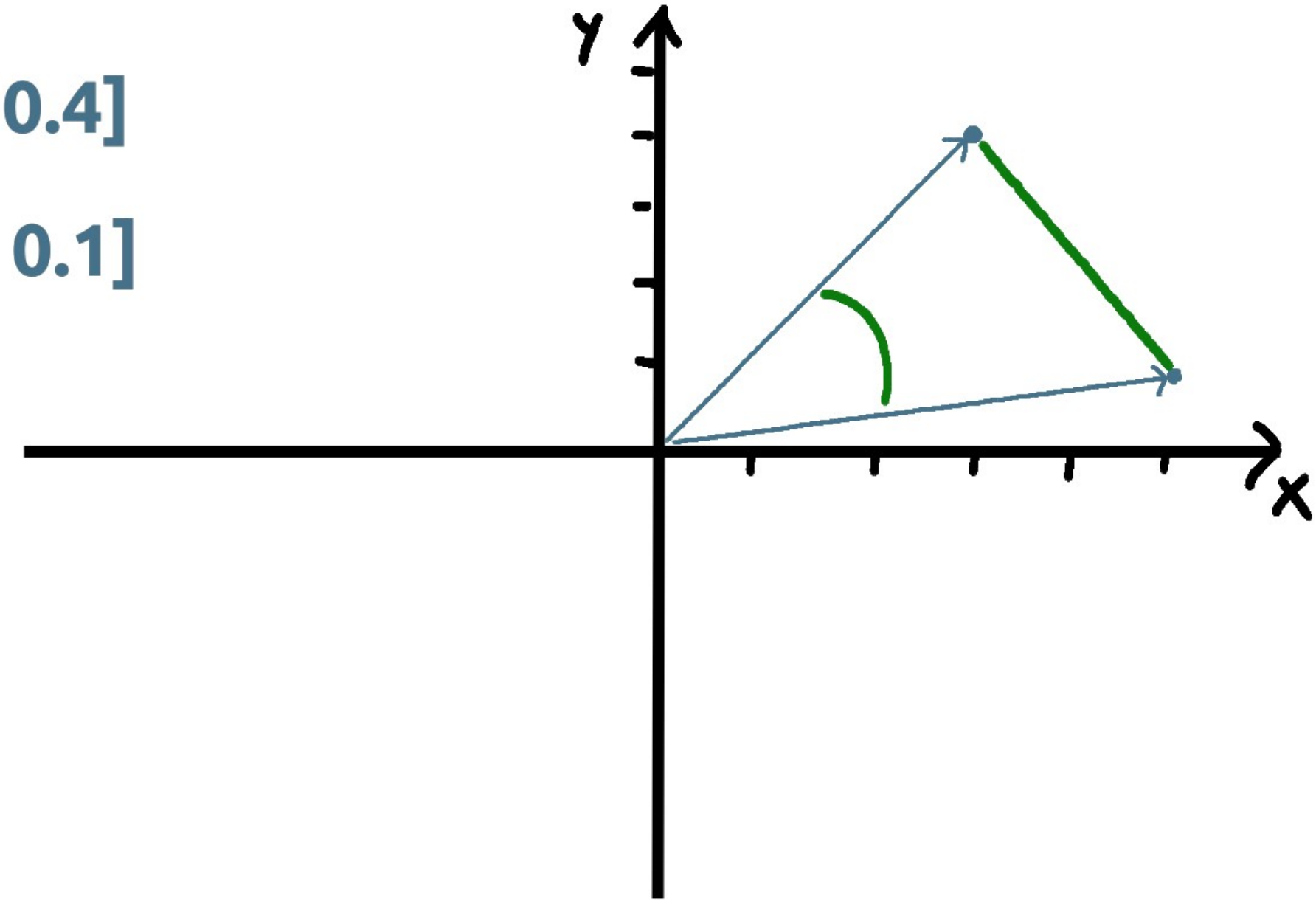
[0.3, 0.4]

[0.5, 0.1]



