# Practical 3
## Arrays and Plotting

**Learning Objectives**

1. Use Python arrays implemented in Numpy
2. Use simple plotting techniques using matplotlib
3. Apply arrays and plotting to more complex systems dynamics problems

**Overview**

In this practical you will be using Numpy arrays to store data. We will then plot data from arrays and lists before using arrays and plotting in some more complex systems dynamics models.

## Tasks

### VIM – additional useful commands

| Command | Description |
|---|---|
| `:w` | When editing a file, you can save changes so far using ":w" from command mode. Press "esc" to go from insert to command mode. |
| `:w filename` | If you want to save a file with a new name from vim command mode, type ":w new_file_name" |
| `:wq` | To save the file and quit, type ":wq", which is equivalent to "ZZ" |
| `:q!` | To quit without saving changes, use ":q!" (also good for backing out if you accidentally put the wrong file name in, e.g. vim growth.py) |
| `D` | To delete the rest of a line (from current cursor position in command mode), type "D" |
| `R` | To replace the rest of a line (from current cursor position in command mode), type "R", puts you into insert mode |
| `u` | To undo a command or change, type "u", repeat to undo multiple |
| `xG` | To go to a line 20 in a file, type "20G". To go to the last line of a file, type "G' |
| `A` | Appends after the end of the current line, puts into insert mode |

On occasion, you may accidentally hit "`ctrl-z`" when using vim. This closes vim, but the program is still running in the "background". Type "`fg`" to bring it back into the foreground. When this happens, or if you close your machine without saving the files, a temporary file that vim creates is left behind (when you save and quit normally, the file is deleted). If you type "`ls -la`", you can see these "hidden" files – they start with a ".", eg. "`.growth.py.swp`". Once you have your file back in order, you can delete the temp files using "`rm .growth.py.swp`".

## 1. Plotting Growth

The lectures notes gave modified code for growth.py to plot the output. Copy growth.py from your Prac01 directory into your Prac03 directory. Rename it growthplot.py and update the documentation at the start of the program. Then make the changes as indicated in the lecture notes. This includes inserting code for importing matplotlib; creating and appending to lists; and plotting the data.

Run the program and confirm that it plots your data.

Make the following modifications to your code (do each modification and confirm it works before moving onto the next one):

1. Change the colour of the plotted line from blue to red
2. Change the symbol for the plotted line to a triangle. Note that the line is formed from many individual data points, these are joined together when we use a line in our plot
3. Change the simulation time from 10 hour to 100 hours, now we can see the exponential growth in the population
4. Change the plotting back to a line
5. Change the plot title to “Prac 3.1: Unconstrained Growth”
6. Save the plot to your Prac03 directory

## 2. Reading Numbers with Arrays

In numbers2.py (Prac01) we read in ten numbers and printed their total. Copy numbers2.py from Prac01 to Prac03/numbersarray.py. We will change this file to use arrays to store the values and then print some summary data. Make the changes below and run the program:

```
#
# numbersarray.py: Read ten numbers give sum, min, max & mean
#
import numpy as np


numarray = np.zeros(10)      # create an empty 10 element array


print('Enter ten numbers...')

for i in range(len(numarray)):
    print('Enter a number (', i, ')...')
    numarray[i] = int(input())

print('Total is ', numarray.sum())
```

Modify the code to:
1. Print the min and max numbers entered
2. Print the average (mean) of the numbers
3. Plot the numbers

## 3. Plotting Growth with Arrays

Copy growthplot.py to growtharray.py. We will change this file to use arrays to store the values and then plot the arrays.

1. First, update the documentation accordingly.
2. To use Numpy arrays, we first need to import numpy: `import numpy as np`. Add the import line to the start of the program.
3. Then, create an array of zeros to hold the calculated values
4. Modify the loop code to put the values into the array
5. Modify the plt.plot call to plot the array
6. If you didn't provide x-values for time (in hours), add code for x-values

## 4. Plotting Subplots

Copy growtharray.py to growthsubplot.py. We will change this program to give multiple plots in the same figure.

