Tutorial IX - Data Visualization

Programming with Python

Today's Tutorial

From now on, it is rather difficult to give you tasks that you cannot solve within minutes by yourself with the help of AI. Therefore, today's tutorial will be more open-ended than usual. All tasks are designed to be solved together with AI, but you are nonetheless the human in the loop. Thus, your responsilities are to control the AI, make the plots meaningful and visually attractive.

Simulation of a random walk for stock prices

In this excercise, you will create a program that is able to simulate a random walk for stock prices. The program should ask the user for the initial stock price, the number of days to simulate and the daily standard deviation of the stock price. Then, the program should simulate 5 random walks of the price and visualize the stock price over time. Feel free to use Zed to help you write the code and visualize the results in an attractive way.

```
# TODO: Create a program that is able to simulate and visualize a random
walk.
# YOUR CODE HERE
```

Plotting data from the World Bank

Now, let's plot some data that we are not creating ourselves. Head to the open data website of the World Bank and find some data you are personally interested in. Then, download the data set and load it into a `pandas` DataFrame. Finally, plot the data.

```
# TODO: Plot some interesting data from the World Bank.
# YOUR CODE HERE
```

Building an investment calculator with PySide6

For this exercise, you'll create a simple investment calculator application using PySide6. This tool will help users calculate the future value of their investments based on different parameters.

Your task is to create a GUI application that:

1. Allows users to input:
   - Initial investment amount
   - Annual contribution
   - Expected return rate (%)
   - Investment period (years)

2. Calculates the future value of the investment
3. Displays the result in a formatted way

Here's a solution for tasks 1-3 to get you started:

```
import sys
from PySide6.QtWidgets import (QApplication, QMainWindow, QWidget,
QVBoxLayout,
                               QLabel, QLineEdit, QPushButton, QGridLayout)
from PySide6.QtCore import Qt

class InvestmentCalculator(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Investment Calculator")
        self.setMinimumSize(400, 300)

        # Create central widget and layout
        central_widget = QWidget()
        self.setCentralWidget(central_widget)
        layout = QGridLayout(central_widget)

        # Create input fields
        self.initial = QLineEdit()
        self.annual = QLineEdit()
        self.rate = QLineEdit()
        self.years = QLineEdit()

        # Create labels
        layout.addWidget(QLabel("Initial Investment ($):"), 0, 0)
        layout.addWidget(self.initial, 0, 1)
        layout.addWidget(QLabel("Annual Contribution ($):"), 1, 0)
        layout.addWidget(self.annual, 1, 1)
        layout.addWidget(QLabel("Expected Return Rate (%):"), 2, 0)
        layout.addWidget(self.rate, 2, 1)
        layout.addWidget(QLabel("Investment Period (years):"), 3, 0)
        layout.addWidget(self.years, 3, 1)

        # Create calculate button
        calc_button = QPushButton("Calculate")
        calc_button.clicked.connect(self.calculate)
        layout.addWidget(calc_button, 4, 0, 1, 2)

        # Create result label
        self.result_label = QLabel()
        self.result_label.setAlignment(Qt.AlignCenter)
        layout.addWidget(self.result_label, 5, 0, 1, 2)

    def calculate(self):
        try:
            initial = float(self.initial.text())
            annual = float(self.annual.text())
            rate = float(self.rate.text()) / 100
            years = int(self.years.text())
```

```
            # Calculate future value
            future_value = initial
            for _ in range(years):
                future_value = (future_value + annual) * (1 + rate)

            # Display result
            self.result_label.setText(
                f"Future Value: €{future_value:,.2f}"
            )
        except ValueError:
            self.result_label.setText("Please enter valid numbers")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = InvestmentCalculator()
    window.show()
    sys.exit(app.exec())
```

Try running this program and then enhance it with additional features such as:

- Including a reset button
- Adding a table showing year-by-year breakdown
- Creating a graph showing the investment growth
- Adding a dropdown menu to select different currencies

Plotting Stocks with a Dashboard

Finally, let's plot some stock data by building a simple Dashboard. To do so, you can either use the `streamlit` library, the `nicegui` library or the `dash` library. All of them are very popular libraries for building Dashboards in Python.

1. Choose a stock of your interest and note down the ticker symbol.
2. Use the `yfinance` library to get the stock data.
3. Plot the stock data in a web app.

```
# TODO: Plot some stock data with Dash, Streamlit or NiceGUI.
# YOUR CODE HERE
```

If you have some time left, try to enhance your Dashboards with additional features.

- Adding a menu to switch between different stocks
- Adding a graph to compare multiple stocks

That's it!

After a week, you can find the solutions to these exercises online in the associated GitHub repository, but we will also quickly go over them in next week's tutorial. To access the solutions, click on the Github button on the lower right and search for the folder with today's lecture and tutorial. Alternatively, you can ask ChatGPT or Claude

to explain them to you. Remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities.