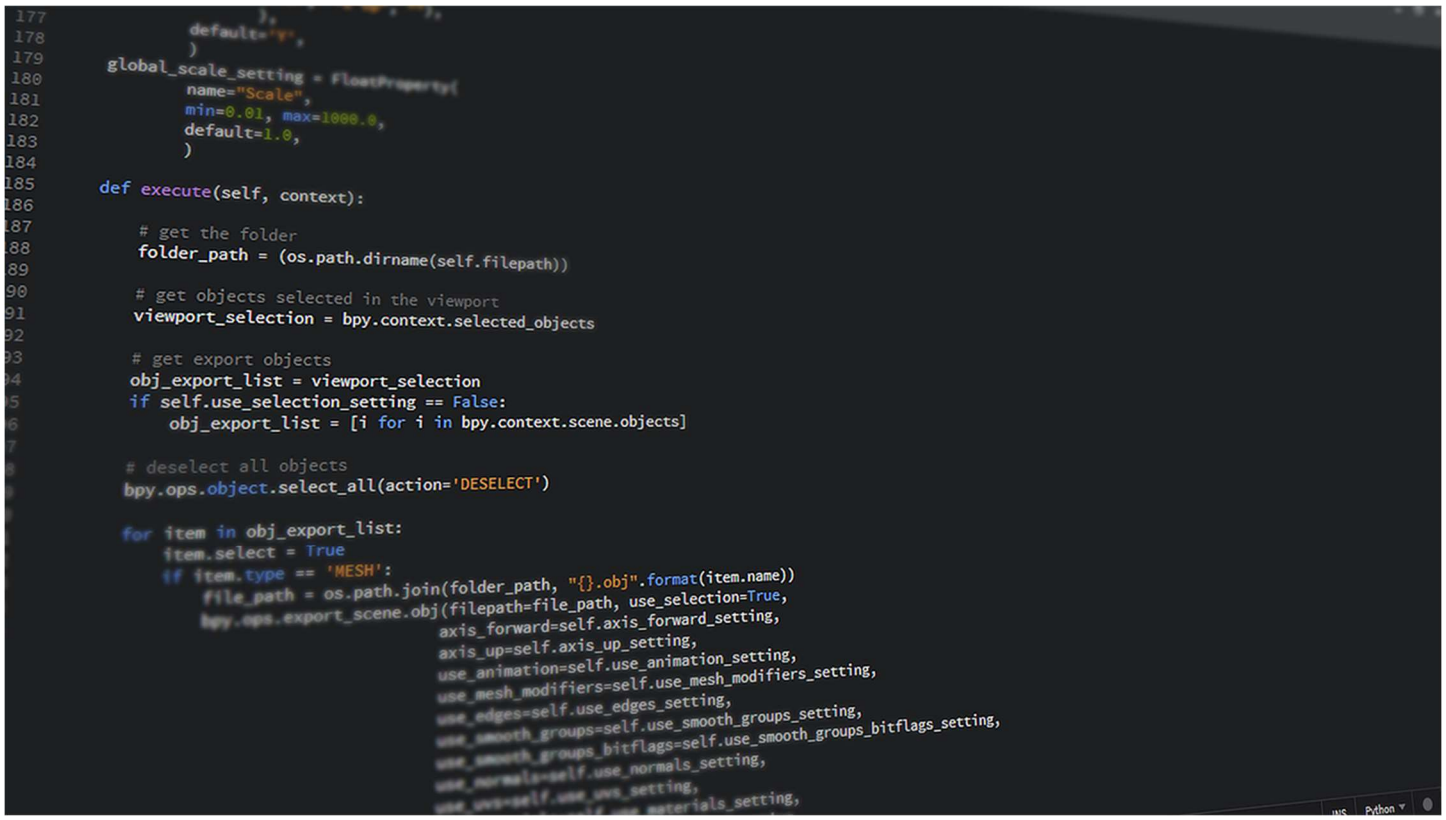
Software Development



Higher Computing Science

ROBERT GORDON'S COLLEGE

Contents

Software Needed:

Python Interpreter (IDLE) : Can be downloaded here

Pycharm: (Windows) and (Mac)

Additional Online Resources

- ❏ Python 3 Documentation (link)
- ❏ TutorialsPoint (link)
- ❏ Snakify (link)
- ❏ Codecademy (link)
- ❏ LearnPython (link)
- ❏ PythonProgramming.net (link)

Pre Knowledge

This course assumes that all of the relevant National 5 Content has been completed.

Exercise 1: Analysis

Learning Aims:

- ✔ To identify the purpose, scope, boundaries and functional requirements of a system
- ✔ To identify the features above in terms of Input, Process and Outputs

Scenario(s)

Required Reading: Analysis Presentation

For the following scenarios identify the likely :

- ❑ Purpose
- ❑ Scope
- ❑ Boundaries
- ❑ Functional Requirements:
  - ❑ Inputs
  - ❑ Processes
  - ❑ Output

Scenario 1

Raw music is a music agency which promotes concerts in various locations across Scotland. The program will have access to all of the Customer information (customer ref, forename, surname, email and tickets bought) as shown below. The information for all customers is currently stored in a text file. The ticket agency has decided to offer a 10% discount to all customers who buy more than two tickets for any concert. All customers eligible for this discount will be entered into a draw to win a backstage pass.

The ticket agency wants you to create a program to calculate the final discounted cost for each eligible customer and select the winner of the backstage pass. These discounted costs should be saved to a new file.

```
3001,Dmitri Petrov,dpetrov@hmail.com,4
```

Purpose	
Scope	
Boundaries	

Functional Requirements:	
Inputs	
Processes	
Outputs	

Scenario 2

The Scandinavian Diving Championship holds qualifying events one each in Sweden, Finland, Denmark, Norway and Iceland. Each country enters three divers to the qualifying events. The same 15 divers take part in all five events. The highest scoring diver from each country goes forward to enter the Scandinavian Diving Championship final.

The organisers require help to work out the finalists, and then how to work out the overall winner of the Championship. Each diver is given a score based on their position in each qualifying event as follows:

Diver position in the event	Score
1	30
2	25
3	20
4	15
5	10
>5	0

You have been asked to create a program that will calculate the total score for each diver, and then find the name and country of the top diver from each country. The details of all 15 contestants are stored in a file as shown below:

```
D1,Jensen A,Denmark
```

Your task is to write a program that will import the data for all of the contestants from the file.

In the final each finalist is given an individual score of between 0 and 10 from each of five judges. Each finalist's total score is created by adding all the five judges' scores together and removing the highest and lowest scores. Your program should find the 5 finalists (the winning person from each country) then calculate each of their total scores, and work out who is the champion.

Purpose	
Scope	
Boundaries	
Functional Requirements:	

Inputs	
Processes	
Outputs	

Scenario 3

Essell Academy is running its annual 'Choral shield' competition. Each of the four houses (Alpha, Beta, Delta and Gamma) enters a team of senior pupils. The show is performed on three evenings — Wednesday, Thursday and Friday.

The cost of a ticket on the Wednesday and Thursday night is £5 and the cost of a Friday night ticket is £10. Tickets can only be ordered for one night at a time. Anyone wishing to attend on more than one night will need to make multiple orders.

When buying their tickets, customers are asked in which of the three areas of the theatre they wish to sit — 1(front), 2(middle) or 3(back). There are 200 seats available in each area of the theatre. The money raised from the concerts will go to the top charity suggested by customers.The school sells tickets via the school website, in addition to them being on sale from the school office.

You have been asked to write a program to find which method of purchase has been the most popular (being the method with most tickets purchased as opposed to the number of transactions) and the total amount of money raised from all orders. Your will be supplied with a file that contains the data from 300 orders.

The orders file has the data in the following order:

Customer ID: this will take the form C001
Ticket ID: this will represent the night (W, T or F) and one of the three positions in the theatre - eg T2 is a Thursday night in the middle section of the theatre.

Number of tickets: an integer value between 1 and 10

Method of purchase: this will represent whether bought in school or via the website and will either be S or W

For example, the first record in the file might look like this:

- `C001,W1,5,S`

Using this file of data, your program should produce the following output (generating the year automatically using a pre-defined function):

```
Essell Academy Choral Shield <year>
The most popular method of sales is XXXXX
The total money raised for charity is £XXXXX
```

Your program should then :

- ❑ export **all data** for **each night (Wednesday, Thursday and Friday)** to separate files(one for each day).
- ❑ When being written to the file each order should be also assigned a unique order ID with the first character of the day and a **unique** 3 digit number e.g. T001

To help you test that your program calculates totals correctly you will be provided with three previous year's data (in 3 csv files.

