Visualisation Workshop

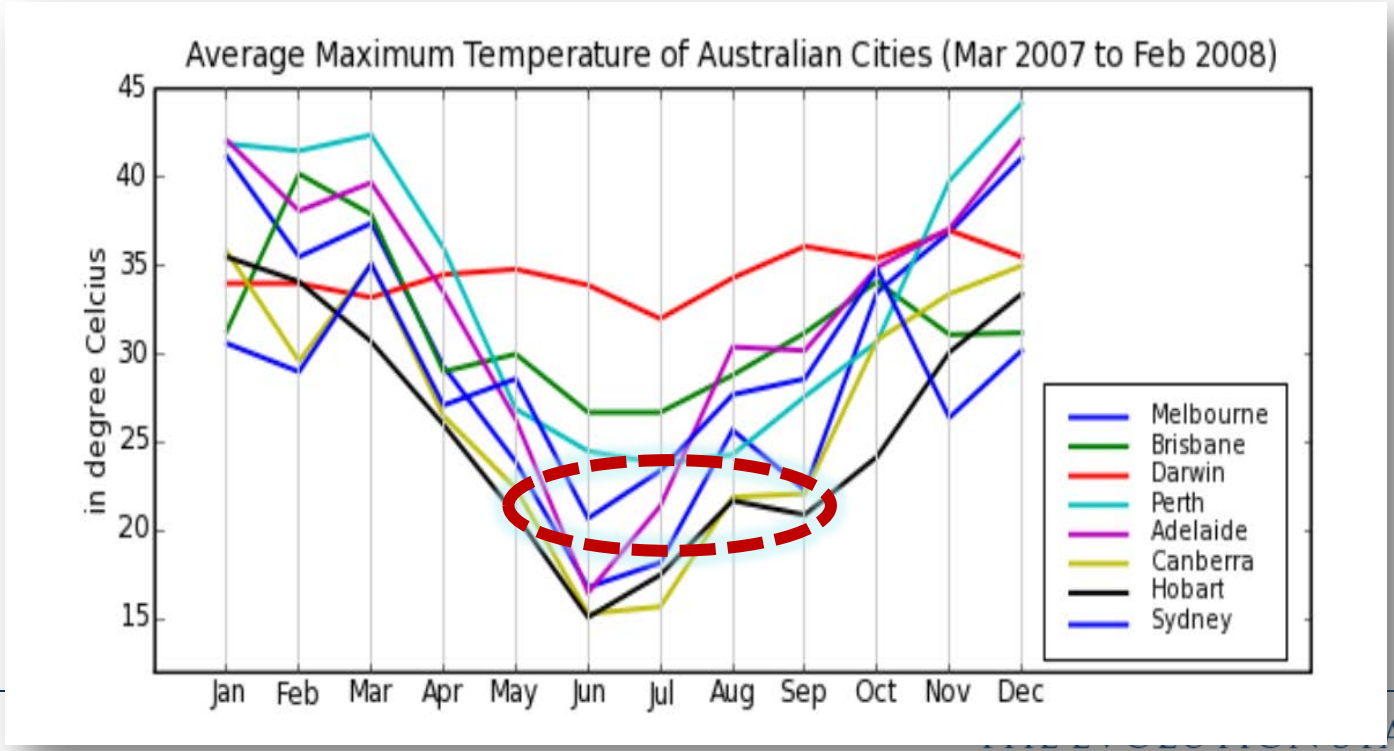
INFO20002: Foundations of Informatics Week 7

Outcomes

- 1. distinguish different forms of visualisation (bar chart; histograms, scatter plots, line plots etc.)
- 2. learn to visualise datasets by using Python library `matplotlib`
- 3. customise the output plot, e.g. texts, plot elements

Exercise 1- Why visualisation

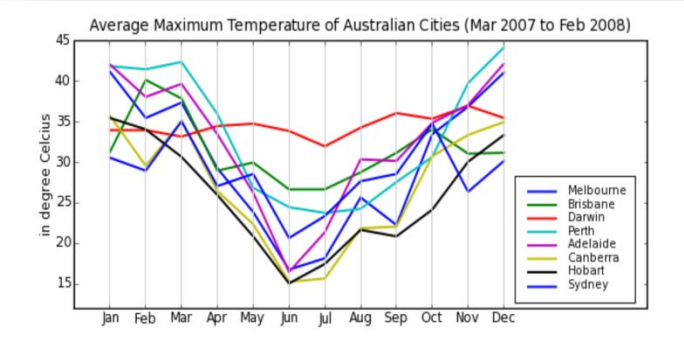
- 1. Average maximum temperature of Australian cities (from March 2007 to February 2008)
- 2. Observe the table and find the “lowest maximum temperature” is in which city during which period?

