Python Programming
Exercises

Control Structures

2.1 Write a program that inputs a 4 digit year and then calculates whether or not it is a leap year. Note that there are 4 cases to consider:

years divisible by 400:	e.g. 2000	is a leap year
years divisible by 100:	e.g. 1900	not a leap year
years divisible by 4:	e.g. 2012	is a leap year
other years :	e.g. 2011	not a leap year

2.2 Using a variation of the above program, calculate the number of days in the inclusive date range *'1st January 2000'* to *'31st December 2999'*.

2.3 Write a program that prints out the square, cubes and fourth power of the first 20 integers.

2.4 Write a program that prints out the first 20 Fibonacci numbers. You can use the web to find out the definition of the Fibonacci sequence.

2.5 Write a program that calculates the ratio of successive pairs of the first 20 Fibonacci numbers. Does the ratio appear to converge to a number?

Data Types

3.1 Write a program that prints out the sum, difference, product and dividend of two complex numbers.

3.3 Use the *range* function to generate two separate tuples containing the list of integers from 10 to 19 and from 30 to 39. Tuples are immutable, so how can you form a tuple that has all the elements of the other two tuples?

3.4 It is not possible to invert dictionaries in the general case, because unlike the dictionary values, keys must be unique and immutable. However the following dictionary can be inverted:

```
salaries = {
     "John"   : 45000,
     "Sheila" : 48000,
     "Vivien" : 31000,
     "Roy"    : 37000,
     "Ruth"   : 50000
 }
```

So write code to interchange the keys and values of the above.

Functions

4.1 Write a function that rotates the values of 3 variables. For example:

```
x = 100
y = 200
z = 300
Rotate( ... )
# x is now 200
# y is now 300
# z is now 100
```

4.2 Write a function that rotates the values of an array. For example:

```
array = [100, 200, 300, 400, 500]
Rotate( ... )
# array is now: [200, 300, 400, 500, 100]
```

4.3 Write a function to calculate Factorials. Try out

```
factorial(1)
factorial(10)
factorial(40)
factorial(100)
```

4.4 Write a function that takes a string and capitalises the first character of the string and ensures the remaining characters are converted to lower case. Use the following test data:

```
UpperFirst("test1")
UpperFirst("mIxEdCaSe")
UpperFirst("UPPER")
UpperFirst("lower")
UpperFirst("oPPOSITE")
```

4.5 Write a function that takes an integer list as a parameter and doubles the value of each element of the array.

4.6 Write a function that takes two *int* arrays (of the same size) as parameters and adds the arrays together, element by element. Return the summed array.

Exception Handling

5.1 Write a program that calculates the factorial of an integer in the range 2 to 10. Add exception handling code to prevent calculating a result where the input number is larger than 10, or any negative integer. Make sure you can handle the case where the input is not an integer.

Classes

6.1 Create a class that represents a bank account. Add methods to allow a customer to **deposit()** and **withdraw()** money and provide a method **getBalance()**. Write a test program to check out your class.

6.2 Modify the previous example to add facilities for providing an overdraft. You will need to define a **setOverdraft()** method.

6.3 An object starts moving at from position s0 with velocity v0 and gains a constant acceleration of a. After time t, compute its position and velocity.

Write a class to define the particle and provide the following methods:

```
    def accelerate(self, a, t):
        # dv/dt = a
        # ds/dt = v
        # v = ∫a.dt = at + s0
        # s = ∫v.dt = ∫at.dt = at^2/2 + s0.t + v0

    def getPosition(self):

    def getVelocity(self):
```

Perform all calculations in 3D space. Use numpy to simplify the calculations

6.4 A relativistic particle, B travels at 3/4 of the speed of light in the x direction as viewed from an observer in frame of reference A.
A second relativistic particle, C travels at 3/4 of the speed of light in the z direction as viewed from an observer in B's frame of reference.
What is speed the of particle C as viewed from the observer in frame A?

Write generalized code to solve the above problem; try some other different velocities for particle C.

The relativistic formulae you require can be found at:
`http://www.math.ucr.edu/home/baez/physics/Relativity/SR/velocity.html`

Files

7.1 Write a program that emulates the word count program: wc. Use the file "zen.txt" as input and print out the number of characters, words and lines in the file.

7.2 Write a program that copies a text file to another file.

7.3 Create a file call TestData.txt with test data consisting of one number per line using your favourite editor. Your job is to read the entire file into memory so that you can compute the sum of all the numbers.

7.4 Try the previous example with other test files that may contain non integral data. Use exception handling to filter out lines that don't contain integers.

7.5 Write a program that concatenates three files into a new file.

7.6 Write a program that reads a file, reverses the order of lines in the file and then saves the changes in a new file.

Numpy

1. Create 4x6 array of floats between 3.5 and 5.5. Display the results to 3 decimal places.

2. Create a 4x6 array of random ints between 10 and 100.

3. Create a 4x6 array of random ints between 10 and 100. Set all the even numbers to 100.

4. Create a 4x6 array of random ints between 10 and 100. Create a new 1D array with all the numbers in the first array that are less than 50.

5. Create a 4x6 array of random integers between 10 and 100. Create a new 4x3 array of the lowest 50% of the original array whilst preserving order.

6. Create a 4x6 array of random integers between 10 and 100. Set the lowest 50% of the array to 0.

7. Create a 6x4 array of ints between 10 and 14 where the first row is filled with 10, the second with 11 and so on.

8. Create a 8x5 array of ints using the "tile" method that looks like:

```
    [[10 11 12 13 10]
     [11 12 13 10 11]
     [12 13 ...
           ... 11 12]
     [13 10 11 12 13]]
```

9. Create a 4x6 array of ints. Swap columns 2 and 4.

10. Create a 3x10 array. Rotate the array by 90 degrees, 4 times, so you get back to the original array.

11. Find the intersection of two 1D arrays.

12. Create two 1D int arrays of equal size. Create a new array of the same size, selecting elements from the first array when the element in this array is even, and selecting elements from the second array when the element in the first array is odd.

13. Create a 4x6 array of random ints between 10 and 100. Look up what it means to define a masked array and then create a masked array where all values in the original array that are less than 50 are masked out.

14. Create a 4x6 array of random ints between 10 and 100. Filter this array and produce a 1D array of the numbers divisible by 3.

15. Create a 4x6 array of random ints between 10 and 100. Then create a 4x6 array of indices of representing rankings of the first array. For example a 2x3 array

```
[[49 56 61]
 [63 39 97]]
```

yields the rankings

```
[[4 0 1]
 [2 3 5]]
```

16. Create a 1D array of random ints of size (30,) containing only the numbers 10, 11, 12, 13 or 14. Now create 30 'one hot encodings' for the original array. For example a set of 'one hot encodings' for:

```
[20, 22, 21, 22, 20, 22]
```

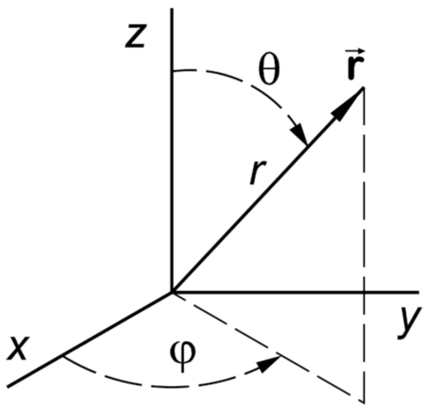
would be:

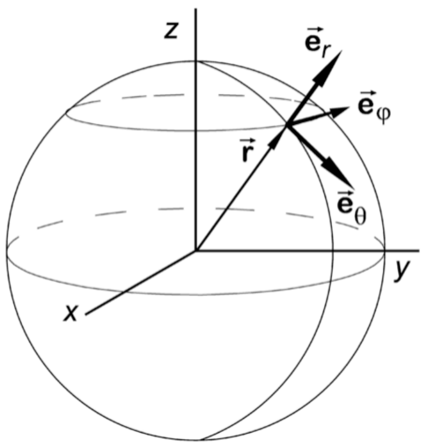
```
[1 0 0]
[0 0 1]
[0 1 0]
[0 0 1]
[1 0 0]
[0 0 1]
```

17. Create two vectors representing points in 3D space. Calculate the distance between the points.

18. Perform a Cartesian to Spherical coordinate transformation on each of the Cartesian unit vectors:

```
iHat = [1, 0, 0]
jHat = [0, 1, 0]
kHat = [0, 0, 1]
```





Note that the transfomation for kHat causes a singularity in Spherical coordinates, so try unit coordinates, so try vectors very near kHat instead:

```
kHat_plus  = [+0.0001,0,1])
kHat_minus = [-0.0001,0,1])
```

Pandas

In all these questions use the same source file:

```
../MiniProject/wtk_site_metadata.csv
```

1. Print the column names of the csv file.

2. Create a new dataframe with just the 'longitude' and 'latitude' columns.

3. Create a new series from the 'State' column of the dataframe and then remove duplicates.

4. Print out the row with the largest 'longitude'.

5. Create a new dataframe containing rows with state equal to 'Rhode Island'. Retain the 'longitude' and 'latitude' columns. Reset the index such that it starts at 1.

6. Create the same dataframe 'Rhode Island' as in the previous exercise. Print the dataframe without the index, display every row to a precision of 2 decimal places

7. Create the same dataframe 'Rhode Island' as in the previous exercise. Now add an extra column called 'power' that is computed as the product of the columns 'capacity_factor' and 'wind_speed'.

8. Work with the dataframe 'Rhode Island' as in the previous exercise with the extra column called 'power'. Reorder the columns such that the first three columns are 'capacity_factor', 'wind_speed' and 'power' and print the first 5 rows of the dataframe.

9. Work with the dataframe 'Rhode Island' as in the previous exercise with the extra column called 'power' and the column headings resorted. Now sort the dataframe in descending order of 'power'.

10. Work with the dataframe 'Rhode Island' as in the previous exercise with the extra column called 'power' and the column headings resorted. Create a new sorted dataframe with rows from the 'Rhode Island' dataframe with a 'power' greater than 3.

