Topic 11

Data Visualisation Using Matplotlib

Murdoch
UNIVERSITY

Last Topic

- Numy package
- Ndarray
- Create ndarrays
- Numpy functions for creating ndarray: zeros, ones, empty, arange, linspace
- Ndarray type
- Ndarray indexing
- Copy, view and concatenate
- Element-wise operations
- Matrix multiplication
- Universal functions

This Topic

Murdoch
UNIVERSITY

- Matplotlib package
- Figure and axes
- Pyplot interface
- Plot function
- Histogram
- Bar Charts
- Pie Charts
- Object-oriented plotting interface

Matplotlib

- Matplotlib is a very popular Python library for graphing and data visualisation
- The original purpose of Matplotlib was to recreate MatLab-like plotting facilities for Python.
- It is often used together with Numpy for data visualisation.
- To use this package, you need to install it on your computer:

```
pip install matplotlib
```

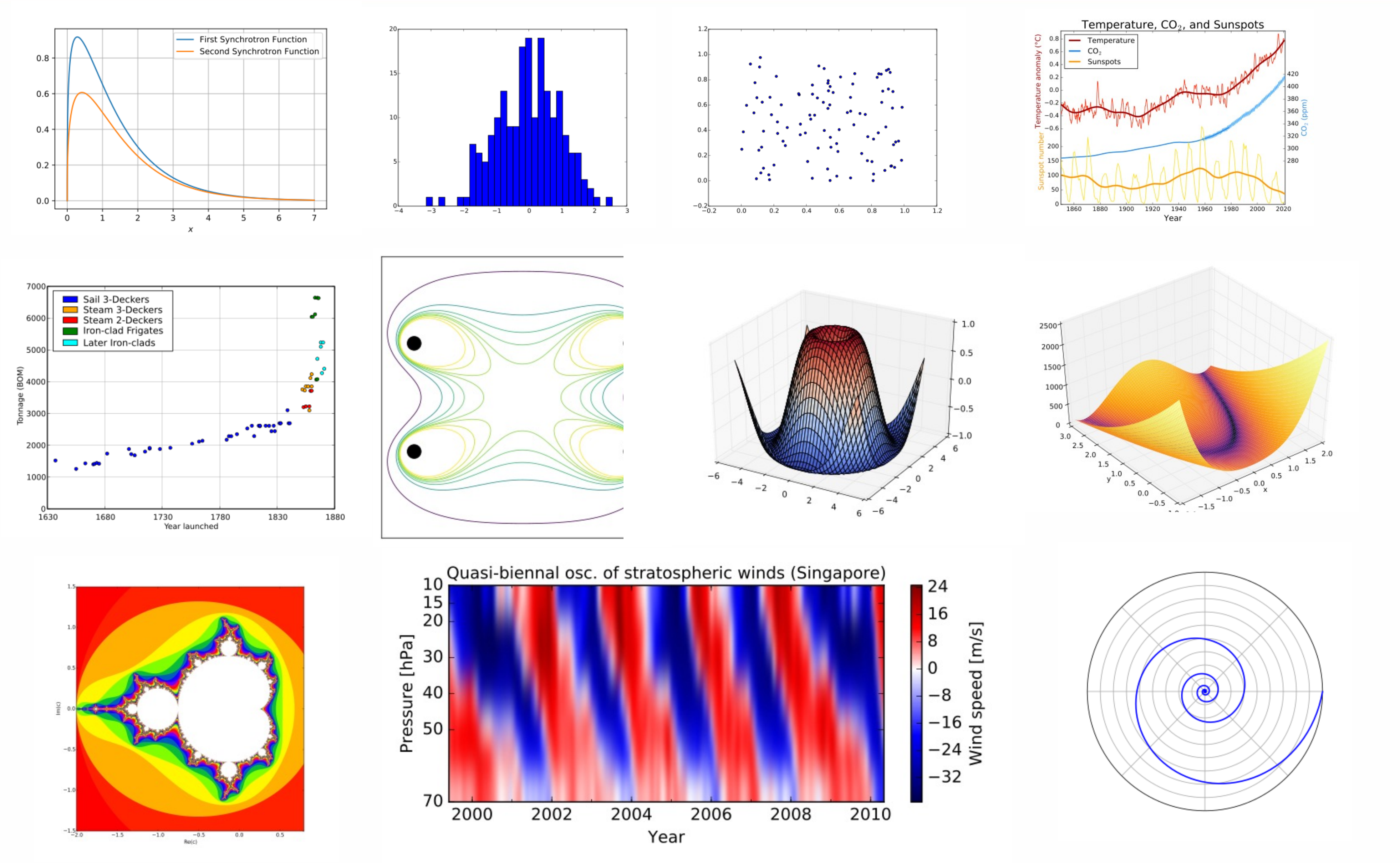
Pyplot

- One of the most frequently used module in Matplotlib package is `pyplot`.
- This module is often imported into our program as `plt`.

```
import matplotlib.pyplot as plt
```

