Dynamic Modelling Course - TEP4290: Warm-up 10

The point of this exercise is for you to practice what you have learned in the recommended video:

- https://www.youtube.com/watch?v=0P7QnlQDBJY (especially between time codes 3:48 and 21:52). The rest of the video is well done as well so do not hesitate to jump to other parts to have a look.
- cheat sheets: https://matplotlib.org/cheatsheets/

Good luck!

About Matplotlib

Matplotlib is a plotting library for the Python programming language.

Tasks

Import the package

To use the functionalities offered by the Matplotlib.pyplot package, you must first load it. Usually, it is imported under the alias **plt**.

Import Matplotlib.pyplot under the alias "**plt**".

In [1]:
```
import matplotlib.pyplot as plt
```

Recap of Warm-up 09

In the recommended video, the plots are based on the columns of a Pandas dataframe imported via a csv file. In this warm-up, we will work with NumPy arrays computed in your work on Warm-up 9. The functions seen in the video work in the same way for both types of objects, so this will not be a problem.

As a reminder, in Warm-up 9, you extracted information from a csv file which contained information about inflows of cars over time. From this, and based on a sharp lifetime of four years, you computed the outflows of cars per year.

There are various ways to use the output of another Jupyter notebook including (but not restricted to):

- Pickling
- Importing the other notebook. Note that this would require to write the entire other notebook as a class or a function.
- Exporting the result of the other notebook to a csv file and reading it in this notebook.
- Writing the instruction to run this other notebook in your code.

For this work, we will make things simple and just create two variables with the correct results that were expected so you can use the results even if you did not manage to reach the last task in Warm-up 09. Feel free however to use any of the other methods to import your own output in this notebook.

In [2]:

```
import numpy as np

inflows = np.array([1.15786405e+03, 3.08565102e+03, 1.31464458e+03, 4.84354551e+03,
                   3.57574517e+03, 2.61054240e+03, 6.04938817e+03, 3.19706580e+03,
                   5.75397706e+03, 6.52657867e+03, 7.19272301e+02, 3.13287589e+03,
                   3.67418264e+03, 3.54452598e+03, 6.04292880e+03, 9.36723546e+03,
                   6.21229886e+03, 4.26659473e+03, 4.41997355e+03, 4.86332309e+03,
                   1.07891040e+04, 1.21971060e+04, 2.02880069e+04, 2.33739903e+04,
                   2.75679497e+04, 3.47901283e+04, 3.47669415e+04, 2.84238818e+04,
                   3.81884417e+04, 4.40034827e+04, 6.10080218e+04, 6.29519281e+04,
                   4.54718391e+04, 4.69895658e+04, 9.79297024e+04, 2.46926421e+05,
                   1.89560034e+05, 1.69567029e+05, 2.13787842e+05, 1.89120253e+05,
                   1.90843253e+05, 2.69124686e+05, 4.54130659e+05, 6.50893682e+05,
                   6.99527911e+05, 7.54783148e+05, 7.87853661e+05, 1.04674934e+06,
                   8.45216718e+05, 9.97514940e+05, 1.30459904e+06, 1.60256501e+06,
                   2.32064613e+06, 3.04394180e+06, 2.90134964e+06, 4.35522219e+06,
                   5.33073384e+06, 6.30587946e+06, 7.06752064e+06, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                   0.00000000e+00])

outflows = np.array([0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00
                   0.00000000e+00, 3.08565102e+03, 1.31464458e+03, 4.84354551e+03,
                   3.57574517e+03, 2.61054240e+03, 6.04938817e+03, 3.19706580e+03,
                   5.75397706e+03, 6.52657867e+03, 7.19272301e+02, 3.13287589e+03,
                   3.67418264e+03, 3.54452598e+03, 6.04292880e+03, 9.36723546e+03,
                   6.21229886e+03, 4.26659473e+03, 4.41997355e+03, 4.86332309e+03,
                   1.07891040e+04, 1.21971060e+04, 2.02880069e+04, 2.33739903e+04,
                   2.75679497e+04, 3.47901283e+04, 3.47669415e+04, 2.84238818e+04,
                   3.81884417e+04, 4.40034827e+04, 6.10080218e+04, 6.29519281e+04,
```

```
                        4.54718391e+04, 4.69895658e+04, 9.79297024e+04, 2.46926421e+05,
                        1.89560034e+05, 1.69567029e+05, 2.13787842e+05, 1.89120253e+05,
                        1.90843253e+05, 2.69124686e+05, 4.54130659e+05, 6.50893682e+05,
                        6.99527911e+05, 7.54783148e+05, 7.87853661e+05, 1.04674934e+06,
                        8.45216718e+05, 9.97514940e+05, 1.30459904e+06, 1.60256501e+06,
                        2.32064613e+06, 3.04394180e+06, 2.90134964e+06, 4.35522219e+06,
                        5.33073384e+06, 6.30587946e+06, 7.06752064e+06, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                        0.00000000e+00])
```

Plotting

We would like to plot the inflows and the outflows for each year on a same graph. Let's start by defining a NumPy array called *time* containing all the years from 1950 to 2050 (included).