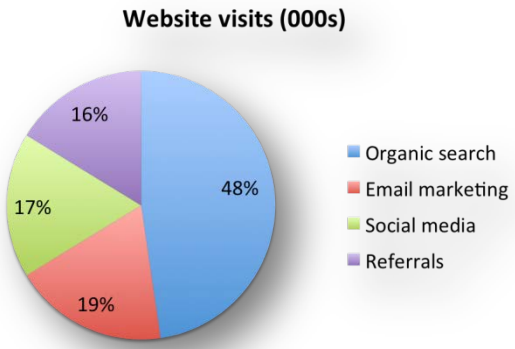


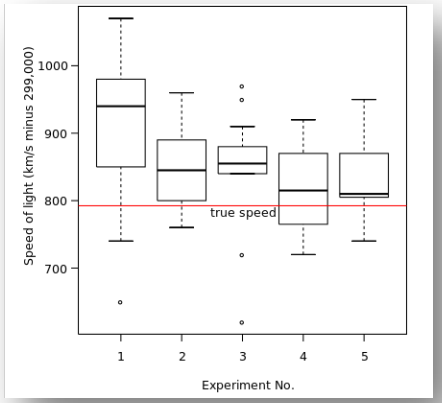
Selection of visualization

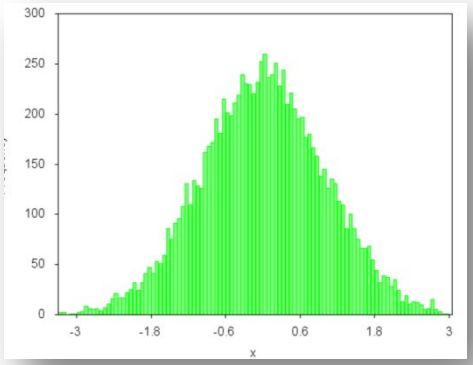
- Graphs are describing general relationships – routers, family trees, vocabulary, hierarchical HR structure
- Maps are stressing the location/position/function – google map, instructions etc.
- **Charts** are stressing the scientific rules identified from data (mathematical) – pie/bar/histogram
- **Plots** are the most basic charts, drawing lines and markers by using coordinates - line/scatter