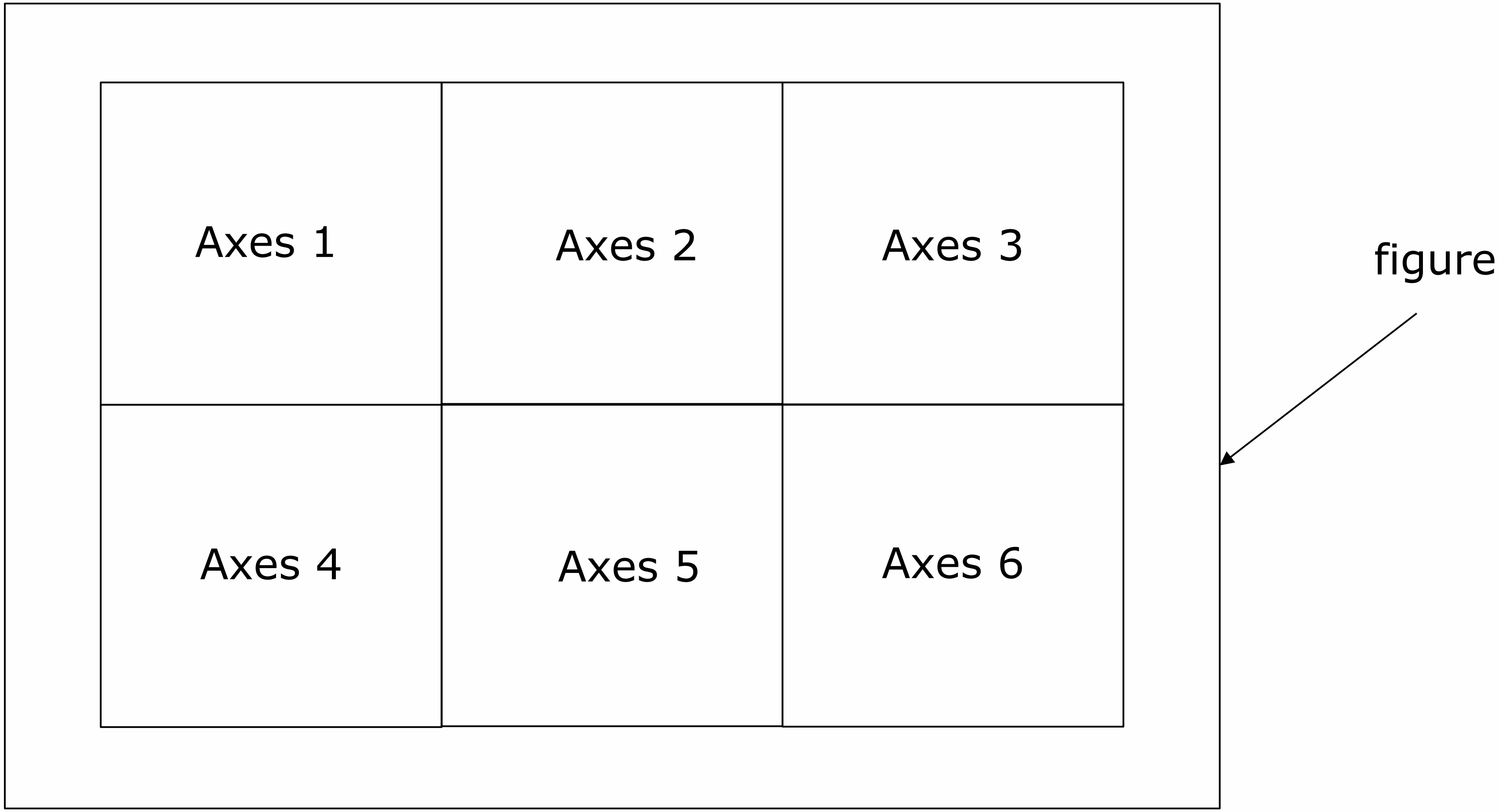
- `Pyplot` provides a MatLab-like interface for plotting graphs and charts, eg,
  - Line plot
  - Scatter plot
  - Histogram
  - 3D plot
  - Time series plot
  - and many many more

