

|  |
| --- |
| Teaching guide |
| Python: Next steps |
| pg_online_normal_ver-02 |

Copyright

© 2017 PG Online Limited

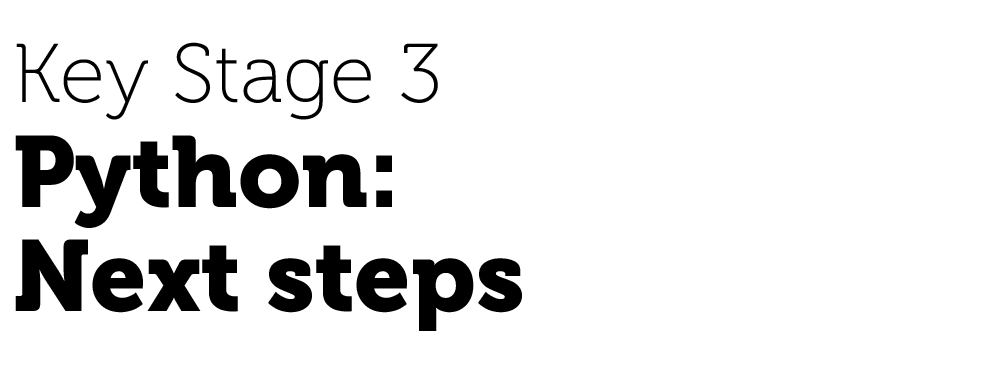
The contents of this unit are protected by copyright.

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. 

(Suggested for teaching in Year 9)

Teacher’s Guide

Introduction

This teacher’s guide contains detailed lesson plans to accompany the five sets of PowerPoint slides and worksheets.

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:

* 5 PowerPoint presentations, each designed to cover one lesson
* 8 worksheets, many with example responses
* 5 homeworks
* An end-of-unit learning portfolio for assessment purposes

### *Summary*

This unit assumes that pupils already have some prior experience in Python or a similar language, and the first lesson has a series of tasks designed to revisit the basic skills already covered. Pupils then use **for** loops and compare their use with **while** loops, before moving on to arrays (lists), which are introduced as a new data structure and are used in conjunction with **for** loops. Functions with and without parameters are covered to help pupils understand the concept and benefits of modular programming. This unit is designed to take pupils right up to a point where a GCSE in Computing can pick up from and should provide ample experience of programming in order to confirm any decision to pursue Computing as a GCSE option.

### *New Attainment Targets (partially covered in this Unit)*

* Use two or more programming languages, one of which is textual, to solve a variety of computational problems; make appropriate use of data structures; design and develop modular programs that use procedures and functions
* Understand several key algorithms that reflect computational thinking (for example, ones for sorting and searching); use logical reasoning to compare the utility of alternative algorithms for the same problem.

### *Learning Outcomes for the unit*

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

* Use data types correctly and convert between them when necessary
* Write programs that use a loop to repeat a section of code
* Write programs that use lists (known as ‘arrays’ in some languages)
* Create and use a function with or without parameters
* Find and debug syntax errors
* Look at a given section of code and describe its function

**Most pupils will be able to:**

* Select the most suitable type of loop (**for** or **while**) for a given problem
* Use counters correctly in conjunction with **for** loops
* Create a list and append or change elements of the list
* Explain the advantages of functions for reusable sections of program code

**Some pupils will be able to:**

* Use loops to populate, interrogate and print lists, using a counter as an index to an array element
* Devise their own functions to create a modular program
* Create a program that is easy to use, caters for user input errors,has explicit error messages telling the user what the correct form of entry is and produces output with suitable headings or explanation

### *Previous Learning*

Pupils are expected to have had some previous experience of Python programming, including basic inputs and outputs, converting inputs to numbers using the **int()** function, selection (**if**-**elif**-**else** statements) and **while** loops. This unit follows on from the PGOnline unit ‘Introduction to Python’.

### *Suggested Resources:*

Python 3.3 or later version (free)

Worksheets 1 to 8

Solutions to exercises

### *Vocabulary*

Vocabulary associated with programming and particularly Python, such as:

Integrated development, IDLE, variable, string, assignment statement, data type, casting, integer, float, round, BIDMAS, selection, iteration, loop, syntax error, logic error, debug, list, array, index, procedure, function, call, argument, parameter, return value, modular program, dry run.