Purpose	
Scope	
Boundaries	
Functional Requirements:	
Inputs	
Processes	
Outputs	

Exercise 2.1 : First Modular Program

Learning Aims:

- ✔ To create a modular program with a parameter passing
- ✔ To identify the data flow between subprograms

Required Reading: Parameter Passing Presentation

You are to create a program to ask for two numbers and add them together (showing data flow)

Top Level Design and Data Flow

Main Steps		In	Out
1	Get Input From User		number1,number2
2	Perform Calculation	number1,number2	answer
3	Display Output	answer	

Refinements

1.1 Get number1 FROM KEYBOARD

1.2 Get number2 FROM KEYBOARD

2.1 answer = number1 + number2

3.1 SEND “Answer is “ & ANSWER TO DISPLAY

Implementation

1. Create a new program, calling it **FirstProcedure**.
2. Now we will **define** and create our first function, at the top of the program line type in the following code:

```
#Function to ask the user to input two numbers
def GetInput():
    value1 = int(input("Please Enter Number 1: "))
    value2 = int(input("Please Enter Number 2: "))
    return value1,value2
```

3. Now we need to repeat this to define our function to calculate the answer and return it:

```
#Function to add two numbers together
def CalculateAnswer(firstvalue,secondvalue):
    answer = firstvalue + secondvalue
    return answer
```

4. Now using this technique write your final subroutine to display the answer. We will only refer to this as a subroutine as it doesn't return any values.

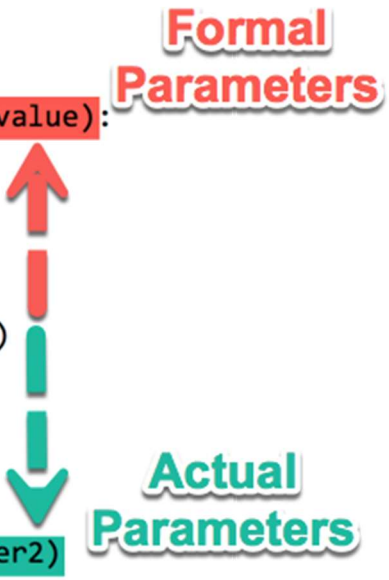
```
#Subroutine to display the answer
def DisplayAnswer(answer):
    print("Answer is :" + str(answer))
```

5. Now we must call the two subroutines/functions from our main program. You will notice that the DisplayAnswer subroutine doesn't return a value so we can refer to this as a subroutine

```
#Main Program

number1,number2 = GetInput()

answer = CalculateAnswer(number1,number2)

DisplayAnswer (answer)
```

6. Your program should now look like this:

```
#Function to ask the user to input two numbers
def GetInput():
    value1 = int(input("Please Enter Number 1: "))
    value2 = int(input("Please Enter Number 2: "))
    return value1,value2

#Function to add two numbers together
def CalculateAnswer(firstvalue,secondvalue):
    answer = firstvalue + secondvalue
    return answer

#Subroutine to display the answer
def DisplayAnswer(answer):
    print("Answer is :" + str(answer))

#Main Program

number1,number2 = GetInput()

answer = CalculateAnswer(number1,number2)

DisplayAnswer (answer)
```



Challenge Task 1

Now adapt your program saving it as **My First Procedure Version 2** so that the output from your program is similar to the following:

Enter your first number: **5**
Enter your second number: **7**

5 + 7 = 12

Amend the Data Flow Table which is given below:

Main Steps		In	Out
1	Get Input From User		number1,number2
2	Perform Calculation	number1,number2	answer
3	Display Output	answer	

Code Listing

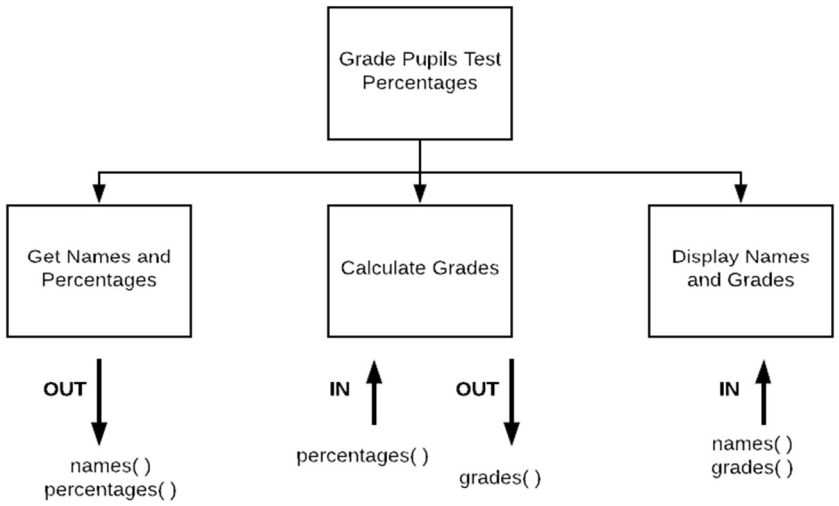
(copy and paste your code below)

Exercise 2.2 : Implementing a Modular Design

Learning Aims:

- ✔ To implement a given structure design

In this example you will be given a structure diagram and the top level design in pseudocode. The program should ask the user to input a list of 10 names and valid percentages and create a new list of grades. The names and grades will then be output to the user. 50% is a “Pass” and less than 50% is a “Fail”



Top Level Design and Data Flow

Main Steps		In	Out
1	Get Names and Valid Percentages		names(),percentages()
2	Calculate Grades	percentages()	grades()
3	Display Names and Grades	names(),grades()	

What to do now:

Complete the refinements for the main steps as required using either pseudocode or a structure diagram.

Implement and save your program as **2.2 Marks**

Code Listing

Test Yourself: Modular Design Tasks

Task 1 VAT Calculator

Create a program that will allow a user to enter a price (exc VAT) and then display the price inclusive of VAT (rounded to two decimal places). **At present VAT is 20%** This should be written in a modular manner and data flow identified. The top level design below should be refined as appropriate.

The output should be in the form:

```
Enter the price excluding VAT: £99.99
The price including VAT is £119.99
```

Top Level Design and Data Flow

Main Steps		In	Out
1	Get Price		
2	Calculate VAT		
3	Display price including and excluding VAT		

Code Listing

Task 2 Fahrenheit Converter

You are going to create a program that uses a function to take in the value in Celsius and convert it into Fahrenheit. The output will be formatted to two decimal places. The formula is: (celsius * 1.8) + 32. The top level design below should be refined as appropriate.

The output should be in the form:

Enter the temperature in Celsius: **100**
100 degrees centigrade is **212 fahrenheit**

Top Level Design and Data Flow

Main Steps		In	Out
1	Get celsius temperature		
2	Calculate fahrenheit		
3	Display both temperatures		

Code Listing

Task 3 Gas Bill Calculator

When you submit a gas meter reading it converts the units used to kilowatt hours (kWh). When receiving a bill your previous and current reading is used to calculate the number of imperial units and then this number is converted to kilowatt hours. Each kilowatt hour costs 3.74p. VAT is also charged on the total due at 5%.

Create a function that will allow you to pass in the previous reading, current reading , calculate the number of kilowatt hours used (see formula below) and return the amount due (rounded to two decimal places).

Previous Reading: **4805**
Current Reading: **5061**

You have used **256** units
This is **8,087.96** kilowatt hours
Your bill is: **£317.61**

The formula for it is:

256.00 imperial units used
x 1.022640 volume correction
x 2.83 to convert to metric
= 740.88 metric units
x 39.3 calorific value
÷ 3.6 to convert to kWh
= 8,087.96 kWh

Top Level Design and Data Flow

Main Steps		In	Out
1	Get Input Data		
2	Calculate Amount Due		
3	Display BIll Details		

Code Listing

Task 4 Month Converter

You are to write a program that will allow the user to enter the number of the month and display which a message stating whether it has 28(or 29), 30 or 31 days. Input should be validated so that only input between 1 and 12 (inclusive) is accepted. The top level design below should be refined as appropriate.

Top Level Design and Data Flow

Main Steps		In	Out
1	Enter Valid Number of Month		
2	Find number of days from array		
3	Display number of days		

A partial design for step 2 is shown below:

2.1 DECLARE days INITIALLY [31,28,31,30,31,30,31,31,30,31,30,31]
2.2 RETURN number of days in month

Code Listing

Extension

The user has now requested a modification to the program. They want the program to display the actual month as well.

For example:

Please enter the number of the month: **5**
Output: **May has 31 days.**

Code Listing

Exercise 2.3 : Prime Number Function

Learning Aims:

- ✔ To use the modulus function to check if a number is a prime number or not.

A number is a prime number if it is only divisible by itself and 1. Write a function that will use the modulus function to allow you to return TRUE if a number is a prime number or FALSE if it is not. An example of the modulus function is shown below:

Example:

`5 % 2 = 1` (as 5/2 = 2 remainder 1)

Ex 2.3 Code Listing

Exercise 2.4 : Converting Numbers

Learning Aims:

- ✔ To use the int function to validate if a number is an integer or real number.