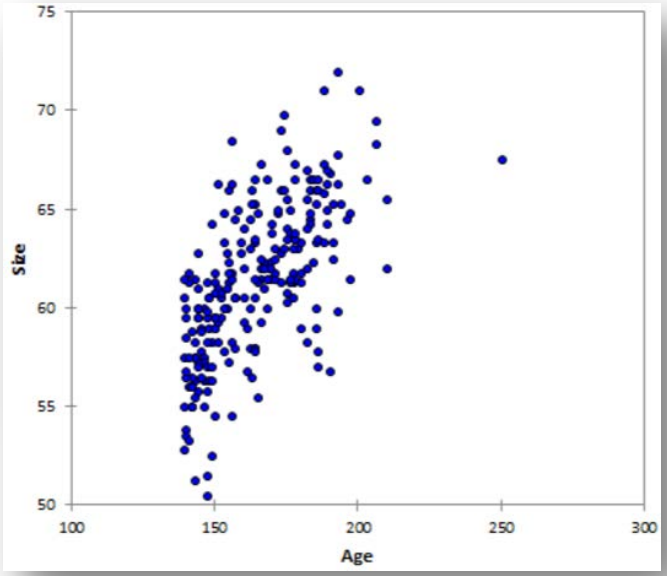
Types of visualization

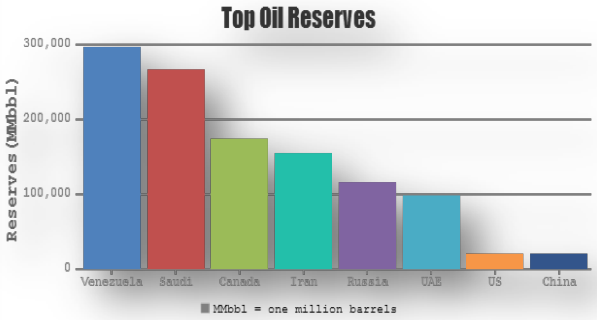












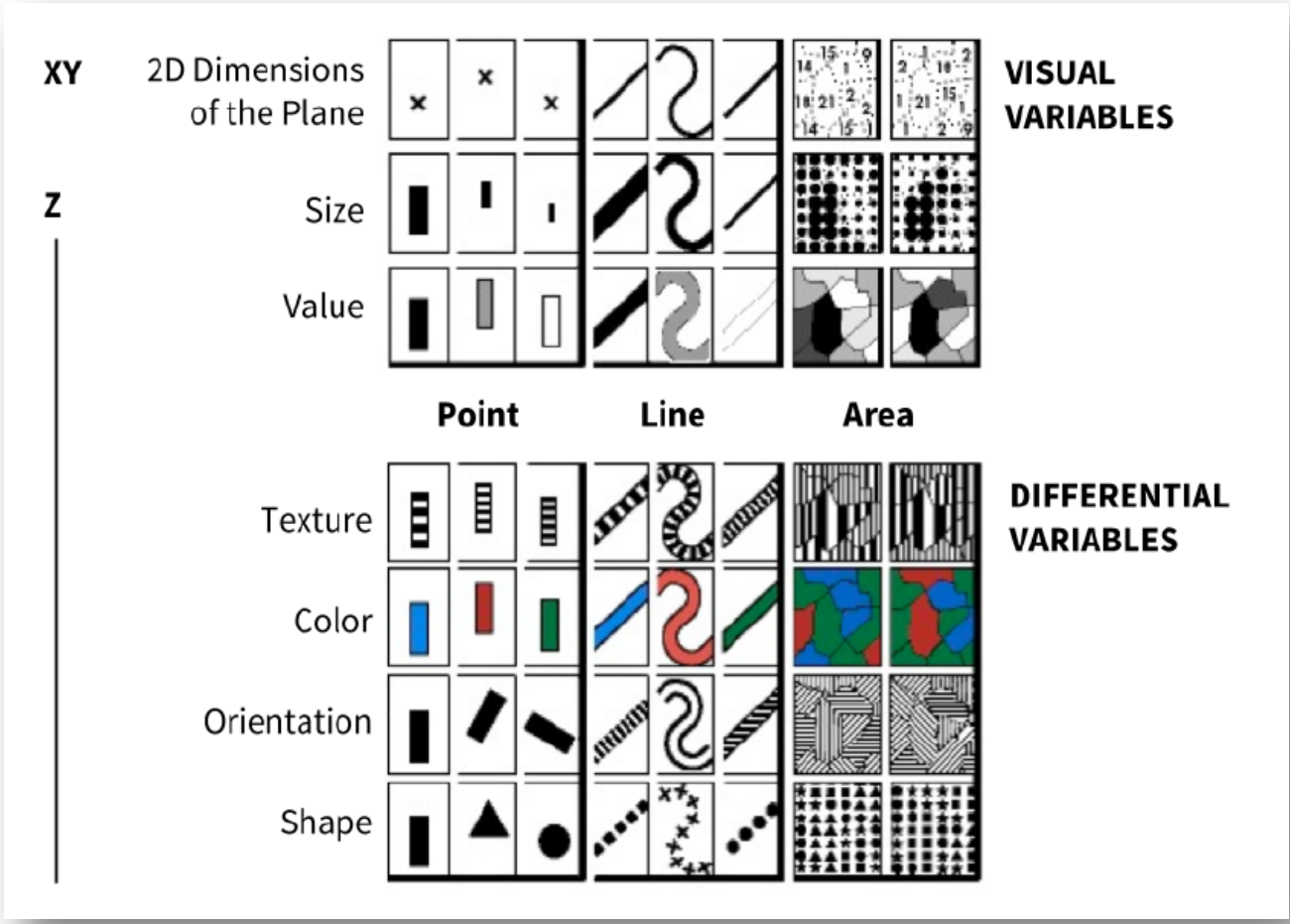
Selection of visualization

- Line plot – relationship (and trends)
- Scatter plot- relationship (and clusters)
- Pie chart – comparison
- Bar chart – comparison
- Histogram – distribution
  - x-axis variable is continuous or categorical
  - sum all the y-axis values
    - Tweets posting during 24h
    - GDP of Australia, US, China and Japan