Pyplot



https://en.wikipedia.org/wiki/Matplotlib

Figure and Axes



Axes 1

Axes 2

Axes 3

Axes 4

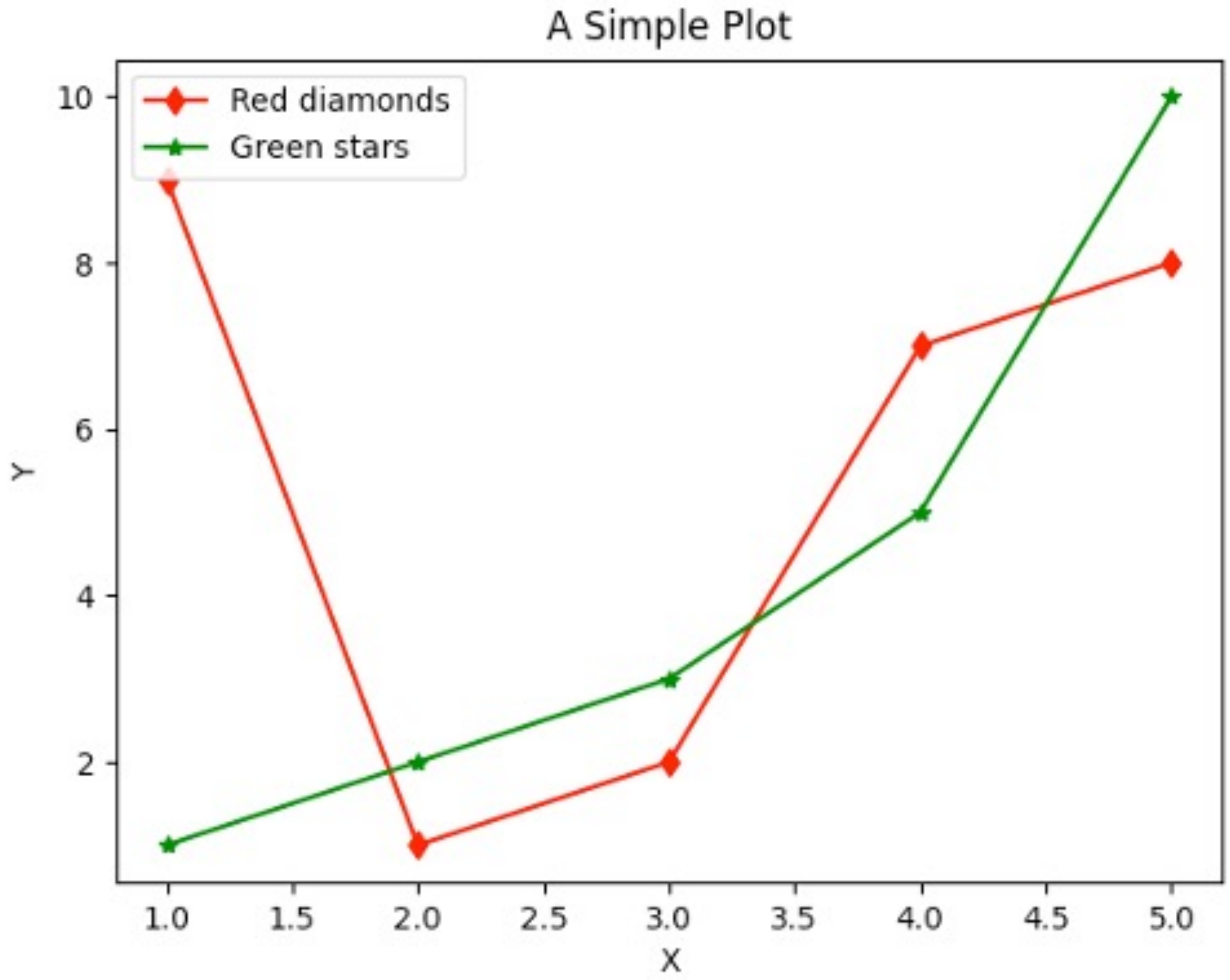
Axes 5

Axes 6

figure

Example: a figure with 2x3 axes

Axes



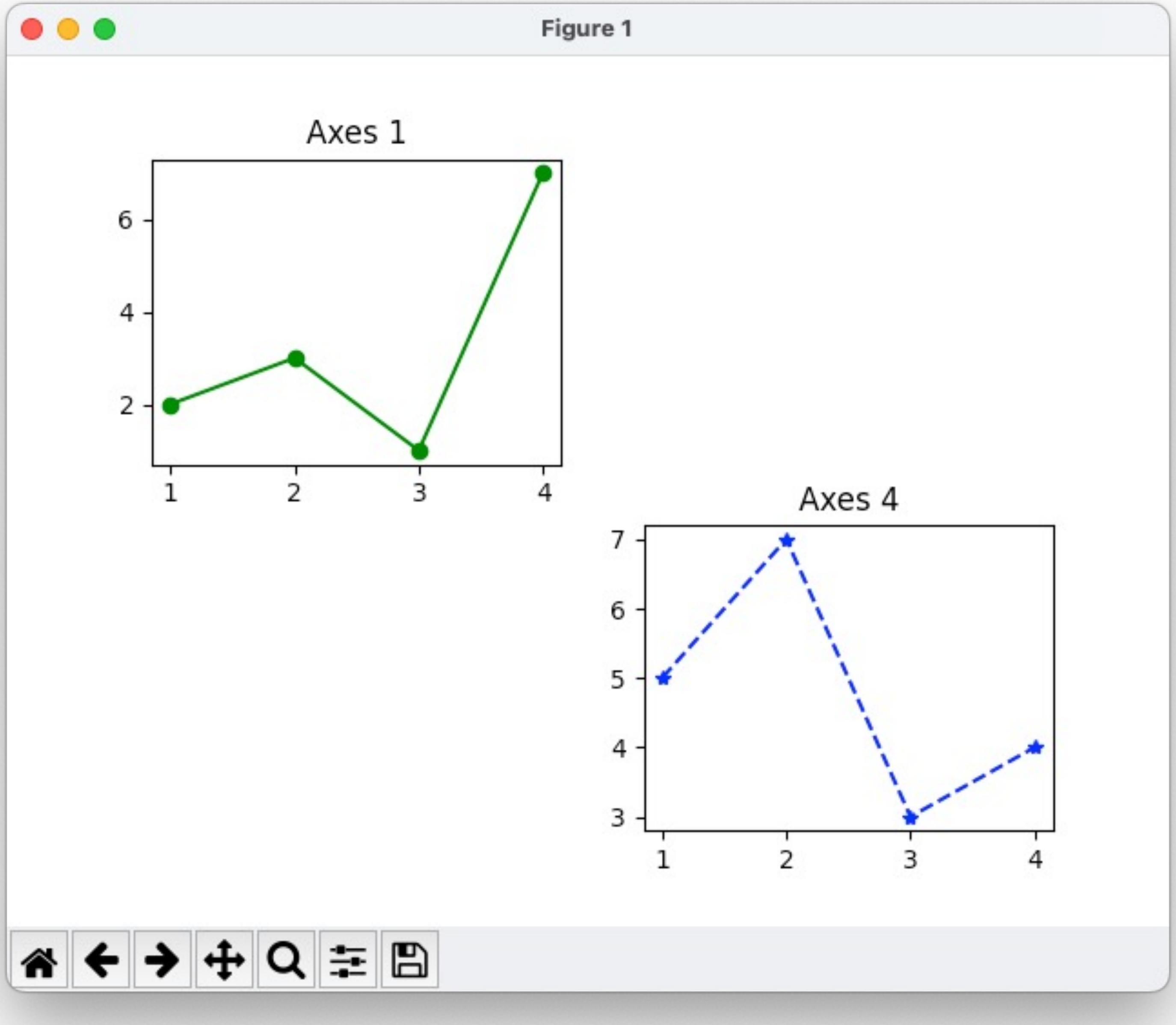
Example: an axes with title, x-axis, y-axis and legend

Plotting with Pyplot

- `Pyplot` mimics the MatLab interface. It is *stateful* – at any time, it remembers the *current axes* and a plotting operation is always for the current axes.

```
import matplotlib.pyplot as plt

plt.subplot(2,2,1) # 2x2 axes, select axes 1
X = [1,2,3,4]; Y = [2,3,1,7]  # X,Y define 4 points
plt.plot(X, Y, 'go-')  # green dots, solid line
plt.title("Axes 1") # write title

plt.subplot(2,2,4) # 2x2 axes, select axes 4
X = [1,2,3,4]; Y = [5,7,3,4]  # X,Y define 4 points
plt.plot(X, Y, 'b*--')  # blue stars, dashed line
plt.title("Axes 4")  # write title

plt.show() # display the figure.
```

Plotting with Pyplot



Plotting with Pyplot

- `plt.subplot(`*`nrows, ncols, index`*`)`
  - Divide the figure into *nrows* x *ncols* axes and select the current axes whose index is *index*
- `plt.plot(`*`X, Y, format`*`)`
  - Plot the points defined by (X,Y) using format
    - 'go-': 'g': green color, 'o': round dots, '-': solid line
    - 'b*--': 'b': blue color, '*': stars, '--': dashed lines
  - For further infomation about the `plot` function and the format, see: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

