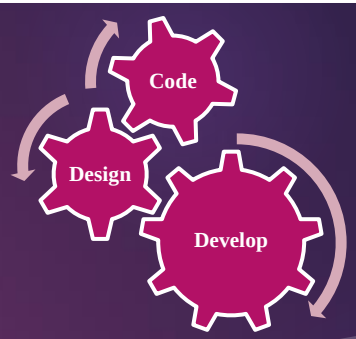
Information Analysis and Visualisation

Lecture 3
Design Principles of Data Visualisation

University of Greenwich
Konstantin Kapinchev

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

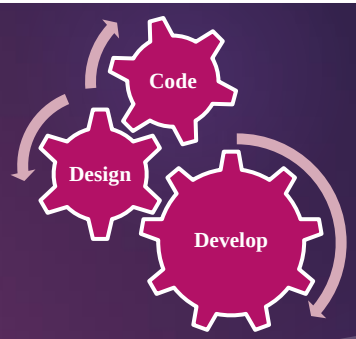


In this lecture:

- Design Principles Overview
- RGB Colour Model
- Visualising Charts
- Line Chart
- Line Chart with Multiple Series
- Bar Chart
- Box Plot Chart
- Conclusions

Konstantin Kapinchev

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

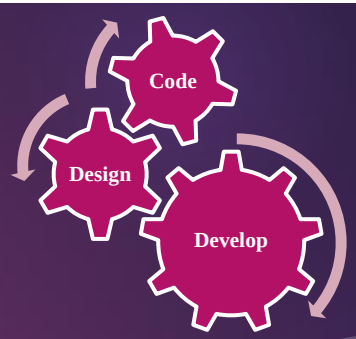


Design Principles Overview

- **Data Visualisation is a multi-disciplinary subject**

- **It has applications in wide range of areas, examples include:**
  - **Media**
  - **Finance**
  - **Health Care**
  - **Politics**
  - **Any private or state-owned enterprise**
  - **And many others ...**

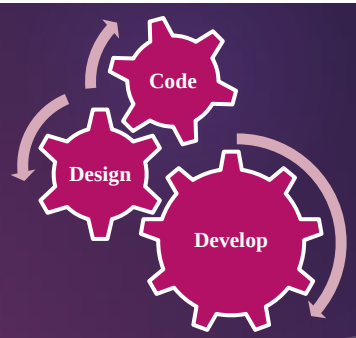
- **Data Visualisation is expected to follow a number of design principles, which ensure its quality**



Design Principles Overview *continued*

- **Key Design Principles[1]:**
  - **Trustworthiness**
  - **Accessibility**
  - **Aesthetics**

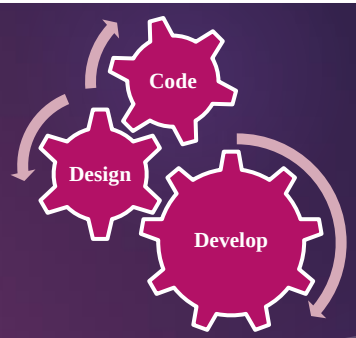
[1]Andy Kirk, "Data Visualisation", page 57



Design Principles Overview *continued*

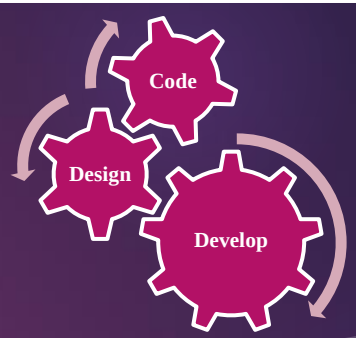
- **Trustworthiness in the context of Data Visualisation**

- **Levels of trustworthiness can be summarised as follows:**
  - **Honest:**
    - **Factually accurate**
    - **Impartially represented**
  - **With agenda:**
    - **To convey specific message**
    - **To enforce an opinion**
  - **Misleading**
    - **Subjective data**
    - **Factually incorrect data**



Design Principles Overview *continued*

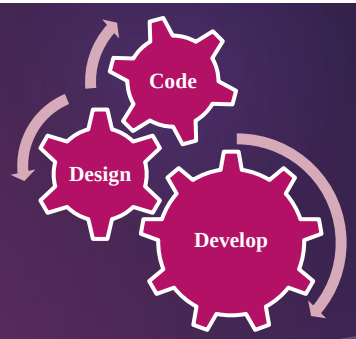
- **Accessibility of Data Visualisation**
  - **Understandable**
    - **Clarity and structure make data visualisation easy to understand**
  - **Relatable**
    - **Data visualisation which is relatable to the intended audience improves its accessibility**
  - **Interactive**
    - **Depending on the circumstances interactive features might improve or reduce the accessibility of the data visualisation**



Design Principles Overview *continued*

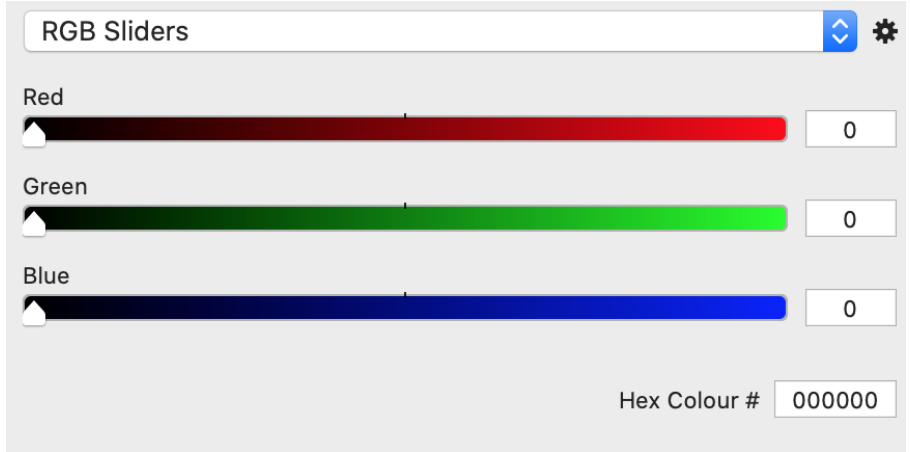
- **Aesthetics in Data Visualisation**
  - **The choice of the following components can have positive or negative effect on the aesthetics of data visualisation:**
    - **Graphical elements**
    - **Colour scheme**
    - **Space utilisation**
    - **Labels (textual or numerical)**

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation



RGB Colour Model

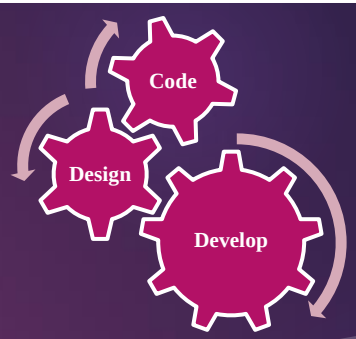
- Setting an appropriate colour of the fonts and the graphics elements is key for successful data visualisation
- Some environments provide set of predefined colours to the users/developers, for example "red", "green", "light blue"
- Other environments support the RGB Colour Model



color = "#FF0000"