Elements of visualization

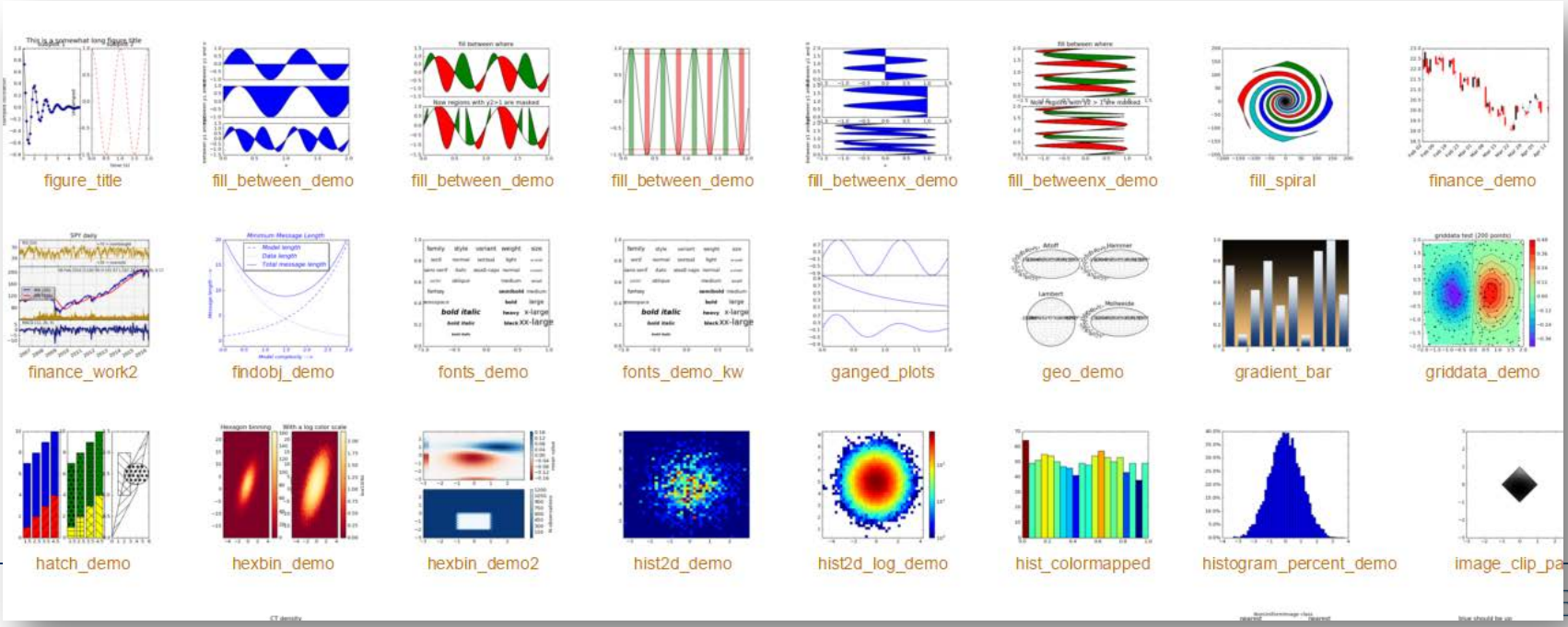
- In principle, visualization is transforming the data into basic visual elements:

- 1. Location (x, y)
- 2. Color (hue)
- 3. Texture
- 4. Shape (marker)
- 5. Line/point/area
- 6. Text



Matplotlib – visualise with python

- Plotting library for Python
- http://matplotlib.org
- Tutorial - http://matplotlib.org/users/pyplot_tutorial.html
- Cookbook - http://wiki.scipy.org/Cookbook/Matplotlib
- Gallery - http://matplotlib.org/gallery.html



matplotlib – library for rendering

- **Part one - The matplotlib APIs**

```
>>>Import matplotlib
```

- **Part two - device dependent backend**
  - Specify a **drawing engine** that can render the visual to a file or a display device.
    - **'PS**' for creating postscript file
    - **'SVG**' for creating scalar vector graphics (SVG file)
    - **'Agg**' for creating PNG file

```
>>> matplotlib.use('Tkinter') – Windows OS rendering
>>> matplotlib.use('Agg')  - save and then print/read - IVLE
```

To display the plot result in a web page, put this c

```
import matplotlib
matplotlib.use('Agg')
```

and this code at the end:

```
savefig("plot.png", dpi=100)
print 'Content-Type: text/html'
print
print '<html><body>'
print '<img src="plot.png"/>'
print '</body></html>'
```

To send the plot result directly to the browser, put this code at the start of your script:

```
import matplotlib
matplotlib.use('Agg')
```

and this code at the end:

```
savefig("plot.png", dpi=100)
print 'Content-Type: image/png'
print
print open("plot.png").read()
```

matplotlib – library for rendering

- **Part three -** The **pylab** interface
  - **function collection**
  - Convenience choice

```
>>> from pylab import *
>>> plot(x,y)  # boxplot(), and bar()
Alternatively, ----------
>>> import matplotlib.pyplot as plt
>>>plt.plot()
>>>plt.show()
# oop plotting library
# For non-interactive plotting, we suggest using
  pyplot
```

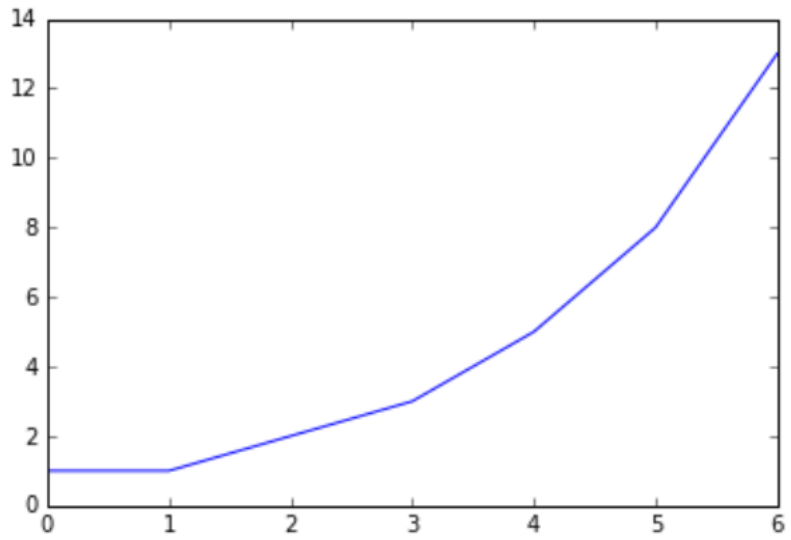
Function "plot()"

