

Copyright

© 2017 PG Online Limited

The contents of this unit are protected by copyright.

This unit and all the worksheets, PowerPoint presentations, teaching guides and other associated files distributed with it are supplied to you by PG Online Limited under licence and may be used and copied by you only in accordance with the terms of the licence agreement between you and PG Online Limited. Except as expressly permitted by the licence, no part of the materials distributed with this unit may be used, reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic or otherwise, without the prior written permission of PG Online Limited.

Licence agreement

This is a legal agreement between you, the teaching institution, and PG Online Limited. PG Online Limited grants to you a non-exclusive, non-transferable, revocable licence to use this unit and all the worksheets, PowerPoint presentations, teaching guides and other associated files distributed with it in the course of teaching by your teachers and/or employees.

The materials distributed with this unit may be copied and used by your teachers and/or employees on a single site only in the course of their teaching. You warrant that you shall not, and shall procure that each of your teachers and/or employees shall not, share in any way any of the materials or part of the materials with any third party, including users on another site or individuals who are teachers and/or employees of a separate institution. You acknowledge and agree that the materials must remain with you, the teaching institution, and no part of the materials may be transferred to another institution. You also warrant that you shall not, and shall procure that each of your teachers and/or employees shall not, procure, authorise, encourage, facilitate or enable any third party to reproduce these materials in whole or in part without the prior permission of PG Online Limited.

In consideration of the licence granted to you, you shall indemnify PG Online Limited against all liabilities, costs, expenses, damages and losses (including but not limited to any direct, indirect or consequential losses, loss of profit, loss of reputation and all interest, penalties and legal costs and all other professional costs and expenses) suffered or incurred by PG Online Limited arising out of or in connection with the exercise by you of your rights granted under this licence.

Teacher’s Guide

Introduction

This teacher’s guide contains detailed lesson plans to accompany the eleven sets of PowerPoint slides, worksheets and practical problems.

The lessons are designed to form a strong base for ideas for the teacher and should be adapted to suit the teaching style and preferences of the individual teacher, and the resources and nature of the individual school or Computing / ICT department.

The material supplied for this unit includes:

* 10 PowerPoint presentations, each designed to cover one lesson
* Worksheets, with answers provided, for the first ten lessons
* 10 homework tasks, each with answers
* 60 sample Python programs and associated text files
* Reference sheet on reading and writing text files in Python
* Reference sheet on list handling in Python
* Five longer exemplar practical problems and Python solutions which recap the skills learned in the unit

### Summary

Although this unit assumes that students have had some exposure to programming in Python, all the basics are covered and students with no experience should quickly be able to catch up.

Lessons 1 to 9 cover all the basic programming syntax that students will require at GCSE level, giving numerous examples of how to write Python programs to solve different types of problem. Lesson 10 summarises some of the most common programming techniques used, such as validating data entry and creating a menu system with separate functions for each of the menu options.

It is not expected that each slideshow will match up to a single one-hour lesson. Some topics may come more naturally and other topics may take more time and practice for students to grasp comfortably.

### Learning Outcomes for the unit

**At the end of this Unit all students should be able to:**

* Use basic programming structures of sequence, selection and iteration
* Use a regular expression to validate an input
* Create, manipulate and interrogate lists
* Read from, write to and append simple text files
* Write, test and debug programs to solve simple problems

**Most students will be able to:**

* Write error-free, well-documented programs
* Use modular programming techniques to break down a problem into its component parts and write well-structured programs using separate functions called from a main program

**Some students will be able to:**

* Write error-free programs that extend advanced techniques (e.g. using 2D arrays)

### Previous Learning

Students should have some previous experience of Python programming, including basic inputs and outputs, converting inputs to numbers, selection (if-elif-else statements), loops (both while and for loops) and modules (e.g. random numbers). These skills may have been acquired through the PG Online units ‘Introduction to Python’ and ‘Python: Next Steps’.

The first three lessons are a revision of these basic techniques.

### Vocabulary

Vocabulary associated with programming and particularly Python, such as:

Integrated development, IDLE, variable, string, syntax, assignment statement, data type, integer, float, round, BIDMAS, selection, iteration, regular expression, list, two-dimensional list, text file, syntax error, logic error, debug, procedure, function, call, argument, parameter.

