Cheatsheet for schweizer jugend forscht project

Tkinter

Tkinter is Python's standard GUI (Graphical User Interface) library, providing tools to create simple and interactive desktop applications. It offers:

- Widgets: Pre-built components like buttons, labels, text boxes, and menus for building interfaces.
- Geometry Management: Methods like .pack(), .grid(), and .place() to arrange widgets in the application window.
- Event Handling: Mechanisms to respond to user actions like clicks or keypresses.

Tkinter is lightweight, easy to use, and well-suited for creating small to medium desktop applications. It comes pre-installed with Python on most platforms.

Additional cheatsheet: https://cdn.activestate.com/wp-content/uploads/2021/02/Tkinter-CheatSheet.pdf

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

You can create a variety of different plots, for example line plots (like shown in the minimal example), bar charts, scatter plots etc.

You can find additional information and tons of examples at matplotlib.org.

Plotting:

Minimal example:

```
import matplotlib.pyplot as plt
import numpy as np

# use numpy library to generate data
times = np.linspace(0.0, 10.0, num=50)
values = np.sin(times)

# plot the data using matplotlib
plt.plot(times, values, label="Test
Data")

# show legend automatically
plt.legend()

# setup labels
plt.xlabel("Time")
plt.ylabel("Values")

# after a plot is created, it will not
automatically be shown. You can either
show it or save it
plt.show()
# plt.savefig("res.pdf")
# canvas.draw() -> tkinter
```

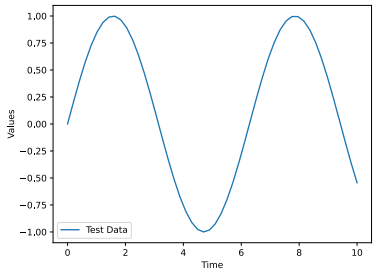

Figure 1: output of minimal example

Combination of Matplotlib and Tkinter:

```
import tkinter as tk
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg
import FigureCanvasTkAgg

# initialize tkinter object
root = tk.Tk()

# create some tkinter content and a
frame to include the plot
tk.Label(root, text="This is some
tkinter content").pack()
figure_frame = tk.Frame(root)
figure_frame.pack()
tk.Label(root, text="This is other
tkinter content").pack()

# initialize plots
fig, ax = plt.subplots()

# initialize canvas
canvas = FigureCanvasTkAgg(fig,
master=figure_frame)

# pack canvas in tkinter (also possible
with grid)
canvas.get_tk_widget().pack()

# create some values to plot
times = np.linspace(-1.0, 2.0, num=200)
values = np.exp(times)

# create some figures
ax.plot(times, values,
label="exponential function")

# show legend
ax.legend()

# update canvas
canvas.draw()

# run mainloop to start tkinter
root.mainloop()
```

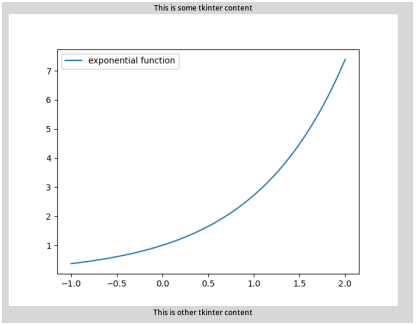

Figure 2: output of tkinter matplotlib example

Paho-mqtt

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed for efficient communication in low-bandwidth or resource-constrained environments. It follows a publish-subscribe model, where:

- Broker: A central server (e.g., Mosquitto) manages all message exchanges.
- Publishers: Devices that send messages to specific "topics."
- Subscribers: Devices that receive messages by subscribing to relevant topics.

The broker ensures that messages published to a topic are delivered to all subscribed clients.

There are three different QoS (Quality of Service) levels:

- level 0, at most once:

Each message is sent without further follow-up or confirmation. Therefore, it is possible that messages will get missing. The advantage of this option is that it is very lightweight. If the messages are not that important (for example if they are observations and missing ones don't matter that much), this can be used. Therefore, we will use it in this project

- level 1, at least once:

The sender keeps a copy of the messages and sends them until he receives a notification that the subscriber got the message. In this case, it is guaranteed that the subscriber will receive the message, but it needs a bigger overhead to do so. This could also be used in out project.

- level 2, exactly once:

A more complicated algorithm ensures that each message is sent exactly once. It needs more overhead than QoS 1, but it does not need to send messages several times.

Our Usecase

In our project, the broker is already set up and can be reached over the student_project_network network on 192.168.2.1:1883

You will use subscribers to get messages (temperature and energy information) as well as a publisher to change the relay state.

We will use the paho-mqtt library for subscribers and publishers.

This is a lightweight python library, which which you can connect to a broker and handle the messages with very few lines of code.

We will not be setting up a broker because this would be outside of the scope of this project.

Following minimal examples can be expanded for our usecases:

Subscriber

Minimal example:

```
import paho.mqtt.client as mqtt

""" handle incoming messages, which are
received as bytestring in
message.payload.
Without writing an on_message function,
the message gets ignored.
For example, you could encrypt and
store messages here  """
def own_on_message_function(client,
userdata, message):
    if message.topic == "own_topic":
        encrypted_message =
encrypt(message.payload)
        store_data(encrypted_message)
    else:
        print(
            f"message on topic
{message.topic} and message
{message.payload} received!"
        )

""" this function gets called whenever
the client connects to a broker.
You can subscribe to topics here """
def own_on_connect_function(client,
userdata, flags, rc):
    client.subscribe("own_topic")
    print(f"connected with result code
{rc}")

# initialize client
client = mqtt.Client()

# assign own function
client.on_connect =
own_on_connect_function
client.on_message =
own_on_message_function

""" connect to the broker. keepalive
means how long a client has to try to
reconnect when it lost connection or
something failed """
client.connect(host="192.168.2.1",
port=1883, keepalive=60)
```

Publisher

The publisher works similarly to the subscriber, with the difference that it can control by itself when to publish stuff.
Therefore it is not necessary to define the on_connect and on_message functions (You could add on_connect if it is important to do something whenever a connection is established)

Minimal example:

```
import paho.mqtt.client as mqtt
#comments?
client = mqtt.Client()
client.connect(host="192.168.2.1",
port=1883, keepalive=60)

if send_some_message:
    client.publish(
        topic="some_topic",
        payload=str(some_values),
        qos=0,
    )
```

Sending messages

Messages are always sent as bytestrings in mqtt (can be interpreted as string, therefore text)

Therefore, you have to somehow convert your messages to strings and be able to convert them back.

One way to do this is to use the struct package.

You can simply use `struct.pack(format, ...)` to convert several numbers into a string and `struct.unpack()` to convert them back.

Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

It simplifies working with structured data like CSVs, databases, or Excel files.

Therefore it is useful to store our results data into a file and handle non-numeric data (like date strings)

Numpy

NumPy (Numerical Python) is an open source Python library that's widely used in science and engineering.
The NumPy library contains large library of functions that operate efficiently on data structures of numerical data like matrizes or arrays.
For example, the library can handle fast matrix operations, statistical functions or numerical integration, which unfortunately all lies outside of the scope of this project.

It can be useful in combination with pandas, for example to analyze a pandas dataframe

---

Debugging in python

Common Errors:

- IndentationError: Check for consistent spacing. For example, after each if statement, there has to be a block with indentation (one tab more)
- NameError: Ensure variables are defined before use. Is the variable only defined outside of this function / scope?

Tips:

- Use IDEs like pycharm to autodetect errors. They also have debugging tools included
- Use built-in functions instead of writing everything by yourself
- Understand what you are writing (expecially if you are using ChatGPT or similar models)