CS3237 Introduction to IoT
Lab 4
ANSWER BOOK

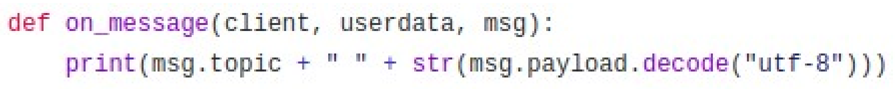
Student ID: A0183217H	Name: Ng Jing Kiat
Student ID: A0184893L	Name: Ng Wei Jie, Brandon

<u>Question 1</u> (2 MARKS)

According the Python3 documentation, bytes literals are always prefixed with ‘b’ or ‘B’. The prefix ‘b’ indicates that the object is an instance of bytes type instead o string type. The ‘b’ prefix is there because MQTT is a binary based protocol where the control elements are binary bytes and not text strings. The payload received in the on_message callback is binary data.

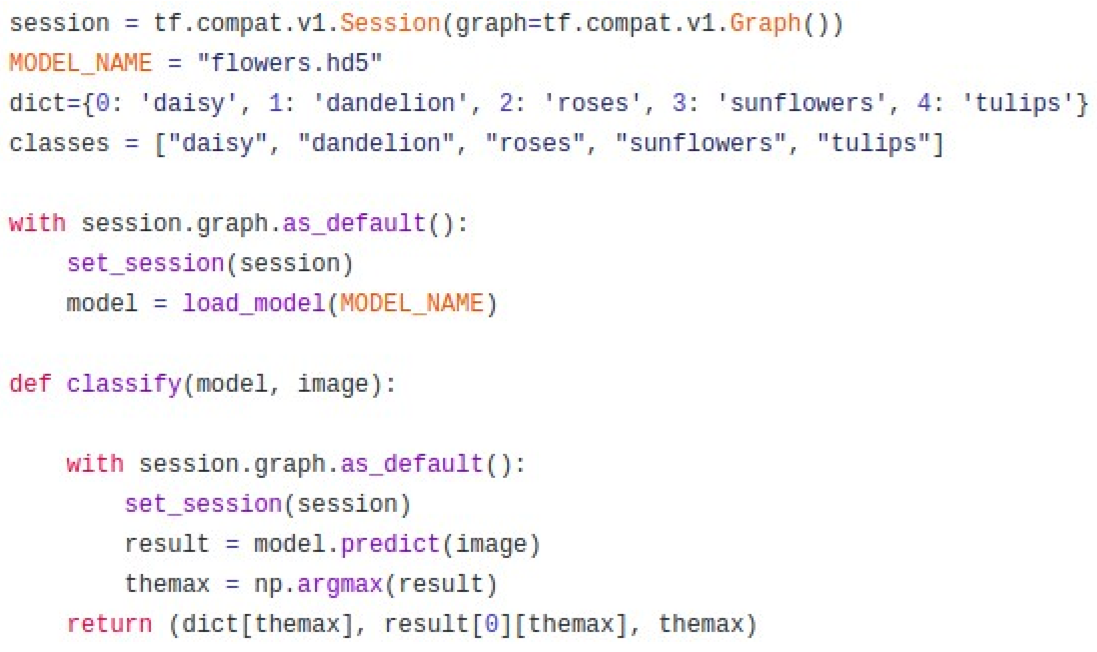
<u>Question 2</u> (3 MARKS)

To remove the prefix ‘b’, the payload needs to be converted from bytes to utf-8 before print it.



```
def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.payload.decode("utf-8")))
```

<u>Question 3 (2 marks)</u>



```
session = tf.compat.v1.Session(graph=tf.compat.v1.Graph())
MODEL_NAME = "flowers.hd5"
dict={0: 'daisy', 1: 'dandelion', 2: 'roses', 3: 'sunflowers', 4: 'tulips'}
classes = ["daisy", "dandelion", "roses", "sunflowers", "tulips"]

with session.graph.as_default():
    set_session(session)
    model = load_model(MODEL_NAME)

def classify(model, image):

    with session.graph.as_default():
        set_session(session)
        result = model.predict(image)
        themax = np.argmax(result)
    return (dict[themax], result[0][themax], themax)
```

I place the lines of code to load the model outside a function such that when I run the python file, the model will get loaded before the MQTT connection is made. Everything a prediction is needed, the same model that was loaded at the start of the program is used.

Question 4 (2 mark)

In section3, we froze the weights when we loaded our model because we want to freeze the weights in those layers but update the weights of the layers that we added.

In the current case, I do not need to freeze the weights because I am not training the model. I am loading the model to predict and classify images so the weights are not updated in the process.