- Blue (00 .. FF)
- Green (00 .. FF)
- Red (00 .. FF)

Konstantin Kapinchev



RGB Colour Model *continued*

- **Setting the colour of fonts and graphics elements**
  - **The RGB function with decimal values:**

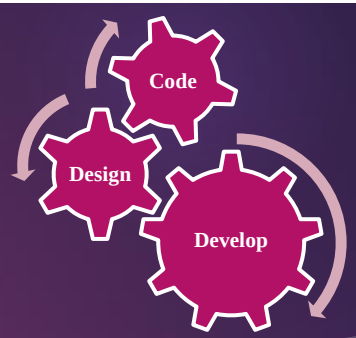
```
color='#{:02x}{:02x}{:02x}'.format(20, 110, 60)
```

Blue (0 .. 255)

Green (0 .. 255)

Red (0 .. 255)

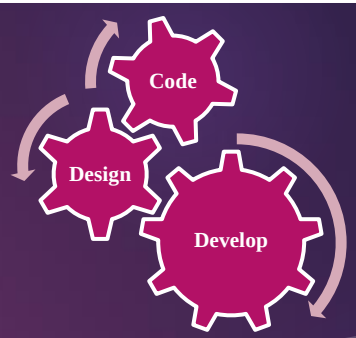
Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation



Visualising Charts

- Charts provide graphical representation of numerical data
- Use various graphics elements to illustrate data points
- Multiple Python libraries provide visualisation functionalities
- The Matplotlib library provides wide range of visualisation tools
- The library provides functionality for visualising different types of charts
- For example: line chart, bar chart and box plot chart

Konstantin Kapinchev

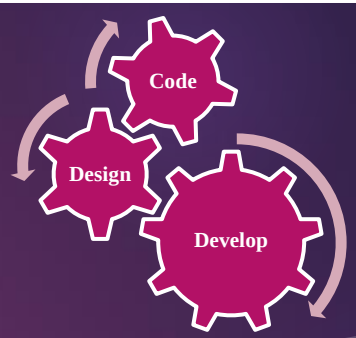


Visualising Charts *continued*

- **This lecture will take a closer look at the following types of charts:**
  - **Line Chart**
  - **Bar Chart**
  - **Box Plot**
- **The charts will visualise the average monthly temperatures of three cities, London, Miami and Anchorage[1]**

[1]Source: Wikipedia

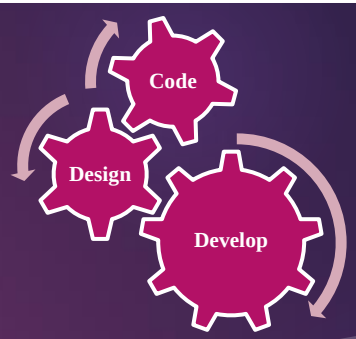
Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation



Line Chart

- One of the most popular type of charts
- Uses lines to connect data points
- Can visualise multiple series of data
  - Each series can be visualised with different colour
  - Data points from each series can be visualised with different graphics element

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

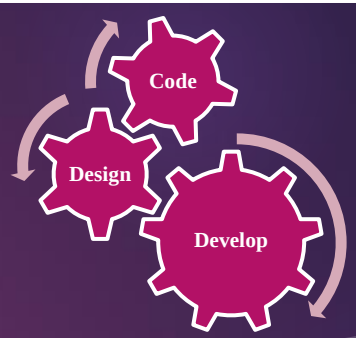


Line Chart

- **Source code:**

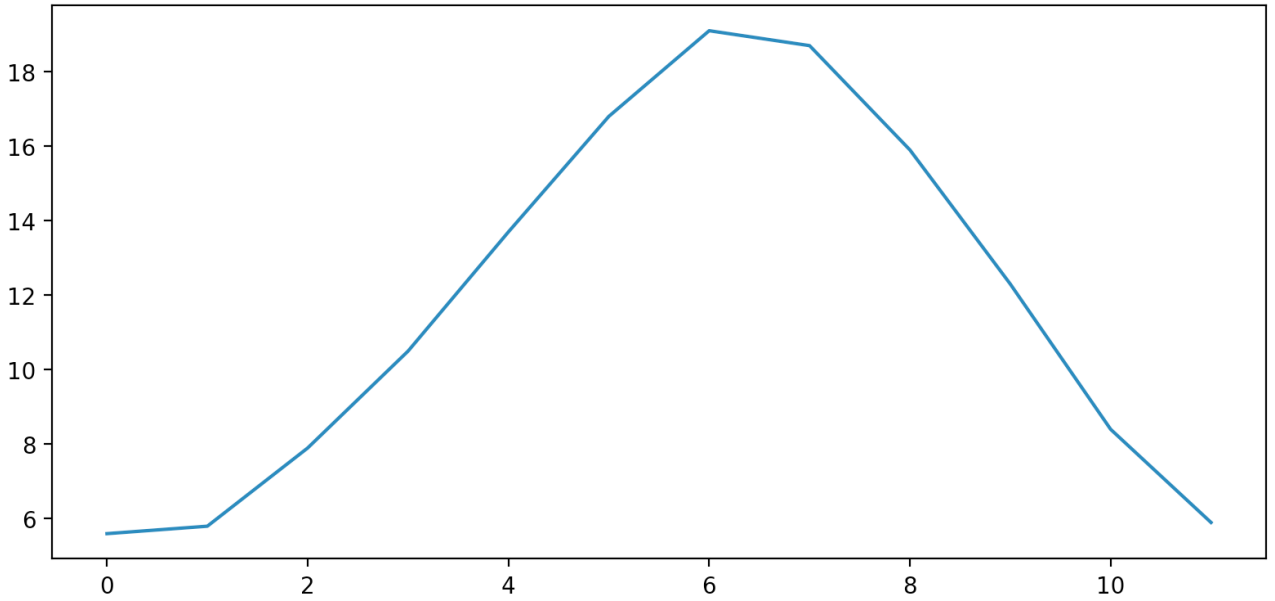
```
import numpy as np
import matplotlib.pyplot as plt
# Average monthly temperatures in London
T = np.array([5.6, 5.8, 7.9, 10.5, 13.7, 16.8, 19.1, 18.7, 15.9, 12.3, 8.4, 5.9], dtype=float)
# Plot the temperatures
plt.plot(T)
# Visualise the chart
plt.show()
```