Plotting with Pyplot

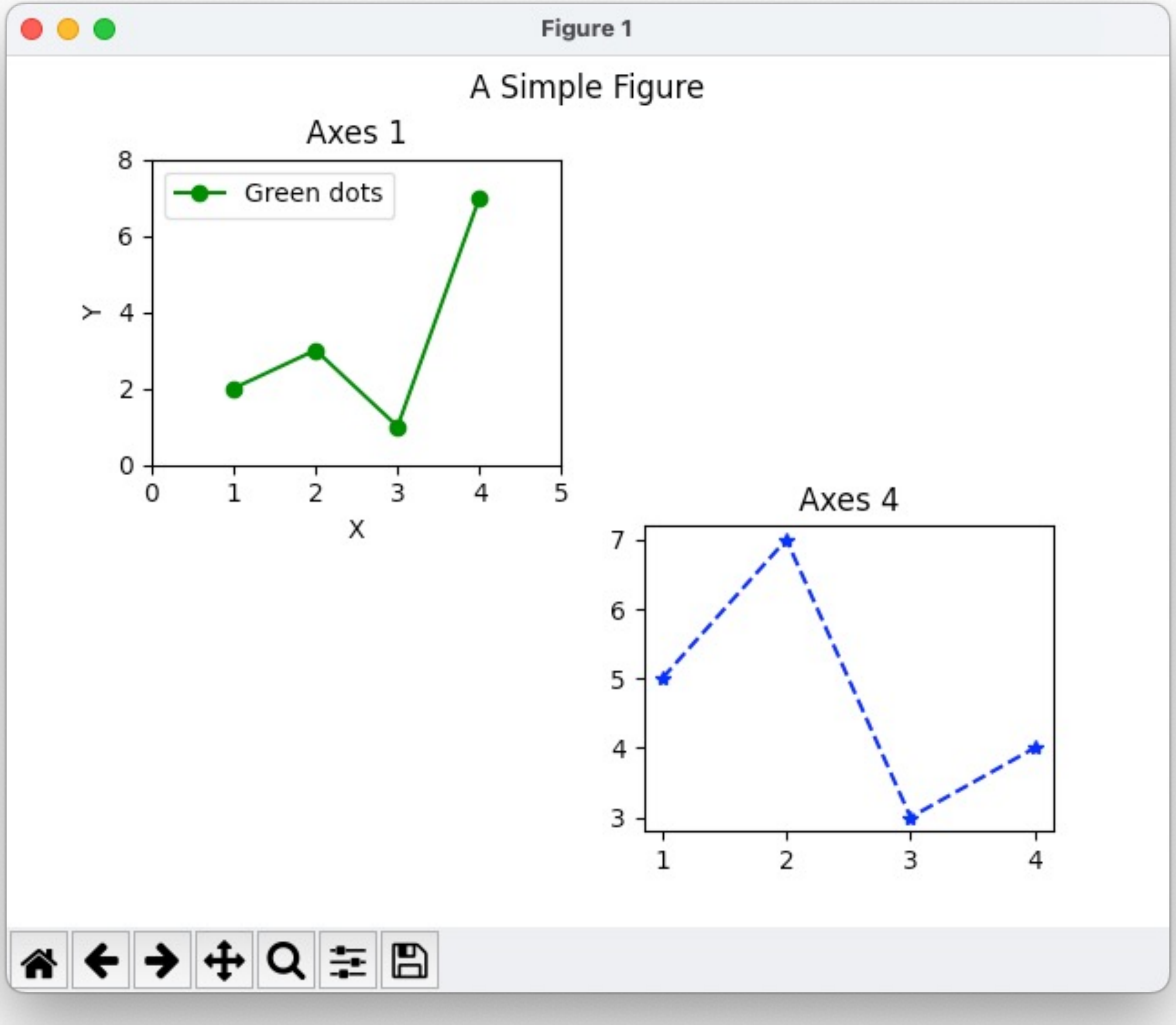
- `plt.title(`*`string`*`)`
  - Plot the title on the current axes
- `plt.show()`
  - Display the figure
  - Note: this function would not return until the window displaying the figure is closed.

Axis' Label, Limit and Legend

Murdoch
UNIVERSITY

```
import matplotlib.pyplot as plt

plt.subplot(2,2,1)
X = [1,2,3,4]; Y = [2,3,1,7]
plt.plot(X, Y, 'go-', label='Green dots' )
plt.title("Axes 1")
plt.xlabel('X')
plt.ylabel('Y')
plt.xlim(0,5)
plt.ylim(0,8)
plt.legend(loc='best')

plt.subplot(2,2,4)
X = [1,2,3,4]; Y = [5,7,3,4]
plt.plot(X, Y, 'b*--', label='Blue stars')
plt.title("Axes 4")

plt.suptitle("A Simple Figure")
plt.show()
```