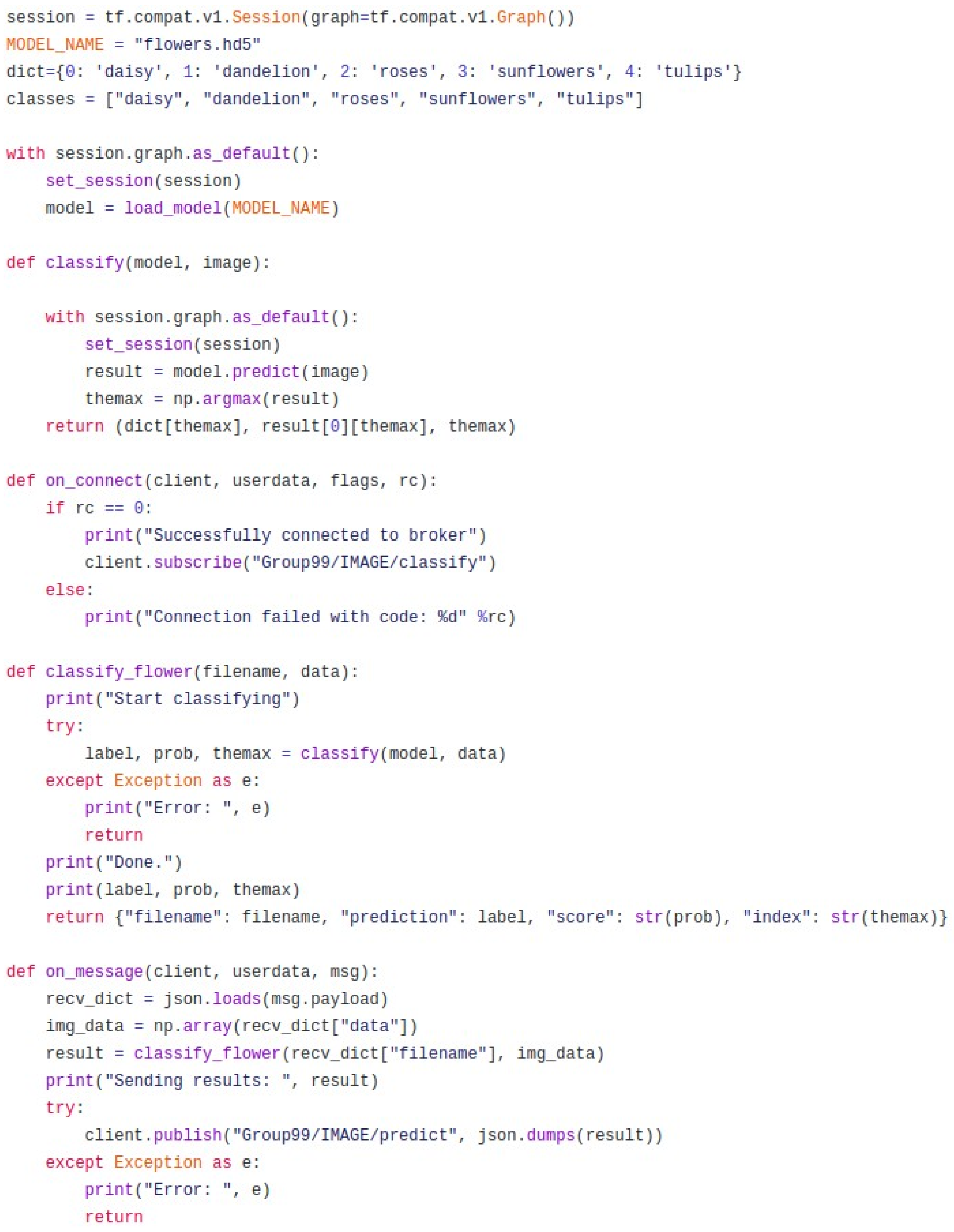
Question 5 (4 marks)

When a message is received, the callback is triggered and will call the function classify_flower that is responsible for returning the predictions of the image.

Inside the function classify_flower, a try and catch exception is added for the function classify as the MQTT failed silently if an exception occurs. The actual predictions are made in the function classify.

In the function classify, I first set the session as it is to resolve a asynchronous bug issue in Keras as mentioned in the lab. I then used the model that was loaded when the program is first run before MQTT connection is setup. The model is used to predict the class for a given image. The results an array of 10 values containing the statistical distribution of the probabilities that the image belong to the class. The index of the high probabilities is obtained. The actual label in string, probability and the hot encoding of the label (index in the array).

