

Summer Practical Challenges in Python
Workbook & Tracker



Cardinal Griffin Catholic College

Name: _____

Table of Contents

Note to Students!	4
Programming Challenge 1: Comparing 2 numbers	8
Programming Challenge 2: Weekday checker.....	9
Programming Challenge 3: Calculating Area of a Triangle.....	11
Programming Challenge 4: Calculating a dogs age	12
Programming Challenge 5: Totalling up an array	14
Programming Challenge 6: Perimeter Calculator	16
Programming Challenge 7: Five-A-Day	18
Programming Challenge 8: Fibonacci	20
Programming Challenge 9: Sort It Out!	22
Programming Challenge 10: Registration!.....	24

Python Challenges Checklist & Tracker

This section is solely for you to keep track of all of the challenges that have been set for you to Design, Code and Test in Python. You should keep track of what you have completed as you work through your challenges.

<u>Challenge Number:</u>	<u>Designed</u>	<u>Coded</u>	<u>Tested</u>
<i>Challenge 1: Comparing 2 numbers</i>			
<i>Challenge 2: Weekday Checker</i>			
<i>Challenge 3: Calculate Area of a Triangle</i>			
<i>Challenge 4: Calculating a dogs age</i>			
<i>Challenge 5: Totalling up an array</i>			
<i>Challenge 6: Perimeter Calculator</i>			
<i>Challenge 7: Five-A- Day</i>			
<i>Challenge 8: Fibonacci Sequence</i>			
<i>Challenge 9: Sort-It- Out!</i>			
<i>Challenge 10: Registration!</i>			

Note to Students!

The challenges in this booklet have been put together to help you plan, design, create and then test a range of programs. When you start your programming journey, experience and practice are absolutely vital. Knowing when and how to use certain features of a programming language, such as conditions and loops, are often difficult at first. Thinking about your program BEFORE you create any code is highly recommended, and indeed, vital at GCSE and AS/A2 Level for the higher grades. You can use the grids on the inside covers to help monitor your progress and track the programming skills you have begun to master.

BEFORE creating a program you should:

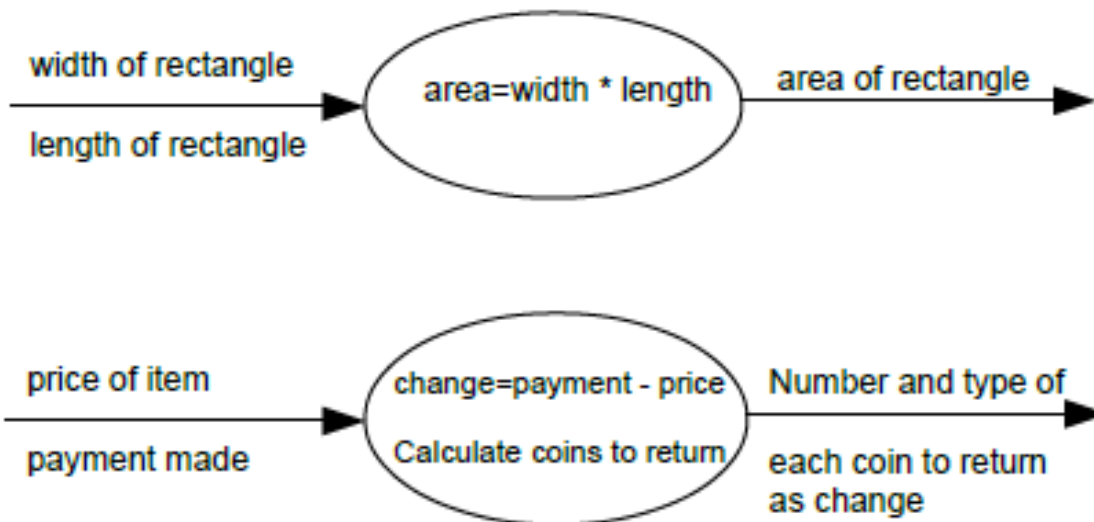
- ***Identify the Inputs, Processing and Outputs***
- ***Identify key variables and other data structures***
- ***Design a labelled interface (for non-console apps)***
- ***Think about the steps the program should execute (using flowcharts and/or pseudocode)***
- ***Draw up a test plan***

After you have created your program, it should be thoroughly tested using a test plan that covers valid, invalid and extreme data. Only when all of the challenge is complete should you mark it off on your tracker grids. The challenges were originally written with the Python language in mind. The wording of some of the written questions may not be 100% familiar to users of other languages, although similar features and concepts will be available in their language of choice.

I → P → O Diagram

This simple diagram helps you focus on what goes **into** the program and what should come **out**. Inputs and Outputs should be stated as clearly and accurately as possible and should help you start to think about the variables needed for the program. The processing may be expressed simply at this stage.

Examples:



Variables and other Data Structures

Nearly all programs will need to store and manipulate data in some way. You should work out the main requirements before you create the program. Think about the structures needed to store the inputs and outputs of the program. Think about the actual processing involved – does it need to be done in stages and if so, what extra data structures are needed? Would an array or list be appropriate?

BEFORE creating a program, you should think about the following for each data structure needed: name; datatype; typical value; minimum/maximum value (validation).

Flowcharts and Pseudocode

Drawing a flowchart or diagram should help you think about the various stages needed in the program: What are they? In what order should they be executed (sequence)? Which sections involve making decisions (selection)? Which sections involve repeating instructions (iteration)?

Pseudocode should help you get to grips with the hardest part of the program: the processing. How exactly are you going to turn those inputs into the expected outputs? What needs to be done? Is it a mathematical formula? Do you need to manipulate strings?

AVOID POINTLESS PSEUDOCODE

```
Input number1
Input number2
Input number3
Input number4
Input number5
Output average
```

Version 1 doesn't really help...

```
Input 5 numbers
average=(sum of the five numbers) / 5
Output average
```

Although shorter, Version 2 is a bit more helpful

```
numberOfItems=5
total=0
average=0
for c=1 to numberOfItems
    Output "Please enter a number"
    Input number
    total=total+number
next
average=total / numberOfItems
Output "The average is " & average
```

In Version3, nearly all the "thinking" has been done

Test Plan

How will you know if your program works correctly?

How will you know what crashes/kills your program? Create a test plan **BEFORE** you write the code. You should be able to predict the output created by a set of inputs. Try to design tests that cover valid, invalid and extreme inputs. A good test will highlight a problem/bug; there's no point in 10 "it works ok" tests.

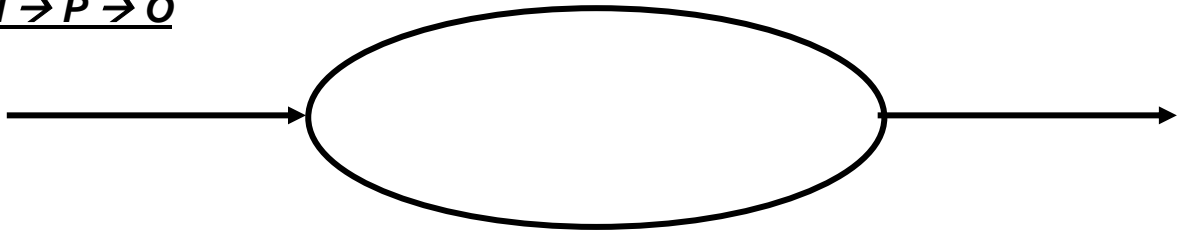
In some of the challenges, a few starting tests have been given but feel free to add more!

Programming Challenge 1: Comparing 2 numbers

Design and implement a program that will:

- *Ask the user to input two numbers*
- *Outputs the larger number of the two numbers along with a suitable message.*

I → P → O



Pseudocode:

Test Plan:

How do you know that your program is working?

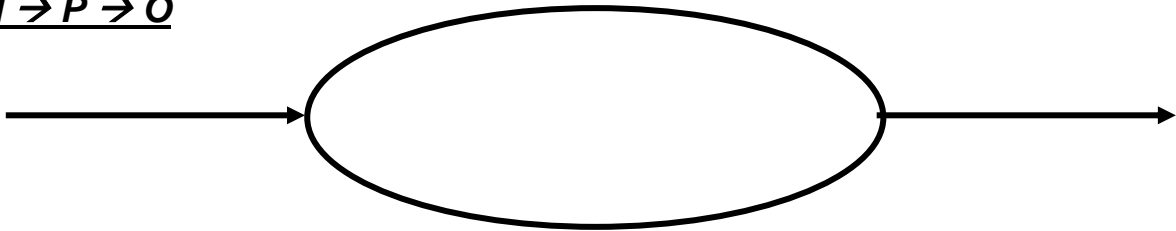
Programming Challenge 2: Weekday checker...

If Monday == 0, Tuesday == 1, Wednesday == 3, etc...

Design and implement a program which will:

- **Ask the user for an input.**
- **Outputs the day that number represents.**
- **Validates that number so that the error message displays if the number is not a valid weekday number.**

I → P → O



Variables and other Data Structures

<u>Name</u>	<u>Datatype</u>	<u>Typical Value</u>	<u>Minimum Value</u>	<u>Maximum Value</u>

Pseudocode:

Test Plan

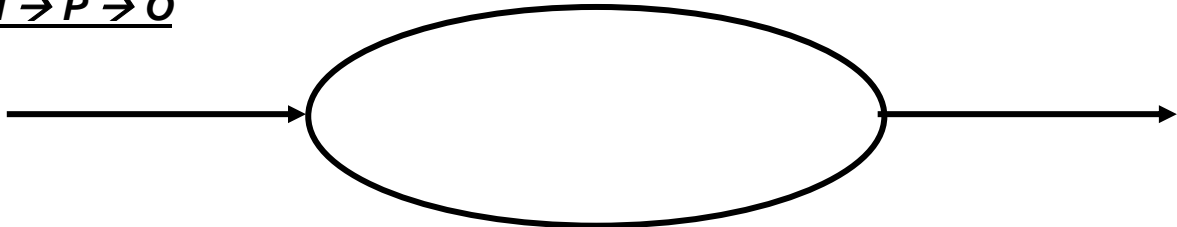
Complete the first 2 columns of the following test plan BEFORE implementation. Add your own tests to the blank lines.

<i>Input</i>	<i>Expected Output</i>	<i>Actual Output</i>	<i>Comments</i>
2, 3			
100, 100			
4.2, 1.8			
1m, 3			

Programming Challenge 3: Calculating Area of a Triangle

Design and implement a program that allows the user to calculate **the area of a triangle**. The user should be prompted to enter 2 variables representing height and width and the program calculates the rest.

$I \rightarrow P \rightarrow O$



Variables and other Data Structures

<u>Name</u>	<u>Datatype</u>	<u>Typical Value</u>	<u>Minimum Value</u>	<u>Maximum Value</u>

Pseudocode:

Test Plan

Complete the first 2 columns of the following test plan BEFORE implementation. Add your own tests to the blank lines.

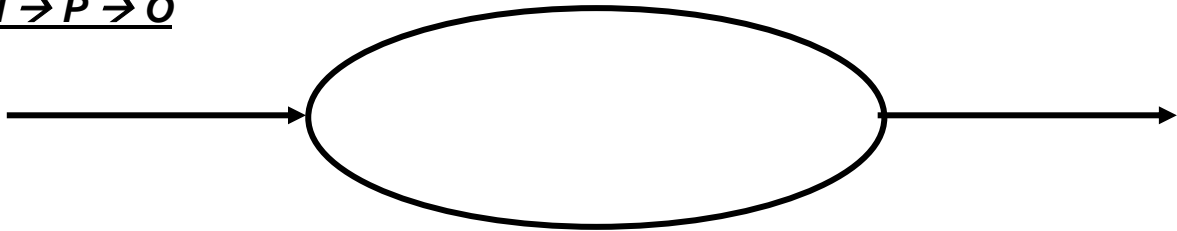
<i>Input</i>	<i>Expected Output</i>	<i>Actual Output</i>	<i>Comments</i>
2, 3			
100, 100			
4.2, 1.8			
1m, 3			

Programming Challenge 4: Calculating a dogs age

A dog that is 5 years old is equivalent to a 42 year old human. You need to write a program that converts the age of a dog to the equivalent of a human. Design and implement a program which:

- **Asks for the age of the dog in years.**
- **If the age is 2 or less, the human equivalent is 12 times the age.**
- **If the age is more than 2, the human equivalent is 24 for the first 2 years, plus 5 for every additional year.**

I → P → O



Variables and other Data Structures

<u>Name</u>	<u>Datatype</u>	<u>Typical Value</u>	<u>Minimum Value</u>	<u>Maximum Value</u>

Pseudocode:

Test Plan

Complete the first 2 columns of the following test plan BEFORE implementation. Add your own tests to the blank lines.