### Preparation

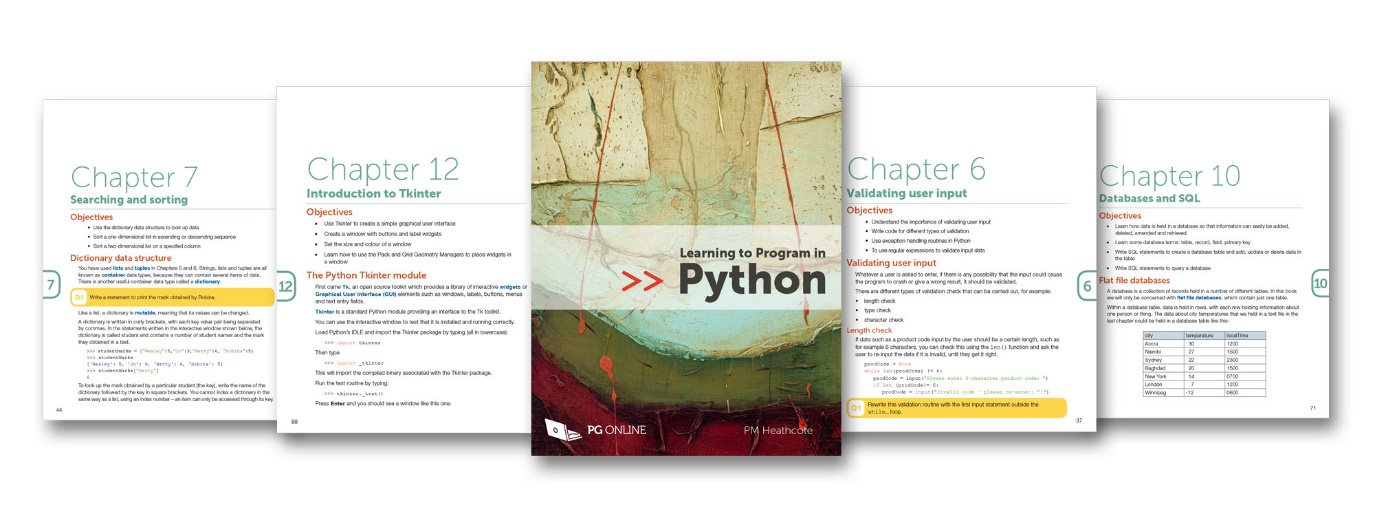
Make sure Python 3.x is installed and running correctly. Run through the programs mentioned in the PP Guides.

### Assessment

Assessment of this work is intended to be formative throughout, with a variety of worksheets and homework tasks to provide feedback to the teacher.

Suggested Resources

No additional resources are required for this unit. However, a complementary book may be a helpful accompaniment as a reference text.

****

The textbook *Learning to Program in Python* by PM Heathcote, published June 2017 (128 pages) provides a straightforward guide to the Python programming language and programming techniques. It covers all of the practical programming skills that may be required up to GCSE level and for those at A Level with limited exposure to Python. It is suitable for experienced programmers as well as students or individuals with very little or no programming experience in other languages.

It teaches basic syntax and programming techniques, and introduces three inbuilt Python modules:

* **Tkinter**, used for building a graphical user interface, which is an option that some users may like to include in their project work.
* **SQLite**, which enables the creation and processing of a database from within a Python program. This provides an alternative to writing to a text file when data needs to be stored and retrieved.
* **pdb**, Python’s debugging module, which can be used to help find elusive logic errors.

Questions and exercises are included in every chapter. Answers to these as well as over 120 Python programs for all the examples and exercises given in the book are provided to students and teachers in a free pack available to download from our website.

The book is published by PG Online both in a printed edition and in electronic format. Please refer to www.pgonline.co.uk for ordering and pricing details and to access a free digital or paperback inspection copy.

Python 3

Python is a free, open-source programming language that was first released in 1991 – named after Monty Python’s Flying Circus. It is a powerful, high-level, object-oriented language which will run on any platform, is easy to learn and used by software developers all round the world. The Python website [www.python.org](http://www.python.org) quotes the following claim:

“I estimate that Python makes our coders 10 times more productive than Java programmers, and 100 times more than C programmers." -- Curt Finch, CEO, [Journyx](http://www.journyx.com)

It can be used for dozens of different applications from Web programming and networking to databases, scientific and numerical applications and games development.

These notes are written with the aim of enabling the teacher to deliver lessons on Python without necessarily giving the students a mountain of indigestible syntax rules, and to enable a teacher unfamiliar with Python to help students with problems they encounter when using the worksheets.

The student worksheets accompanying these notes will need minimal introduction, making the lesson largely a process of the students finding out things for themselves while interacting with the computer.

Installing Python in Windows

Go to the website <http://www.python.org/download/> and select [Python 3 Windows x86 MSI Installer](http://www.python.org/ftp/python/3.3.2/python-3.3.2.msi). Follow instructions to download. You can accept the default configuration.

Python will also install on MacOS and Linux based platforms.

The Integrated Development Environment

Python has its own development environment which is known as IDLE. This is where you will write, save and edit programs. IDLE provides two modes in which you can write programs: an Interactive mode and a Script mode.

The Interactive mode instantly interprets and runs each line of program code as soon as you press Enter. It is a quick way to get started and useful for trying out an instruction to make sure you have the syntax correct. However, in order to save a program so that you can load and run it as often as you like, you need to work in Script mode.