Intermediate Questions

8.1 removeDuplicatesFromLists

How do you find the intersection of 2 lists (i.e. those elements common to both lists)? For example:

```
list1 = [ 1, 3, 5, 7, 5, 4, 3, 2, 1 ]
list2 = [ 2, 4, 6, 8, 6, 4, 3, 2, 1 ]
```

have these elements in common:

```
[1, 2, 3, 4]
```

8.2 palindromes

Write a function that takes a list as a parameter and checks if it is a palindrome. Use the following test data:

```
listA = [ 5, 7, 3, 22, 15, 6, 15, 22, 3, 7, 5 ]         # is palindrome
listB = [ 5, 7, 3, 22, 15, 6, 6, 15, 22, 3, 7, 5 ]      # is palindrome
listC = [ 5, 7, 3, 22, 15, 6, 7, 15, 22, 3, 7, 5 ]      # is NOT palindrome
listD = [ 5, 7, 3, 'A', 15, 'Z', 15, 'G', 3, 7, 5 ]     # is NOT palindrome
listE = [ 5, 'A', 3, 'G', 15, 6, 15, 'G', 3, 'A', 5 ]   # is palindrome
```

8.3 flatten

Write a function that flattens multi-dimensional lists. For example:

```
listA = [[1, 2, 3], [4, 5, 6], 7]
listB = [1, [2, 3], [4, 5, [6, [7]]]]
```

get flattened to:

```
[1, 2, 3, 4, 5, 6, 7]
[1, 2, 3, 4, 5, 6, 7]
```

8.4 compress

Write a function that removes adjacent duplicates in a list, such that for example:

```
compress([4,7,5,5,5,3,4,4,4,2,2])
```

gives the output:

```
[4, 7, 5, 3, 4, 2]
```

8.5 hashTables

Study the data in the file:

```
data/customer.txt
```

Read the data from the file and set up a dictionary such that the key is formed from the FIRSTNAME, INIT, LASTNAME fields and the values from the remaining fields. Note that some keys do not have an initial.

Once the hash is complete perform a search for two different entries:

```
key = ('WILLIAM', 'T', 'JONES')
print key, customer[key]

key = ('DAVID', '', 'BROWN')
print key, customer[key]
```

Advanced Questions

A1. Roman Numerals

Create two subroutines that convert a decimal number in the range 1 to 4000 to Roman numerals, and back. You will need to use lookup tables to simplify the code (see 1.py in the resources folder).
Warning: This question is rather more difficult than it looks.

The Roman numerals are:

```
      I    1
      V    5
      X    10
      L    50
      C    100
      D    500
      M    1000
```

Roman numerals generally go from large values to small values, with one notable exception: a single small numeral can be used before a larger numeral, and is then subtracted from that larger numeral. So XC means 90. However, only the following combinations are valid:

```
      IV   4
      IX   9
      XL   40
      XC   90
      CD   400
      CM   900
```

A2. Stems

Word prefixes are also called stems. Write a program that reads the file ***/usr/share/dict/words*** and finds the most popular stems of size 2 to 29 (if you get a tie, just pick one).

For example, the stems for the following input:

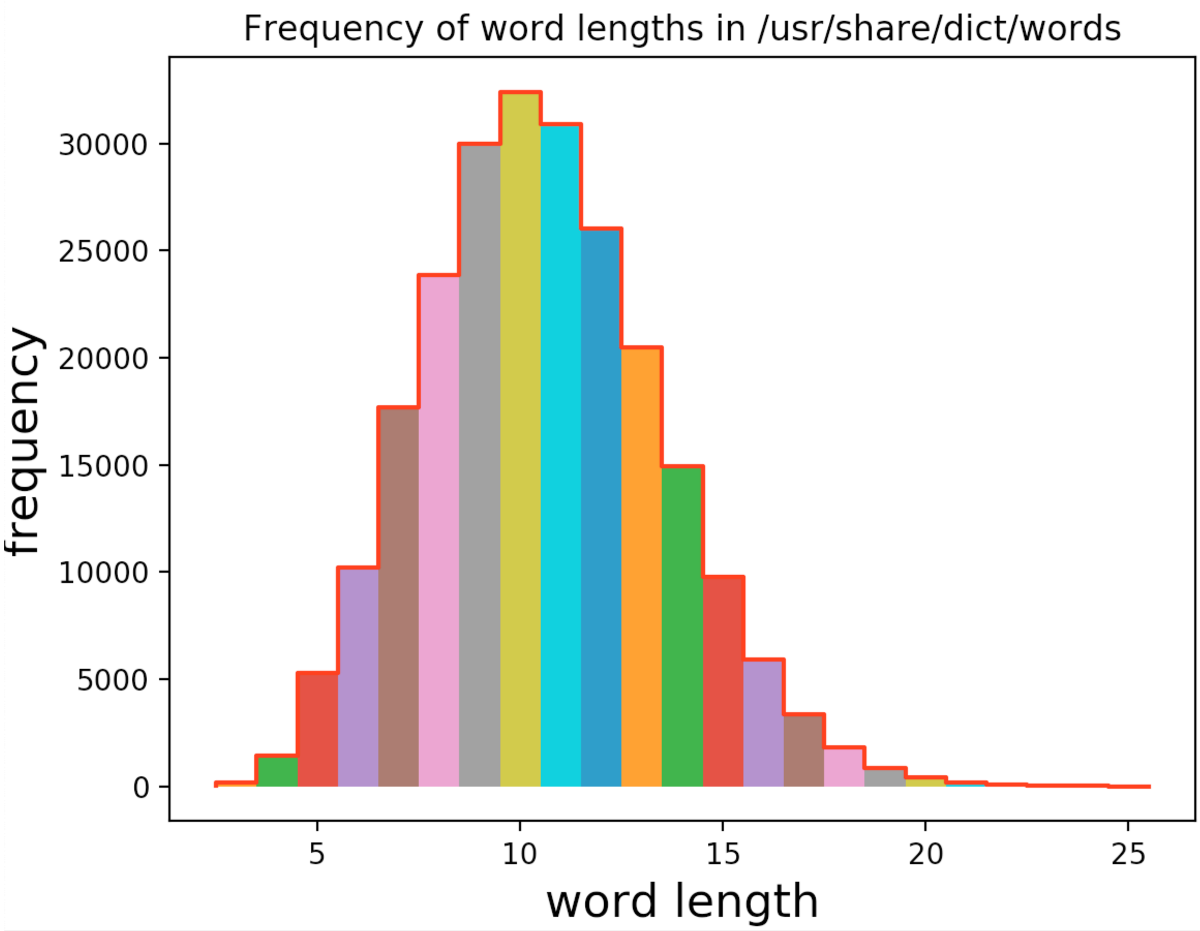
```
test
tester
jest
compute
computer
literate
literal
literacy
continue
collaborate
```