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

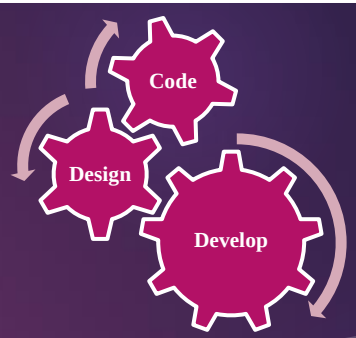


Line Chart *continued*

- **Possible improvements**

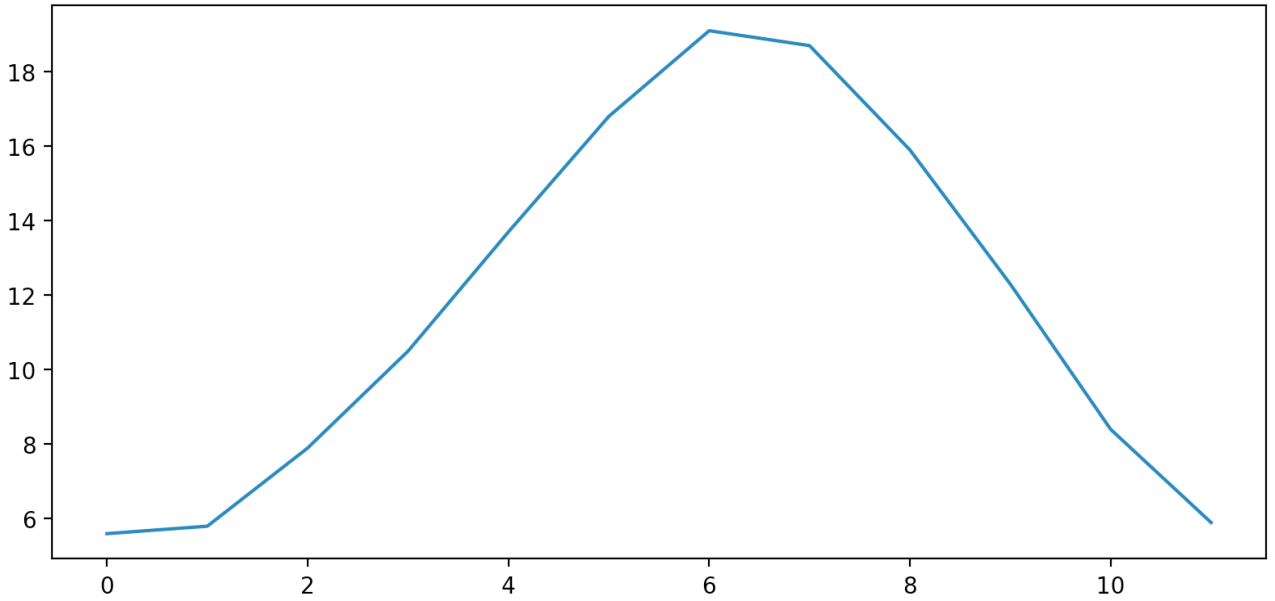


Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

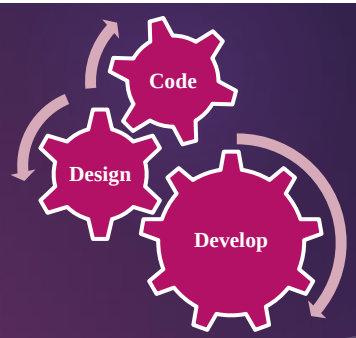


Line Chart *continued*

- **Possible improvements**
  - **Title of the diagram**
  - **Label X axis and Y axis**
  - **X axis indicates months**
  - **Graphics elements for data points**
  - **Display the values on the diagram**



Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

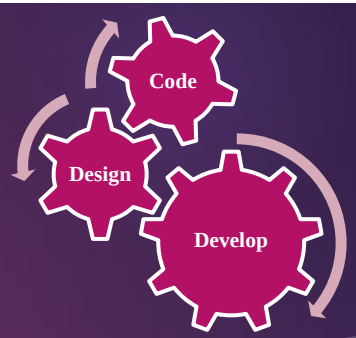


Line Chart *continued*

- **Possible Improvements: Source Code Part 1**

```
import numpy as np
import matplotlib.pyplot as plt
# Array containing the average monthly temperatures in London
T = np.array([5.6, 5.8, 7.9, 10.5, 13.7, 16.8, 19.1, 18.7, 15.9, 12.3, 8.4, 5.9], dtype=float)
# Diagram title
plt.title('Average Monthly Temperatures in London')
# Add months on the X axis
plt.xticks((0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'))
```

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

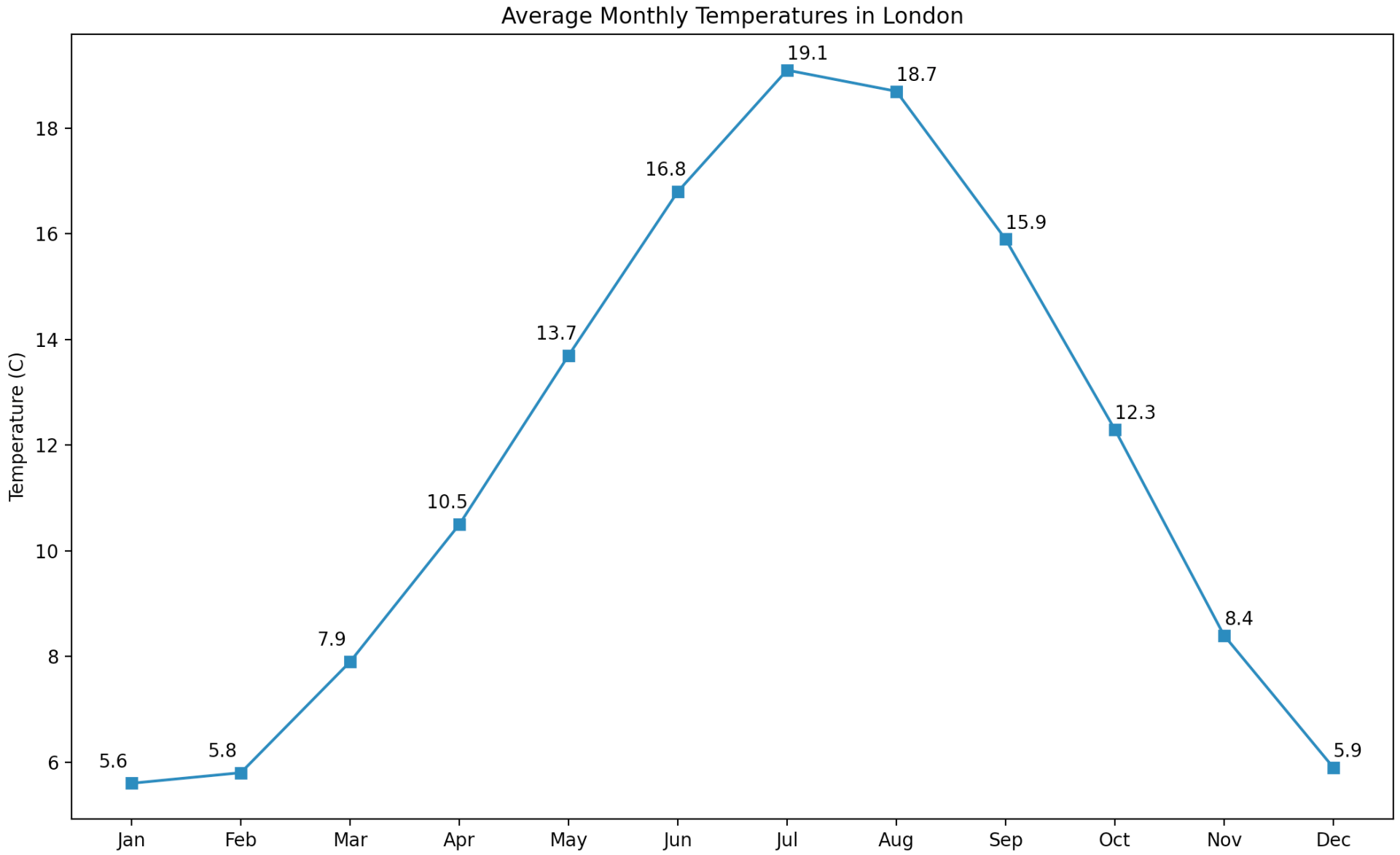


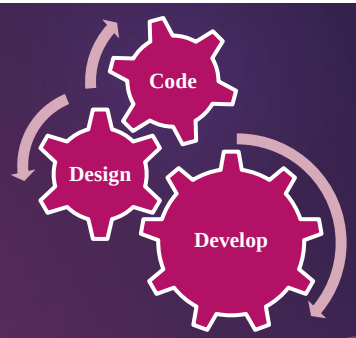
Line Chart *continued*

- **Possible Improvements: Source Code Part 2**

```
# Add label to Y axis
plt.ylabel('Temperature')
# Plot the temperatures
plt.plot(T, marker='s')
# Display values on the diagram
for i in range(0, 6):
    plt.text(i-0.3, T[i]+0.3, str(T[i]))
for i in range(6, 12):
    plt.text(i, T[i]+0.2, str(T[i]))
# Visualise the chart
plt.show()
```

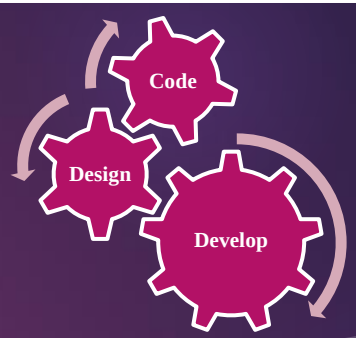
Chart generated by using the PyCharm IDE





Line Chart with Multiple Series

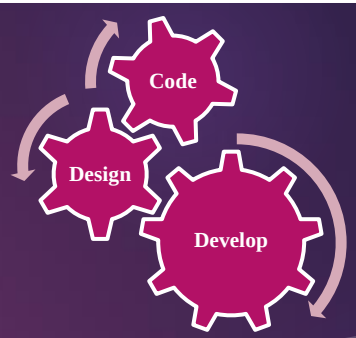
- In many cases in practice data is organised into multiple series
- Visualising multiple series within a single line chart allow to:
  - better illustrate an entire dataset
  - compare data points from different series
  - identify the presence, or the lack of it, of trends and patterns across all series



Line Chart with Multiple Series *continued*

- **Visualising multiple series within a single line chart**
  - **Source code part 1:**

```
import numpy as np
import matplotlib.pyplot as plt
# Array containing the average monthly temperatures in London
T1 = np.array([5.6, 5.8, 7.9, 10.5, 13.7, 16.8, 19.1, 18.7, 15.9, 12.3, 8.4, 5.9], dtype=float)
T2 = np.array([19.7, 20.6, 21.6, 23.7, 25.7, 27.4, 28.3, 28.4, 27.8, 26.1, 23.2, 21.3], dtype=float)
T3 = np.array([-8.4, -5.9, -3.4, 3.1, 8.9, 13.3, 15.3, 14.2, 9.6, 2.4, -4.7, -7.1], dtype=float)
# Diagram title
plt.title('Average Monthly Temperatures in London, Miami and Anchorage')
# Add months on the X axis
plt.xticks((0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11), ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'))
```

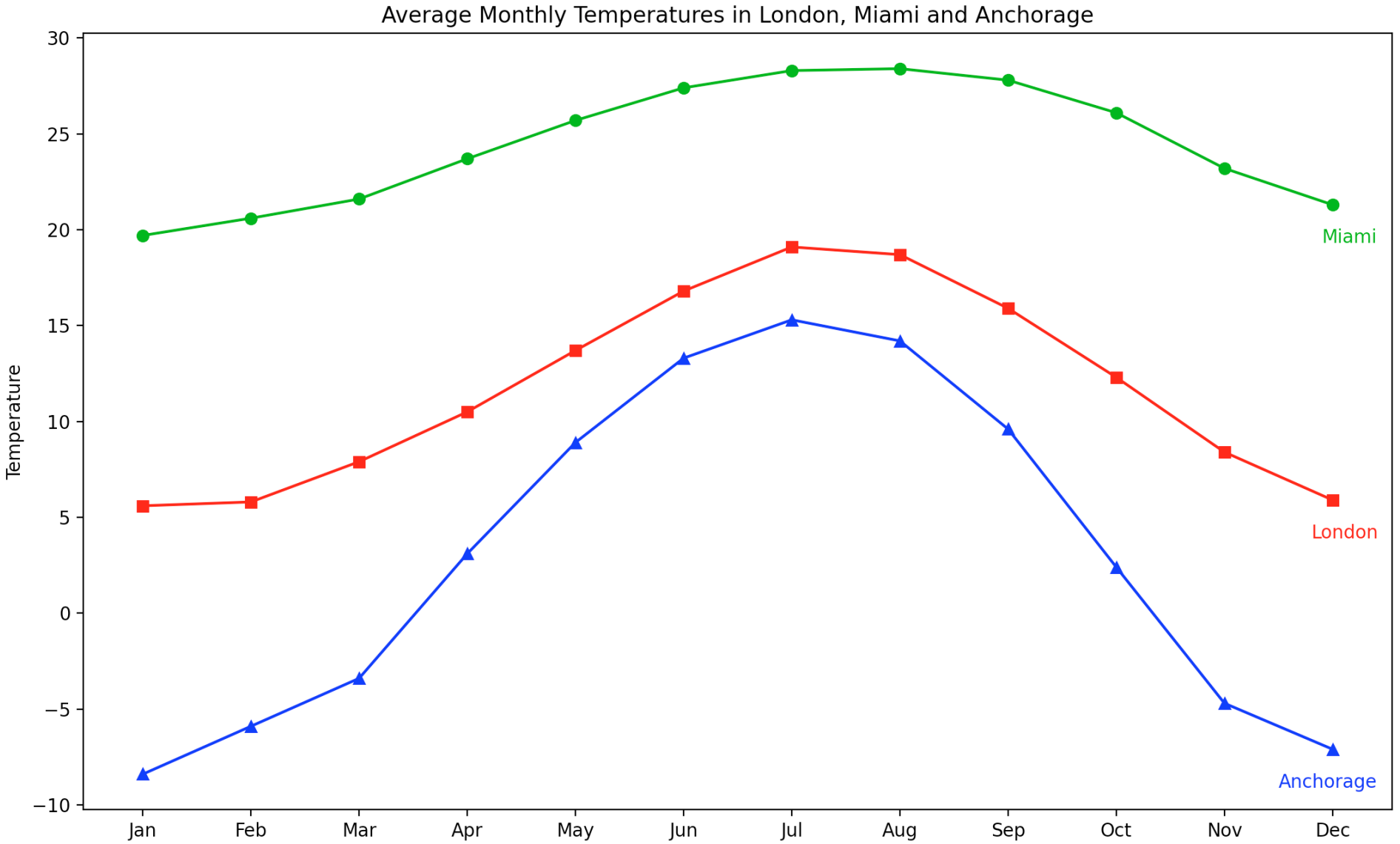


Line Chart with Multiple Series *continued*

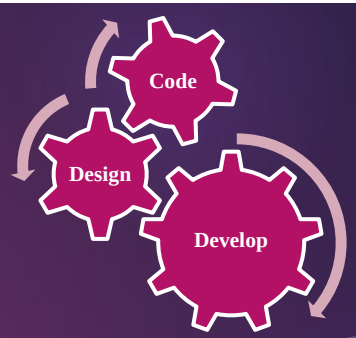
- **Visualising multiple series within a single line chart**
  - **Source code part 2:**

```
# Add label to Y axis
plt.ylabel('Temperature')
# Plot the temperatures
plt.plot(T1, marker='s', color="#FF0000")
plt.plot(T2, marker='o', color="#00AF00")
plt.plot(T3, marker='^', color="#0000FF")
# Print the names of the cities
plt.text(10.8, T1[11]-2, "London", color="#FF0000")
plt.text(10.9, T2[11]-2, "Miami", color="#00AF00")
plt.text(10.5, T3[11]-2, "Anchorage", color="#0000FF")
# Visualise the chart
plt.show()
```

Chart generated by using the PyCharm IDE



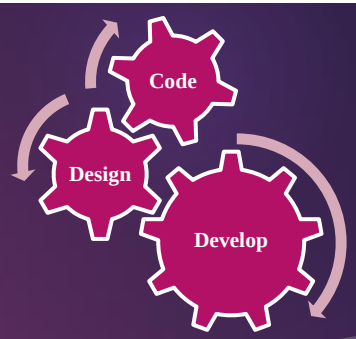
Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation



Bar Chart

- Represents numerical values associated with categorical features
- Numerical values are visualised with bars
- Hights, or lengths, of the bars are proportional to the numerical values
- Can represent both positive and negative values
- Not suitable for data with significant differences in magnitudes

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

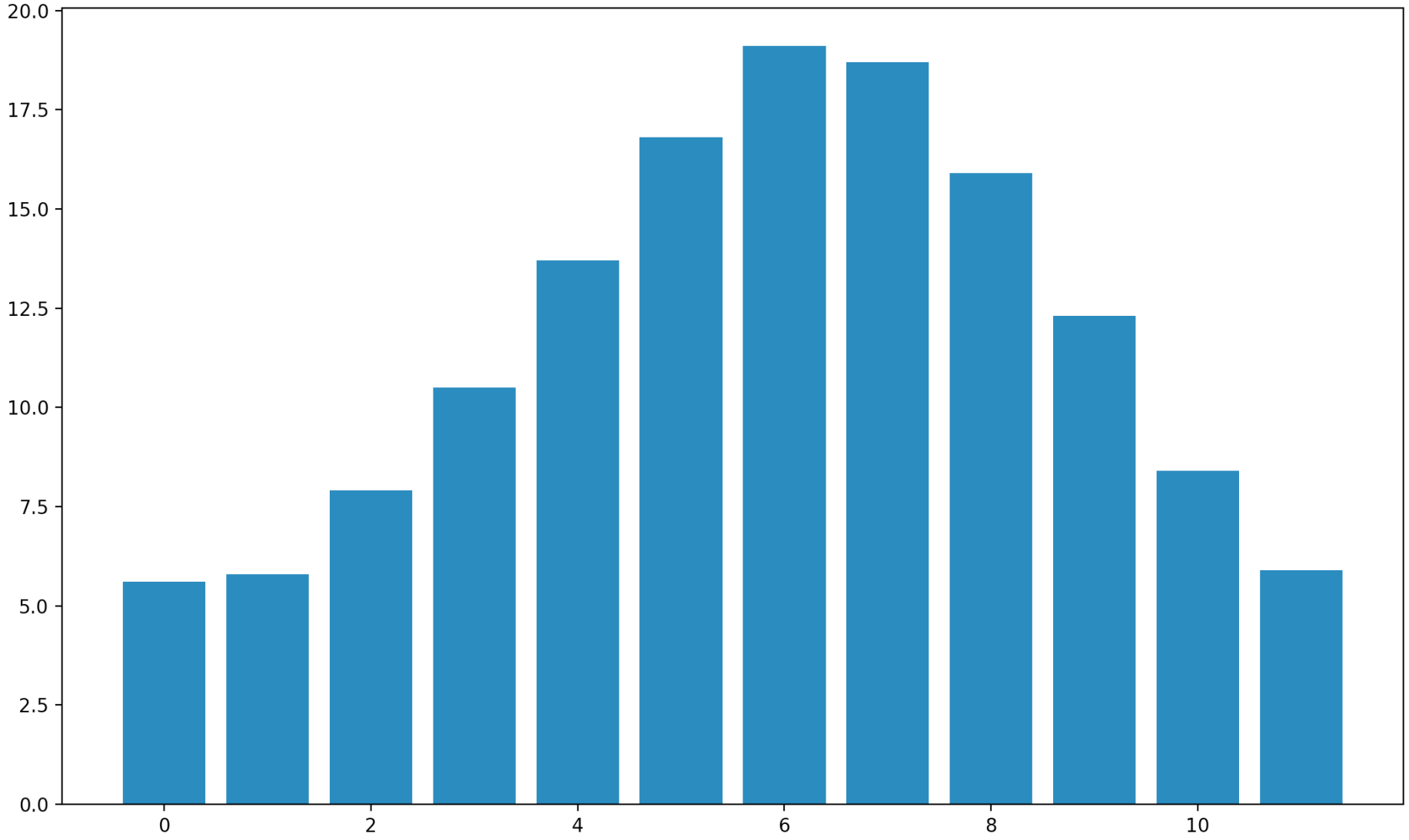


Bar Chart *continued*

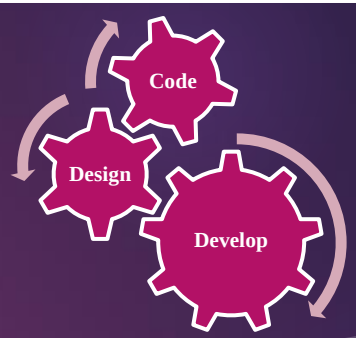
- **Example:**

```
import numpy as np
import matplotlib.pyplot as plt
# Array containing the average monthly temperatures in London
T = np.array([5.6, 5.8, 7.9, 10.5, 13.7, 16.8, 19.1, 18.7, 15.9, 12.3, 8.4, 5.9], dtype=float)
# Bar index
N = np.empty(shape=12, dtype=int)
for i in range(0, 12):
    N[i] = int(i)
# Plot a bar chart
plt.bar(N, T)
# Visualise the chart
plt.show()
```

