Report for HW 1:

1.1 Getting the Data

```
# Formula 1 Tracks Data from  https://en.wikipedia.org/wiki/List_of_Formula_One_circuits as Formula 1
f1_tracks_data = {
    'Circuit': ['Adelaide Street Circuit', 'Ain-Diab Circuit', 'Aintree Motor Racing Circuit',
                'Albert Park Circuit *', 'Algarve International Circuit', 'Autódromo do Estoril',
                'Autódromo Hermanos Rodríguez *', 'Autódromo Internacional do Rio de Janeiro',
                'Autodromo Internazionale del Mugello', 'Autodromo Internazionale Enzo e Dino Ferrari *',
```

I took different sources of data and created data objects with them. The source is mentioned as a comment of every data.

1.2 Saving the data and scaling

```
2 usages
def save_data(df, filename_prefix):
    df.to_xml(f"{filename_prefix}.xml", index=False)
    df.to_json(f"{filename_prefix}.json", orient="records")

1 usage
def fill_and_scale_data(df):
    from sklearn.preprocessing import StandardScaler
    df = df.fillna(df.mean(numeric_only=True))
    numeric_cols = df.select_dtypes(include=[np.number])
    if not numeric_cols.empty:
        scaler = StandardScaler()
        df[numeric_cols.columns] = scaler.fit_transform(numeric_cols)
    return df
```

Afterwards I scale the data using the StandardScaler and save them as a XML and JSON

1.3 Creating the pivot table

```
1 usage
def create_pivot_table(df, index, values, aggfunc):
    pivot_table = df.pivot_table(index=index, values=values, aggfunc=aggfunc)
    save_data(pivot_table, filename_prefix: f"pivot_{index}_vs_{values}")
    return pivot_table
```

I reuse the save_data method to save the newly created pivot table

2. Basic analysis and save to txt and pdf

```
1 usage
def statistical_analysis(df):
    stats_summary = df.describe()
    correlation_matrix = df.corr()

    with open("stats_summary.txt", "w") as file:
        file.write(f"Statistical Summary:\n{stats_summary}\n\nCorrelation Matrix:\n{correlation_matrix}\n")

    pdf = FPDF()
    pdf.add_page()
    pdf.set_font( family: "Arial", size=12)
    pdf.cell( w: 200,  h: 10, txt="Statistical Summary and Correlation Matrix", ln=True, align="C")
    pdf.cell( w: 200,  h: 10, txt=str(stats_summary), ln=True, align="L")
    pdf.cell( w: 200,  h: 10, txt=str(correlation_matrix), ln=True, align="L")
    pdf.output("stats_summary.pdf")
```

3. Create visualisation

```
3 usages
def plot_visualization(df, x_column, y_column, plot_type, filename):
    plt.figure(figsize=(10, 6))
    if plot_type == 'bar':
        sns.barplot(x=x_column, y=y_column, data=df)
    elif plot_type == 'histogram':
        sns.histplot(df[x_column], kde=True)
    elif plot_type == 'scatter':
        sns.scatterplot(x=x_column, y=y_column, data=df)

    plt.title(f'{plot_type.title()} of {x_column} vs {y_column}')
    plt.xlabel(x_column)
    plt.ylabel(y_column)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig(f"{filename}.jpg")
    plt.close()

    # Save plot to PDF
    pdf = FPDF()
    pdf.add_page()
    pdf.image( name: f"{filename}.jpg", x=10, y=8, w=180)
    pdf.output(f"{filename}.pdf")
```

I created a bar, hist and scatter plot fir the data and save it to a pdf and jpg

Unfortunatly the track data is somewhat wrong therefore only the driver data is working correctly .