<i>Input</i>	<i>Expected Output</i>	<i>Actual Output</i>	<i>Comments</i>
1			
7			
13			
11			

Programming Challenge 5: Totalling up an array

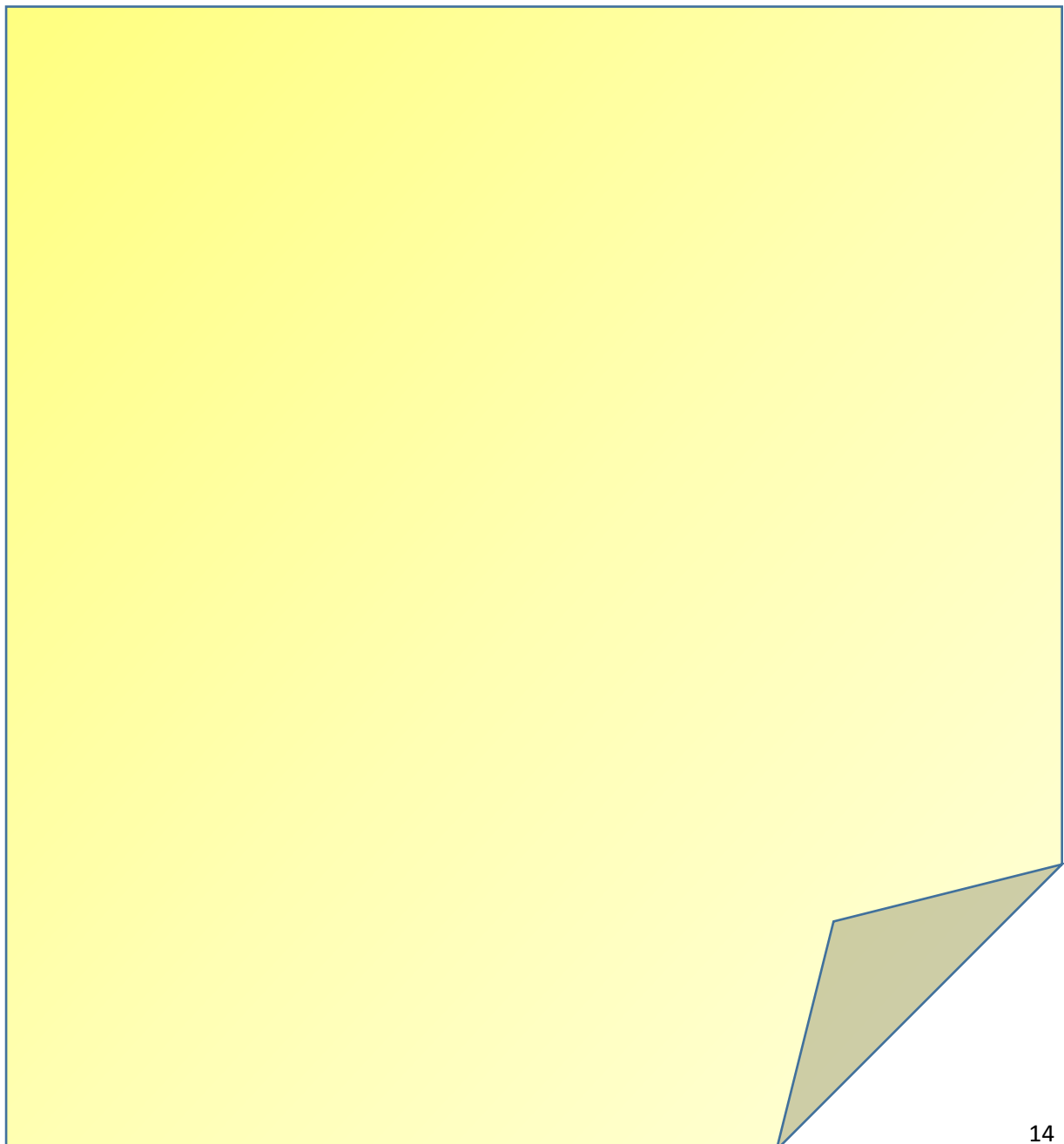
Consider the following array **numbers** = [2, 5, 6, 2, 1, 7, 2, 8, 9, 1] where the value of numbers[2] is 6.

You are to design and implement a program that will:

- **Add up all of the numbers in the array.**
- **Display the result with a suitable output.**

Mr O'Neill's TOP TIP: To complete this program you need to make sure that you are using iteration.