In [5]:
```
time = np.arange(1950, 2051, 1)
print(time)
```

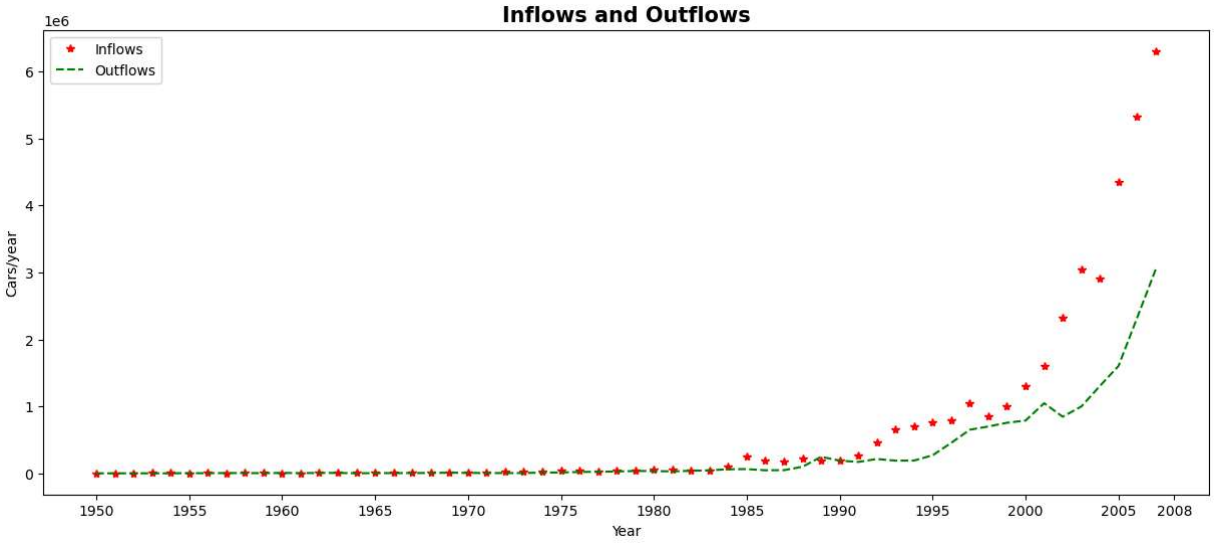
```
[1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963
 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977
 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991
 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005
 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019
 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033
 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047
 2048 2049 2050]
```

Based on these vectors, plot on a same graph the inflows and the outflows as a variable of time. The graph should respect the following formatting:

- The figure size should be 15 by 6
- There shoudl be a title in bold font size 15
- There should be a legend located in the top left corner
- The inflows should be represented in as stars ("******") in red
- The outflows should be represented in a dashed line ("---") in green
- The time range should be restricted 1950 to 2008 only
- Each axis should be named
- The x axis should show years with an increment of 5 years, starting from 1950. 2008 should be added as the last date.

In [10]:
```
start = 1950 - 1950
end = 2008 - 1950
```

```
plt.figure(figsize=(15, 6))
plt.plot(time[start:end], inflows[start:end], 'r*', label='Inflows')
plt.plot(time[start:end], outflows[start:end], 'g--', label='Outflows')
plt.title('Inflows and Outflows', fontsize=15, fontweight='bold')
plt.legend(loc='upper left')
plt.xlabel('Year')
plt.ylabel('Cars/year')
# add 2008 as the last date
plt.xticks(np.append(np.arange(1950, 2009, 5), 2008))
plt.show()
```



Matplotlib also has plenty of other types of charts.To exemplify one of them, we now would like to plot the outflows in a bar chart based on the five-years time period where the cars exited the stock.

For that, we need to group the values by periods of five-year time from 1950 to 2008 (note that the last one will be 2005-2008).