Chart generated by using the PyCharm IDE



Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

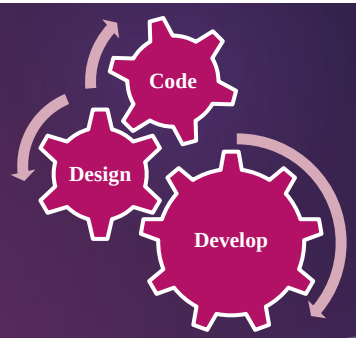


Bar Chart *continued*

- **Possible improvements: source code part 1**

```
import numpy as np
import matplotlib.pyplot as plt
# Array containing the average monthly temperatures in London
T = np.array([5.6, 5.8, 7.9, 10.5, 13.7, 16.8, 19.1, 18.7, 15.9, 12.3, 8.4, 5.9], dtype=float)
# Indices
N = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], dtype=int)
# Months
M = np.array(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'], dtype=str)
# Diagram title
plt.title('Average Monthly Temperatures in London')
```

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

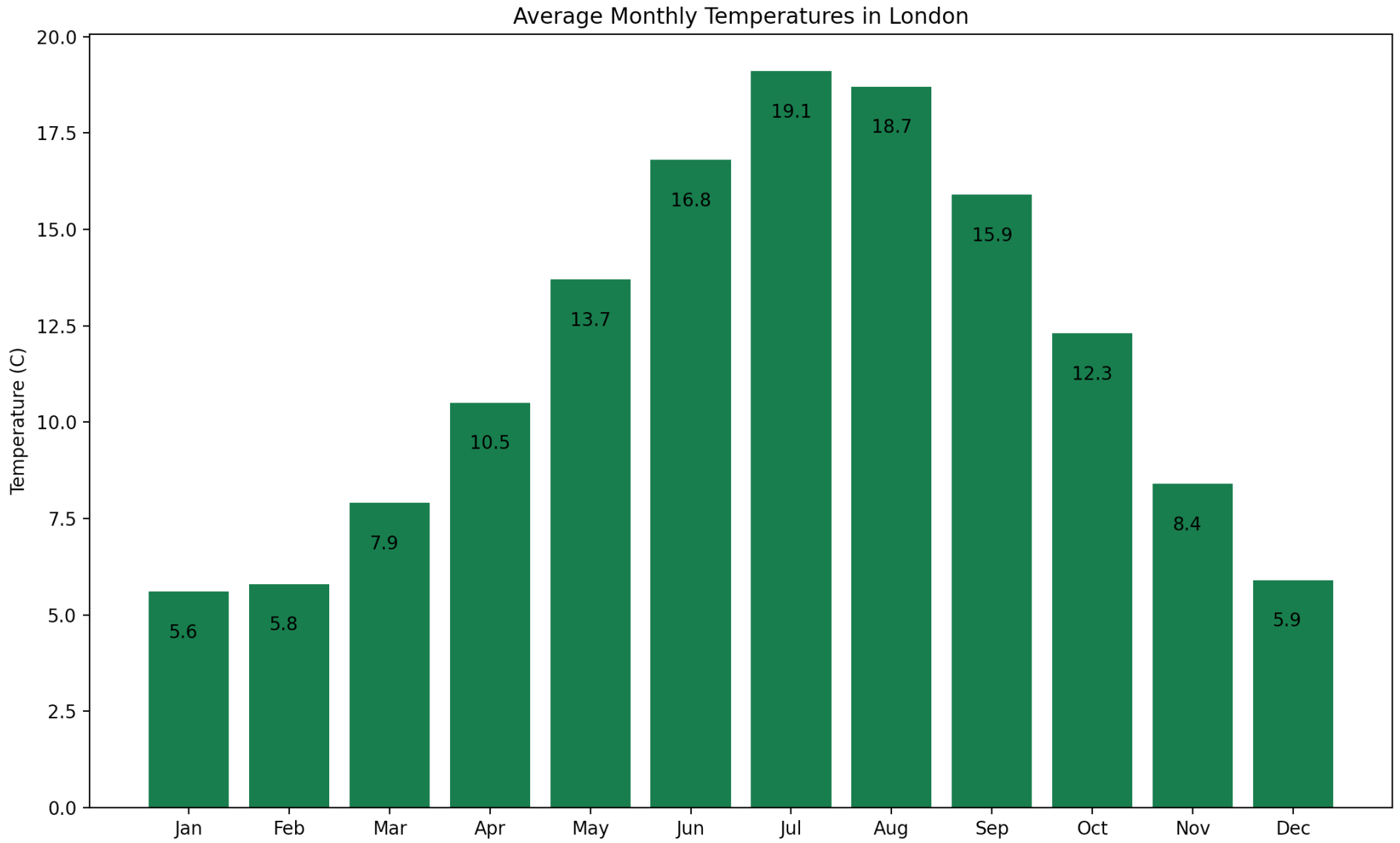


Bar Chart *continued*

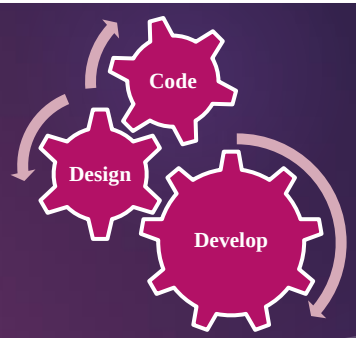
- **Possible improvements: source code part 2**

```
# Labels on X axis
plt.xticks(N, M)
# Label on Y axis
plt.ylabel('Temperature (C)')
# Plot a bar chart
plt.bar(N, T, width=0.8, color='#{:02x}{:02x}{:02x}'.format(20, 110, 60))
# Display data points on the chart
for i in range(0, 12):
    plt.text(i-0.2, T[i]-1.2, str(T[i]))
# Visualise the chart
plt.show()
```