We have used this before but the **`int()`** function will convert a value into an integer variable. There is no rounding performed the decimal portion of any real number is simply truncated.

See the example below:

```
>>> mynumber = 12.34
>>> mynewnumber = int(mynumber)
>>> print(mynewnumber)
12
```

Your task is to create a function to check if a number that has been entered is an integer or not.

Example:

Please enter a number to test: **5.0**
The number is an Integer

Please enter a number to test: **5.1**
The number is a real number.

Extension

Adapt your function to return a single parameter but be able to use this to state if a number is positive or negative and use this to implement a program similar to below:

Please enter a number to test: **5.0**
The number is a **positive Integer**

Please enter a number to test: **-5.1**
The number is a **negative real number**.

Ex 2.4 Code Listing

Exercise 3.1: Writing to a plain text file

Learning Aims:

- ✔ To write text to a plain text file

Required Reading: 4 File Handling and Structures Presentation

Write a program that will ask the user to enter a message and then store it in a new text file.

Top Level Design and Data Flow

Main Steps		In	Out
1	Enter Message		
2	Write message to file	filecontents	

1. Create a function called **WriteFile** that will have one parameter called filecontents. This is the text that will be written to the file

```
def WriteFile(filecontents):

    with open("newfile.txt") as writefile:
        writefile.write(filecontents)
        print("File updated...")
```

2. We will now ask the user to enter their message and pass it to the WriteFile function.

```
message = input("Please enter your message ")

WriteFile(message)
```

3. Run the program to test.

You will see that you get an error message.

```
FileNotFoundError: [Errno 2] No such file or directory: 'newfile.txt'
```

This is because the file does not exist and we haven't given Python the instructions to make it.

4. So we will alter the mode of the file to w - modify your writefile subroutine as shown below and retest the program.

```
def WriteFile(filecontents):

    with open("newfile.txt","w") as writefile:
        writefile.write(filecontents)
        print("File updated...")
```

5. You will notice that the file is now created in the same folder where you Python program was saved.
6. If you run the program twice you will notice that the contents of the file are being overwritten.
7. We will modify the program so that the contents of the file will be appended (written onto the end of the file) and each message will be on a new line.
8. Save a new version of your file as (**WriteFile2**)
9. Modify your subroutine to the following:

```
def WriteFile(filecontents):

    with open("newfile.txt","a") as writefile:
        writefile.write(filecontents + "\n")
        print("File updated...")
```

Changes the file mode to append

Adds a new line character

10. Now run and test your program.

Exercise 3.2: Reading from a csv file

Learning Aims:

- ✔ To read a CSV file and store it within two **parallel arrays**

Your task is to read the contents of a file which contains details of 20 pupil names and their guidance house.
The structure of the file is as follows:

```
Nichola,Blackfriars
James,Collyhill
Nichola,Blackfriars
James,Collyhill
```

You will read the file and import the contents of the existing file (PupilDetails.csv) into two parallel arrays, one storing the pupil names and one for guidance houses.

The file has already been made for you. You will then output the file in the format below:

Pupil Name:	**Guidance House**
Nichola	Blackfriars
James	Collyhill

Top Level Design and Data Flow

Main Steps		In	Out
1	ReadFile		names(), houses()
2	DisplayContents	names(), houses()	

Refinements

1.1 OPEN FILE PupilDetails.csv for READ ACCESS
1.2 WHILE file.txt IS NOT BLANK
1.3 READ next line of file
1.4 SET newarray = split line on “,”
1.5 SET names(counter) = newarray(0)
1.6 SET houses(counter) = newarray(1)
1.7 READ next line of file
1.8 END WHILE
1.9 CLOSE FILE

1. Create the subroutine definitions from the data flow given above.
2. Implement the following function which will read the file and split it into an array by using the comma as a separator. As it will be used many different times it is heavily commented to explain the function of each line.

```
def ReadPupilDetails():
    namesarray = [None] * 20
    housesarray = [None] * 20
    counter = 0

    with open("PupilDetails.csv") as readfile:
        #read the first line of the file
        line = readfile.readline().rstrip('\n')
        #While this line of code contains any characters (file isnt empty)
        while line:
            #split the line on the comma (and store in an array)
            #each part of the array contains the parts between the commas
            items = line.split(",")
            #First part contains the pupils name
            namesarray[counter] = (items[0])
            #Second part contains the pupils house
            housesarray[counter] = (items[1])
            #read the next line of the file and continue if contains text
            line = readfile.readline().rstrip('\n')
            counter += 1
    input("File read...Press any key to continue")
    return namesarray,housesarray
```

Now we will have to iterate through these arrays to display the content.

3. The **DisplayContents** function is shown below:

```
def DisplayContents(names,houses):

    for counter in range(len(names)):
        print(names[counter],houses[counter])
```

4. When you run this program you will notice that the output isn't formatted correctly. One way to do this is to format the string to be a fixed length (add some padding to the right hand side. Alter your display subroutine to the following:

```
def DisplayContents(names,houses):
    print("{0:<10s}".format("Name"),"{0:<10s}".format("House"))
    for counter in range(0,len(names)):
        print("{0:<10}".format(names[counter]),"{0:<10}".format(houses[counter]))
```

Now run and retest your program.

Exercise 3.3: Creating a CSV file

Learning Aims:

- ✔ To create a modular program with a parameter passing

A teacher is writing a program to automate some SQA processes for their school. They want a program that will allow them to enter Forenames, Surnames and Scottish Candidate Numbers (SCN's). The program should then export the data into a Comma Separated File which could be used to import into a database.

Top Level Design and Data Flow

Main Steps		In	Out
1	InputDetails		forenames(), surnames(), SCN()
2	ExportFile	forenames(), surnames(), SCN()	

Refinements

1.1 GET numpupils FROM USER
1.2 DECLARE forenames[] AS ARRAY OF STRING * numpupils
1.3 DECLARE surnames[] AS ARRAY OF STRING *numpupils
1.4 DECLARE SCN[] as ARRAY OF STRING *numpupils

1. Create the subroutine/function definitions from the data flow given above.
2. For the InputData function to ask the user how many pupils' details are to be entered. This will then be used to initialise the arrays as the appropriate length.

```
def inputdetails():

    pupils = int(input("Please enter the number of pupils: "))

    forenames = [None] * pupils
    surnames = [None] * pupils
    SCN = [None] * pupils
```

3. We will then ask the user to enter the names and candidate numbers into three parallel arrays:

```
for i in range(pupils):
    forenames[i] = input("Please enter the forename for pupil " + str(i+1) + " : ")
    surnames[i] = input("Please enter the surname for pupil " + str(i+1) + " : ")
    SCN[i] = input("Please enter the SCN for pupil " + str(i+1) + " : ")
```

4. The function will then return the 3 arrays

```
return forenames,surnames,SCN
```

5. The subroutine to export the file will use the 3 arrays and write each record to a file named **SCN.csv**, with each field separated by a comma. The "\n" character being concatenated onto the end of the line string is a new line character that will force each record to be on a new line

```
def writefile(forenames,surnames,SCN):
    with open("SCN.csv","w") as writefile:
        for i in range(len(forenames)):
            line = forenames[i] + "," + surnames[i] + "," + SCN[i] + "\n"
            writefile.write(line)

    print("File written...")
```

6. Now add the code to call your subroutine and function and test that your program works as intended and that the file is created correctly.

What to do now

Create a new function to read and display the file contents in columns as in the previous program.

Ex 3.3 Code Listing

Exercise 3.4: Reading a file (single line)

Learning Aims:

- ✔ To parse and process a single line CSV file.