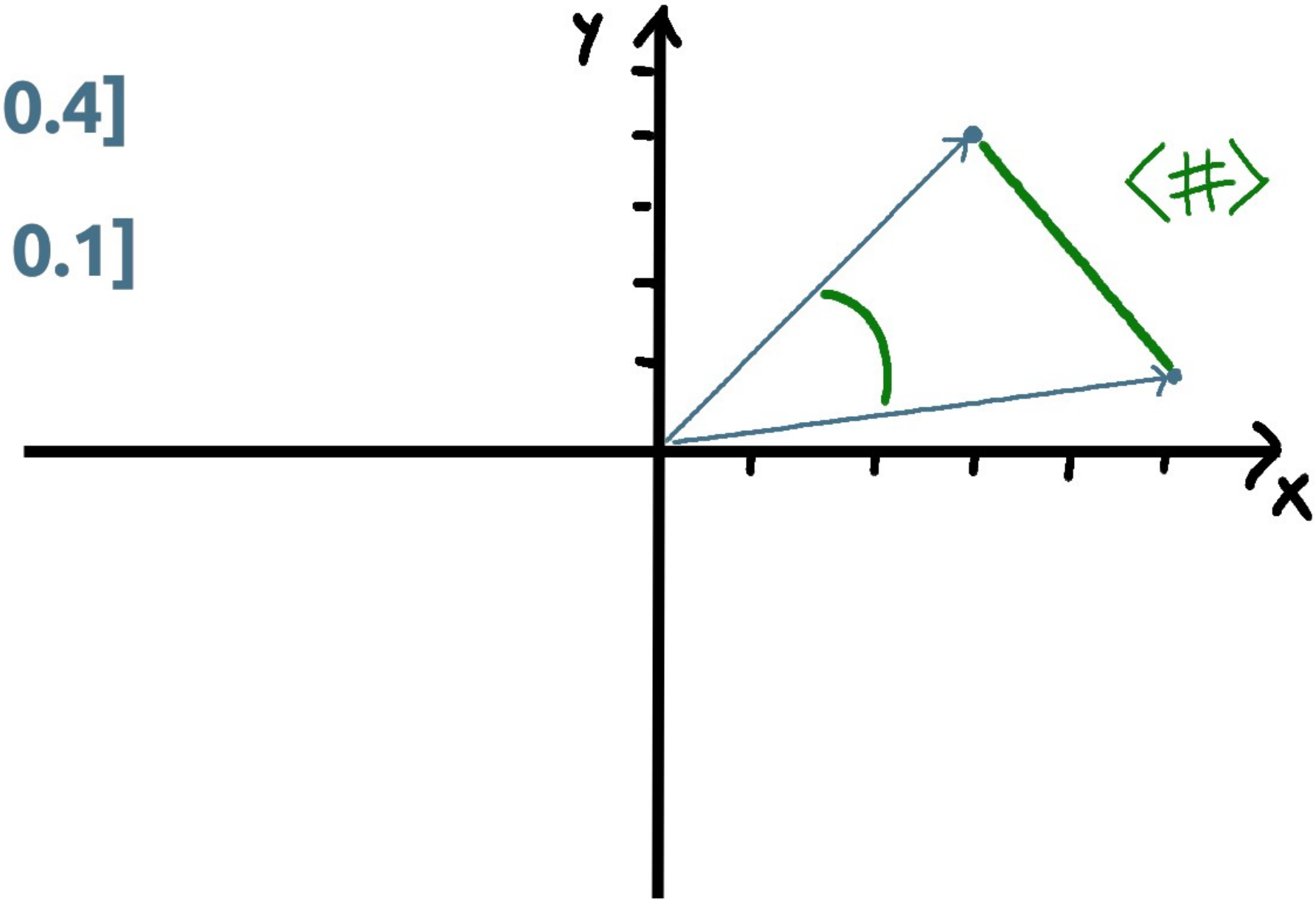
[0.3, 0.4]

[0.5, 0.1]