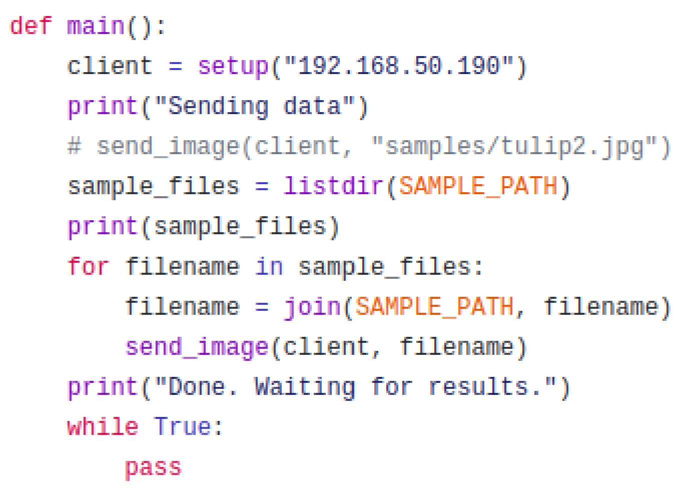
The function classify_flower will return a json of the label, probability and the index with the values casted to string; otherwise, there will be a json decoder error.



```python
session = tf.compat.v1.Session(graph=tf.compat.v1.Graph())
MODEL_NAME = "flowers.hd5"
dict={0: 'daisy', 1: 'dandelion', 2: 'roses', 3: 'sunflowers', 4: 'tulips'}
classes = ["daisy", "dandelion", "roses", "sunflowers", "tulips"]

with session.graph.as_default():
    set_session(session)
    model = load_model(MODEL_NAME)

def classify(model, image):

    with session.graph.as_default():
        set_session(session)
        result = model.predict(image)
        themax = np.argmax(result)
    return (dict[themax], result[0][themax], themax)

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Successfully connected to broker")
        client.subscribe("Group99/IMAGE/classify")
    else:
        print("Connection failed with code: %d" %rc)

def classify_flower(filename, data):
    print("Start classifying")
    try:
        label, prob, themax = classify(model, data)
    except Exception as e:
        print("Error: ", e)
        return
    print("Done.")
    print(label, prob, themax)
    return {"filename": filename, "prediction": label, "score": str(prob), "index": str(themax)}

def on_message(client, userdata, msg):
    recv_dict = json.loads(msg.payload)
    img_data = np.array(recv_dict["data"])
    result = classify_flower(recv_dict["filename"], img_data)
    print("Sending results: ", result)
    try:
        client.publish("Group99/IMAGE/predict", json.dumps(result))
    except Exception as e:
        print("Error: ", e)
        return
```

Question 6 (4 marks)