Lesson plans

|  |  |
| --- | --- |
| Lesson 1 - Fundamentals | |
| Preparation   * Make sure Python 3.x is installed and running correctly. * Run through the programs mentioned in the PP Guide. Many of them can be found in the Sample Python Programs folder   Learning Objectives:   * Be able to identify and correct common errors in computer programs * Be able to use inputs, outputs, arithmetic and string handling | |
| Content | Resources |
| **Starter**  Students will need to load Python in Interactive mode, so that they can try out different Python statements. In order to test new programs in Script mode, they need to select File, New File from the menu, type, save and run the program.  The lesson starts by asking students to identify errors in a short section of Python code. This will help the teacher to discover what, if any, programming experience the students have had. Some of the errors such as missing closing brackets, closing quotes or the logic error “less than” symbol rather than “greater than” symbol should be detectable even by students who have not used Python before. Other errors will need explaining.  **Main**  Explain the three types of error, syntax, logic and runtime. Ask for other examples of each – for example, division by zero will cause a runtime error, which means the program will crash. Give some simple examples of programs for the students to try out in Script mode.  Give out **Worksheet 1** and ask the students to do **Question 1** – they can type the code into Python, save, run and debug their programs.  Input and output statements, data types and casting are covered next. Students can then do **Question 2** on the worksheet.  Go over the arithmetic statements and operators, and ask students to complete **Questions 3**. Some students may like to try the **Extension question 4**.  **Plenary**  A final code snippet containing errors is used to recap the contents of this lesson. Give out **Homework 1**. | PowerPoint Guide: Lesson 1 Fundamentals  Worksheet 1  Worksheet 1 Answers  Sample Python programs  Homework 1  Homework 1 Answers |
| Lesson 2 – Selection and iteration | |
| Preparation   * Make sure Python 3.x is installed and running correctly. * Run through the programs mentioned in the PP Guide. Many of them can be found in the Sample Python Programs folder   Learning Objectives:   * Describe the different comparison operators * Be able to use selection statements * Be able to use counter controlled (for) loops * Be able to use condition controlled (while) loops | |
| Content | Resources |
| **Starter**  Show the first slide as a Starter activity. The answers are on the next slide. Point out the symbols for equality (==) and inequality (!=).  **Main**  Summarise all the relational operators. The next few slides give examples of the use of different comparison operators. Students can try out Python statements and code snippets in IDLE using interactive mode.  Give out **Worksheet 2a** and ask students to complete **Questions 1 and 2**.  **Selection statements**  Go over the three types of selection statement described on the next few slides, allowing students to try them out in IDLE’s interactive mode.  Demonstrate the pitfalls of wrongly constructed if statements, as shown on the slides “**if vs elif**”.  Ask students to do **Questions 3 and 4** on the Worksheet.  **Count controlled loops (for)**  Point out that the statements in the loop  for count in range(5)  are performed 5 times, but count goes from 0 to 4, not 1 to 5.  You can include the starting value in the range, for example range(1,5). Ask, how many times would this loop be performed? (Answer on next slide).  **Running total**  The technique of calculating a total of several values entered by the user, using a loop and a running total, is very important in programming. Make sure all students understand this technique and why it is used. Suppose you had 1000 values to add up, input from a file. The “running total” technique is equally applicable.  Give out Worksheet 2b and ask students to complete questions 1 and 2.  **Condition controlled loops (while)**  Explain the while loop. Note that if the condition is False when the while statement is executed, the loop will be skipped altogether.  The condition is tested after the last statement in the loop has been executed, not as soon as it becomes false. This becomes relevant in longer loops.  Correct indentation is crucial as it indicates where the loop ends.  Ask students to complete questions 3 and 4.  **Plenary**  Recap the two different types of loop covered in this lesson. Note that there is no repeat..until loop in Python.  Give out Homework 2. | PowerPoint Guide: Lesson 2 Selection and iteration  Worksheet 2a  Worksheet 2a Answers  Worksheet 2b  Worksheet 2b Answers  Sample Python programs  Homework 2  Homework 2 Answers |

|  |  |
| --- | --- |
| Lesson 3 – Functions | |
| **Preparation:**  Make sure that the programs L3 WS3a Ex 2 tic\_tac\_toe.py and  L3 WS3a Ex 3 tic\_tac\_toe with loop.py are available for students to download. They are provided in the Sample Programs folder.  Learning Objectives:   * Be able to use decomposition to help solve a larger problem * Be able to use subroutines (called procedures or functions) to help make your programs easier to create and more efficient | |
| Content | Resources |
| **Starter**  The starter activity is designed to get students used to the format of a function and the calling statement.  **Main**  **Functions**  Some programming languages distinguish between functions and procedures and write them in a slightly different way but Python has just one type of subroutine, the **function**. Sometimes a function returns a value, and other times it does not.  Discuss the syntax of the function header and the call statement, noting the parentheses.  Give out **Worksheet 3a** and ask students to do **Question 1**.  **Rock paper scissors**  Let students work in pairs to work out what functions are needed in a program to play this game against a computer. Put these steps into a flowchart.  A second exercise asks students to decide on the functions needed in a game of Tic-Tac-Toe. Once again, ask the students to produce a flowchart showing the steps.  Ask the students to complete, in pairs, question 2, which is a manual run-through of the steps they have put in their flowcharts. Are any adjustments necessary?  When they are satisfied that it works, let them load program  **L3 WS3a Ex 2 tic\_tac\_toe.py**, and complete the program by uncommenting the statements in the main program.  An alternative version of the program is also provided for students to try out.  **Conversion tables**  Give out Worksheet 3b and let them read the story of the 1999 Mars probe – lost because of a failure to convert to or from metric units.  The next slide describes how to define and call a function which performs the conversion from inches to cm and prints the result.  **Question 1** on **Worksheet 3b** asks students to write functions which will perform a range of similar conversions of this kind. This is quite a lengthy exercise and will probably take some time.  **Functions**  This slide and the next one shows a function which returns a value; the function is called slightly differently.  **Returning more than one value**  Show how to write the call statement for a function which returns more than one value.  Ask students to try Question 2 in Worksheet 3b.  **Parameters**  Explain what is meant by a parameter – a value passed to a function to be used in some way – and ask students to write the program which calls a function to return and print the square of a number provided as a parameter.  Ask students to write and test the programs for questions 3 and 4 on the worksheet.  **Plenary**  A summary of the various terms used in connection with subroutines is given. Revise these with the students before showing the final slide.  Give out **Homework 3**. | PowerPoint Guide: Lesson 3 Functions  Worksheet 3a  Worksheet 3a Answers  Sample Python programs  L3 WS3a Ex 2 tic\_tac\_toe.py  L3 WS3a Ex 3 tic\_tac\_toe with loop.py  Worksheet 3b  Worksheet 3b Answers  Python Homework 3  Python Homework 3 Answers |