Pseudocode:



Test Plan

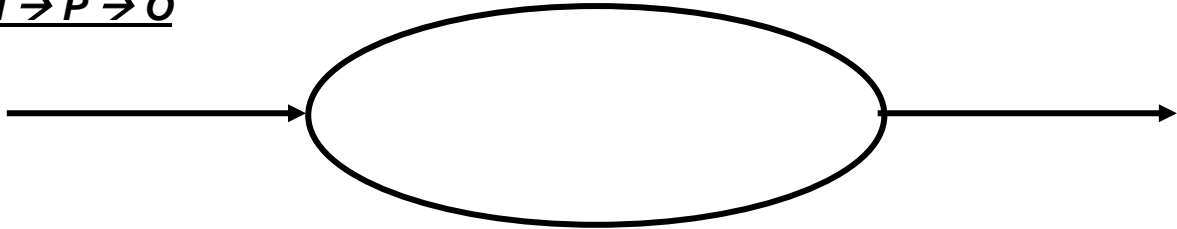
Complete the first 2 columns of the following test plan BEFORE implementation. Add your own tests to the blank lines.

<i>Input</i>	<i>Expected Output</i>	<i>Actual Output</i>	<i>Comments</i>
<i>numbers</i> = <i>[2, 5, 6]</i>			
<i>numbers</i> = <i>[2, 4, 7, 6]</i>			
<i>numbers</i> = <i>[2, 1, 3, 5, 4]</i>			
<i>numbers</i> = <i>[2, 5, 6, 2, 1,</i> <i>7, 2, 8, 9, 1]</i>			

Programming Challenge 6: Perimeter Calculator

Design and implement a program that allows the user to enter 2 numbers representing the width and length of a rectangle. The program calculates and displays the perimeter of the rectangle.

$I \rightarrow P \rightarrow O$



Variables and other Data Structures

<u>Name</u>	<u>Datatype</u>	<u>Typical Value</u>	<u>Minimum Value</u>	<u>Maximum Value</u>

Pseudocode:

Test Plan

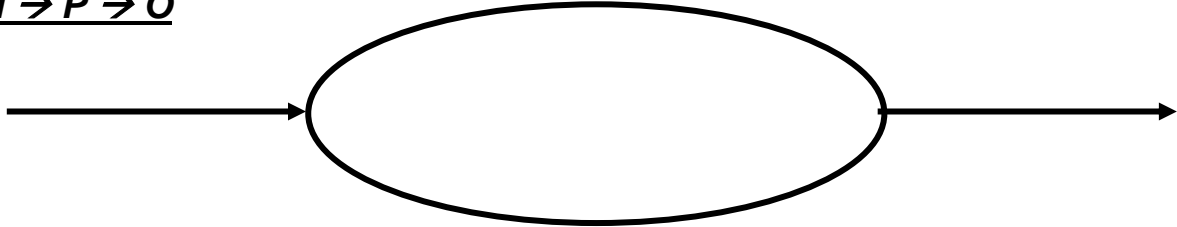
Complete the first 2 columns of the following test plan BEFORE implementation. Add your own tests to the blank lines.

<i>Input</i>	<i>Expected Output</i>	<i>Actual Output</i>	<i>Comments</i>
2, 3			
100, 100			
4.2, 1.8			
1m, 3			
1, 3m			

Programming Challenge 7: Five-A-Day

Design and implement a program that allows the user to enter a positive number. The program displays all of the numbers divisible by 5 up to the number entered. Eg the user enters 30, the program displays 5, 10, 15, 20, 25, 30

$I \rightarrow P \rightarrow O$



Variables and other Data Structures

<u>Name</u>	<u>Datatype</u>	<u>Typical Value</u>	<u>Minimum Value</u>	<u>Maximum Value</u>

Pseudocode:

Test Plan:

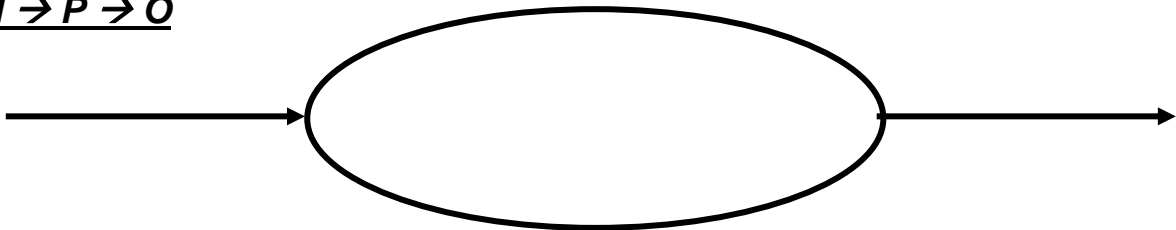
Complete the first 2 columns of the following test plan BEFORE implementation. Add your own tests to the blank lines.