First step: create a list of string in the form of "19XX - 19XX" or "20XX-20XX" corresponding to the five-years time period where the cars exited the stock, starting from 1950.

In [14]:
```
x = []
i = 0

#For that, we need to group the values by periods of five-year time from 1950 to 20
start_year = 1950
end_year = 2008
step = 5

while start_year + i < end_year:
    year_start = start_year + i
    year_end = min(year_start + step, end_year)
    x.append(f'{year_start}-{year_end}')
    i += step

print(x)
```

```
['1950-1955', '1955-1960', '1960-1965', '1965-1970', '1970-1975', '1975-1980', '1980
-1985', '1985-1990', '1990-1995', '1995-2000', '2000-2005', '2005-2008']
```

Second step: create a list containing the outflows grouped by five-years time period (e.g. sum over 1970-1974).

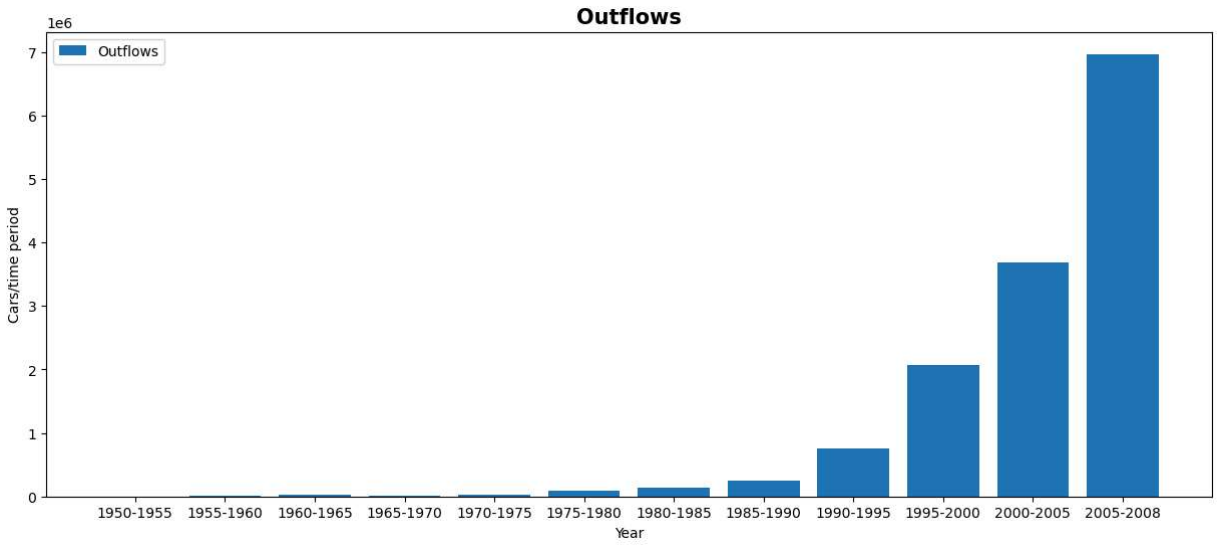
In [18]:
```
outflows_grouped = []

i = 0
for year in x:
    year_start = start_year + i
    year_end = min(year_start + step -1, end_year)
    outflows_grouped.append(outflows[year_start - start_year:year_end - start_year]
    i += step

print(outflows_grouped)
```

```
[np.float64(0.0), np.float64(12819.58628), np.float64(21527.0097), np.float64(16394.
513310000002), np.float64(19762.19023), np.float64(83427.0529), np.float64(145382.74
77), np.float64(253343.0354), np.float64(762035.158), np.float64(2073676.938), np.fl
oat64(3677334.659), np.float64(6967152.9399999995)]
```

Plot the corresponding bar chart. Don't forget to include an adequate formatting (title, axis, figure size etc).

Some extra content on bar chart can be found online (eg. https://www.w3schools.com/python/matplotlib_bars.asp).

In [ ]:
```
plt.figure(figsize=(15, 6))
plt.bar(x, outflows_grouped, label='Outflows')
plt.title('Outflows', fontsize=15, fontweight='bold')
plt.legend(loc='upper left')
plt.xlabel('Year')
plt.ylabel('Cars/time period')
plt.show()
```



```
The Kernel crashed while executing code in the current cell or a previous cell.

Please review the code in the cell(s) to identify a possible cause of the failure.

Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a> for more info.

View Jupyter <a href='command:jupyter.viewOutput'>log</a> for further details.
```

Well done!