would be:

```
Most popular stem of size 2: co (occurs 4 times)
Most popular stem of size 3: lit (occurs 3 times)
Most popular stem of size 4: lite (occurs 3 times)
Most popular stem of size 5: liter (occurs 3 times)
Most popular stem of size 6: litera (occurs 3 times)
```

A3. Word Frequencies

Read the file ***/usr/share/dict/words*** and print a histogram of the occurrence of each size (in percent), giving output similar to the one below:



A4. Threading

This exercise uses Events to synchronize two worker threads that are working with share prices. Take a look at the following class:

```
class Share:
    # prices are stored in an internal list

    def __init__(self, startPrice):
        '''set up initial list of prices (just one)'''
        self.prices = [int(startPrice)]

    def nextPrice(self):
        '''generate a new share price'''

        # simulate taking some time to get a new share price
        time.sleep(0.01 * random.random())

        # calculate change in price and append to list
        delta = int(random.random() * 100 - 50)
        currentPrice = self.prices.pop() # easy way to get last price
        self.prices.append(currentPrice) # must push it back again
        currentPrice += delta
        self.prices.append(currentPrice) # append new price

    def getPrices(self):
        '''get list of recent prices'''
        prices = self.prices
        currentPrice = self.prices.pop()
        self.prices = [currentPrice]
        return prices # return list as it was before reset

    def reset(self):
        '''reset list of prices'''
        currentPrice = self.prices.pop()
        self.prices = [currentPrice]
```

Objects of this class simulate individual share prices. The share price is in integral units and can vary by plus or minus 50 units each time nextPrice() is called. The class maintains a history of recent prices in self.prices, but the history can be cleared and reinitialized with the current share value by the method reset(). The getPrices() method returns this history list.

1. To simplify this exercise, you will only be working with one share price object. Initialize this share price with the value 1000.

2. Now create two worker threads.

3. The first worker (ticker) should call the nextPrice() method several times (say 10) to generate a history of prices.

4. The second worker (monitor) should call getPrices() to get the history of prices after the ticker thread has generated a complete history. After retreiving this history, print out the average price and then call reset() to clear the history.

5. The two worker threads should repeat their tasks 25 times, ensuring that the ticker thread doesn't start generating a new history until the monitor has printed the average for the

previous history set and the monitor thread doesn't attempt to retreive the history until is has been completely generated by the ticker thread.

6. Use global variables to define your event object references. Recall that to create an event use:

```
    myevent = Event()
```

to wait for an event use:

```
    myevent.wait()
```

to set an event use:

```
    myevent.set()
```

and to clear an event use:

```
    myevent.clear()
```

A5. Scrape Football Results

Write a web scraping program to extract Football results and produce a table that looks like:

```
Arsenal                  5-0 3-0 0-0 2-1 2-0 2-2 2-1 4-1 2-2 1-2 4-1 2-1 1-0 3-0 0-0 0-1 1-1 4-1 3-0
Aston Villa          0-3     0-1 1-2 0-0 3-2 2-1 2-1 0-2 0-2 1-1 0-0 3-3 1-1 1-2 0-0 0-1 1-2 2-1 1-0
Burnley              0-1 1-1     1-3 2-3 1-3 1-0 0-1 0-1 1-0 0-0 1-1 2-1 1-0 0-0 0-0 0-1 0-0 2-2 1-3
Chelsea              2-0 3-0 1-1     1-0 1-0 2-0 2-0 1-1 1-1 1-0 2-0 2-1 1-1 2-1 3-1 4-2 3-0 2-0 2-0
Crystal Palace       1-2 0-1 0-0 1-2     0-1 0-2 2-0 3-1 2-1 1-2 1-1 3-1 1-3 1-1 1-3 1-0 2-1 0-2 1-3
Everton              2-2 3-0 1-0 3-6 2-3     1-1 2-2 0-0 1-1 3-0 3-0 3-1 1-0 0-1 0-2 0-0 0-1 0-0 2-1
Hull City            1-3 2-0 0-1 2-3 2-0 2-0     0-1 1-0 2-4 0-0 0-3 2-1 0-1 1-1 1-1 0-1 1-2 0-0 2-2
Leicester City       1-1 1-0 2-2 1-3 0-1 2-2 0-0     1-3 0-1 5-3 3-0 5-1 2-0 0-1 0-0 2-0 1-2 0-1 2-1
Liverpool            2-2 0-1 2-0 1-2 1-3 1-1 0-0 2-2     2-1 1-2 2-0 2-1 2-1 1-0 0-0 4-1 3-2 2-1 2-0
Manchester City      0-2 3-2 2-2 1-1 3-0 1-0 1-1 2-0 3-1     1-0 5-0 6-0 2-0 0-1 3-2 2-1 4-1 3-0 2-0
Manchester United    1-1 3-1 3-1 1-1 1-0 2-1 3-0 3-1 3-0 4-2     3-1 4-0 0-1 2-1 2-0 1-2 3-0 0-1 2-1
Newcastle United     1-2 1-0 3-3 2-1 3-3 3-2 2-2 1-0 1-0 0-2 0-1     1-0 1-2 1-1 0-1 2-3 1-3 1-1 2-0
Queens Park Rangers  1-2 2-0 2-0 0-1 0-0 1-2 0-1 3-2 2-3 2-2 0-2 2-1     0-1 2-2 1-0 1-1 1-2 3-2 0-0
Southampton          2-0 6-1 2-0 1-1 1-0 3-0 2-0 2-0 0-2 0-3 1-2 4-0 2-1     1-0 8-0 0-1 2-2 0-0 0-0
Stoke City           3-2 0-1 1-2 0-2 1-2 2-0 1-0 0-1 6-1 1-4 1-1 1-0 3-1 2-1     1-1 2-1 3-0 2-0 2-2
Sunderland           0-2 0-4 2-0 0-0 1-4 1-1 1-3 0-0 0-1 1-4 1-1 1-0 0-2 2-1 3-1     0-0 2-2 0-0 1-1
Swansea City         2-1 1-0 1-0 0-5 1-1 1-1 3-1 2-0 0-1 2-4 2-1 2-2 2-0 0-1 2-0 1-1     1-2 3-0 1-1
Tottenham Hotspur    2-1 0-1 2-1 5-3 0-0 2-1 2-0 4-3 0-3 0-1 0-0 1-2 4-0 1-0 1-2 2-1 3-2     0-1 2-2
West Bromwich Albion 0-1 1-0 4-0 3-0 2-2 0-2 1-0 2-3 0-0 1-3 2-2 0-2 1-4 1-0 1-0 2-2 2-0 0-3     1-2
West Ham United      1-2 0-0 1-0 0-1 1-3 1-2 3-0 2-0 3-1 2-1 1-1 1-0 2-0 1-3 1-1 1-0 3-1 0-1 1-1
```

Scrape data from:

```
https://en.wikipedia.org/wiki/2018-19_Premier_League#Result_table
```

A6. Home Results League Table

Use the results in the previous question to print out a league table for the home results of each team:

```
             Chelsea                   19 15  4  0 36  9 49
             Manchester City           19 14  3  2 44 14 45
             Manchester United         19 14  2  3 41 15 44
             Arsenal                   19 12  5  2 41 14 41
             Southampton               19 11  4  4 37 13 37
             Liverpool                 19 10  5  4 30 20 35
             Stoke City                19 10  3  6 32 22 33
             Tottenham Hotspur         19 10  3  6 31 24 33
             Swansea City              19  9  5  5 27 22 32
             West Ham United           19  9  4  6 25 18 31
             Everton                   19  7  7  5 27 21 28
             Leicester City            19  7  5  7 28 22 26
             Newcastle United          19  7  5  7 26 27 26
             West Bromwich Albion      19  7  4  8 24 26 25
             Queens Park Rangers       19  6  5  8 23 24 23
             Aston Villa               19  5  6  8 18 25 21
             Crystal Palace            19  6  3 10 21 27 21
             Hull City                 19  5  5  9 19 24 20
             Sunderland                19  4  8  7 16 27 20
             Burnley                   19  4  7  8 14 21 19
```

A7. Orbiting Particle

Write a program with a Particle class that plots the position of the particle as it orbits a central point under the influence of an inverse square force (F).