So far all of your files have been comma separated or plain text on seperate lines. Sometimes you may have a single file such as below where all of the text is on a single line

The file below for example has a list of Ticket Sales (sales.csv) showing the name, date of purchase, number of tickets and the price for 100 people. We will be writing a program which will read this file and process it into parallel arrays to display the information for each purchase.

```
Joe Bloggs,14/3/18,5,150.00,Anna Smith,14/3/18,3,180.00
```

Top Level Design and Data Flow

Main Steps		In	Out
1	ReadFile		names(), dates(), tickets(),price()
2	DisplayContents	names(), dates(), tickets(),price()	

Refinements (partial)

1. ReadFile(names(), dates(), tickets(),price())
   - 1.1. INITIALISE names, dates, tickets, prices AS ARRAYS [] * 100
   - 1.2. OPEN filename FOR READ ACCESS
   - 1.3. SET line = file.readline()
   - 1.4. SET items = line.split(“,”)
   - 1.5. FOR purchasecounter = 0 TO LEN(items),STEP 4
   - 1.6. namesarray[counter] = items[purchasecounter]
   - 1.7. datesarray[counter] = items[purchasecounter+1]
   - 1.8. ticketsarray[counter] = items[purchasecounter+2]
   - 1.9. pricearray[counter] = items[purchasecounter+3]
   - 1.10. SET counter = counter+1
   - 1.11. RETURN namesarray,datesarray,ticketsarray,pricearray

What to do now

1. Create the function to read the file as shown below:

```
def ReadFile(filename):
    namesarray = [None] * 100
    datesarray = [None] * 100
    ticketsarray = [None] * 100
    pricearray = [None] * 100
    counter = 0

    with open(filename) as readfile:
        #Read the (only) line in the file
        line = readfile.readline().rstrip('\n')
        #Split on the comma
        items = line.split(",")
        #Set a loop that will increase in steps of 4
        for purchasecounter in range(0,len(items),4):
            #The counter variable is used to ensure that
            #each element in the parallel arrays is assigned
            #correctly, the purchase counter is used to step
            #through the items array
            namesarray[counter] = items[purchasecounter]
            datesarray[counter] = items[purchasecounter+1]
            ticketsarray[counter] = items[purchasecounter+2]
            pricearray[counter] = items[purchasecounter+3]
            counter += 1

    #File now closed

    return namesarray,datesarray,ticketsarray,pricearray
```

2. Now create a function to display the contents of the 4 arrays using previous examples as a reference.
3. Call and test your functions

Ex 3.4 Code Listing

Exercise 3.5: Test Yourself Parking Times

An automated car park records the Registration Number, Entry Time and Departure time of cars that use the park. The files are limited to 100 records per file.

The prices for parking are as follows:

- ❏ Up to 1 Hour: £1.00
- ❏ Up to 2 Hours: £2.20
- ❏ 2 to 3 Hours: £3.30
- ❏ 3 to 4 Hours: £4.40
- ❏ Up to 12 hours £12.00
- ❏ Over 12 hours £20.00

Times have been converted into a decimal format . For example 18:30 is stored as 18.5.

Write a modular program that will read the file named **times.csv** (in the format below)

```
N843 HYW,18/9/18,09.0,11.5
```

The program should then write the Car Registration Number with the entry time, exit time and the cost of parking to a new file.

Complete the Data flow and any refinements required. These can be shown as pseudocode or a structure diagram **with data flow**.

Top Level Design and Data Flow

Main Steps		In	Out
1			
2			
3			

Refinements (as appropriate)

Extension

It is possible that there are times when a car enters the car park late at night they leave the next day as in the example below:

```
J365 CVH,25/03/2018,23.08,8.25
```

Using the new file called **times2.csv** modify your program to process these times. **Assume that all times are valid**.

Ex 3.5 Code Listing

Exercise 4.1: Linear Search

Learning Aims:

- ✔ To implement a linear search algorithm

Required Reading: 5 Standard Algorithms Presentation

You are to design and create a program that will allow the stock file for a local garden centre to be imported and searched (see example file below)

```
Yosemite Onion, 18.27
```

The file is ordered in the order items appear in aisles. So an item at position 0 is located in Aisle 1.

Your program will read the stock file of 200 items (maximum) and then prompt the user to search for an item name (non case specific). It will then return the aisle number that the item is in.

Top Level Design and Data Flow

Main Steps		In	Out
1	ReadFile		items(), prices()
2	SearchFile	searchitem,items()	

Refinements (partial)

2.1 PROCEDURE SearchFile (searchitem,items(),prices())
2.2 FOR counter = 0 TO LEN(items())
2.3 IF items[x] = searchitem THEN
2.4 SEND "Match found at aisle " & (counter + 1) TO DISPLAY
2.5 END IF
2.6 NEXT counter
2.7 END PROCEDURE

What to do now

1. Create a subroutine to read the file PlantStockFile.csv and store in two parallel arrays called items and prices. The arrays should be initialised to 200 elements.
2. In your SearchFile subroutine add the following code to search through an array and display the position found:

```
def SearchFile(names):
    searchitem = input("Please enter the item to be searched for: ")

    #Loop through the entire array
    for x in range(len(names)):
        #If item is found
        if names[x] == searchitem:
            print("item found in aisle " + str(x+1))
```

3. Now call the readfile function and search subroutine

```
items,prices = ReadFile("PlantStockFile.csv")
SearchFile(items)
```

4. Run and test your program.

Ex 4.1 Code Listing

Exercise 4.2: Linear Search + Found Flag

Learning Aims:

- ✔ To create a linear search algorithm that will stop when an item is found.

The previous program wasn't very efficient in that it continued to search even if an item had been found. We will adapt your previous program to use a conditional loop that will continue to iterate until an item is found or there are no other items in the list.

Refinements (partial)

2. SearchFile
    - 2.1. SET counter = 0
    - 2.2. SET found = False
    - 2.3. WHILE counter < len(names) AND found = FALSE
    - 2.4. IF names[counter] = searchitem THEN
    - 2.5. SET found = True
    - 2.6. SET matchindex = counter
    - 2.7. END IF
    - 2.8. SET counter = counter + 1

What to do now

1. Create a copy of your previous program.
2. Modify your **SearchFile** subroutine to the following. We will use the Boolean value to signify if a match has been found. You will notice that the conditional loop has been modified to only continue whilst there are more items in the array and a match hasn't been found.

```
def SearchFile(names):
    counter = 0
    found = False
    searchitem = input("Please enter the item to be searched for: ")

    #Loop through the entire array
    while(counter <len(names) and found == False):
        #If item is found
        if names[counter] == searchitem:
            found = True
            foundposition = x
        counter += 1

    print("item found in aisle " + str(foundposition+1))
```

What to do now

- ❑ You will notice that there is no output if the item is found, adapt your program so that a suitable message is displayed if the item has **not** been found.
- ❑ Try searching for "yosemite onion" you will find it shows as False. This is due to "ABC" being compared to "abc" which fails.
- ❑ Use the **`.upper()`** string method as shown below to that it will find item regardless of the case.

For example

```
string1 = "mr Hay"
string2 = "Mr Hay"
if string1.upper() = string2.upper():
      print ("names match")
```

Ex 4.2 Code Listing

Exercise 4.3: Find Max

Learning Aims:

- ✔ To implement a find maximum algorithm

Required Reading: 5 Standard Algorithms Presentation

Write a program which displays an existing list of scores in the file called scores.csv which are read from a file The program should then display a message showing the highest score in the list.

The scores are integers between 0 and 9999.

```
Jessalyn Keeri,3186
```

Top Level Design and Data Flow

Main Steps		In	Out
1	ReadFile		names(),scores()
2	FindMax	names(),scores()	

Refinements (partial)

2.1 SET max = scores[0]
2.2 FOR EACH score in scores()
2.3 IF score(counter) > max THEN
2.4 SET max = scores(counter)
2.5 END IF
2.6 NEXT item

What to do now

1. Create a subroutine to read the file Scores.csv and store in two parallel arrays called **names** and **scores**. The arrays should be initialised to **50** elements.
2. When reading from the file ensure that you cast the score to an integer value so that when using it later in the program it is not treated as a string:

```
with open("scores.csv") as readfile:
        line = readfile.readline().rstrip('\n')

        while line:
            items = line.split(",")
            names[counter] = items[0]
            scores[counter] = int(items[1])
            counter += 1
            line = readfile.readline().rstrip('\n')
```

3. In your **FindMax** subroutine add the following code to search through an array and display the maximum score found:

```
def MaxScore(scores):
    maxscore = scores[0]
    for counter in range(0,len(scores)):
        if scores[counter] > maxscore:
            maxscore = scores[counter]
    print ("Highest score is",maxscore)
```

4. Now call the **MaxScore** function and then run and test your program.

What to do now

Use this function as a basis to find the **minimum** score.

Ex 4.3 Code Listing (copy and paste your code below)

Exercise 4.4: Pupil Marks

Learning Aims:

- ✔ To implement a find min and max algorithm
- ✔ To implement a counting occurrences algorithm

Required Reading: 5 Standard Algorithms Presentation

A school keeps a track of the prelim marks and the courseworks for their 5 classes of 20 Higher Computing Students.

The prelim is out of 110 and the coursework is out of 50.

A sample of the file is shown below showing the name, form class, prelim mark and coursework mark:

```
Paul Kincade,5B1,8,3
```

The teachers want to find each pupils total percentage(exam +coursework) /160
The minimum and maximum mark obtained by the class for each of the 2 parts of the course.

Top Level Design and Data Flow (incomplete)

Main Steps		In	Out
1	ReadFile		names(), forms(),prelims(),cworks()
2	FindMin	prelims()	
3	FindMax		

What to do now

1. Create a subroutine to read the file ComputingMarks.csv and store in **four parallel arrays** called names,forms,prelims and courseworks. The arrays should be suitably initialised.