### *Preparation and common errors or misconceptions*

Make sure Python 3.3.x or higher is installed and running correctly. Run through the programs mentioned in the PowerPoint Guides.

### *Assessment*

Pupils will answer some questions relating to a short Python program, and write some pseudocode for a short program. They will write and run a program and submit the code and screenshots of the program running in a learning Portfolio.

The assessment describes grades as Basic, Intermediate, Advanced or Expert. It is expected that teachers will map these onto their own school assessment structure for Computing and ICT.

Please note that the Assessment Portfolio is protected in Word to enable form-filling. If you wish to edit it, you need to remove protection as follows: (There is no password.)

* Show the **Developer** tab in the ribbon using **File** > **Options**
* Click on **Restrict** **Editing**
* Click the **Stop Protection** button at the bottom of the **Restrict Formatting** **and Editing** panel.

To reapply protection:

* Click on **Developer** > **Restrict** **Editing**
* Ensure that the **Editing Restriction** check box is checked in the **Restrict Formatting** **and Editing** panel
* Select **Filling in forms** from the dropdown menu
* Click **Yes, Start Enforcing Protection**
* It is advisable not to set a password in case you forget it!

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 pupils a mountain of indigestible syntax rules, and to enable a teacher unfamiliar with Python to help pupils with problems they encounter when using the worksheets.

The pupil worksheets accompanying these notes will need minimal introduction, making the lesson largely a process of the pupils 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 the Python 3 Windows Installer. 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 | The basics |  |
| Preparation   * Make sure Python 3.3.2 (or the latest version) is installed and running correctly. Run through the programs mentioned in the PP Guide.   Learning Objectives:   * Correctly read and understand an existing Python program * Recall different data types * Use the **int()**, **float()** and **str()** functions to convert data types * Write an if-else statement | | |
| Content | | Resources |
| **Starter**  Give the pupils a minute or two to describe what the first program in the PowerPoint slideshow does.  **Main Activity**  Once pupils have attempted this, run through each section explaining and reminding them of the core concepts.  **Section 1**  Variables (objects used to store values) are initialised.  **Section 2**  This section repeats as long as the guess is not equal (!=) to the value of the correct password variable. Within the loop the user is asked to guess the password and the *guesses* variable counts up by 1 each time.  At the end of the loop (i.e. once the password is guessed correctly) the success message is printed.  **Section 3**  A **print** function is used to print out how many guesses were needed. The **if** statement allows for the singular ‘guess’ to be printed instead of ‘guesses’. Because the plural version uses the value of the *guesses* variable it is necessary to cast this as a string in order to successfully print it.  **Data Types & Casting**  “Casting” is the technical term for converting a variable from one data type to another. Remind pupils of the need for different variable types. Pupils should carry out the example code and see how adding two strings produces a different result from adding two numbers.  Pupils should then try converting the string value to a number to show how this solves the problem of dealing with numbers that may be stored as strings. You can also convert decimal numbers to integers, and convert numbers to strings in order to print them out.  Pupils should then be in a position to complete the Data Types **Worksheet 1a: Data Types**.  Task 1 reinforces the three main data types.  “Float” can be described as “A number with a decimal point”  “A piece of text” is a data type “string”  The solution for Task 2 can be found in the file **bill.py**.  The source code and solution for Task 3 are in **age.py**and**ageFixed.py**.  **If Statements**  Ask pupils to raise their hands based on the questions in the slideshow – this helps to reinforce the logical structure behind **if** statements. Compare this natural language version to the Python syntax, reminding pupils of the inequalities (<, >, ==, !=, <=, >=), indentation and colons.  Pupils should now be able to complete **Worksheet 1b: If statements**.  Task 1 reminds pupils of the 6 inequalities.  The solution for Task 2 can be found in the file **cash.py*.***  A sample solution for Task 3 can be found in the file **insult.py**.  Give out **Homework 1**. | | PowerPoint Guide: Python Next Steps L1 The Basics  password.py  Python Next Steps Worksheet 1a Data Types.docx  bill.py  age.py  ageFixed.py    Python Next Steps Worksheet 1b If statements.docx  cash.py  insult.py  Python Next Steps Homework 1  Python Next Steps Homework 1 Answers |