Define a the class along the lines:

```
class Particle:
    def __init__(self, name, x0, v0): ...
    def getPosition(self): ...
    def next(self, dt): ...
```

where x0 and v0 are the initial position and velocity of the particle. The ***next*** method calculates the new position (***x***) and velocity (***v***) of the particle a time interval ***dt*** later.

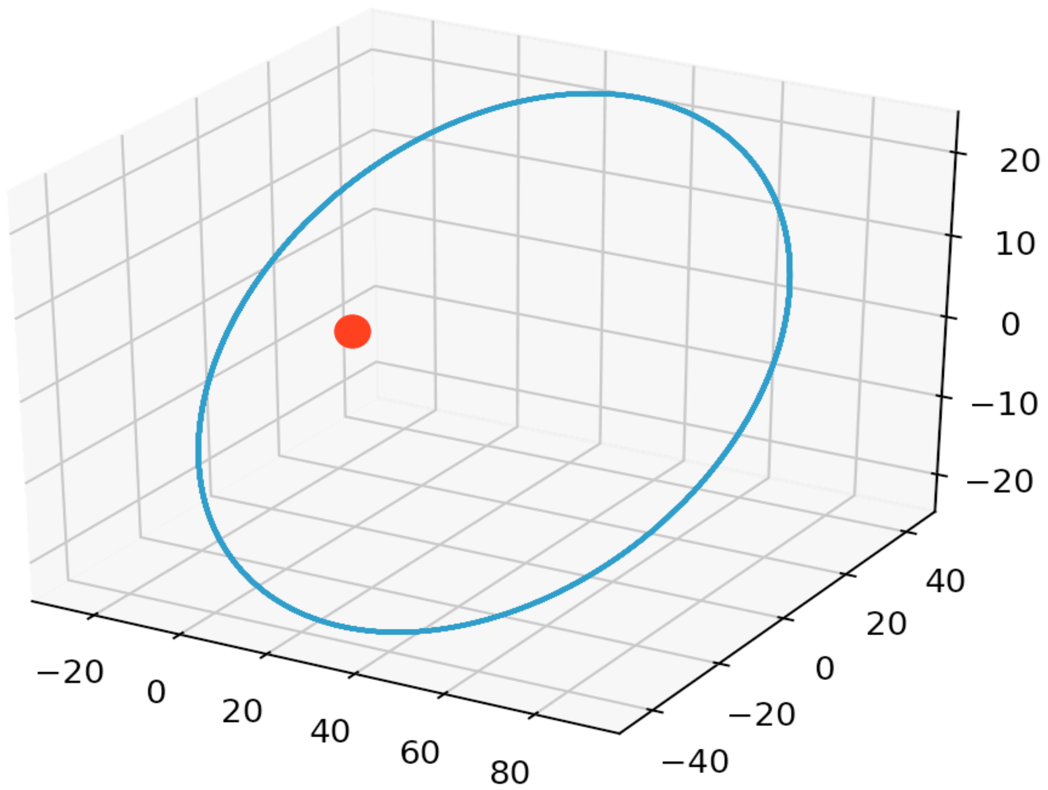
Call the next method repeatedly and use the formulae:

```
dv = F * dt / m
dx = v * dt
```

to calculate the new position (x) and velocity (v) of the particle:

```
v = v + dv
x = x + dx
```

Plot the resulting orbit in 3D with matplotlib.



A8. Simulation

Convert the previous example into a simulation using MatPlotLib.

A9. Image Processing

Write a Python program that generates a chessboard JPEG file. Use Numpy to create a 3D array representing the pixels of the chessboard. The array should alternate between black and white pixels and look like:

```
[[
  [  0   0   0]
  [  0   0   0]
  [  0   0   0]
  [  0   0   0]
  [  0   0   0]
  [  0   0   0]
  [  0   0   0]
  [  0   0   0]
  [  0   0   0]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]
  [255 255 255]
  [255 255 255]
  [255 255 255]
  [255 255 255]
  [255 255 255]
  [255 255 255]]
  ...
 [[  0   0   0]
  [  0   0   0]
  [  0   0   0]
  ...
```

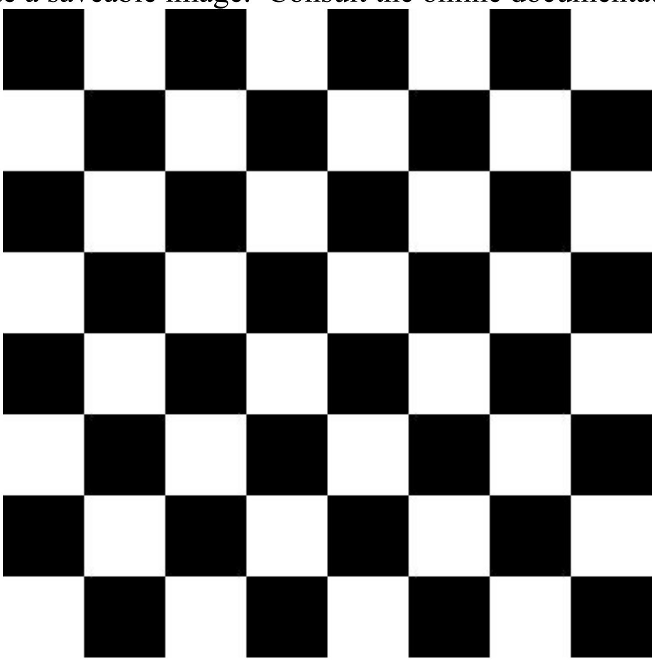
Once you have created the Numpy array, you can use:

```
    PIL.Image
```