- Plot(x, y) – draw lines or markers based on a series of coordinates

- Plot(x) – a single list of arguments will be considered as a list of y-values (default x-values)

```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot([1,1,2,3,5,8,13]) # a plot of fibonacci sequence
```

```
[<matplotlib.lines.Line2D at 0xee549b0>]
```

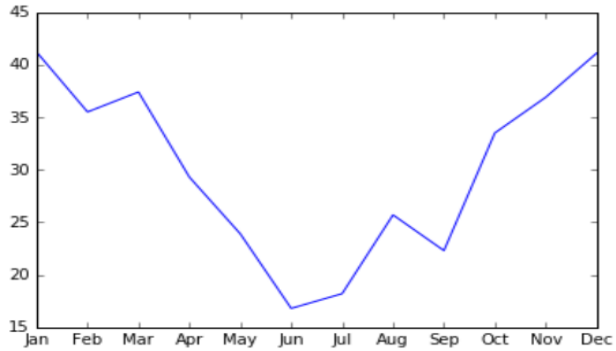


Function “plot()”

- You can supply the **xticks**() with a list of values and a list of texts

```
%matplotlib inline
import matplotlib.pyplot as plt
import calendar

# Melbourne maximum temperature Mar 2007 - Feb 2008
t = [41.2,35.5,37.4,29.3,23.9,16.8,18.2,25.7,22.3,33.5,36.9,41.1]
# print 'jan' - 'dec'
plt.xticks(range(12),list(calendar.month_abbr)[1:])
plt.plot(t)
```

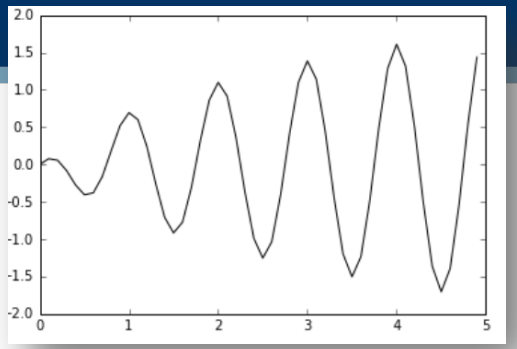
```
[<matplotlib.lines.Line2D at 0xec889b0>]
```




```
>>> plt.xticks(range(12), list(calendar.abbr_month()))
```

  - [0,1,2,3,4,5…,11]
  - [Jan, Feb, …,Dec]

Rendering formula - Cosine



See the example of “Cosine”

```
>>> from numpy import arange
# numpy – a package for scientific computing
```

- **arange** (start, end, step)
  - Return evenly spaced values given an **interval**.

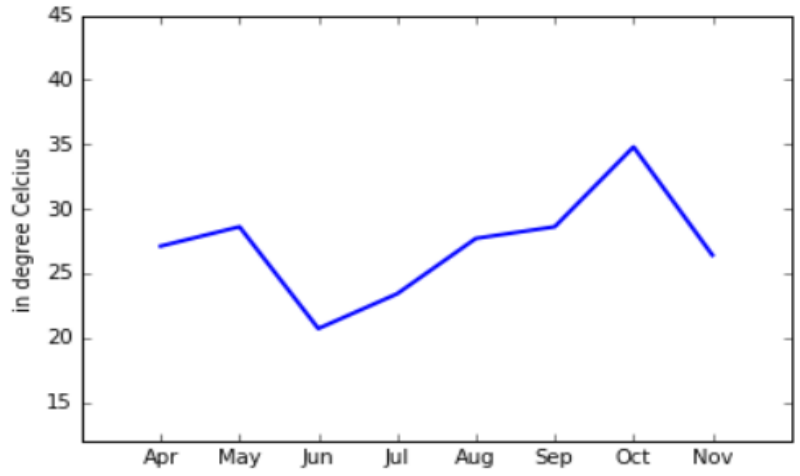
- What if change the precision to ‘0.02’? – see the change
- In the mathematical formula example, change the definition of f(t) to `sin(2*pi*t)*exp(-t)` and **see the impact of the change to the result.**