|  |  |
| --- | --- |
| Lesson 4 - Regular expressions | |
| Preparation   * Make sure Python 3.x is installed and running correctly. * Run through the programs mentioned in the PP Guide. Many of them can be found in the Sample Python Programs folder   Learning Objectives:   * Review the purpose of validation, one of the programming techniques that may be helpful to complete the controlled assessment tasks * Understand the purpose of a regular expression * Be able to use a regular expression to validate an input | |
| Content | Resources |
| **Starter**  The aim of the lesson is to match up patterns in a way that is very similar to using an input mask in an Access database.  Recap the purpose of validation (to ensure that inputs that are clearly wrong can be spotted early – e.g. if someone mistypes and claims they are 200 years old instead of 20 or confuses a zero with a letter ‘o’ in a postcode).  **Main**  Ask students to come up with other data input that has a pattern – e.g. a car registration, a name must begin with an uppercase letter, an email address must contain @.  Explain that regular expressions are a way of validating that an input matches a given pattern – just like an input mask in a database.  Students should type up and run the first program. Read through the explanatory notes in order to identify how a basic regular expression works (importing the library and then matching against the pattern in speech marks).  Students should then extend the program to ask for a town name. The only change here is in the wording of the prompt displayed in the input statement, and possibly in the naming of the variable used to store the town rather than a name.  The correct program should have a line something like: valid = re.match(“[A-Z]”,homeTown)  *Tasks from the slideshow are also included on the worksheet provided.*  **Upper Case, Lower Case and Numbers**  Introduce the next example program, which checks that the answer starts with a lower case letter “[a-z]” rather than an upper case letter “[A-Z]”. It is also possible to check for a number using “[0-9]”.  Students should now be able to create a program (or edit a previous program) to ask for a phone number and validate that it starts with a digit.  The correct program should have a line something like: valid = re.match(“[0-9]”,phoneNumber)  By this point it may become clear that the structure of a validation check remains very much the same – only the input prompt and the pattern to be matched change significantly.  Ask students to complete **Tasks 1** and **2** of **Worksheet 4.**  **Matching postcodes**  Ask students to identify the rules that make up a postcode. Discuss the nature of the patterns used in postcodes.  Explain that a good regular expression will look at more than just the first character. Demonstrate using the next program that students can use three codes together to match a more detailed pattern (e.g. [A-Z][A-Z][0-9] will produce a positive match with AB1, DJ7, etc. but NOT with M62.  Students will often add spaces in between each code, but the regular expression match will treat this literally and the pattern  ‘[A-Z] [A-Z] [0-9]’ will only match positively with ‘A B 1’, etc.  Explain that it is also possible to match specific letters (for example most motorways will match the pattern ‘M[0-9]’ and this is slightly more effective than using ‘[A-Z][0-9]’ which would accept an input of ‘N62’ which contains a typographical error).  **Multiple Characters**  The next thing to add is that a ‘+’ symbol in a regular expression pattern standards for “one or more”. Using this, ‘[A-Z]+’ means “one or more upper case letters”. This allows some more flexibility in terms of spotting postcodes that can have one or two letters at the start.  It is important to make it clear that the ‘+’ refers to the character or code immediately before it.  **Question 3** of the worksheet asks the students to extend a regular expression check for the first three characters of a typical UK postcode. A “crib sheet” is given above the question and they can use this to form a new regular expression.  **Question 4** goes one step further for a full validation of a postcode. They could use the expression  "[A-Z]{1,2}\d{1,2}[A-Z]?\s\d[A-Z][A-Z]"  or a variation.  You will find Python programs for these exercises in the **Sample Programs** folder.  **Plenary**  To check students’ understanding of regular expression codes, ask them to identify which values would pass or fail the validation check  “[A-Z]+[0-9]+ [0-9][A-Z]+”  The correct responses are provided in the slideshow with explanations for each test that will fail.  Using the first section of the postcode example [A-Z]+[0-9]+, this would allow ABC123 as well as other valid postcodes, but [A-Z]{1,2}[0-9]{1,2} will restrict the number of entries to at least one letter or number and a maximum of two. There are many more rules that some students may enjoy trying using **Link Regular expression operations**.  **Homework**  Hand out **Homework 4**. | PowerPoint Guide: Lesson 4 Regular expressions  Worksheet 4  Worksheet 4 Answers  Sample Python programs  Link Regular expression operations  Homework 4  Homework 4 Answers |