to display the image and

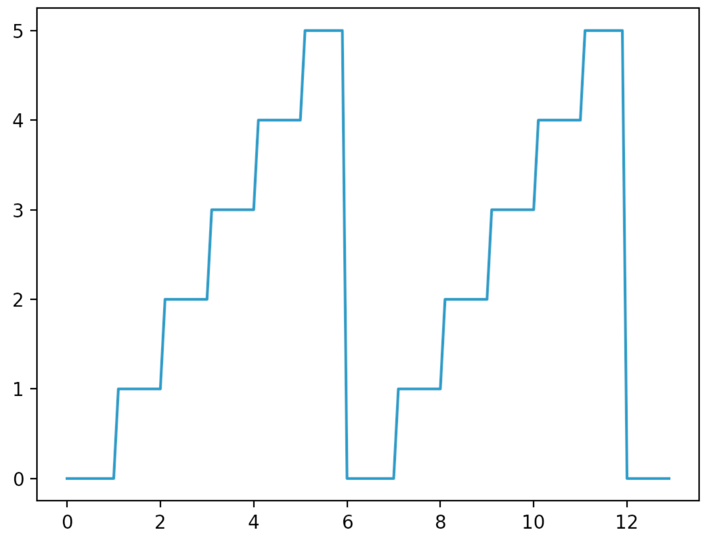
```
    PIL.Image.fromarray
```

to create a saveable image. Consult the online documentation for more details.



A10. Fourier Series

Define a stair function and use MatPlotLib to plot this function.



Now calculate the Fourier coefficients corresponding to this function, so that you can calculate the Fourier approximation function. Refer to:

http://www.thefouriertransform.com/series/coefficients.php

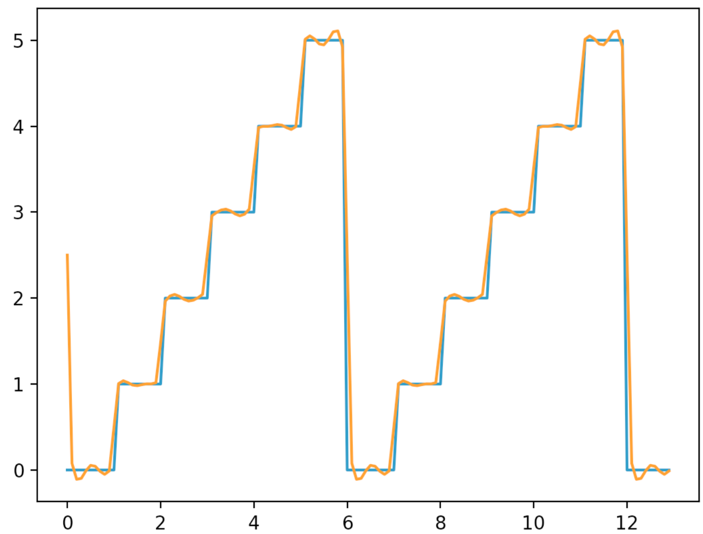
as a guide to the formule to be used. Your stair function will be f(t) and the Fourier approximation function will be g(t).

You will need to calculate the Fourier coefficients a[0], a[m] and b[m] where m in theory varies from 1 to infinity. In practice you get a good approximation by calculating coefficients between m=1 and 50. Use:

```
scipy.integrate.quad
```

to perform numerical integration required to calculate the coefficients (don't use analytical integration).

You will now be able to plot both the stair function and the Fourrier approximation:



A11. Gaussian Fitting

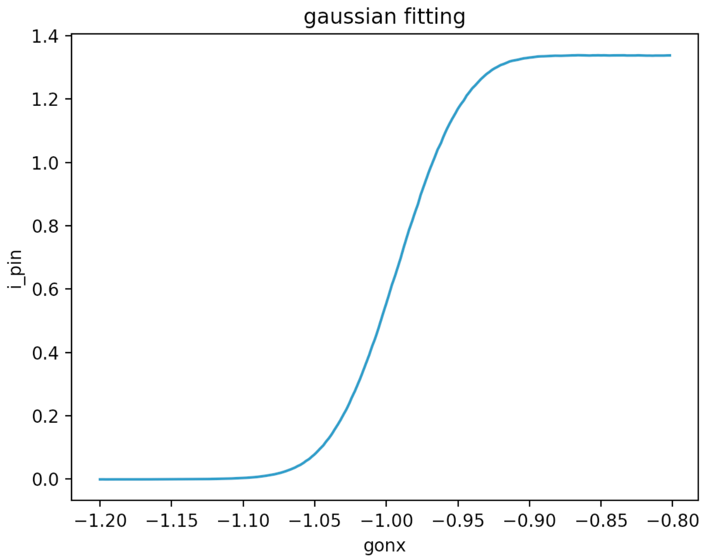
Use Pandas to open the file:

```
    data/14763.dat
```