<i>Input</i>	<i>Expected Output</i>	<i>Actual Output</i>	<i>Comments</i>
30			
100000			
86			
0			
-22			

Programming Challenge 8: Fibonacci

The first few numbers in the Fibonacci sequence are: 0, 1, 1, 2, 3, 5, 8, 13, 21... Design and implement a program to display N Fibonacci Numbers, where N is entered by the user.

I → P → O



Variables and other Data Structures

<u>Name</u>	<u>Datatype</u>	<u>Typical Value</u>	<u>Minimum Value</u>	<u>Maximum Value</u>

Pseudocode:

Test Plan:

Complete the first 2 columns of the following test plan BEFORE implementation. Add your own tests to the blank lines.

<i>Input</i>	<i>Expected Output</i>	<i>Actual Output</i>	<i>Comments</i>
3			
7			
9			
0			

Explain these terms:

Iteration:.....
.....

Loop:.....
.....

Stopping Condition:.....
.....

What is the difference between a FOR...NEXT loop and a REPEAT....UNTIL loop?
.....
.....

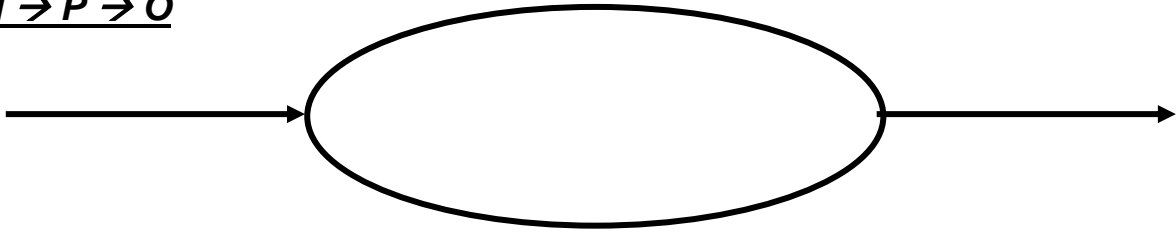
What is the difference between a FOR...NEXT loop and a WHILE....DO loop?
.....
.....

What is the difference between a REPEAT...UNTIL loop and a REPEAT....UNTIL loop?
.....
.....

Programming Challenge 9: Sort It Out!

Write a program that allows the user to enter 5 words. The words are sorted and displayed in alphabetical order.

I → P → O



Variables and other Data Structures

<u>Name</u>	<u>Datatype</u>	<u>Typical Value</u>	<u>Minimum Value</u>	<u>Maximum Value</u>

Pseudocode:

Test Plan:

Complete the first 2 columns of the following test plan BEFORE implementation. Add your own tests to the blank lines.

<i>Input</i>	<i>Expected Output</i>	<i>Actual Output</i>	<i>Comments</i>
And, but, car, egg, dust			
Dust, egg, car, and, but			
Egg, dust, car, but, and			
And, able, but, bit, bat			
And, but, car, car, dust			

Mr O'Neill's TOP TIPS:

- It is very easy to over-complicate this one. Don't!
- Be careful with your interface – there are up to 30 test scores to enter, do you really want 30+ textboxes?
- Do you want to use more than one array?
- Do you want to use iteration? A ***FOR...NEXT*** loop? A ***REPEAT...UNTIL*** loop?