|  |  |
| --- | --- |
| Lesson 5 - Using lists |  |
| Preparation   * Make sure Python 3.x is installed and running correctly. * Run through the programs mentioned in the PP Guide. Many of them can be found in the Sample Python Programs folder   Learning Objectives:   * Understand why lists and arrays are useful * Be able to read and edit data in a list * Know how to declare and append to a list | |
| Content | Resources |
| **Starter**  Ask students how many variables would be needed to hold eight values.  They may spot that this is a trick question, but using techniques excluding lists (very similar to arrays in other languages) there is no easy way to use fewer than 8 variables.  Explain that a list is just what it sounds like – a list of values. Analogies might include numbered bullet points, or a chest of drawers.  List items are indexed (addressed) using numbers in square brackets (students must not leave a space between the list name and the index). It is important to point out that the numbering starts with 0 (just as it does with FOR loops). One analogy that might help is that we count storeys in this way – with Ground, 1st floor, 2nd floor, etc.  **Lists**  Students should copy and test a simple program to create a list of hard-coded values (where the values are written into the code and not input by the user). Using a simple print statement to print the entire list in one go will produce an output in square brackets, with each value separated by a comma.  Explain how to address an individual list item using its index and demonstrate how to change a single value – printing the whole list before and after in order to demonstrate how it has changed.  Demonstrate that students can create an empty list by putting a single value into square brackets and using the ‘\*’ operator to create the required number of spaces (or drawers, or bullet points).  **Worksheet 5a**  Students should now be able to complete **Worksheet 5a**,**Task 1**  They may want to refer back to the slideshow in order to help them with the tasks – providing a digital copy on a VLE or similar may prove helpful.  There is a document called Python Reference List Handling in the root folder which you could give to the students to help them with their programs. It has sample programs to create and update: a list, append values, and sort.  **Appending a new element to a list**  Ask students to add a new value to the end of a list (e.g. in a list of 4 names [positions 0, 1 2 and 3], add a new value in position 4). The program will fail because there is no position 4.  Instead, explain that if a student doesn’t know exactly how big their list will be at the start they can use the code  listName.append(‘new value’) to add a new value onto the end of a list – whatever size it is.  **List methods**  Explain to students that there are various other methods that can be used to manipulate values in a list such as **pop()**, **append()**, **insert()** and **remove()**. Ask pupils to figure out the printed word in the following slide based on the list methods. A trace table is shown below of the output at each line:   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **List index** | **0** | **1** | **2** | **3** | **4** | **5** | | word =["c","b","e","g","h","d"] | c | b | e | g | h | d | | word[0] = "e" | e | b | e | g | h | d | | word.pop(2) | e | b | g | h | d |  | | word.remove ("g") | e | b | h | d |  |  | | word.insert(0,"z") | z | e | b | h | d |  | | word.pop(3) | z | e | b | d |  |  | | word.pop() | z | e | b |  |  |  | | word.insert(3,"r") | z | e | b | r |  |  | | word.append("a") | z | e | b | r | a |  | | print(word) | z | e | b | r | a |  |   Ask students to complete **Task 2** on **Worksheet 5a** where they can use a blank trace table to work through a different example, similar to the ‘Zebra’ one above. A table of method actions is also given to help students with the task. This would also be useful for them in their programming as a handy reference.  **Stepping Through Lists**  Remind students that in a simple FOR loop it is possible to print out the value of the counter. Using the loop counter variable as the index value it is possible to step through each value in a list, one at a time and print out the value. Note that in the statement  for loop in range(4):  the range goes from 0 to 3 – not from 1 to 4 as you might expect.  It is important for students to understand how this process works, although there is a slightly more straightforward method to search within a list using the code:  if "Paul" in friends:  print("Found him")  This won’t necessarily identify if there are two identical values, although it is possible to find out exactly which position a value is in using the code friends.index("Paul") - but only if the value is definitely in the list – and so the two blocks of code should usually be used together.  **Worksheet 5b**  Ask students to complete **Worksheet 5b**.  Again, access to the slideshow will be helpful for the students. A solution to the extension task on 2D lists is available in the Sample Programs folder with comments to help describe the code L5 WS5b Extension.py.  **Homework**  A homework sheet is provided to help reinforce the students’ understanding. | PowerPoint Guide: Lesson 5 Using lists  Worksheet 5a  Worksheet 5a Answers  Python Reference List Handling.docx (in root folder)  Worksheet 5b  Worksheet 5b Answers  L5 WS5b Extension.py  Homework 5 Using lists  Homework 5 Using lists Answers |