Axis' Label and Limit and Legend



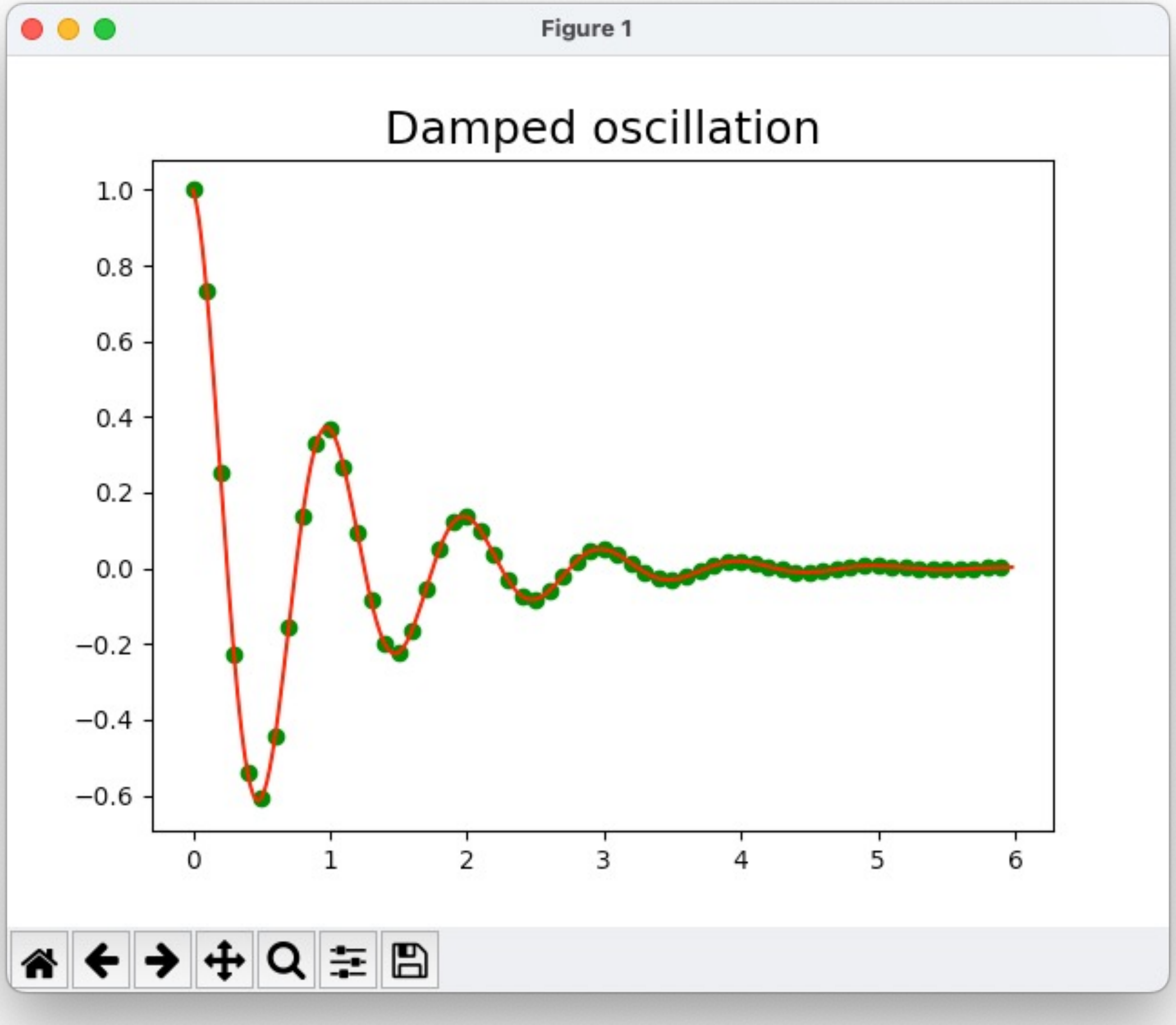
Axis' Label and Limit and Legend

- `plt.xlabel(`*`string`*`), plt.ylabel(`*`string`*`)`
  - The label for the X-axis and Y-axis
- `plt.xlim(`*`start, end`*`), plt.ylim(`*`start, end`*`)`
  - Set X-axis' range and Y-axis' range
- `plt.plot(`*`X, Y, format`*`, label=`*`label`*`)`
  - *`Label`*: the label used for legend
- `plt.legend(loc=`*`location`*`)`
  - *`Location`*: 'upper left', 'upper right', 'lower left', 'lower right', 'upper center', 'lower center', 'center left', 'center right', 'center', 'best'

Visualise the Data in Ndarrays

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

def f(t):
   s = np.cos(2*np.pi*t)
   e = np.exp(-t)
   return s * e

t1 = np.arange(0.0, 6.0, 0.1)
t2 = np.arange(0.0, 6.0, 0.02)

plt.plot(t1, f(t1), 'go')
plt.plot(t2, f(t2), 'r-')
plt.title('Damped oscillation', fontsize=18)

plt.show()
```

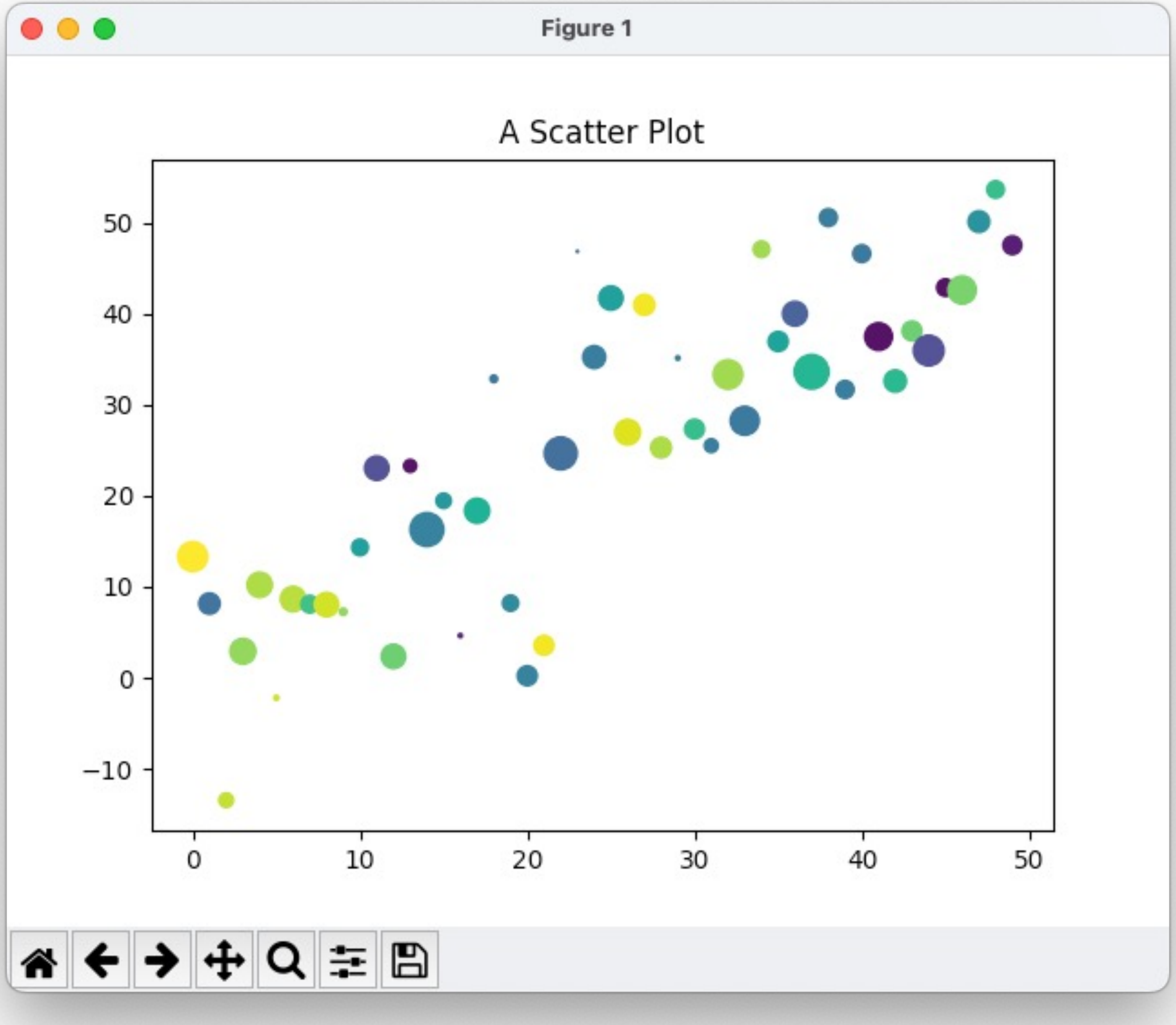
Visualise the Data in Ndarrays



Scatter Plot

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

np.random.seed(10)
data = { 'a': np.arange(50),
         'b': np.arange(50) + 10 * np.random.randn(50),
         'c': np.random.randint(0, 50, 50),
         'd': np.abs(np.random.randn(50)) * 100
       }
plt.scatter('a', 'b', c='c', s='d', data=data)
plt.title('A Scatter Plot')
plt.show()
```

Scatter Plot



Scatter Plot

- `np.random.seed(10)`
  - Seed the random number generator, different seed gives you a different random number sequence
- `np.arange(50)`
  - An ndarray with a sequence from 0 to 49
- `np.random.randn(50)`
  - An ndarray with a sequence of 50 random numbers following the normal distribution, with mean 0 and standard deviation 1
- `np.random.randint(0, 50, 50)`
  - An ndarray with a sequence of 50 random numbers between 0 and 49