Exercise 3 – Plot data

- Modify the example on Melbourne's maximum temperature to display Sydney's Maximum temperature from **April 2007 to November 2007.**
- What you can do:
  - observe the CSV data (data/max_temp.csv), and then read and process
  - add months on the x-axis
- You can do more:
  - label the meaning on axes – xlabel(), ylabel()
  - Set limitation for axes values– xlim(), ylim()

solution

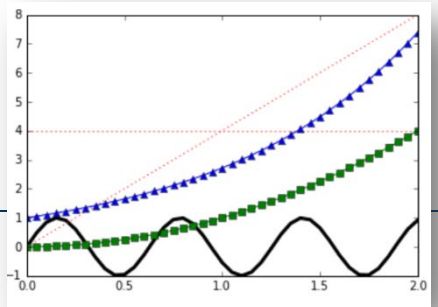
```
import matplotlib
import matplotlib.pyplot as plt
import csv
# cells in your notebook are working together...for instance here the nunmpy
#is not necessary
data = list(csv.reader(open('data/max_temp.csv')))
start_month = 4
months = data[0][start_month:-1]
line = []
for row in data[1:]:
    if row[0] == 'Sydney':
        for col in row[start_month:-1]:
            line.append(col)
        break

plt.plot(line, linewidth=2)
plt.xticks(arange(len(months)), months)
plt.ylim(12,45)
plt.xlim(-1,8)
plt.ylabel("in degree Celcius")
plt.grid(False)
```



Plot properties

- Optional **arguments** to customize the color or “linestyle” of plot output
  - `>>> plot([1,2,3,4], ‘ro’)` - red circle (marker)
  - `>>> plot([1,2,3,4], ‘bs:’)` – blue dotted square (marker)
- Alternatively, you can use **keyword arguments**
  - `>>> plot(x,y,linewidth=3.0) # thickness is 3.0`
- To set properties of multiple lines, you can use `setp()`
  - `>>> line=plot(x1,y1)`
  - `>>> line.set_marker(‘s’)`

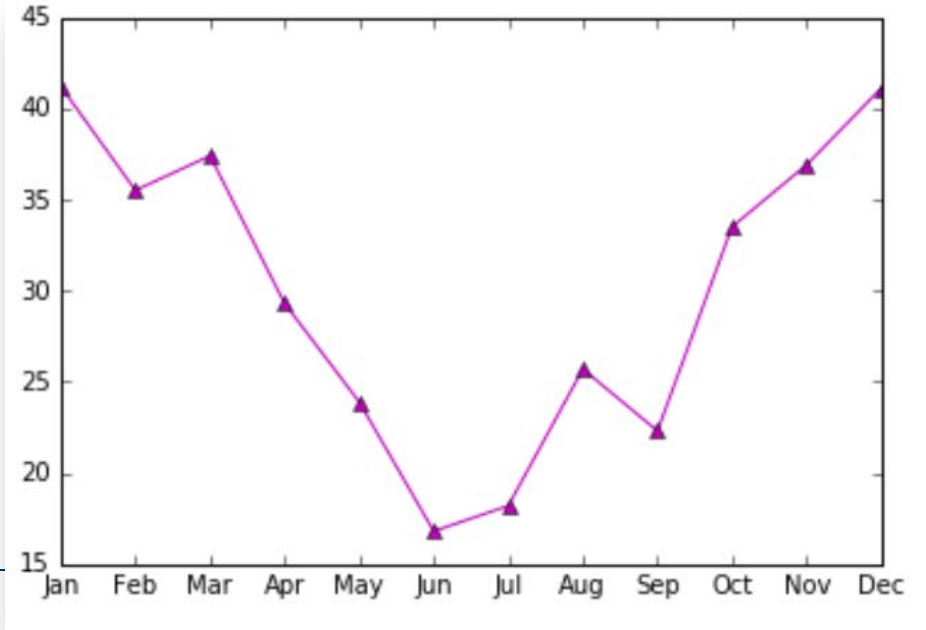
See the example of multi-line plot…



Property	Values
alpha	The alpha transparency on 0-1 scale
antialiased	True or False - use antialised rendering
color	a matplotlib color arg
label	a string optionally used for legend
linestyle	One of -- : -. -
linewidth	a float, the line width in points
marker	One of + , o . s v x > < ^
markeredgewidth	The line width around the marker symbol
markeredgecolor	The edge color if a marker is used
markerfacecolor	The face color if a marker is used
markersize	The size of the marker in points

Exercise 4 – plot line property

- Modify the example on Melbourne maximum temperature in the previous section; produce a plot with magenta colored **triangle** marker. Increase the thickness of the plot line.
  - Tip: check the keyword or set_ functions for certain color and marker



Add Texts…

- xlabel() and ylabel() can be used to add labels to the x and y axis
  - `plt.ylabel("in degree Celcius")`
  - Add more semantic meaning
- xtick() – 2 arguments required
  - A list of values
  - A list of texts go to the value
  - `plt.xticks(range(12), list(ca`
- You also can *customize* the texts