|  |  |
| --- | --- |
| Lesson 6 - Sorting lists |  |
| Preparation   * Make sure Python 3.x is installed and running correctly. * Run through the programs mentioned in the PP Guide. Many of them can be found in the Sample Python Programs folder   Learning Objectives:   * Understand why you might want to sort a list * Know how to sort a list using Python * Be able to use other functions with lists | |
| Content | Resources |
| **Starter**  Ask students to sort the list of famous people in order by the name of their product and again by their net worth.  Once this is complete, challenge students to explain why this is a useful skill to be able to include in a programmer’s toolkit.  While sorting a small list of values is simple to do manually, a large dataset is much more difficult to sort by hand.  **Main**  **Sorting simple lists**  Demonstrate a simple program to sort a **list** of numbers using the ‘sorted()’ function. The resulting list can either be stored as a new variable or overwriting the original list. Demonstrate how to sort in reverse order by adding an extra parameter to the sorted function:  sortedValues = sorted(values,reverse=True)  Further, show that a list of strings can be sorted in exactly the same way. Note that a list containing both numbers and strings cannot be sorted.  **Worksheet 6**  Students should now be able to complete the questions in **Worksheet 6, Task 1**  They may want to refer back to the slideshow in order to help them with the tasks – providing a digital copy on a VLE or similar may prove helpful.  **Other Functions**  Demonstrate that, as with a spreadsheet, Python includes built-in functions to find the largest and smallest values in a list using ‘max()’ and ‘min()’. These functions will work with lists of numbers, but also with lists of strings (finding the first and last values alphabetically).  Two other useful functions are sum() – which works only with lists of numbers to add all of their values together – and len() – which will provide the length of (or number of values in) a given list.  Using these functions it is possible to calculate the mean average from a list of numbers.  Ask students to complete the questions in **Task 2** of **Worksheet 6**.  Access to the slideshow may be helpful for the students. A list of all the functions covered has been included in both tasks of the worksheet for reference.  If the questions in Task 2 are too difficult for the students you could alter the questions by providing the programs with some syntax/and or logic errors and ask the students to find them. Or give them the programs and ask them to run them, then insert a few errors and get a partner to find them.  Programmed solutions to Question 6 and the extension task are available in the Sample Programs folder with comments to help describe the code. The extension exercise is far from trivial and goes beyond the scope of the syllabus.    **Homework**  Give out **Homework 6**. | PowerPoint Guide: Lesson 6 Sorting lists  Worksheet 6 Sorting lists  Worksheet 6 Answers  Sample Python programs  Homework 6 Sorting lists  Homework 6 Answers |

|  |  |
| --- | --- |
| Lesson 7 - Reading from a file |  |
| Preparation   * Make sure Python 3.x is installed and running correctly. * Run through the programs mentioned in the PP Guide. Many of them can be found in the Sample Python Programs folder   Learning Objectives:   * Understand how to read data from a file * Be able to read data from a file one line at a time * Know how to interrogate data | |
| Content | Resources |
| **Starter**  Ask students to complete the table (which will be used to practise file handling techniques during the lesson). The aim Is to make sure students are familiar with the contents of the file and can recognise the relationship between the table and the comma separated value (CSV) data file.  **Reading Files**  Discuss the 3 methods of data collection with the students and identify pros and cons for each (data entry by a user, reading data from a sensor and reading data from a file).  Have students copy and test a simple program to open a file, read the entire contents with “.read()” and display the contents. The file **people.txt** can be opened and viewed in Notepad – it contains 9 records shown in Worksheet 7a and below.  C:\Users\Pat\AppData\Roaming\PixelMetrics\CaptureWiz\Temp\2.jpg  A second version, people2.txt, containing an extra blank field at the end of each record, will be used later in this lesson.  Run through some of the key bits of syntax – such as the “r” parameter in the ‘open()’ function that opens the file in ‘read’ mode. A common error is to forget the speech marks around the “r”.  Explain the importance of having the data file in the same folder as the Python program (it is possible to use a path to a file in a different folder, but this is unnecessarily complicated for this level).  Closing a file once it is finished with is a good habit, especially once students start working on more complex programs with multiple files, but be aware that programs that forget to close the file may not appear to cause any problems initially.  The students should now be able to adapt their existing program to open the second data file – **albums.txt**  The correct program should have a line something like: file = open(“albums.txt”, “r”)  **Reading one line at a time**  Demonstrate how to use the readline() function to read one line at a time. Note that readline(3) doesn’t read the third line, but reads the first 3 characters from the current line. Because Python keeps count of which line it is up to, the best way to get to line 3 would be something similar to:  file.readline() file.readline() line = file.readline()  **Worksheet 7a**  Ask students to complete Worksheet 7a. The file **people.txt** has been included with prepopulated data.  *Expect this section of the “lesson” to take around 1 hour of contact time. It may be necessary to carry out further practice tasks in order to gain sufficient confidence and experience.*  *The second part of the lesson relies heavily on students’ confidence and experience with lists.*  **Splitting and Interrogating Data**  Demonstrate and have students test appropriate code that will take a line from a file and split the data into a list based on any delimiter. This is not entirely dissimilar from importing data into a spreadsheet or database – though the code at this point is becoming a little longer and it may be easy for students to start to gloss over the key functionality of the program.  A simple task in the slideshow is to read the surname. This should reinforce some of the understanding of how to read the resulting list generated by the split() function.  The correct program should have a line something like: print(data[1])  **Interrogating data in a file**  **Problem!**  A complication in splitting data records into fields now becomes apparent. The last field in each record contains an invisible newline character, held internally as **\n**. Therefore, if you look for records which contain “Musician” in the last field, for example, none will be found because the field actually contains “Musician\n”  Provide students with the last sample of code. This program should loop through all 9 lines of the people.txt data file and, within each loop should read the line, split the line into a list and then check if the 3rd value (index [2]) is “Musician”. If it is then the first and surname should be printed (data[0] and data[1]). This is effectively a simple query. Unfortunately, for the reason given above, it will not find any records!  ***There are two solutions:***  (a) look for “Musician\n”  (b) When the file is created, add an extra comma to the end of each record, before the newline character, so that it is in a field of its own and is not part of the last data field. The file **people2.txt** has been amended to contain the extra comma separating the fields from the newline character.  Python programs **L7 interrogating data in a file v1.py** and **L7 interrogating data in a file v2.py** show these two techniques, using data files **people.txt** and **people2.txt**.  Students should now be able to complete Task 1, question 1 of **Worksheet 7b**. This task requires students to annotate some Python code in order to demonstrate their understanding. This could be done either in the table provided, or by printing the last page of worksheet 2 and having students annotate by hand in a more freeform structure.  **Question 2 of Task 1** takes students through a number of practice exercises in order to improve their programming confidence and experience.  **Extension**  Task 2 question 3 on the worksheet shows how to loop directly through the lines of a file using a FOR loop, to print all the records.  Question 4 shows how to read a text file into a list using the readlines() method.  Each line of the file becomes a string element of the list. If you don’t know how many records there are in the text file, you can then get the length of the list using the len() function, which tells you how many records there are.  You can then close the file and reopen it at the beginning and process as normal, knowing the length of the list (now held in a variable e.g. numberOfRecs).  **Homework**  A homework sheet is provided to help reinforce the students’ understanding. | PowerPoint Guide: Lesson 7 Reading from a file  people.txt  albums.txt  Worksheet 7a Reading from a file  Worksheet 7a Answers  people.txt  Worksheet 7b Interrogating a file  Worksheet 7b Answers  L7 interrogating data in a file v1.py  L7 interrogating data in a file v2.py  people.txt  people2.txt  L7 WS7b Task 1 Albums.py  L7 WS7b Task 2 Reading a text file.py  Homework 7 Reading from a file  Homework 7 Answers |