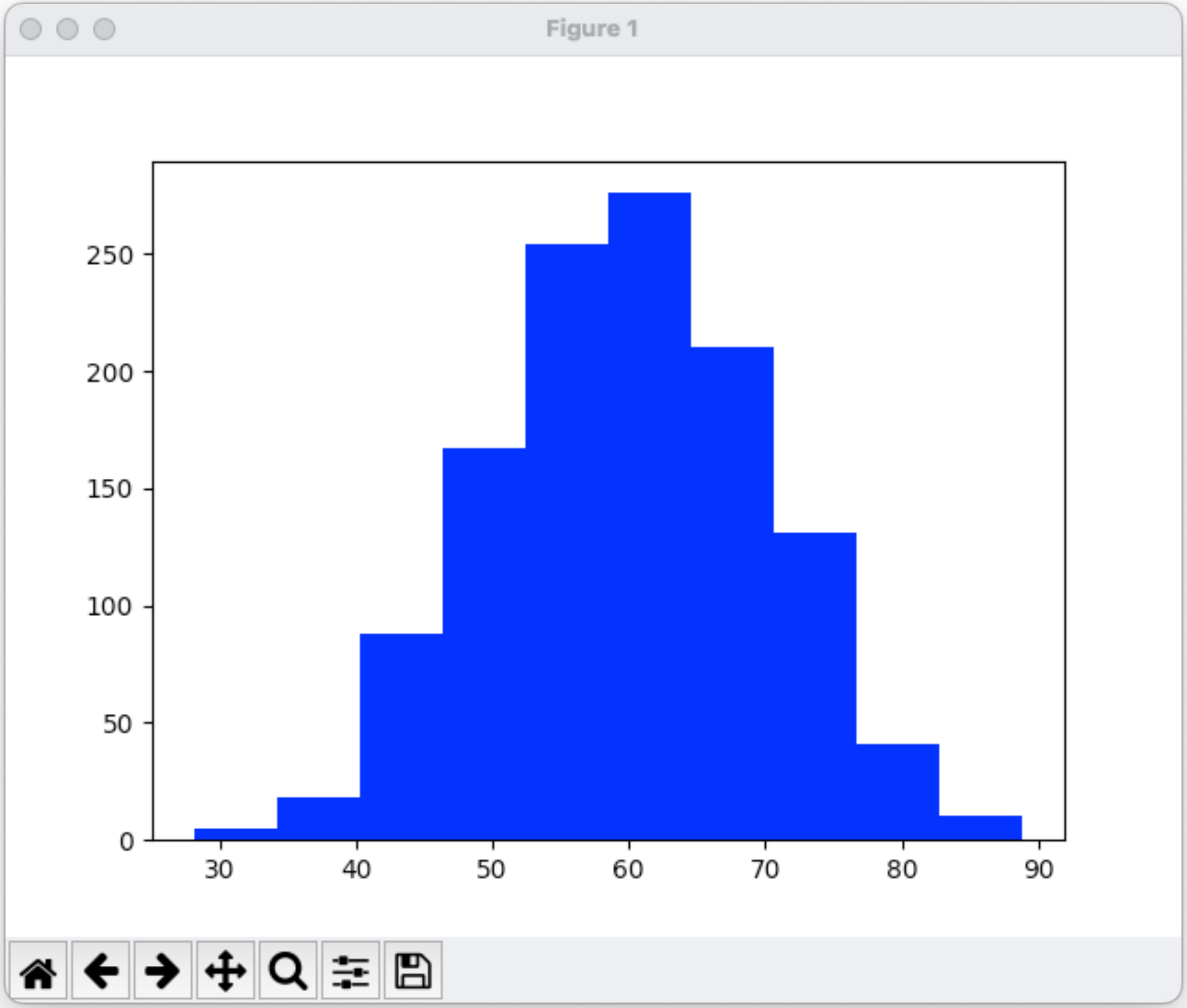
Scatter Plot

- `plt.scatter('a', 'b', c='c', s='d', data=data)`
  - `data['a']`: data for X axis
  - `data['b']`: data for Y axis
  - `data['c']`: data representing the colour of each point
  - `data['d']`: data representing the size of each point

Histogram

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

np.random.seed(20)
marks  = np.random.normal(60,10,1200)
plt.hist(marks, 10, facecolor='blue', alpha=0.5)
plt.show()
```

Histogram



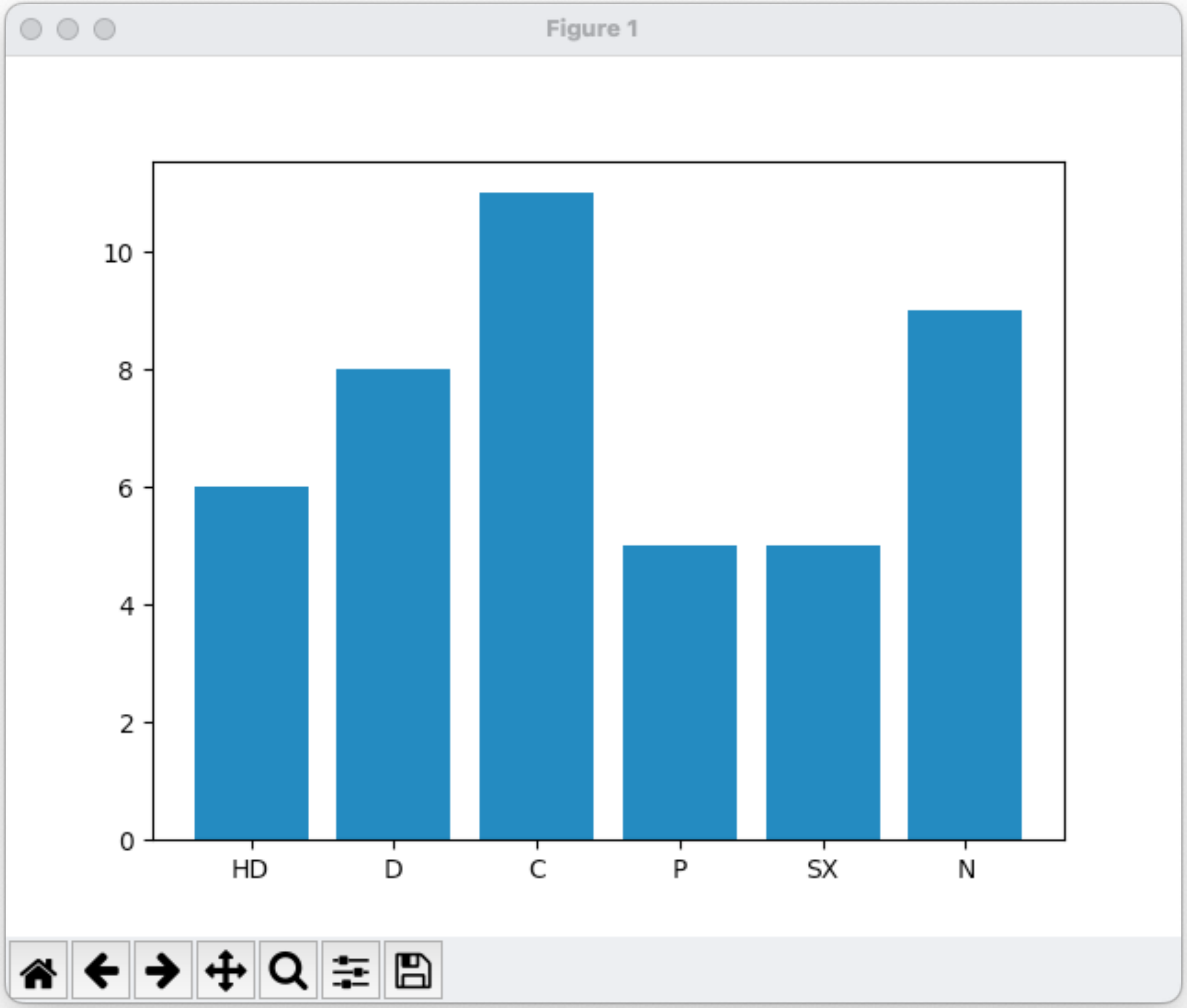
Histogram

- `marks = np.random.normal(60,10,1200)`
  - An ndarray containing 1200 numbers following normal distribution centred on 60 with standard deviation 10.
- `plt.hist(marks, 10, facecolor='blue')`
  - Generate the histogram of marks with 10 bins. The 10 bins are coloured in blue.

Bar Chart

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

X = np.array([ 'HD', 'D', 'C', 'P', 'SX', 'N' ])
Y = np.array([ 6, 8, 11, 5, 5, 9 ])
plt.bar(X,Y)

plt.show()
```