Chart generated by using the PyCharm IDE

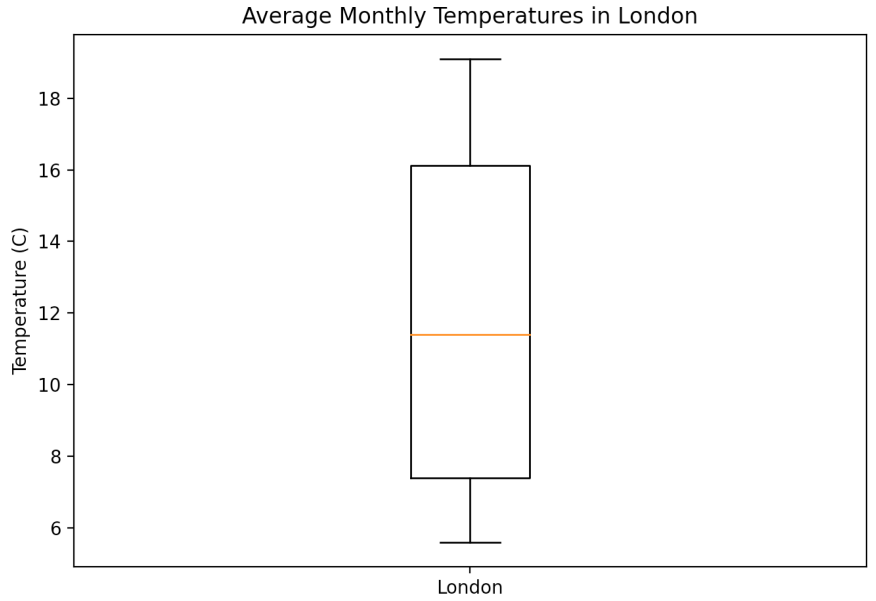


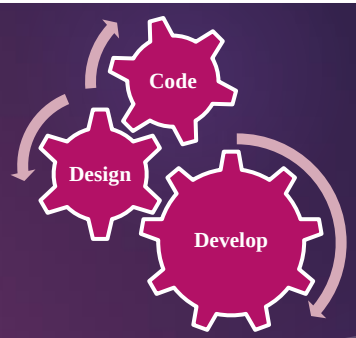
Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation



Box Plot Chart

- This type of chart provides the following statistical informatior
  - Minimum value
  - Lower quartile (25%)
  - Median value (50%)
  - Upper Quartile (75%)
  - Maximum value

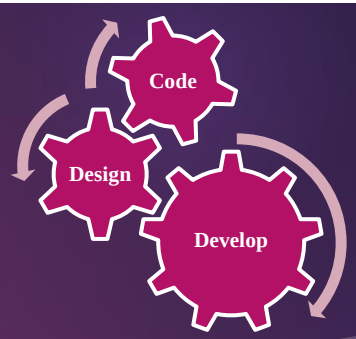




Box Plot Chart *continued*

- **Example: source code part 1**

```
import numpy as np
import matplotlib.pyplot as plt
# Array containing the average monthly temperatures in London
T = np.array([5.6, 5.8, 7.9, 10.5, 13.7, 16.8, 19.1, 18.7, 15.9, 12.3, 8.4, 5.9], dtype=float)
# Diagram title
plt.title('Average Monthly Temperatures in London')
# Add label to Y axis
plt.ylabel('Temperature (C)')
```

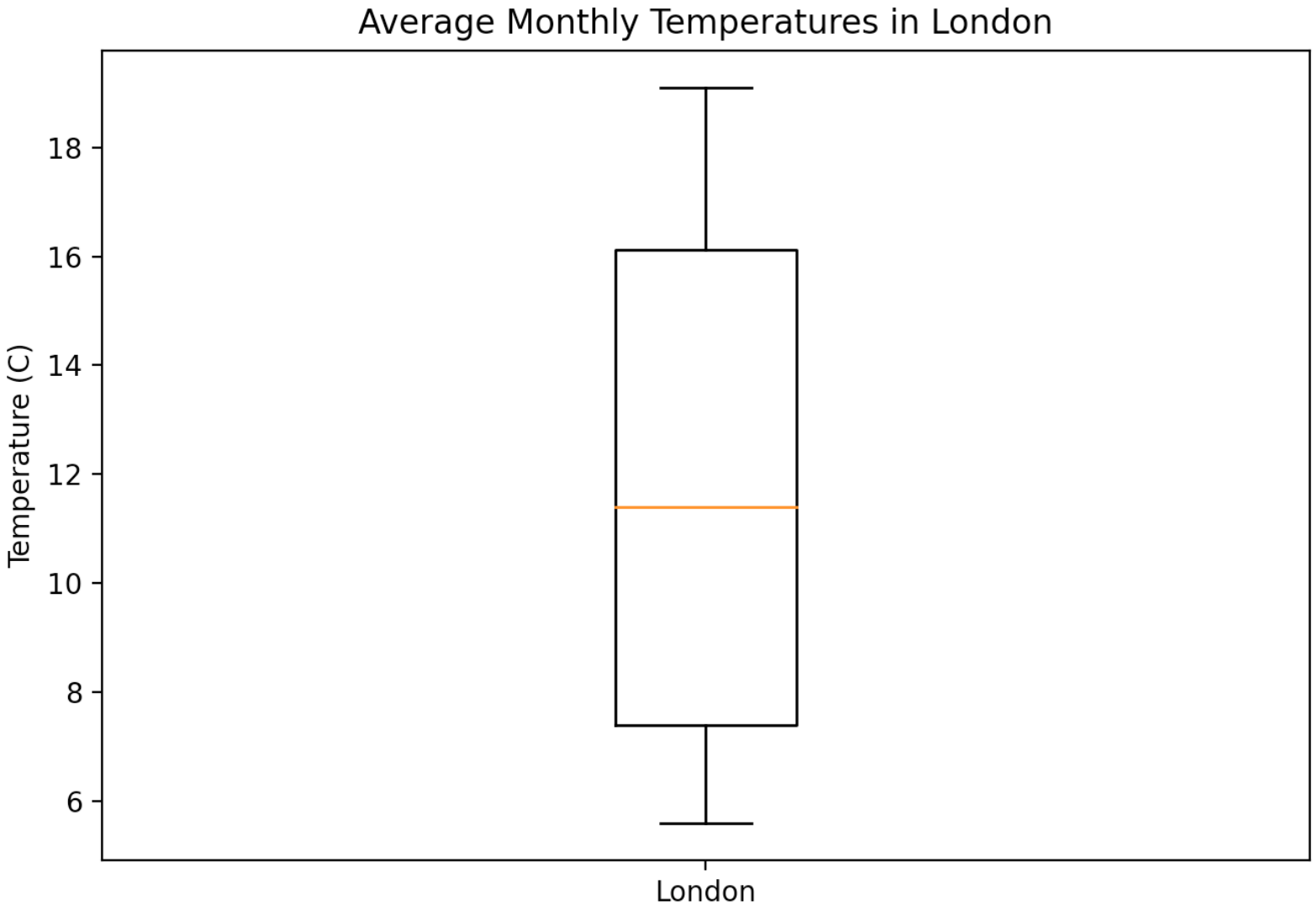


Box Plot Chart *continued*

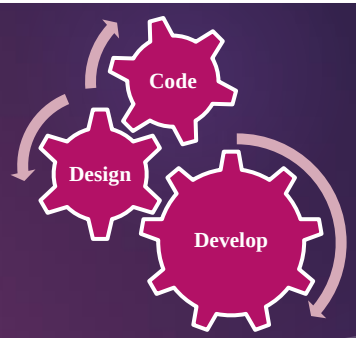
- **Example: source code part 2**

```
# Plot the temperatures
plt.boxplot(T)
# Add the name of the city on X axis
plt.xticks([1], ['London'])
# Visualise the chart
plt.show()
```

