- I will be using VS Code. You can use whatever IDE you prefer. Just setup a *.venv* for it. (See https://code.visualstudio.com/docs/python/environments for VS Code)
  - On VS Code, I added Python extension by Microsoft, from the marketplace.
  - I am using the latest python interpreter
  - From the VS Code search bar I selected *"Show and Run Commands > Python: Create Environment…"* to add *.venv* to my *classcodes* folder.

Big data

Apache Kafka

Consume and Produce from External Codes via API

…Python

- Right click on *.venv/bin/activate* to open on terminal for interaction.
- Run the following commands to install on the .venv environment:
  - *pip install kafka-python-ng* [it should have been *kafka-python* but there is a bug with python 3.12 waiting to be fixed]
- *Add a file kafka/my-python-client-lesson/kafka-consumer.py*
- *Add a file kafka/my-python-client-lesson/kafka-producer.py*
- Add codes for test:
  - [Exercise in class]

Big data

Apache Kafka

Consume and Produce from External Codes via API

…Python

- [Exercise in class]
- 1. Create *kafta/my-python-lesson* folder in your current project environment
- 2. Add *kafka-consumer.py* and *kafka-producer.py* files to the above folder.
- [For your reference on kafka-python, see https://kafka-python.readthedocs.io/en/master/usage.html]

Big data

Apache Kafka

Consume and Produce from External Codes via API

…Python

- …[Exercise in class]
- 3. In *kafka-consumer.py* file, programmatically set up consumer endpoint for your *quickstart-events* topic which you already created for illustration, as shown in the code below:

```
from kafka import KafkaConsumer

consumer = KafkaConsumer('quickstart-events',bootstrap_servers=['localhost:9092'], auto_offset_reset='earliest')

for msg in consumer:
        print (msg.value)
```

Big data

Apache Kafka

Consume and Produce from External Codes via API

…Python

- …[Exercise in class]
- 4. Run the above code from terminal (*python kafka-consumer.py*) and you should see all the events in the topic, printed in the console, as the code instructs.
- 5. Also programmatically produce events with codes in *kafka-producer.py*.

```
from kafka import KafkaProducer


producer =
KafkaProducer(bootstrap_servers=['localhost:9092’])


producer.send('quickstart-events', b'message-from-python’)


for n in range(3):
        message = 'additional message {} from
        python'.format(n)
        producer.send('quickstart-events', bytearray(message,
        'utf-8’))


producer.flush() # producer.send() is asynchronous. Flush all
messages to topic.
```

- …[Exercise in class]
- Notice that so far, for serialization and deserialization, we have only worked with strings which are serialized as *bytearrays*.
- We can also work with other formats e.g. *json*, *msgpack* (see https://msgpack.org/index.html).
- For serializing and deserializing more structured data e.g. to and fro dbms, we need tools like *avro* (https://avro.apache.org/), *protocol buffer* (https://protobuf.dev/).

Big data

Apache Kafka

Consume and Produce from External Codes via API

…Python

- …[Exercise in class]
- Let's work with *json*, *msgpack*. We will work with *avro* and perhaps *protocol buffer* later, when we have dealt with how to register the schemas, using suitable schema registry.
- 5. From console, create the topics named *json-events* and *msgpack-events* which we can use for illustration.

bin/kafka-topics.sh --create --topic json-events --bootstrap-server localhost:9092

bin/kafka-topics.sh --create --topic msgpack-events --bootstrap-server localhost:9092

Big data

Apache Kafka

Consume and Produce from External Codes via API

…Python

- …[Exercise in class]
- 6. From your terminal with *.venv* activated, *pip install msgpack*
- 7. Add the following codes respectively to your python files
  - *Kafka-producer.py*
    - *import msgpack*
    - *import json*
    - *# encode objects via msgpack. See https://msgpack.org/index.html*
    - *producer2 = KafkaProducer(value_serializer=msgpack.dumps)*
    - *producer2.send('msgpack-events', {'msgpack_key': 'msgpack_value'})*
    - *producer2.flush() # block until all async messages are sent*
    - *# produce json messages*
    - *producer3 = KafkaProducer(value_serializer=lambda m: json.dumps(m).encode('ascii'))*
    - *producer3.send('json-events', {'json_key': 'json_value'})*
    - *producer3.flush() # block until all async messages are sent*

Big data

Apache Kafka

Consume and Produce from External Codes via API

…Python

- …[Exercise in class]
- 8. Each consumer needs its own process. Create files for each of json and msgpack and add the respective codes
  - ***kafka-consumer-json.py***

```
import json
from kafka import KafkaConsumer

consumer = KafkaConsumer('json-events',value_deserializer=lambda m:
json.loads(m.decode('ascii')), auto_offset_reset='earliest')
for msg in consumer:
        print (msg.value)
```

  - ***kafka-consumer-msgpack.py***

```
import msgpack
from kafka import KafkaConsumer

consumer = KafkaConsumer('msgpack-events',
value_deserializer=msgpack.unpackb, auto_offset_reset='earliest')
for msg in consumer:
                print (msg.value)
```

- 9. Exercise: Run each of the above in separate terminals to see the events displayed. Take snapshots and send to me via eLearning portal

Big data

Apache Kafka

Consume and Produce from External Codes via API

…Python

- Use cases discussion up to this point.
  - Consume, transform and send to another topic
  - Consume, transform and send to DBMS
  - Custom Application Process generate and produce to queue
  - …