Bar Chart



Pie Chart

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

y = np.array([ 3, 3, 4, 20, 30, 40 ])
assessment = ['Lab3', 'Lab7', 'Lab10', 'Assignment1', 'Assignment2',
'Exam']
plt.pie(y, labels=assessment)
plt.title('ICT365 Assessments')

plt.show()
```

Pie Chart

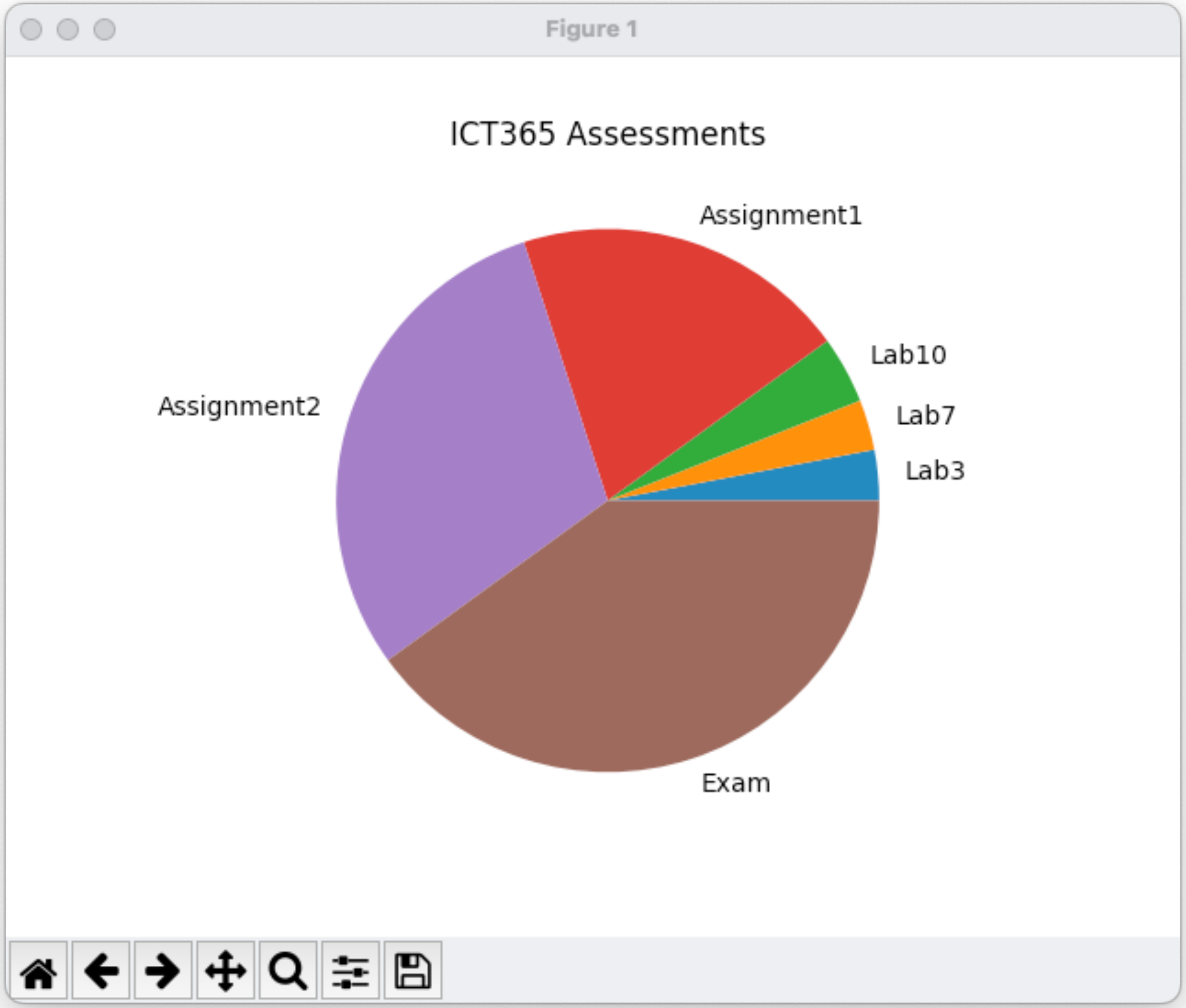


Figure 1
ICT365 Assessments
Assignment1
Lab10
Lab7
Lab3
Assignment2
Exam

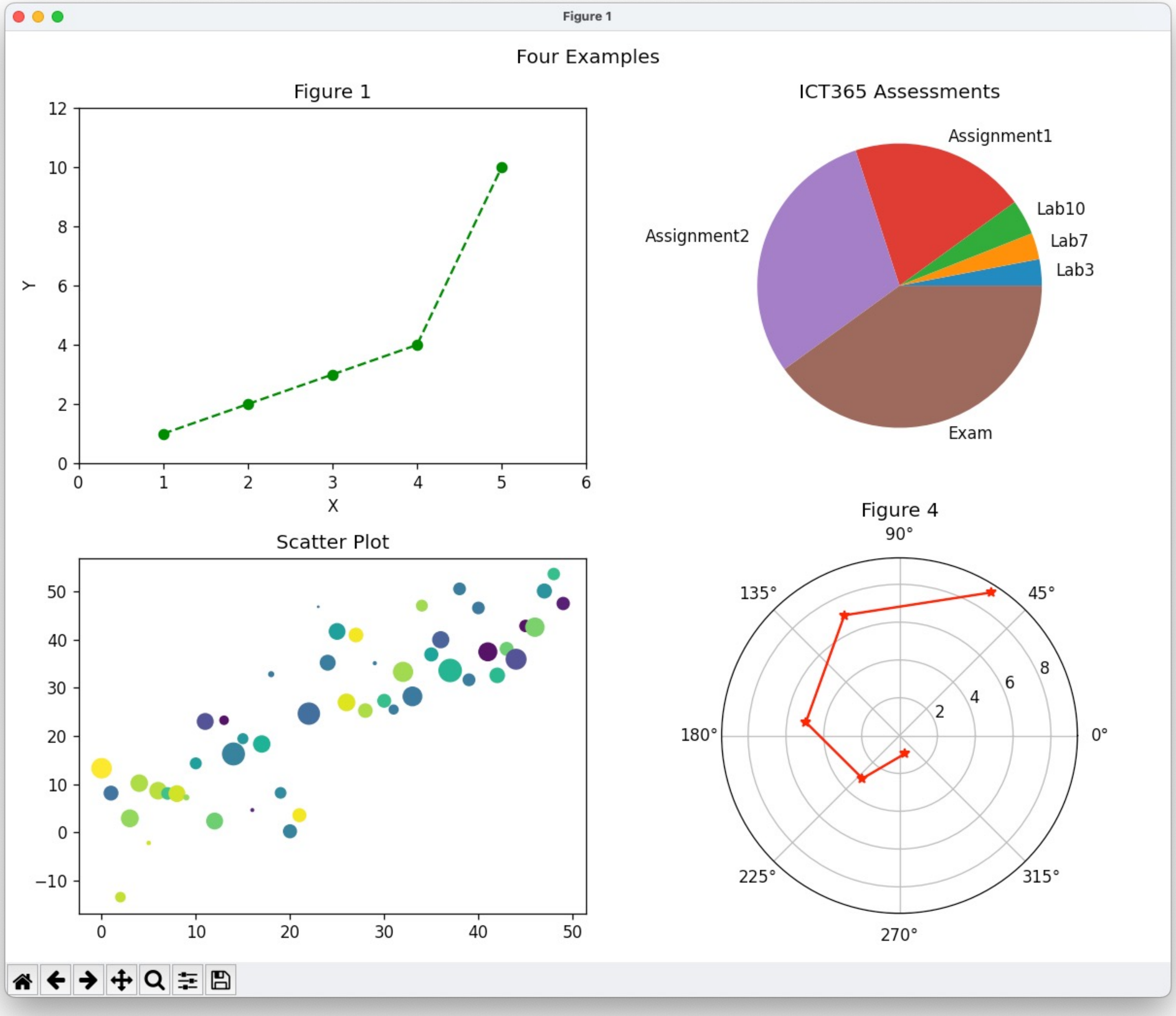
Object Oriented Interface

- Matplotlib provided the MatLab-like interface, via `matplotlib.pyplot` module, which is stateful – it always remembers which axes is the current axes and the plot operations are always targeted at the current axes.
- Matplotlib also provided an object-oriented interface centred on the class `Axes`.
- The class `matplotlib.axes.Axes` provided nearly the same set of plotting operations as `pyplot`, with slightly different function/method names.

Object-Oriented Interface

matplotlib.pylot	matplotlib.axes.Axes
.plot()	.plot()
.scatter()	.scatter()
.bar()	.bar()
.hist()	.hist()
.pie()	.pie()
.step()	.step()
.contour()	.contour()
.boxplot()	.boxplot()
.quiver()	.quiver()
.steampot()	.steampot()
.grid()	.grid()
.title()	.set_title()
.legend()	.legend()
.xlabel()	.set_xlabel()
.ylabel()	.set_ylabel()
.xlim()	.set_xlim()
.ylim()	.set_ylim()

Object Oriented Interface



Object-Oriented Interface

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

fig, axes = plt.subplots(2,2, figsize=(10,8), dpi=120)

# Subplot 1
ax1 = axes[0,0]
ax1.plot([1,2,3,4,5], [1,2,3,4,10], 'go--')