|  |  |
| --- | --- |
| Lesson 8 - Writing to a file |  |
| Preparation   * Make sure Python 3.x is installed and running correctly. * Run through the programs mentioned in the PP Guide.   Learning Objectives:   * Be able to write data to a file * Understand what “append” means * Know how to append data to a file | |
| Content | Resources |
| **Starter**  Ask students to recall the syntax for opening a file in read mode and ask them to guess what they might need to do to open a file in write mode. Read mode is something like:  file = open("filename.txt","r")  Write mode is something like:  file = open("filename.txt","w")  **Writing a new file**  Show students the example program on the slide that will ask for their name, open a file in write mode and write the data to a file. Students should copy and test the program – they should find a new file in the same folder as the program that contains the data they entered.  Students should try to guess what will happen if they run the same program again and type in a different value when prompted. They will probably expect the new data to be added to the file, but opening a file in "w" (write) mode clears any content already in the file – so when they check the file they will only see the new data.  Explain to students that it is the open() function that clears the file – so once it is open they can enter data as often as they like.  **Writing several records**  Show students the extended version of the program that asks for two names and writes them both to the file. Again, students should guess and then test how the program works. They will probably expect that each write() function will write the new data on one line, without any line breaks. This might be a familiar situation if you have tried to demonstrate HTML without using <p> or <br> tags.  **The newline character**  If you haven’t introduced it previously, explain to students how to use the "\n" character to add a newline to a string.  Note that, by using the following alternative line of code, the students can add the contents of a variable to a file and move to a new line ready for the next piece of data:  file.write(variable + "\n")  Note: If you want to have the newline character in a field of its own, with a comma at the end of each line, the code would be  file.write(variable + ",\n")  or  file.write(variable + ",” + “\n")  Doing this will mean that the last data field does not have \n as the last character.  See programs **L8 WS8 Ex4 write classmates data v1.py** and  **L8 WS8 Ex4 write classmates data v2.py** in Sample programs folder  **Worksheet 8 Task 1**  Students can now be asked to complete some or all of the questions in Worksheet 8, Task 1.  There is a reference sheet called “Python Reference File Handling” root folder, which you could give out to students, or which they could copy into their own areas. They may find it useful when doing the practical tasks.  **Appending data to a file**  Based on the dangers of overwriting files by opening in write mode, explain that the append mode is the solution if you are adding data to an existing file.  Provide students with some sample code similar to opening a file in write mode, but using an "a" parameter to indicate append mode.  Encourage students to test this out, still using the "\n" newline character to add new data to an existing file.  **Worksheet 8 Task 2**  Students should now be able to complete some or all of the questions in Task 2 of the worksheet.  **Plenary**  Discuss any problems encountered. Review the three modes Read, Write, Append.  Note that it is not easy to edit existing data within a file, this requires the programmer to extract the data, alter it and prepare it for re-writing to a fresh version of the file. There are several Python modules that can make this process more straightforward. File handling at this level is not covered in this unit, although capable students may well be able to tackle this as a stretch and challenge exercise.  Students may wish to combine their knowledge that data can be read into a program using CSV (comma separated value) format with their new knowledge of writing data to files and write data using commas in order to make that data more usable in the future.  **Homework**  Give out **Homework 8**. | PowerPoint Guide: Lesson 8 Preparation – Writing to a file  Worksheet 8 Writing to a file  Worksheet 8 Answers  Python Reference File Handling.docx  L8 WS8 Ex4 write classmates data v1.py  L8 WS8 Ex4 write classmates data v2.py  L8 WS8 Ex5 read and write classmates.py  L8 WS8 Ex11.py  L8 WS8 Ex12.py  Homework 8 Writing to a file  Homework 8 Answers |