2. In your **FindMin** subroutine add the following code to search through an array and return the minimum mark

```
def FindMin(marks):
    minmark = marks[0]
    for counter in range(len(marks)):
        if marks[counter] < minmark:
            minmark = marks[counter]

    return minmark
```

3. Now call the FindMin function and test it

```
minmark = FindMin(prelims)

print("the mininum prelim mark was :",minmark)
```

You should have successfully found the lowest mark. (Use Excel to quickly verify this). But it would be more beneficial if we store the position as we can then then use this to display all of the other details in the parallel arrays.

4. Modify your subroutine to the following:

FROM	TO

def FindMin(marks):
  minmark = marks[0]
  for counter in range(len(marks)):
  if marks[counter] < minmark:
  minmark = marks[counter]
 	
  return minmark
	def FindMin(marks):
	  minposition = 0
	  for counter in range(len(marks)):
	  if marks[counter] < marks[minposition]:
	  minposition = counter
	
	  return minposition

5. This means that we can use the value held in the **minposition** variable to then display all the other details as below:

```
minindex = FindMin(prelims)

print("the mininum prelim mark was :",prelims[minindex],"from",names[minindex])
```

6. Run and test your program.

What to do now

Now adapt your program to perform the following:

- Display the name and form class of the person with the biggest coursework mark
- Display the name and form class of the pupil with the highest combined mark

Challenge:

The display function should display **all** of the details requested.

Ex 4.4 Code Listing

Exercise 4.5: School Trip List

Learning Aims:

- ✔ To read from a file and implement a counting occurrences algorithm

A school is organising a projects week and a list has been collated of the **100** names, genders of the pupils and the trip they have chosen.

You have to write a program to read in the contents of the file and count the amount on each trip. The school trips on offer are: **London,Paris,BMX,Mountain Biking and Gorge Walking.**

A sample of the file (schooltrips.csv) is shown below:

```
Tarah Gaudon,F,Paris
Dawn Antonchik,F,Gorge Walking
```

The output from your program should be the total amount of pupils on each trip should be displayed. A file should then be created for **each** trip leader containing the **number** of pupils on that respective trip.

Top Level Design and Data Flow (incomplete)

Main Steps		In	Out
1	Readfile()		names(),genders(),trips()
2	Countpupils	trips()	
3	WriteFile		

This is a partial counting occurences for one trip only, you will have to adapt/reuse this in order to deal with the other 4 trips.

Refinements (partial)

2. CountPupils
   - 2.1. SET londoncounter = 0
   - 2.2. FOR counter = 0 to length(trips)
   - 2.3. IF trips(counter) = “London”
   - 2.4. SET londoncounter = londoncounter +1
   - 2.5. END IF
   - 2.6. NEXT loop

Ex 4.5Code Listing (copy and paste your code below)

Exercise 4.6: 5k Running Times

A local running club is running a 5k run competition. Over 2 days the runners run two 5k races. The runners times are recorded in a file below. Their name, category (Junior, Senior or Elite) and bib number along with their times are stored. A sample of the file runnertimes.csv is shown below:

```
Lyven Natale,Elite,512,12:07:49,12:07:14
```

Your task is to read and process the file. The data should be stored in suitable data structures and written in a modular manner.

The program must offer the following functions:

- ❏ Allow the user to enter a bib number **or** name (non case specific) to display the runners details (with an appropriate message if not found)
- ❏ Display the quickest and slowest times for the Elite category on both days.
- ❏ Display the total amount of all Junior or Senior contestants with a qualifying time of 22min (0:22:00) for **either** race and also save these details to a new file. When writing the file you can simplify the algorithm by opening a file before traversing an array and then closing it at the end. This will mean that you do not have to repeatedly open and close files

HINT: You can use < and > on strings just as with numbers.

Top Level Design and Data Flow (incomplete)

Main Steps		In	Out
1			
2			
3			

Refinements (as required)

Ex 4.6 Code Listing

Exercise 5.1 : ISBN Version 1 (Parallel Arrays)

Required Reading: 4.1 Data Structures Structure Presentation

You are to write a program that will allow the user to read the contents of a file - books.txt which will contain the Title, Author, ISBN and Price of **20 books**. A sample of the file is below:

When the file has been read the program will allow the user to search the contents of the file on the ISBN field and display the book details if a book has been found.

Top Level Design and Data Flow

Main Steps		In	Out
1	ReadFile	filename	titles(), authors(), ISBN(), prices()
2	SearchBookDetails	titles(), authors(), ISBN(), prices()	

Refinements (partial)

2. ReadFile
    - 2.1. SET counter = 0
    - 2.2. INITIALISE ARRAYS
    - 2.3. OPEN FILE (filename) FOR READING
    - 2.4. line = READ line from FILE
    - 2.5. WHILE NOT EOF
    - 2.6. SET items = line.split(“,”)
    - 2.7. titles[counter] = items[0]
    - 2.8. authors[counter] = items[1]
    - 2.9. ISBN[counter] = items[2]
    - 2.10. prices[counter] = items[3]
    - 2.11. line = READ line from FILE
    - 2.12. INCREMENT COUNTER
    - 2.13. END WHILE
    - 2.14. RETURN titles,authors,ISBN,prices

1. Declare the functions using the data flow given earlier.
2. Create the function to read and parse the file as below:

```
def ReadBooks(filename):
    titles = [None] * 20
    authors = [None] * 20
    ISBN = [None] * 20
    prices = [None] * 20
    counter = 0

    with open(filename) as readfile:
        line = readfile.readline().rstrip('\n')
        while line:
            items = line.split(",")
            titles[counter] = items[0]
            authors[counter] = items[1]
            ISBN[counter] = items[2]
            prices[counter] = items[3]

            line = readfile.readline().rstrip('\n')

            counter += 1
    input("File read...Press any key to continue")
    return titles,authors,ISBN,prices
```

3. Now we will pass these parallel arrays to another subroutine to perform a Linear Search on the ISBN field.

```
def SearchBooks(titles, authors, ISBN, prices):
    found = False
    choice = input("Please enter the ISBN to search for: ")
    for counter in range(0,len(ISBN)):
        if ISBN[counter] == choice:
            found = True
            position = counter
    if found == False:
        print("No book found in the list")
    else:
        print("Book found:\n")
        print("Title: ",titles[position])
        print("Author: ",authors[position])
        print("Price :",prices[position])
```

4. Now call the functions/subroutines using the code below and test the program:

```
filename = "books.txt"

titles,authors,ISBN,prices=ReadBooks(filename)

SearchBooks(titles,authors,ISBN,prices)
```

Exercise 5.2 : ISBN (Records)

Learning Aims:

- ✔ To declare and create an array of records to use in order to simplify parameter passing

Required Reading: 4.1 Data Structures Structure Presentation

We will now adapt your previous program so that we declare a class to mimic records to hold our data about each book. We will then create an array of 20 books. The data flow below has been adapted - you should notice there are less parameters being passed.

Top Level Design and Data Flow

Main Steps		In	Out
1	ReadFile	filename	books()
2	SearchBookDetails	books()	

Refinements (partial)

```
RECORD Book:
{
        Title AS String
        Authors AS String
        ISBN AS String
        Price AS REAL
}
```

1. ReadFile
    1.1. SET counter = 0
    1.2. INITIALISE books as BOOK
    1.3. OPEN FILE (filename) FOR READING
    1.4. line = READ line from FILE
    1.5. WHILE NOT EOF
    1.6. SET items = line.split(“,”)
    1.7. books[counter].title = items[0]
    1.8. books[counter].authors[counter] = items[1]
    1.9. books[counter].ISBN[counter] = items[2]
    1.10. books[counter].price[counter] = items[3]
    1.11. line = READ line from FILE

1.12. INCREMENT COUNTER
1.13. RETURN books

1. Create a new class definition at the top of your program as below (we will use this to mimic records as Python's dictionary syntax isn't very intuitive.)

```
class Book():
    def __init__(self):
        self.Title = ""
        self.Author = ""
        self.ISBN = ""
        self.Price = 0.0
```

2. Change your declarations inside your **ReadBooks** subroutine as shown below. This will create an array of **20 Book** records:

Before	After

`titles = [None] * 20`
`authors = [None] * 20`
`ISBN = [None] * 20`
`prices = [None] * 20`
`counter = 0`
	`Books = [Book() for x in range(20)]`
	`counter = 0`

3. Now change the lines where you assign the items to the parallel arrays so that we can use our record type structure using the following code.