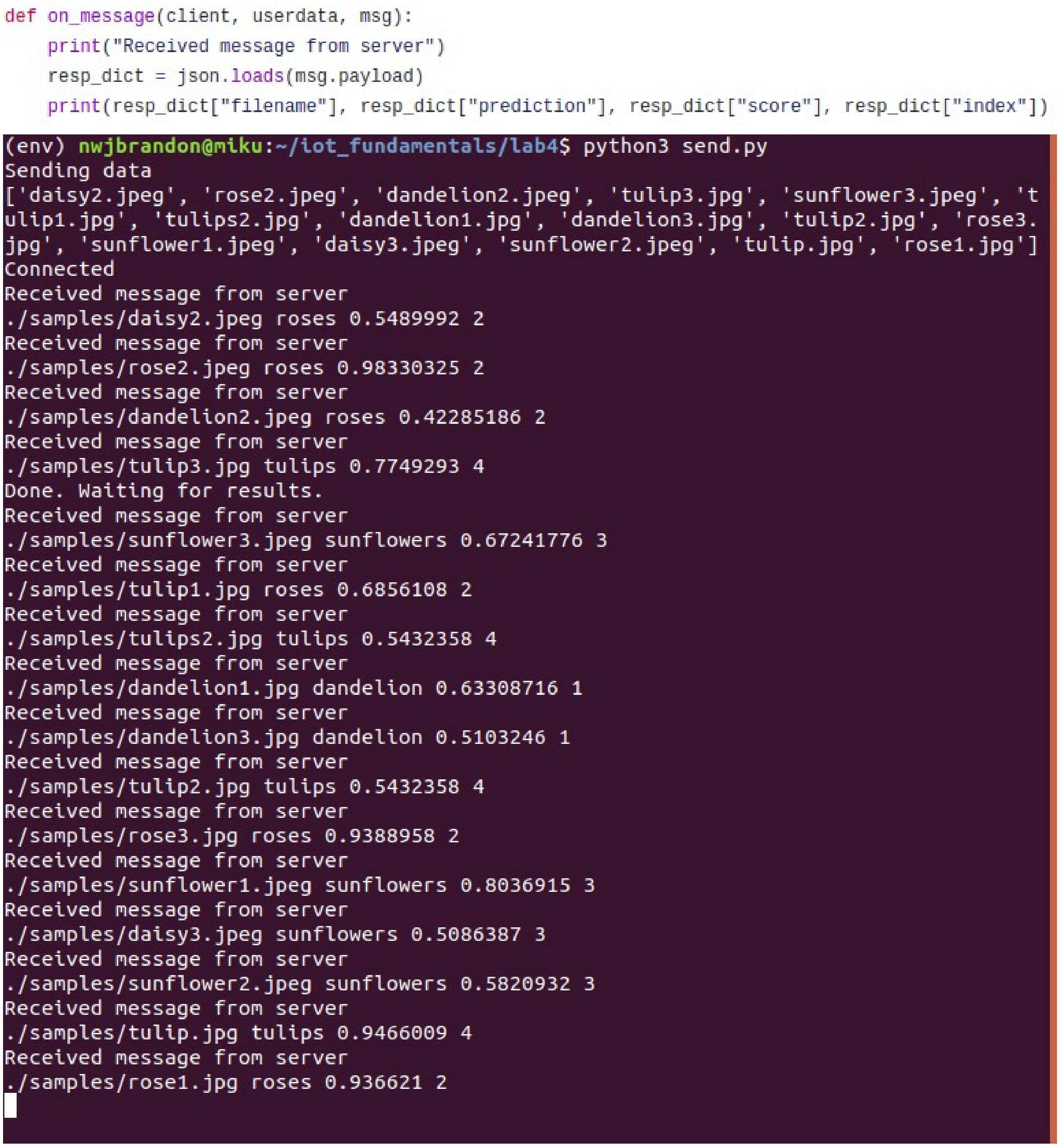
The sample images are stored in the directory samples/ in the same project directory level as the sender.py file. I first list the items in the directory samples/ and store the file names in the sample_files variable. I then iterate through each file name and join the path to the file name to get the full path to the file so that the send.py can read the image and send the file name and image through the send_image function.



```
def main():
    client = setup("192.168.50.190")
    print("Sending data")
    # send_image(client, "samples/tulip2.jpg")
    sample_files = listdir(SAMPLE_PATH)
    print(sample_files)
    for filename in sample_files:
        filename = join(SAMPLE_PATH, filename)
        send_image(client, filename)
    print("Done. Waiting for results.")
    while True:
        pass
```

Question 7 (3 marks)

The console output using the on_message in send.py is stated below. It prints the filename, label, probability and the index.



```python
def on_message(client, userdata, msg):
    print("Received message from server")
    resp_dict = json.loads(msg.payload)
    print(resp_dict["filename"], resp_dict["prediction"], resp_dict["score"], resp_dict["index"])
```

```
(env) nwjbrandon@miku:~/iot_fundamentals/lab4$ python3 send.py
Sending data
['daisy2.jpeg', 'rose2.jpeg', 'dandelion2.jpeg', 'tulip3.jpg', 'sunflower3.jpeg', 't
ulip1.jpg', 'tulips2.jpg', 'dandelion1.jpg', 'dandelion3.jpg', 'tulip2.jpg', 'rose3.
jpg', 'sunflower1.jpeg', 'daisy3.jpeg', 'sunflower2.jpeg', 'tulip.jpg', 'rose1.jpg']
Connected
Received message from server
./samples/daisy2.jpeg roses 0.5489992 2
Received message from server
./samples/rose2.jpeg roses 0.98330325 2
Received message from server
./samples/dandelion2.jpeg roses 0.42285186 2
Received message from server
./samples/tulip3.jpg tulips 0.7749293 4
Done. Waiting for results.
Received message from server
./samples/sunflower3.jpeg sunflowers 0.67241776 3
Received message from server
./samples/tulip1.jpg roses 0.6856108 2
Received message from server
./samples/tulips2.jpg tulips 0.5432358 4
Received message from server
./samples/dandelion1.jpg dandelion 0.63308716 1
Received message from server
./samples/dandelion3.jpg dandelion 0.5103246 1
Received message from server
./samples/tulip2.jpg tulips 0.5432358 4
Received message from server
./samples/rose3.jpg roses 0.9388958 2
Received message from server
./samples/sunflower1.jpeg sunflowers 0.8036915 3
Received message from server
./samples/daisy3.jpeg sunflowers 0.5086387 3
Received message from server
./samples/sunflower2.jpeg sunflowers 0.5820932 3
Received message from server
./samples/tulip.jpg tulips 0.9466009 4
Received message from server
./samples/rose1.jpg roses 0.936621 2
```

The accuracy of my classify is 0.75

TOTAL: _______ / 20