1. Update the documentation accordingly
2. Modify the plotting code to do the do the ***equivalent*** of the subplot code in the lecture slides (variable names and labels/titles will need to be changed)

```
plt.figure(1)

plt.subplot(211)
plt.plot(dates, march2017, '--')
plt.title('March Temperatures')
plt.ylabel('Temperature')

plt.subplot(212)
plt.plot(dates, march2017, 'ro')
plt.ylabel('Temperature')
plt.xlabel('Date')
plt.show()
```

Save the resulting plot in your Prac03 directory.

## 5. Plotting a Bar Chart

Copy numbersarray.py to numbersbar.py. Update numbersbar.py to print a bar chart of the numbers. In the lecture notes, we saw how to plot a bar chart from a list. We will use similar code to plot the numbers entered into numbersbar.py

```
plt.title(Numbers Bar Chart')
plt.xlabel('Index')
plt.ylabel('Number')
plt.bar([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], numarray, 0.9,
color='purple')
plt.show()
```

Add this code to numbersbar.py to print a purple bar chart. Remember to import matplotlib!

Save it to your Prac03 directory.

## 6. Systems Dynamics Revisited

In `growth.py` we implemented a simulation of unconstrained growth. We can use the same approach to simulate decay – using negative growth. In this example, we can look at a dosage of a drug, e.g. Aspirin for pain and Dilantin for treating epilepsy.

Download `dosage.py` from the Practical 3 area on Blackboard and save it into your Prac03 directory. Run the program and see if you can understand what it is doing. Look at chapter 2 of the text for background. The program `dosage4hr.py` is a variation of dosage.py where another two tablets are taken after 4 hours.

Next download `repeatdosage.py` from Blackboard and run it. MEC and MTC are values for effective and toxic concentrations, respectively. Note how it takes multiple doses to get up to an effective level. Download `skipdosage.py` and see the impact of skipped pills on the concentration.

For more background information, this exercise is based on p45-50 Chapter 2 of the Shiflet & Shiflet textbook - http://press.princeton.edu/chapters/s2_10291.pdf .

## 7. Update README file

Update and README file to include growthplot.py, numbersarray.py, growtharray.py, growthsubplot.py, numbersbar.py, dosage.py and repeatdosage.py, along with any additional programs and charts you have created.

## Submission

All of your work for this week's practical should be submitted via Blackboard using the link in the Week 3 unit materials. This should be done as a single "zipped" file – Prac03_<student_id>.zip.

## Reflection

1. **Knowledge**: What are the names of the two Python packages we use for arrays and for plotting?
2. **Comprehension**: What changes if we replace plt.xlabel('Count') with plt.xlabel('Time')
3. **Application**: What value (??) would you give to plt.subplot(??) to set up the 2nd plot in a 2x2 set of subplots?
4. **Analysis**: What type of file is created when we save a plot?
5. **Synthesis**: Each week we create a README file for the Prac. How is this file useful?
6. **Evaluation**: Compare the use of lists and arrays in the growth*.py programs. Name two advantages of using lists, and two advantages of using arrays

## Challenge

For those who want to explore a bit more of the topics covered in this practical. Note that the challenges do not need to be submitted but may be used as questions in tests and the exam.

1. Modify growthsubplot.py to print four subplots (2x2) – the additional plots should print green squares and black triangles. (hint: subplot 1 is subplot(221))
2. Modify growthsubplot.py to print nine subplots (3x3) – the additional plots should print green squares, black triangles, black circles, black squares, blue triangles, blue circles and blue squares. (hint: subplot 1 is subplot(331))
3. Extend the aspirin simulation length in dosage4hr.py to see what happens over time with repeated dosages
4. Modify dosage4hr.py to see the impact of having doses every 2 hours