Before	After

`while line:`
`  items = line.split(",")`
`  titles[counter] = items[0]`
`  authors[counter] = items[1]`
`  ISBN[counter] = items[2]`
`  prices[counter] = items[3]`
 	
`  line = readfile.readline().rstrip('\n')`
 	
`  counter += 1`
	`while line:`
	`  items = line.split(",")`
	`  Books[counter].Title = items[0]`
	`  Books[counter].Author = items[1]`
	`  Books[counter].ISBN = items[2]`
	`  Books[counter].Price = items[3]`
	
	`  line = readfile.readline().rstrip('\n')`
	
	`  counter += 1`

The new code will create a new record and then assign the values to the record fields using **.Fieldname**	

4. We will also need to alter the return line and call lines

Return Lines
FROM
`return titles,authors,ISBN,prices`
TO
`return Books`

Call Lines
FROM
`titles,authors,ISBN,prices=ReadBooks(filename)`
TO
`books = ReadBooks(filename)`

5. Now we will need to modify the display subroutine to use the new array of records.

From

```
def SearchBooks(titles, authors, ISBN, prices):
    found = False
    choice = input("Please enter the ISBN to search for: ")
    for counter in range(0,len(ISBN)):
        if ISBN[counter] == choice:
            found = True
            position = counter
    if found == False:
        print("No book found in the list")
    else:
        print("Book found:\n")
        print("Title: ",titles[position])
        print("Author: ",authors[position])
        print("Price :",prices[position])
```

To

```
def SearchBooks(books):
    found = False

    choice = input("Please enter the ISBN to search for: ")

    for counter in range(0, len(books)):
        if books[counter].ISBN == choice:
            found = True
            position = counter

    if found == False:
        print("No book found in list")

    else:
        print("Book found:\n")
        print("Title: ", books[position].Title)
        print("Author: ", books[position].Author)
        print("Price: ", books[position].Price)
```

You will notice that we have changed the parameters being passed in and now only have to pass in a single array of books.

What to do now:

Modify your Exercise 4.5 Program to use an array of records as opposed to the parallel arrays.

Modified Ex 4.5 Code Listing (copy and paste your code below)

Exercise 5.3: Baking Competition

A group of local schools took part in a competition to find the best amateur baker.

There are **20** contestants **each** from **5** different Schools

- ❏ AiryBrae
- ❏ Inverlamond Academy
- ❏ Linlathen
- ❏ St Saviours
- ❏ Aberdeen Academy.

AiryBrae and Linlathen are Primary Schools and the rest are Secondary schools. Each contestant will have a maximum score of 30). They were recorded as below

A sample of the file is shown below:

```
Hale Cowton,1,Aberdeen Academy,Secondary
```

Your task is to open and read the file called Baking.csv. **The data must be stored in an appropriate data structure.** The program must:

- ❏ Display the details of the best primary and secondary pupil.
- ❏ Display the School with the Highest Average Score
- ❏ Write the pupil name and school name of the best primary and best secondary pupil to a new text file

Your output should be similar to below:

```
Best Primary Pupil: Ado Riach,Linlathen
Best Secondary Pupil: Bentlee Forty, St Saviours
Highest Average Score:  18.8 from Inverlamond Academy
```

Top Level Design and Data Flow (incomplete)

Main Steps		In	Out
1	ReadFile		primary(), secondary()
2			
3			

Refinements (as required)

1. When reading the file it would be useful if you could read them into an array that held either primary or secondary pupils at the beginning of the process.

   Declare the following record structures to use when reading the file.

```
class secondary():
    def __init__(self):
        self.name = ""
        self.score = ""
        self.school = ""
        self.type = "Secondary"

class primary():
    def __init__(self):
        self.name = ""
        self.score = ""
        self.school = ""
        self.type = "Primary"
```

2. Use the following code for your **ReadFile** function:

```
def ReadFile():

    primarypupils = [primary() for x in range(40)]
    primarycounter = 0

    with open("baking.csv") as readfile:

        line = readfile.readline().rstrip('\n')
        while line:
            items = line.split(",")
            if items[3] == "Primary":
                primarypupils[primarycounter].name   = items[0]
                primarypupils[primarycounter].score  = int(items[1])
                primarypupils[primarycounter].school  = items[2]
                primarypupils[primarycounter].type  = items[3]
                primarycounter +=1

            line = readfile.readline().rstrip('\n')

    #File now closed
        return primarypupils
```

3. Now **alter the above code** to deal with Secondary Pupils - storing them in an array called **secondarypupils**:
4. You can now attempt the rest of the program.

Ex 5.3 Code Listing

Exercise 5.4: Skiing Competition

In a local downhill skiing competition the competitors have to pass through a maximum of 9 gates. There are 4 teams

- ❏ Aberdeen Juniors, Slalom Fun, Dundee Dry and Glasgow Juniors

There are 5 competitors from each team

For every gate missed or hit there is a 0.5 second penalty. The last piece of information in the file e.g. 1269 is a list of the gate numbers they faulted at. So for example 1269 would mean that there were 4 gates missed or hit so there is a 2.0 second penalty to be added.

The competitors name, race number, team, race time along with the penalties are stored in the file **skiing.csv**

A sample of data is shown below:

```
Joe Bloggs,2,Aberdeen Juniors,48.2,1269
```

Create a program that will

- ❏ Read and process the file to calculate the total time for each competitor (incorporating any penalties)
- ❏ Store the data in an appropriate record structure
- ❏ List the final times per team
- ❏ Display the individual winner (individual best time inclusive of penalties)

Your output should be similar to below:

```
Final Total Times
Aberdeen Juniors: 392.9 seconds
Dundee Dry: 409.1 seconds
Glasgow Juniors: 360.6 seconds
Slalom Fun: 360.6 seconds
Individual Winner: Brendis Woodland of Glasgow Juniors
```

Extension

- ❏ Display the winning team (best total time) and in the event of a tie the team with the lowest amount of penalties will win.

For example:

```
The winner was: Glasgow Junior with a total time of 360.6 seconds
```

Ex 5.4 Code Listing (copy and paste your code below)

Exercise 5.5: Diving Competition

In an university synchronised diving competition there are 10 participants with 11 judges scoring each between 0 and 10. The results are stored in a file called Diving.csv and an example of the file is shown below:

```
8.9,5.8,1.7,7,2.8,1.7,3.9,1.7,3.8,6.6,2.9
```

To calculate the final score the lowest score **and** the highest score will be removed. The remaining scores are added and multiplied by 0.6 to give the final result.

For the first record above the program should output the total score similar to below:

```
Score 1 = 21.72
```

Complete the Top Level Design and Data flow for the program, refining where appropriate.

Top Level Design and Data Flow (partial)

Main Steps		In	Out
1			
2			
3			

Ex 5.5 Code Listing

Exercise 5.6: Vote Counter

A local council is trialling an electronic voting system.

There are 4 parties in the election with the following candidates:

- ❏ Abolish exams! - Candidate J Smith
- ❏ Apple Party - Candidate G Bloggs
- ❏ Party against Homework - Candidate A N Other
- ❏ Rampant Vikings - Candidate T Thunderson

Each voting console will save a file in the following format:

Voter ID, Party Name, Candidate Name and whether the vote was VALID or SPOILED.

An example from the file is shown below:

```
1,Apple Party,G Bloggs,VALID
2,Abolish Exams,J Smith,SPOILED
3,Abolish Exams,J Smith,VALID
```

For the trial there are 5 files produced from the 5 electronic voting machines, each file will contain **100** records. The results are stored in the files **voting1.csv**, **voting2.csv**, **voting3.csv**, **voting4.csv** and **voting5.csv**

Your task is to create a program that will process the 5 files created by the machines and display the total for each candidate and the winning candidate and number of votes. Any votes marked as SPOILED should not count towards a party.

The output should be similar to below:

Machine files read ok…

Apple Party **66 votes**
Rampant Vikings **61 votes**
Party against Homework **67 votes**
Abolish Exams! **63 votes**

The winner is **A N Other** from the **Party against Homework** party with **67** votes.

Ex 5.6 Code Listing

Exercise 6 Creating Substrings

Required Reading: 4.2 String Handling Presentation

Python refers to this as string slicing but when we want to extract a particular string or substring in Python we use indexing such as below:

mystring = "Computing"			
Command	**Explanation**	**Example**	**Output**
`mystring[i]`	Returns the character at position **i** of the string	print(mystring[3])	p
`mystring[i:j]`	Returns a substring for index **i** to **before** index **j**	print(mystring[3:6])	put
`mystring[i:j:k]`	Returns a substring from index **i** to **before** index **j** with a step of **k**	print(mystring[5:2:-1])	tup

Code Comprehension: String Manipulation

Assuming the variable **`mystring = "This string is an example"`**

Write the Python code below to create the relevant substrings below:

Output	**Code**
`g`	
`This`	
`example`	
`exam`	
`this string`	
`elpmaxe na si gnirts sihT`	

Exercise 6.1 : Memorable Information

Your task is to create a program that could be used in a call centre or similar to answer certain characters from your 'memorable information'.