Property	Values
alpha	The alpha transparency on 0-1 scale
color	a matplotlib color argument
fontangle	italic | normal | oblique
fontname	Sans | Helvetica | Courier | Times | Others
fontsize	an scalar, eg, 10
fontweight	normal | bold | light | 4
horizontalalignment	left | center | right
rotation	horizontal | vertical
verticalalignment	bottom | center | top

Exercise 5 – Adding texts

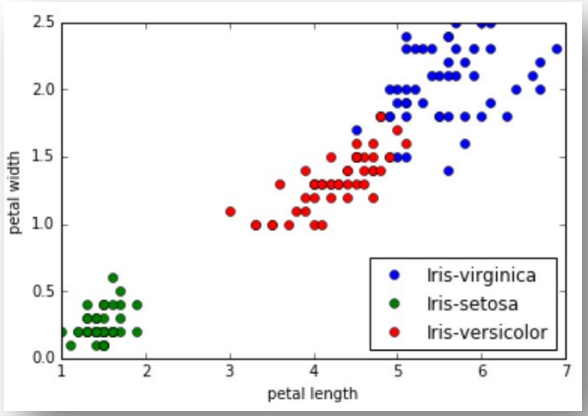
- Add the following lines of code to the example on Melbourne maximum temperature. Replace the xticks() command with the supplied code. Run to see the effect.

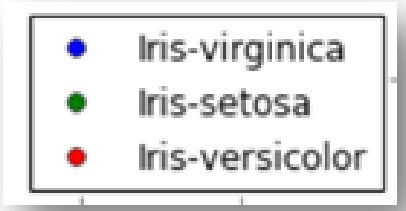
```
>>> plt.xticks(range(12), list(calendar.month_abbr)[1:], rotation=40)
>>> plt.title("Melbourne maximum temperature (Dec 07 - Feb 08)", fontsize=18)
>>> plt.ylabel("temperature in celcius", colors='red')
>>> plt.xlabel("months", fontsize=14)
```

Scatter plot – example Iris flowers

`Find the data/iris.csv`

- Scatter plot is used to display the relationship between x and y

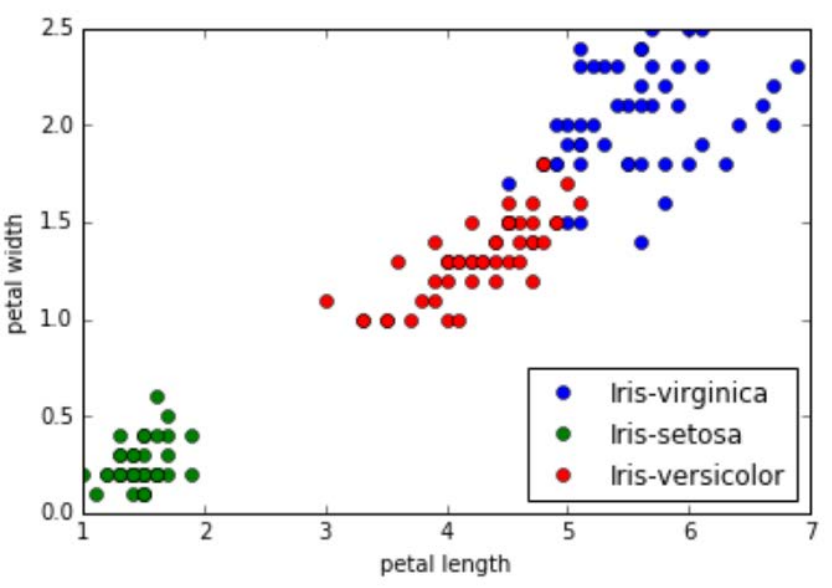




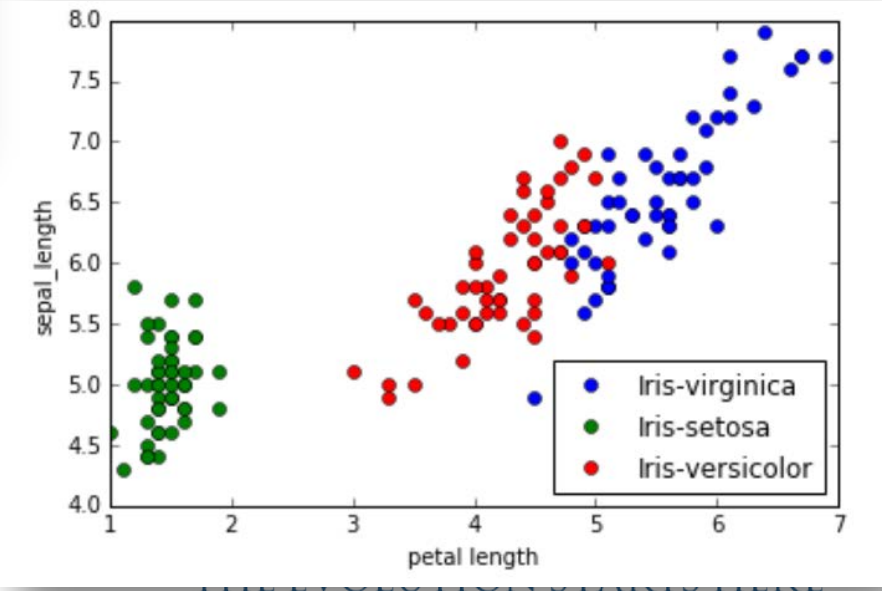
- Legend is useful in multi-variable cases
  - `plt.legend(`**`Keys`**`, loc='lower right', numponint = 1)`