|  |  |  |  |
| --- | --- | --- | --- |
| Lesson 2 | Loops |  | |
| Learning Objectives:   * Use a **while** loop to repeat a section of code * Use a **for** loop to repeat a section of code * Make a choice about which loop to use, and why | | | |
| Content | | Resources | |
| **Starter**  Ask the pupils what the program will print the first time the **print** statement in the **while** loop is executed (“This line of code has been run 1 times”).  Ask the pupils how many times the print statement will be executed (9 times – it will stop when counter = 10).  **Main activity**  **While loops**  Explain that **while** loops work like **if** statements that repeat. Explain how the natural language version of a while loop can be written in Python code. Pupils should open and complete the **guessGame.py** program, adding the 3 lines from the slideshow. The solution is in the file **guessGameFinished.py**.  **For loops**  Explain that **for** loops are better if you know how many times the loop will repeat. Pupils should try the sample code and note that it is shorter and simpler than an equivalent **while** loop (no need to manually count each step).  Pupils should try writing **for** loops that repeat a different number of times and, as an extension, demonstrate a **for** loop wth two arguments that will count from (in the example) 5 to 9.  **Worksheet 2a: Pocket Money**  Worksheet 2a will take pupils through a program that uses a **for** loop within the context of getting a good deal out of pocket money. The starting code and an example solution are available in the files **pocketMoney.py**and **pocketMoneyComplete.py**.  **Worksheet 2b: Loops**  Worksheet 2b focuses more on the theory and understanding of loops and asks pupils to complete a series of short-answer questions. Solutions are included.  Give out **Homework 2**. | | PowerPoint Guide: Python Next Steps L2 Loops  whileExample.py  guessGame.py  guessGameFinished.py  forExample.py  forExampleComplete.py  Python Next Steps Worksheet 2a Pocket Money.docx  pocketMoney.py  pocketMoneyComplete.py  Python Next Steps Worksheet 2b Loops.docx  Worksheet 2b Loops Answers.docx  Python Next Steps Homework 2  Python Next Steps Homework 2 Answers | |
| Lesson 3 | Lists | |  |
| Learning Objectives:   * Be able to store and update values in a list * Be able to append data to a list * Be able to use a **for()** loop to step through a list * Understand why using a list can be more efficient than using single variables | | | |
| Content | | | Resources |
| **Starter**  Ask pupils to identify why the sample code is inefficient – the reason is that there are lots and lots of variables and to print out the entire list requires many lines of code.  **Main activity**  **(Introducing the idea of a list, which uses a single variable name and an index to refer to individual elements of the list.)**  It would be simpler to have one list of values than can be looped through.  **Variables**  Explain the thinking above to pupils and have them try out the sample code. The first print statement will print the entire list, the second will print out the first value and the next line will print the second variable.  Make the point that lists (also known as arrays) start counting from 0.  The number in the square brackets is referred to as the **index**.  Pupils should try printing the other two values from the list – **highScore[2]** and **highScore[3]**.  **Updating Lists**  Have pupils try the sample code – which will overwrite the first value (125) using the number 127. Explain that, just as the individual list element can be printed, it can be altered. The pupils should try to update all four values using the numbers provided on the slide.  **Appending Lists**  Pupils should try the sample code, but it will give an error because there is no **highscore[4]** (remember that the index starts from 0).  The next sample code shows how to use the **append()** function to add a new value to the list.  **Worksheet 3a: Superheroes**  Pupils should follow the instructions to manage a list of superheroes. The coded solution is included.  **Using Loops**  Pupils should try the sample code which uses a **for** loop (because we know how many times the loop will need to run) to run through and print each value individually. (Trivia: The image shows the super villain ‘Venom’ from the Marvel Spiderman series.)  Run through the slides to demonstrate how the loop repeats, using a different value for the counter each time.  Pupils can now complete **Worksheet 3b: Super villains** which gives them opportunities to try using a **for** loop to interrogate a database of villains. A coded solution is provided.  Give out **Homework 3**. | | | PowerPoint Guide Python Next Steps L3 Lists  listExample.py  Python Next Steps Worksheet 3a Superheroes.docx  Superheroes Complete.py  listLoop.py    Python Next Steps Worksheet 3b Supervillains.docx  Supervillains Complete.py  Python Next Steps Homework 3  Python Next Steps Homework 3 Answers |