Your program will use a predefined memorable word and ask the user to enter the characters at two **random** positions. The user has to get **both** of these correct. They are only allowed three attempts, and they answers are **not** to be case specific.

Such as below: for this example we will assume the memorable information word is "`doorjamb`"

```
Enter character at position 1 = d
Enter character at position 5 = g
Sorry that was incorrect, please try again.

Enter character at position 1 = d
Enter character at position 5 = j
Verification accepted
```

Ex 6.1 Code Listing (copy and paste your code below)

Exercise 6.2 : Substrings (Email)

Learning Aims:

- ✔ To manipulate strings in terms of substrings using the **`.index()`** function

Top Level Design and Data Flow

Main Steps		In	Out
1	GetEmail		emailaddress
2	ProcessEmail	emailaddress	mailbox,domain
3	DisplayDetails	emailaddress,domain	

1. Create a new program, calling it **DomainExtractor**.
2. Declare the appropriate subroutines/functions using the Data Flow above.
3. Inside your **GetEmail** function we will ask for the email address to be input. The email address will then be validated before proceeding to ensure that it is a sensible entry. The requirements will be that it can only have a single @ sign in and there has to be at least 3 characters before the @ sign.

```
#Local variables that will be returned
mailbox=""
domainname=""

inputemail = str(input("Please enter the email address: "))
#While there is less than 3 characters before the @ sign
#or there is more than one @ sign
while inputemail.count("@") !=1 or ( int(inputemail.index("@"))<3):
    print("invalid email please re-enter")
    inputemail = str(input("Please enter the email address:  "))
finalemail = inputemail.strip(" ")
return finalemail
```

4. Add the following code to your Process Email subroutine to find the index (position) of the @ symbol so we can extract the domain name. Position is a local variable in this subroutine so its scope will be limited to the subroutine ProcessEmail and then it will be destroyed.

```
def ProcessEmail(emailaddress):
    #find the position of the @ sign
    position = int(emailaddress.index("@"))
    #extract this string before the @ sign
    mailbox = emailaddress[0:position]
    #extract this string from after the @ sign until the end
    domainname = emailaddress[position:]
    return mailbox,domainname
```

5. Add the following code to your **DisplayDetails** subroutine so that the output will be displayed.

```
def DisplayDetails(emailaddress,domainname):
    print("Emailaddress = " + emailaddress)
    print("Domain Name = " + domainname)
```

6. Now we will need to declare variables to pass into the functions and then call your subroutines and functions.

```
#Get the Email Address
emailaddress = GetEmail()
#Pass to process it
mailbox,domainname = ProcessEmail(emailaddress)
#Display the details
DisplayDetails(emailaddress,domainname)
```

Challenge

1. Now run and test your program, you will notice that in the domain it is showing the '@' symbol, alter your code to fix this.

Ex 6.2 Code Listing

Characters - ASCII and vice versa

When manipulating strings it may be required to convert to an ASCII character or to a character from an ASCII value. Python has two predefined functions to achieve this.

Command	Explanation	Example	Output
`ord("a")`	Returns the ASCII code for a character	`ord("a")`	`97`
`chr(65)`	Returns the character for the specified ASCII code	`chr(65)`	`A`

You may find this ASCII table useful when completing the following programming tasks:

Exercise 6.3 : String Handling Exercises

Learning Aims:

- ✔ To create an algorithm to validate a phone number and format it as an international number.

Exercise 1

The mobile number **07840 235678** if converted to a international dialling number would be **+44 7840 235678**

Your task is to write a program to allow a user to enter a valid mobile number. A valid mobile number is one that begins with 0 and has a maximum of 11 numeric digits. It will then be formatted as shown below:

The phone number should be entered as either: 07840 235678 (with space) or 07840235678 (without spaces)

This should produce the output similar to below:

```
Enter the mobile number: 0784 235678
Invalid Phone Number
Enter the mobile number: 07840 235678
Formatted number is: +44 7840 235678
```

HINT: `string = string.replace(" ","")` will remove any spaces in your input string.

Ex 6.3 TASK 1 Code Listing (copy and paste your code below)

Exercise 2

A score for judging a competition is allowed be between 0 and 10 and to have no more than one decimal place. Write and test a function that will check whether a score is valid. The parameters should be the min score, max score and the score itself.

Hint: you may need to look at the function: **`str.index()`**

Ex 6.3 TASK 2 Code Listing

Exercise 6.4 : Order Code Generator

When a customer placed an order with an online supplier they are assigned a unique order ID.

The ID will consist of:

- ❏ First 3 chars of surname,
- ❏ last two digits of postcode (that may be entered in the format AB101FE or AB10 1FE)
- ❏ random 3 digit number.

Although these would be held in a database for the purposes of this program they will be entered.

Previous order ID's are held in a file called OrderHistory.txt.

As an order ID has to be unique the order ID will have to be validated against the previous ones. If there is a matching order such as **LSAFE010** then that order number would be incremented to **LSAFE011**.

Develop a program that will allow the user to

- ❏ Read the file

  As you will be appending the records to the file you will need to count the amount of records there are before creating your arrays. The following code will count the amount of lines and you can then use this to create arrays with the required size:

```
noLines = 0

with open("BankStatement.csv ") as readfile:
    line = readfile.readline().rstrip('\n')
    while line:
        noLines += 1
        line = readfile.readline().rstrip('\n')
```

- ❏ Enter the relevant information
- ❏ Create a unique orderID
- ❏ Display this orderID
- ❏ Append this new orderID to the file

When testing this program you will need to set the digits to create repeated order ID's to test your algorithm .

Ex 6.3 TASK 1 Code Listing (copy and paste your code below)

Exercise 6.5 : Password Generator

A system administrator requires a program that will generate an initial single use default password for a user.

Your task is to create a function that will generate and return this.

- ❏ It should be 8 characters in length.
- ❏ The first character should be an uppercase letter
- ❏ Any vowels in the password should be replaced with the following numbers/symbols (a=@, e = 3, o = 0, i = 1, u = !).
- ❏ There should also be an non alphanumeric character at a random position in the password (NOT the first position)

Ex 6.5 Code Listing

Exercise 6.6 : Password Generator Part 2

A system administrator requires a program that will allow a user to change their password. They should have to enter their old password first and then verify their password by entering their new choice twice.

A log of the previous passwords for the single user are contained in the file: **encryptedfile.txt**

Your function should ensure that:

- ❏ The password matches the most recent password in the file (they are simply encrypted by using a Ceaser Cypher of +2 eg. A has been coded as C. Examples can be seen here
- ❏ Their new password has been entered correctly twice (including case sensitive).
- ❏ Their new password choice has not been used already and is not the same as the current password.
- ❏ The new password must be encrypted and appended to the file (look into appending file access)

Ex 6.6 Code Listing (copy and paste your code below)

Exercise 6.7 Part 1: Binary Converter

A program is being written to help pupils study. It will allow them to enter a binary number (up to 32 bits) and display the decimal value. An example is shown below:

```
Enter Binary Number: 0100 0100
Converted to Decimal = 68
```

You are required to write a function which will return the converted Binary decimal number.

HINT: `binaryvalue = binaryvalue.replace(" ","")` will remove any spaces in your input binary value.

Ex 6.7 Part 1 Code Listing

```

```

Exercise 6.7 Part 2 : Binary Converter

A program is being written to help pupils study. It will allow them to enter a decimal integer number up to 65,535 and display the binary value. An example is shown below:

```
Enter Decimal Number:  68
Converted Binary Number =100 0100
```

You are required to write a function which will return the converted Binary number.

Hint you may want to use the **modulus (%)** operator and the **`int()`** function

Extension:

Change the program so you can set the number of binary digits requires. E.g.

```
Enter Decimal Number:  68
Enter Number of Bits required: 16
Converted Binary Number = 0000 0000 0100 0100
```

Ex 6.7 Part 2 Code Listing

```

```

Exercise 6.7 Part 3 : Two's complement

A program is being written to help pupils study. It will allow them to enter a decimal integer number (from -128 to 127) and display the 8 bit Two's complement binary value. An example is shown below:

Decimal =-29
Two's complement number: **1110 0011**

You are required to write a function which will return the converted Binary number.

Hint: use your previous binary function.

Extension: You can import a function from other files (further reading here)

Ex 6.7 Part 3 Code Listing

Extension: Floating Point Rep

A program is being written to help pupils study. It will allow them to enter a decimal number with a decimal portion (assume **1 bit** for the sign bit, **8** bits for the exponent and **15** for the mantissa) and display the number in floating point representation.