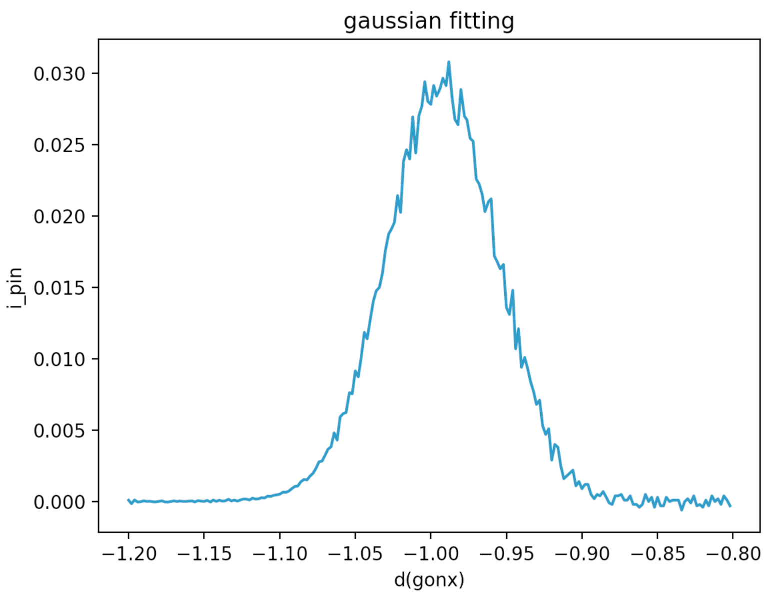
and then print the data and the keys from this file:

```
         gonx    i_pin   Time
0    -1.7000   1.3456    0.0
1    -1.6980   1.3456    0.0
2    -1.6959   1.3461    0.0
3    -1.6941   1.3460    0.0
4    -1.6920   1.3457    0.0
..       ...      ...    ...
```

Now plot this data using MatPlotLib:



As you can see this is a scan showing change in intensity of the beam. You need to compute and plot differences in ***gonx*** to obtain and plot a Gaussian:



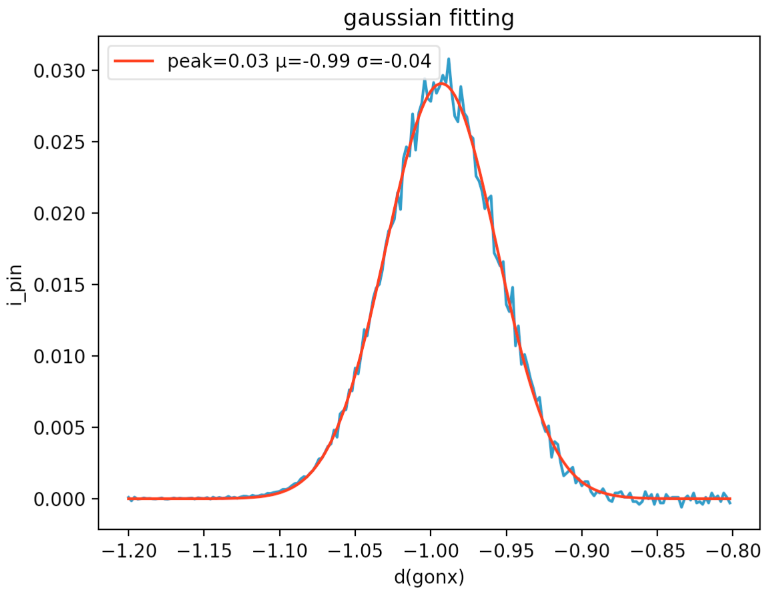
Now use

```
    scipy.optimize.curve_fit
```

to fit the Gaussian and calculate the peak, μ (center) and σ (standard deviation). Use the function:

```
    def gauss(x, peak, □, □):
        return peak*np.exp(-(x-□)**2/(2.*□**2))
```

as the function parameter to ***curve_fit***:



A12. Knights Random Chessboard Walk

Of all the chess pieces, the knight moves in the strangest way; each move is two squares in one direction and one square in an orthogonal direction. Thus if the knight is placed in the center of the board it has 8 possible moves.

Given an empty chess board and a single knight placed on the board, the knight can reach any part of the board by a suitable sequence of moves. If the knight makes a random set of moves, eventually it will visit every square on the board.

Perform a Monte Carlo simulation of a knight performing a random walk across the board. Represent the board by a Numpy array, initially filled with zeros:

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
```

Fill in squares with ones as the knight moves across the board. After a few moves the board will look like:

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0]
 [0 0 0 1 0 0 0 0]
 [0 0 0 0 1 0 1 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
```

Eventually the board will end up filled:

```
[[1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]]
```

The simulation should reveal that the knight will take a long time to visit each square on a random walk. How many moves will it take to more or less guarantee the knight visits every square of the board.

You might find it helpful to use a generator to calculate each new move.