Chart generated by using the PyCharm IDE



Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

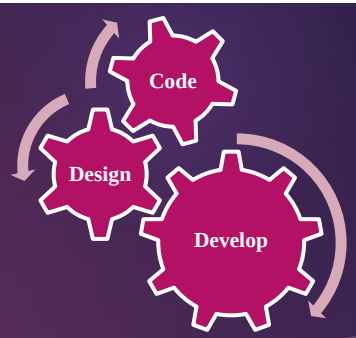


Box Plot Chart *continued*

- **Visualising three cities: source code part 1**

```
import numpy as np
import matplotlib.pyplot as plt
# Arrays containing the average monthly temperatures in London, Miami and Anchorage
T1 = np.array([5.6, 5.8, 7.9, 10.5, 13.7, 16.8, 19.1, 18.7, 15.9, 12.3, 8.4, 5.9], dtype=float)
T2 = np.array([19.7, 20.6, 21.6, 23.7, 25.7, 27.4, 28.3, 28.4, 27.8, 26.1, 23.2, 21.3], dtype=float)
T3 = np.array([-8.4, -5.9, -3.4, 3.1, 8.9, 13.3, 15.3, 14.2, 9.6, 2.4, -4.7, -7.1], dtype=float)
# Initialise a matrix containing the temperatures
T4 = np.array([T1, T2, T3], dtype=float)
# Transpose the matrix
T = np.transpose(T4)
```

Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation

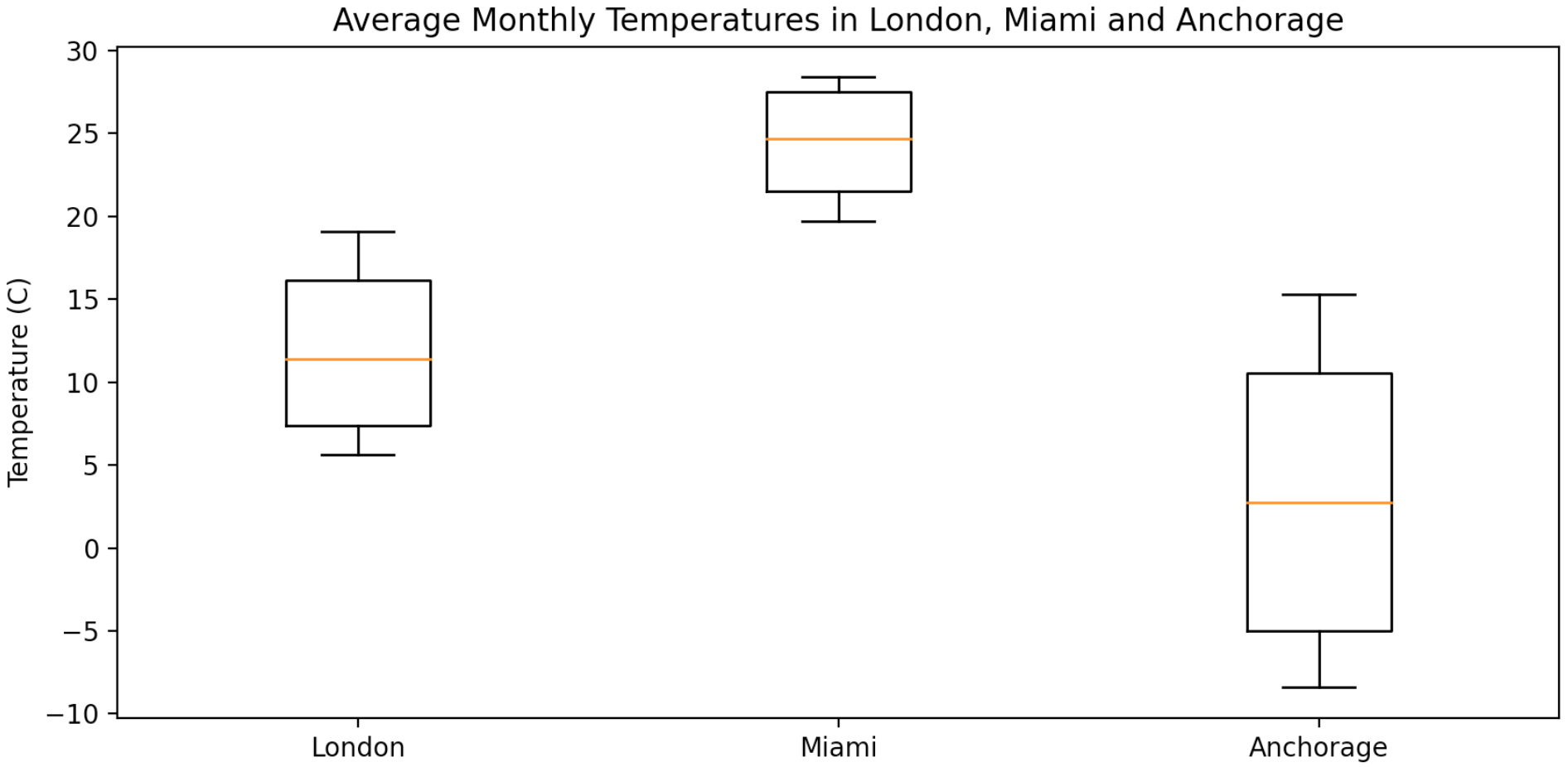


Box Plot Chart *continued*

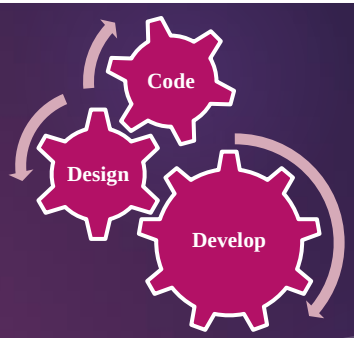
- **Visualising three cities: source code part 2**

```
# Chart title
plt.title('Average Monthly Temperatures in London, Miami and Anchorage')
# Add label on Y axis
plt.ylabel('Temperature (C)')
# Plot the box plot charts
plt.boxplot(T)
# Add the names of the cities on X axis
plt.xticks([1, 2, 3], ['London', 'Miami', 'Anchorage'])
# Visualise the chart
plt.show()
```

Chart generated by using the PyCharm IDE



Information Analysis and Visualisation
Lecture 3: Design Principles in Data Visualisation



Conclusions

- **This lecture discussed the main principles of Data Visualisation**
- **How these principles relate with the quality of the charts was demonstrated**
- **Examples introducing main types of charts were demonstrated and discussed**