|  |  |
| --- | --- |
| Lesson 9 – 2D Lists | |
| Learning Objectives:  Preparation   * Make sure Python 3.x is installed and running correctly. * Run through the programs mentioned in the PP Guide.   Learning Objectives:   * Understand the nature of a 2D list * Be able to use a 2D list to solve a problem | |
| Content | Resources |
| **Starter**  Data about rooms in a hotel could be held in a table, or two-dimensional list, with rows representing floors (starting at 0) and columns representing rooms.  **Main**  Compare one- and two-dimensional lists. Show how each element in a 2-D list is referenced with [row][column] indices.  Ask **students to do Question 1 on Worksheet 9a** which gives practice at referencing individual elements in a 2-D list.  **Creating a 2-D list**  Demonstrate how a 2-D list can be defined in a Python program. It is basically a “list of lists”  You cannot insert an extra element into an existing list, but you can use the **append** method to add a new element to the end of a list.  Ask students to do Questions 2-5 on the worksheet.  **Finding the size of a list**  It is possible to find the number of rows or columns in a 2-D list. The number of rows is the number of sublists in the 2-D list.  Hand out **Worksheet 9b** and ask students to do **Question 1.**  **Stepping through a list**  Go through these slides and then ask students to try Questions 2-and 3 on the worksheet.  Demonstrate how to find a running total of a given column in a 2-D list.  Students can now do questions 4 and 5 on the worksheet.  **Sorting a 2-D list**  A 2-D list can be sorted on any particular column. One way of doing this is to use a bubble sort, and this is shown on the slides.  A simpler method is to use the lambda function, shown next.  Ask students to do the three questions in Worksheet 9c.    **Plenary**  Go over the answers to the first three questions. Some students may have time to try writing a program for the extension task.  Make sure that all students know how to refer to an individual element in a 2-D list, and can give the length of the list (the number of columns).  Give out **Homework 9**. | PowerPoint Guide: Lesson 9 two-dimensional lists  Worksheet 9a  Worksheet 9a Answers  Sample Python programs  Homework 9  Worksheet 9b  Worksheet 9b Answers  Worksheet 9c  Worksheet 9c Answers  Homework 9  Homework 9 Answers |

|  |  |
| --- | --- |
| Lesson 10 – Programming techniques | |
| Learning Objectives:   * Use programming ’set pieces’ for common problems   + Use a ‘flag’   + Use a while loop to validate data entry   + Create a menu system | |
| Content | Resources |
| **Starter**  Do a quick warm-up, revising lists and how to sort them, recapping the previous lesson.  **Main**  **Flag**  A flag is a variable used to indicate that something has happened or that some condition is True or False. The variable can then be tested in a conditional statement and appropriate action taken.  Give out Worksheet 10a and ask students to do Questions 1, 2 and 3. Give assistance where required and go over answers as a class exercise.  **The break statement**  The next slide shows the use of the **break** statement.  This can be a useful way of terminating a loop if some condition has been met. In general it is not considered good practice in structured programming - the same effect can always be achieved by writing an appropriate **while** loop.  Students should be encouraged to write code which not only works correctly, but which does the job efficiently and is easy to follow.  **Validation**  Validation of user input is essential, and is used to ensure that the user enters something reasonable and that the program will check for anything which will cause the program to crash, for example an alphabetic character when a numerical value is expected. Validation normally involves a loop which repeatedly asks the user to re-enter a value if they make an invalid entry, until they enter an acceptable value or run out of attempts.  Set Worksheet 10b, which has several questions giving practice at writing validation routines. Notice the technique in the answer to Question 4, of converting the user’s answer to lowercase before checking it, which ensures that whatever case individual letters are typed in, a correct answer will be accepted.  **Menu system**  Many programs which offer different options will have some kind of menu system, with each option being written as a separate function. This is a good way of structuring a program.  **Worksheet 10c** has practical exercises which give students practice at developing, structuring and coding programs in this way.  **Plenary**  Recap the useful techniques covered in this lesson.  Give out **Homework 10**. | PowerPoint Guide: Lesson 10 Programming techniques  Worksheet 10a  Worksheet 10a Answers  Sample Python programs  Worksheet 10b  Worksheet 10b Answers  Worksheet 10c  Worksheet 10c Answers  Homework 10  Homework 10 Answers |

|  |  |
| --- | --- |
| Practical problems | |
| Learning Objectives:   * Review the programming techniques that may be helpful to complete the controlled assessment tasks | |
| Content | Resources |
| The document **Practical Problems - Introduction.doc** describes five longer programming projects which students could tackle in advance of any practical task required to be completed as part of their course.  Folders for each of these tasks show how a program can be developed in stages, with exemplar solutions for each task being provided. | Practical problems |

Acknowledgements

The authors and publisher would like to thank all contributors for their kind permission to reproduce any copyright material in the PowerPoint Guides.

All images unless otherwise stated © Shutterstock.com

Every effort has been made to trace the copyright holders and we apologise in advance for any unintentional omissions. We would be pleased to insert the appropriate acknowledgement in any subsequent edition of this publication.

This material may contain links to relevant websites featured in the teacher’s guide. Every effort has been made to ensure that at the time of distribution, the links remain unbroken, the material remains up-to-date and that links are not inadvertently linked to sites that could be considered offensive. PG Online cannot be held responsible for the content of any website mentioned in this material. It is sometimes possible to find relocated sites by typing the original URL into a browser. Any errors should be reported directly to support@pgonline.co.uk and changes will be made in any subsequent editions of the material.

Artwork



Fosse No. 3 © 2015 Barbara Burns

Oil on linen, 30x30cm

www.slaneart.com