Exercise 6 – scatter plotting – Irise flowers

- Can you identify some rules from the clusters in following plot?

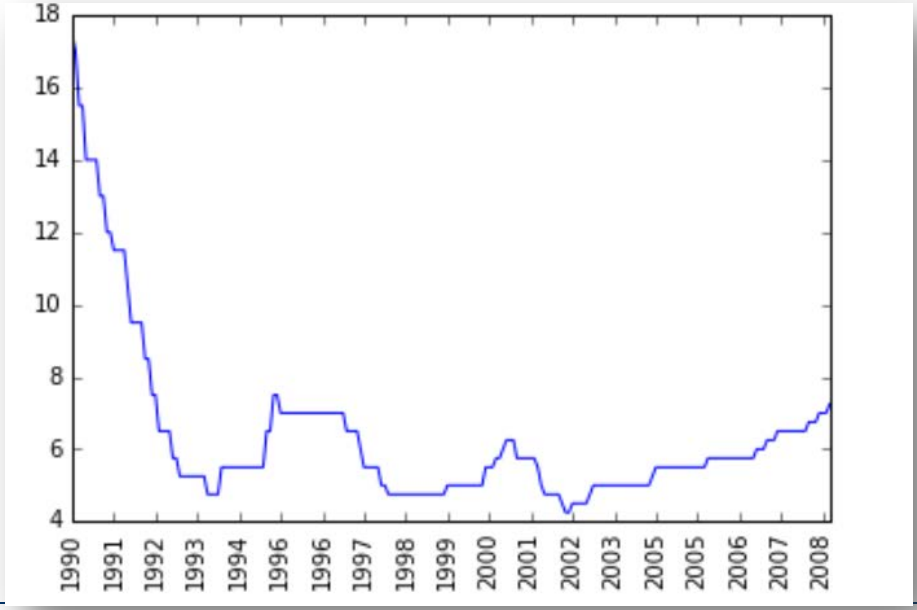


- Modify the example of 'Iris flowers
of **petal length** and **sepal length**



Time series

- Historical information
  - Deduce trends
  - Predict future
  - Observe relations
- See the example of “Reserve Bank of Australia every month”



Bar chart – showing comparison and contrasts

```
>>>import matplotlib.pyplot as plt
......
plt.bar(a,b)
```

Clustered bar charts – always based on my “hypothesis”

Horizontal bar chart - rotate your bar chart by 90 degrees

```
plt.barh()
```

Histogram – distribution

```
plt.hist()
```

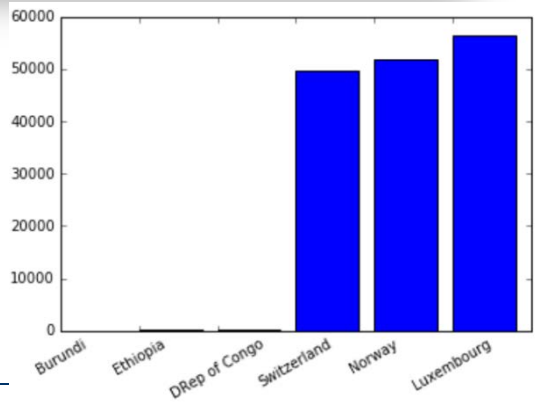
Pie chart – comparison of proportions

```
plt.pie()
```

Please see the examples on Hands-on as the recipe of your work

- For example, the bar chart below displays the GNP per capita of the three poorest and the three richest countries in the world (based on 2004 GNP per capita):

```
%matplotlib inline
import matplotlib.pyplot as plt
import calendar

countries = ['Burundi','Ethiopia','DRep of Congo','Switzerland','Norway','Luxembourg']
gnp = [90,110,110,49600,51810,56380] # GNP per capita (2004)
plt.bar(arange(len(gnp)),gnp)
plt.xticks( arange(len(countries)),countries, rotation=30)
```



THE UNIVERSITY OF

MELBOURNE