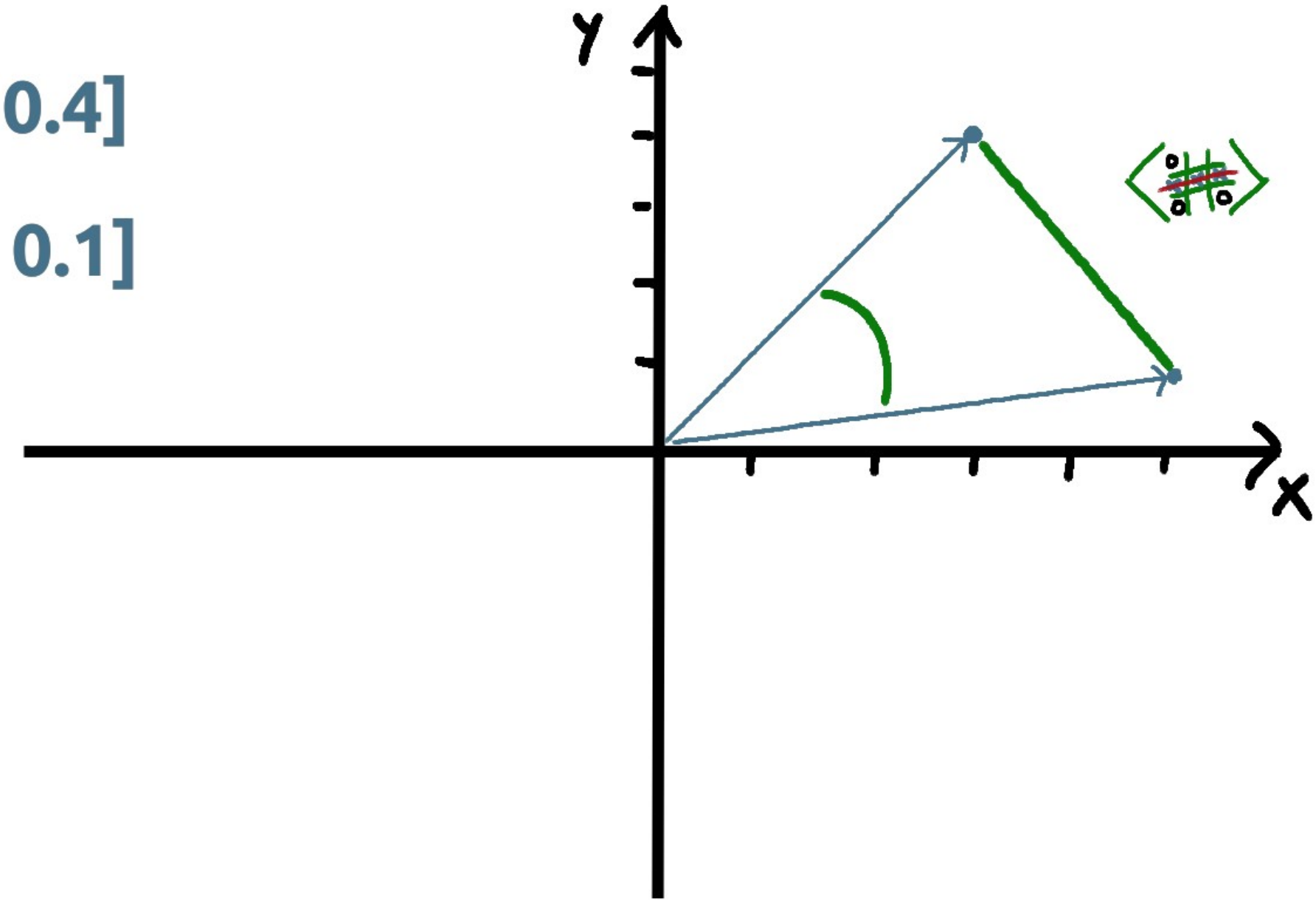
[0.3, 0.4]

[0.5, 0.1]



[0.3, 0.4]

[0.5, 0.1]



Querying

Querying

```
SELECT embedding <-> '[0.2, 0.1, 0.0]' AS distance
FROM p2d2.pictures;
```

Querying

```
SELECT embedding <-> '[0.2, 0.1, 0.0]' AS distance
FROM p2d2.pictures;

SELECT embedding <+> '[0.2, 0.1, 0.0]' AS l1_distance
FROM p2d2.pictures;
```

Querying

```
SELECT embedding <-> '[0.2, 0.1, 0.0]' AS distance
FROM p2d2.pictures;

SELECT embedding <+> '[0.2, 0.1, 0.0]' AS l1_distance
FROM p2d2.pictures;

SELECT embedding <=> '[0.2, 0.1, 0.0]' AS cosine_simil
FROM p2d2.pictures;
```

Querying

```
SELECT embedding <-> '[0.2, 0.1, 0.0]' AS distance
FROM p2d2.pictures;

SELECT embedding <+> '[0.2, 0.1, 0.0]' AS l1_distance
FROM p2d2.pictures;

SELECT embedding <=> '[0.2, 0.1, 0.0]' AS cosine_simil
FROM p2d2.pictures;

SELECT embedding <#> '[0.2, 0.1, 0.0]' AS inner_prod
FROM p2d2.pictures;
```

Querying

```
SELECT id, filename FROM p2d2.pictures
WHERE filename ~ '/chihuahua/' LIMIT 10;

SELECT id, filename FROM p2d2.pictures
WHERE filename ~ '/muffin/' LIMIT 10;


SELECT embedding <-> (SELECT embedding
                      FROM p2d2.pictures WHERE id=1)
FROM p2d2.pictures WHERE id = 2600;
```