ax1.set_title("Figure 1")
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_xlim(0,6)
ax1.set_ylim(0,12)

# Subplot 2
ax2 = axes[0,1]
y = np.array([ 3, 3, 4, 20, 30, 40 ])
assessment = ['Lab3', 'Lab7', 'Lab10', 'A1', 'A2', 'Exam']
ax2.pie(y, labels=assessment)
ax2.set_title('ICT365 Assessments')
```

Object-Oriented Interface

```
# Subplot 3
ax3 = axes[1,0]
np.random.seed(10)
data = { 'a': np.arange(50),
         'b': np.arange(50) + 10 * np.random.randn(50),
         'c': np.random.randint(0, 50, 50),
         'd': np.abs(np.random.randn(50)) * 100
       }
ax3.scatter('a', 'b', c='c', s='d', data=data)
ax3.set_title('Scatter Plot')

# Subplot 4
ax4 = plt.subplot(2,2,4, projection='polar')
ax4.plot([1,2,3,4,5], [9,7,5,3,1], 'r*-')
ax4.set_title("Figure 4")

plt.suptitle("Four Examples")
plt.tight_layout()
plt.show()
```

Object-Oriented Interface

- `fig, axes = plt.subplots(2,2, figsize=(10,8), dpi=120)`
  - `plt.subplots` returns the object representing the figure, and a 2x2 tuple representing the 4 axes (note: `plt.subplots`, not `plt.subplot`)
  - `figsize` specifies the width and height of the figure in inches.
  - `dpi` specifies the dots per inches
- `plt.subplot(2,2,4, projection='polar')`
  - `projection` specifies that the axes uses the polar coordinate system instead of the cartesian coordinate system.

References

- W3school:

  https://www.w3schools.com/python/matplotlib_intro.asp

- Tutorialspoint

  https://www.tutorialspoint.com/matplotlib/index.htm

- Matplotlib Tutorial

  https://matplotlib.org/stable/tutorials/index