Your program should output the following, giving the final number in the format sign, exponent, mantissa. If the exponent is negative then it should be shown using two's complement notation (**Hint**: use your previous binary two's complement function).

```
Enter Number 44.5
Fixed Point = 101100.1

Sign Bit = 0
Mantissa = 101100100000000
Exponent = 00000110
Final Number = 0 00000110 101100100000000
```

You are required to write a function which will return the converted number in floating point representation

Ex 6.7 Part 4 Code Listing

Exercise 6.8 : Bank Statement Generator

An bank's computer system allows various banking functions to be performed. The file called BankStatement.csv contains the date, deposit/withdrawal amount and the current balance of an account.

A sample of the file is shown:

```
13/3/18,-50.00,350
13/3/18,1000,1350
14/3/18,-10.00,1340
```

You are to write a program that will read the current contents of the file into an appropriate data structure and allow the user to enter either a deposit or withdrawal and update the balance and the file accordingly. The date for the new transaction should be generated automatically using an appropriate predefined function.

There is no need to account for an overdraft facility.

Such as in the example below:

```
Current Balance:£1350.00

Enter + for deposit, - for withdrawal.
Enter transaction: 500
Error: Please enter + or - to show deposit or withdrawal

Enter transaction: +500
Please confirm Deposit of 500  Y/N: F
Invalid - please only enter Y or N
Please confirm Deposit of 500  Y/N: N

Enter transaction: -10.00
Please confirm withdrawal of 10.00  Y/N: Y
Current Balance is £1340.00
```

Top Level Design and Data Flow (incomplete)

Main Steps		In	Out
1	ReadFile		data()
2			
3			

Refinements (as required)

Ex 6.8 Code Listing

Exercise 6.9 : Palindrome Checker

Learning Aims:

- ✔ To create a program to assess whether a word is a palindrome or not

A palindrome is a word which is spelt the same backwards as it is forward such as **(spaces are disregarded)**:

Such as the word "`pullup`" and the phrase "`A man a plan a canal panama`"

You are to write a function which will return **True** if a string is a palindrome or **False** if it is not.

Ex 6.9 Code Listing

Review Task 1: Graphics Card Task

Erica McKenzie runs a business from her home in a small village in the highlands of Scotland designing custom-built computer systems for businesses in her area. A local software design company wants to develop software to help her grade and select suitable processors for her customers.

This software will:

- ❑ **Generate and display a customer code**
  - ❑ ask for the forename and surname of the customer
  - ❑ create the code for the customers by:
    - ❑ extracting the first letter of each name and add these to the code
    - ❑ adding a random number between 0 and 9
    - ❑ adding a random lower case character (a to z)
  - ❑ display the generated customer code
- ❑ **Read the processor details from a file**
- ❑ **Allocate points based on clock speed, data bus width and cache size**
- ❑ Use the points to allocate a star rating to each processor
- ❑ Calculate the number of processors at each star rating
- ❑ Display a table showing the points and rating for each processor
- ❑ Display a table showing the number of processors at each star rating
- ❑ Select the processor with the highest number of points
- ❑ **Display the points and rating for the selected processor and write the points for each processor to a file**

The program will be tested using the details of five processors (see below).

Reading processor details

The data on the five processors which should be used in testing is given below. This is contained in the file processors.txt and an example of the file is shown below. The data in order is Processor Name, Clock Speed, Data Bus Width and Cache size.

```
Intium II,2.8,24,128
```

All data should be stored in a suitable user defined data structure.

Allocating points to each processor

Points will be awarded to each processor as follows:

- ❏ Clock speed is less than 2 GHz, award 10 points.
- ❏ Clock speed is between 2 GHz and 3 GHz inclusive, award 20 points.
- ❏ Clock speed is greater than 3 GHz, award 30 points.

Data Bus Points

1 point for each line on the data bus e.g. 24 bit data bus, award 24 points.

Cache Size Points

1 point for each whole 10 Kb of cache e.g. 128Kb cache, award 12 points.

(128/10 = 12·8, which should be rounded **down** to 12)

Calculate star ratings

The processors should then be given a star rating. Processors with less than 60 points gain one star. Between 60 and 80 points inclusive earns a processor two stars. Processors with more than 80 points gain three stars. The number of processors should be counted for each star rating.

Displaying the results

The output from the program should be formatted similar to the following:

```
Erin McKenzie your order code is EM5f

Processor    Points     Rating
Intium II    56         *
Celerisc I   67         **
Ethloan III  64         **
Powerup II   125        ***
Mitduo III   66         **


Rating      Number of Processors
*           1
**          3
***         1
The Processor selected is the PowerupII with 125 points.
```

Storing the points

The name, points and ratings for each processor should then be stored in a new text file.

Review Task 1 Code Listing

Review Task 2 : Choral Shield

Essell Academy is running its annual 'Choral shield' competition. Each of the four houses (Alpha, Beta, Delta and Gamma) enters a team of senior pupils. The show is performed on three evenings — Wednesday, Thursday and Friday.

The cost of a ticket on the Wednesday and Thursday night is £5 and the cost of a Friday night ticket is £10. Tickets can only be ordered for one night at a time. Anyone wishing to attend on more than one night will need to make multiple orders.

When buying their tickets, customers are asked in which of the three areas of the theatre they wish to sit — 1(front), 2(middle) or 3(back). There are 200 seats available in each area of the theatre. The money raised from the concerts will go to the top charity suggested by customers.The school sells tickets via the school website, in addition to them being on sale from the school office.

You have been asked to write a program to find which method of purchase has been the most popular and the total amount of money raised from all orders. Your will be supplied with a file that contains the data from 300 orders.

The orders file (Choral Shield.csv) has the data in the following order:

Customer ID:	this will take the form C001
Ticket ID:	this will represent the night (W, T or F) and one of the three positions in the theatre - eg T2 is a Thursday night in the middle section of the theatre.
Number of tickets:	an integer value between 1 and 10
Method of purchase:	this will represent whether bought in school or via the website and will either be S or W

For example, the first four records in the file might look like this:

- `C001,W1,5,S`
- `C002,F3,4,W`
- `C003,F1,6,W`
- `C004,T2,2,S`

Using this file of data, your program should produce the following output (generating the year automatically using a pre-defined function):

```
Essell Academy Choral Shield <year>
The most popular method of sales is XXXXX
The total money raised for charity is £XXXXX
```

Your program should then :

- ❏ export **all data** for **each night (Wednesday, Thursday and Friday)** to separate files(one for each day).
- ❏ When being written to the file each order should be also assigned a unique order ID with the first character of the day and a **unique** 3 digit number e.g. T001

To help you test that your program calculates totals correctly you will be provided with three previous year's data (in csv files names ChoralShield1.csv, ChoralShield2.csv **and** ChoralShield3.csv) and results.

	Most popular sales method	Total money
Test 1	School	£10,810
Test 2	Web	£10,925
Test 3	School	£10,840

Review Task 2 Code Listing

Review Task 3 : Crash Logging

Suretest employ people to test computer games. New employees are given a budget to buy a suitable computer system for testing the games. Each tester plays the games on their own desktop computer system and logs the faults they encounter while playing each game. To help collate the test data generated by their testers, Suretest wish to create a program to store and process test results.

Suretest decide to commission a software development company to develop a program to log faults found by their testers. The program will be tested on the game, "Hamster Commanders".

The software will:

- ❏ Read the log files supplied by the beta testers showing the game levels (a-j) which crashed during each test session.
- ❏ Count the number of times each level has crashed.
- ❏ Produce a list of the number of crashes in each level.
- ❏ Calculate which level/levels had the greatest number of crashes.
- ❏ Display any levels that had zero crashes

Each of the beta tester will submit a log file detailing the levels in which faults were found. They are asked to play through the 10 available levels. (indicated by a letter a-j). The results are stored in a file called gamecrashes.txt and the first line of the file is shown below:

```
accicjejji
```

The total crashes should be displayed as below:

Level	Number of Crashes
a	21
b	0
c	12
d	5
…	…
j	21

The information on the level/levels with the most crashes should be displayed as shown below:

```
The level(s) with the most number of crashes were a,j. These
levels crashed 21 times.

The level(s) with no crashes were b.
```

Review Task 3 Code Listing

Extension

Create a function that will allow you to calculate the largest prime factor of a given number.

This site may help: https://www.mathsisfun.com/prime-factorization.html

The largest prime factor of **13195** is **29**.

A prime factor is any factor that is also a prime number. In other words: any of the prime numbers that can be multiplied to give the original number. Example: The largest prime factor of 20 The prime factors of 15 are 3 and 5 (because 3×5=15, and 3 and 5 are prime numbers).

Further Extension Tasks

For any other programming extension tasks create an account on

Project Euler (https://projecteuler.net/)