Lab – Part IV

indexing

pgvector has indexing

pgvector has indexing

- HNSW
- IVFFlat

pgvector has indexing

- HNSW

  Hierarchical Navigable Small Worlds
- IVFFlat

pgvector has indexing

- HNSW

  Hierarchical Navigable Small Worlds
- IVFFlat

  InVerted File with Flat compression

pgvector has indexing

- HNSW

  Hierarchical Navigable Small Worlds
- IVFFlat

  InVerted File with Flat compression

  Inverted lists

Creating indexes for vectors

Creating HNSW indexes for vectors

```
CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_l2_ops);
```

Creating HNSW indexes for vectors

```
CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_l2_ops);

CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_l1_ops);
```

Creating HNSW indexes for vectors

```
CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_l2_ops);

CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_l1_ops);

CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_cosine_ops);
```

Creating HNSW indexes for vectors

```
CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_l2_ops);

CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_l1_ops);

CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_cosine_ops);

CREATE INDEX ON p2d2.pictures
USING hnsw (embedding vector_ip_ops);
```

Creating IVFFlat indexes for vectors

```
CREATE INDEX ON p2d2.pictures
USING ivfflat (embedding vector_l2_ops);

CREATE INDEX ON p2d2.pictures
USING ivfflat (embedding vector_l1_ops);

CREATE INDEX ON p2d2.pictures
USING ivfflat (embedding vector_cosine_ops);

CREATE INDEX ON p2d2.pictures
USING ivfflat (embedding vector_ip_ops);
```

Querying

Querying

```
SELECT embedding <-> '[0.2, 0.1, 0.0]' AS distance
FROM p2d2.pictures;

SELECT embedding <+> '[0.2, 0.1, 0.0]' AS l1_distance
FROM p2d2.pictures;

SELECT embedding <=> '[0.2, 0.1, 0.0]' AS cosine_simil
FROM p2d2.pictures;

SELECT embedding <#> '[0.2, 0.1, 0.0]' AS inner_prod
FROM p2d2.pictures;
```

Lab – Part V

The Challenge

Embedding Model

[0.42, 0.1, -0.7, ..., 0.3]

Embedding Model

[0.42, 0.1, -0.7, ..., 0.3]

Embedding Model

[0.43, 0.1, -0,8, ..., 0.33]

[0.42, 0.1, -0.7, ..., 0.3]

Embedding Model

[0.43, 0.1, -0,8, ..., 0.33]

[0.42, 0.1, -0.7, ..., 0.3]

The Challenge

Check `/home/user42/.the_challenge`

`chihuahua_or_muffin.py`

The Challenge

Check /home/user42/.the_challenge

chihuahua_or_muffin.py

connect

close

cursor

fetchall

Vector operator

The Challenge

```
./chihuahua_or_muffin.py                \
    /home/dataset/chihuahua_muffin/challenge/img1a.jpg \
    /var/run/postgresql 5432 p2d2db42 user42
```

The Challenge

```
./chihuahua_or_muffin.py          \
    /home/dataset/chihuahua_muffin/challenge/img1a.jpg \
    /var/run/postgresql 5432 p2d2db42 user42


python3 -m http.server 8042
```

Closing Words

Why not PostgreSQL?

Why not PostgreSQL?

- Relational database designed for OLTP/OLAP
- Multi-Version Concurrency Control

Why not PostgreSQL?

- Relational database designed for OLTP/OLAP
- Multi-Version Concurrency Control
- Vertical Scalability

Why PostgreSQL?

Why PostgreSQL?

- Shortcomings of specialized vector databases
  - Lack of transactional support
  - Difficult to combine queries with other filters

Why PostgreSQL?

- Shortcomings of specialized vector databases
  - Lack of transactional support
  - Difficult to combine queries with other filters
- Integration with existing software

Why PostgreSQL?

- Shortcomings of specialized vector databases
  - Lack of transactional support
  - Difficult to combine queries with other filters
- Integration with existing software
- Integration with existing data

Why PostgreSQL?

- Shortcomings of specialized vector databases
  - Lack of transactional support
  - Difficult to combine queries with other filters
- Integration with existing software
- Integration with existing data
- Established community

Why PostgreSQL?

- Shortcomings of specialized vector databases
  - Lack of transactional support
  - Difficult to combine queries with other filters
- Integration with existing software
- Integration with existing data
- Established community – to be improved

What’s the buzz?

Take Away

- Multi-disciplinary work

Take Away

- Multi-disciplinary work
- pgvector already turns PostgreSQL into a vector database
- pgvector keeps improving

Take Away

- Multi-disciplinary work
- pgvector already turns PostgreSQL into a vector database
- pgvector keeps improving
- Look at the larger picture and collaborate

Thank you