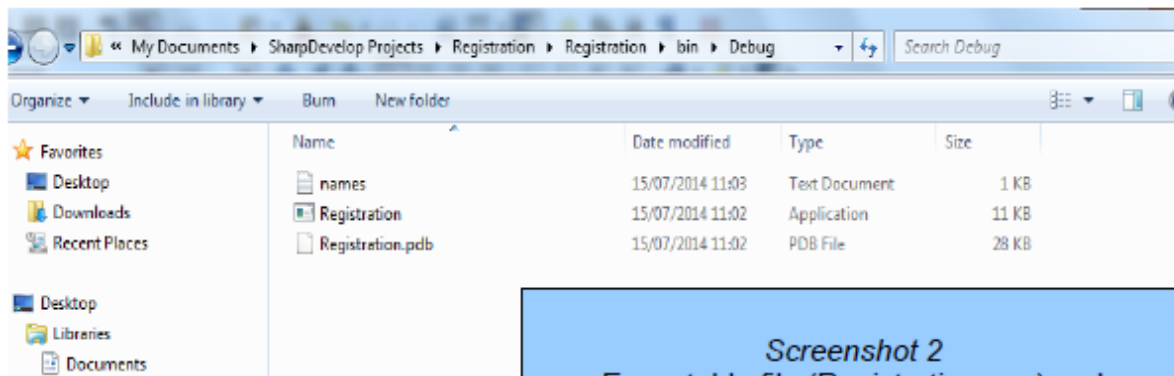
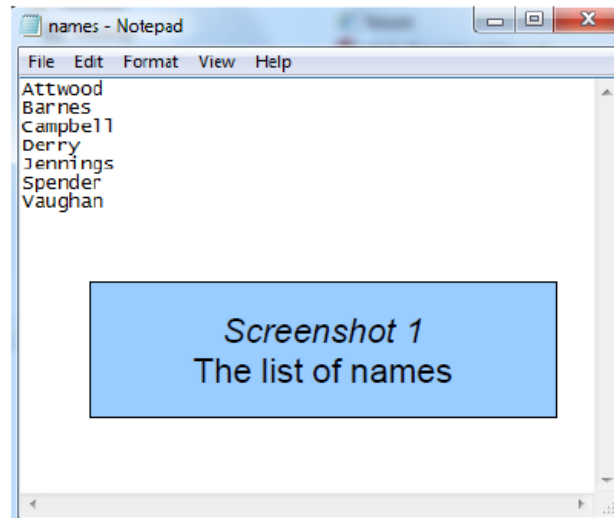
Programming Challenge 10: Registration!

Part 1: Use a text editor (Notepad) to create a text file as in Screenshot 1.

It must be saved as “names.txt”,

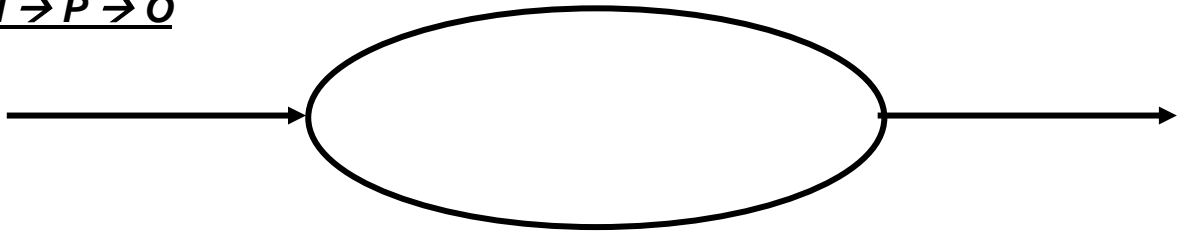
It must then be copied into the same folder as the executable file created by your Python IDE as in Screenshot 2.

This means that, for the moment, you do not have to worry with providing full file-paths in your code. The executable will look in its own folder for the text file needed.



Part 2: Design and write a program that opens, reads and displays the text file "names.txt". Remember that the program won't know in advance how many names are in the text file.

$I \rightarrow P \rightarrow O$



Variables and other Data Structures

<u>Name</u>	<u>Datatype</u>	<u>Typical Value</u>	<u>Minimum Value</u>	<u>Maximum Value</u>

Pseudocode:

A large yellow rectangular area with a folded corner at the bottom right, intended for writing pseudocode.

Test Plan:

How will you know if the program works?

Mr O'Neill's TOP TIPS:

It is very easy to over-complicate this one. Don't!

- Open the file
- Read and display one line at a time
- Close the file

Testing

Test your program. Does it work? Add another name to the text file using a text editor. Does your program still display all the names in the file?

Delete a couple of names from the text file. Does your program still display the full list?

Extension

Create group1.txt, group2.txt and group3.txt files containing different sets of names. Add a feature to your program that allows the user to choose which file to display.

Think about the advantages of disadvantages of each of the following methods of choosing the file:

- ***A text box where the user can type in any filename***
- ***A drop down menu with the 3 text files already listed***
- ***A file-picker control***