|  |  |  |
| --- | --- | --- |
| Lesson 4 | Introducing functions |  |
| Learning Objectives:   * Understand what a procedure is * Be able to define and call a procedure * Understand why procedures are useful * Be able to use parameters in a procedure | | |
| Content | | Resources |
| **Starter**  Pupils should write out an algorithm for tying shoelaces. The exact detail of the response is not the key here, it is about the fact that we say “tie your shoes” rather than running through the entire algorithm. The same is true of tying a school tie or brushing your teeth.  **Main activity**  **(Defining and calling a function)**  Explain that a procedure (called a **function** in Python) is a way of giving one instruction that actually leads to a longer set of instructions in order to save time.  **bbc.py**  Pupils should open the **bbc.py** file which displays a BBC logo and then gives the program information for a BBC program. The logo contains a deliberate error in the letter C (don’t mention it yet and try to pass this off to any eagle-eyed pupils).  A loop would not be an efficient solution here because the information following each copy of the logo is different.  Pupils should follow the instructions to define a function called **bbcLogo()**. Definitions should always go at the top of the program – they must be defined before they can be used or ‘called’.  The BBC Logo code should be pasted into the definition and it must all be indented by one tab.  Once this is done the pupils should call the function by writing **bbcLogo()** wherever the logo appeared originally.  The **bbcImproved.py** file shows what the program should look like.  It is critical that the pupils try this code out to make sure it works before going any further.  Discuss the advantages of using a function to repeat this code. The program is now significantly shorter, some people may consider it easier to read and if you need to change the original logo function then you only need to do it once – not four times over.  Now you can point out the error in the logo and the pupils will only need to fix it once.  **Parameters**  Explain to pupils why parameters are needed in some functions. Lending some money, boiling an egg (3 minutes for soft – 5 minutes for hard) and running a race (100m, 400m, 1500m…) are all examples of the same principle.  Ask the pupils to try the sample code in a new Python program and see how the function **printDouble** doubles whatever value gets passed to it.  **Worksheet 4** tells the pupils to take a Python program **longProgram.py** and improve it using the skills they have been exploring.  Tasks 1 and 2 walk the pupil through replacing the 5 lines that print the program title and time/date – just like the BBC program. The **longProgramStep1.py** file shows what the code should look like.  Task 3 further improves the program by repeating the 5 print lines with the information about that program.  The final extension step involves th use of lists and **for** loops to extract the information for each program. This will be a challenging task for most pupils and is very much an extension activity.  Give out **Homework 4**. | | PowerPoint Guide: Python Next Steps L4 Introducing functions    bbc.py  bbcImproved.py  parametersExample.py  Python Next Steps Worksheet 4  longProgram.py  longProgramStep1.py  longProgramStep2.py  longProgramStep3.py  Python Next Steps Homework 4  Python Next Steps Homework 4 Answers |

|  |  |  |
| --- | --- | --- |
| Lesson 5 | Functions returning values |  |
| Learning Objectives:   * Understand what a function is * Be able to define a function * Be able to call a function and capture the return value | | |
| Content | | Resources |
| **Starter**  Pupils should identify the largest number in each list (104, 29, 583).  **Main activity**  **Functions**  Briefly recap the idea that a function is a piece of code that you can reuse – with an emphasis on doing something.  Explain that a function can return an answer of some kind. Remind pupils of functions such as **int(a)**, which is called by writing, for example **result = int(a)**. Other functions pupils have seen include **input()**, **round()** and **random()**.  Pupils should try the sample code. Explain that the function **calcDouble(amount)** will double whatever value is given to it and will then return that value so that it can be used in the main program.  The function call in the main program captures the value and assigns it to a variable (just like an input statement).  **Defining Functions**  Pupils should add two more functions – **calcHalf(amount)** and **calcTenMore(amount)**. The exact code for the functions is not provided in the slideshow and the pupils should try adapting the existing function to make their new functions return half the value (**amount/2**) and ten more than the amount (**amount + 10**) respectively.  The pupils should then call each function and print out the result when the value 120 is passed to it.  **Programming Task**  The pupils should look at the **calculator.py** file which is already set up to display a menu and add two numbers (assuming the user enters option 1). They should add three more functions and complete the selection statement **(if.. elif**) to allow the program to fully work.  **Worksheet 5: Card Game**  Pupils should carry out the tasks in the worksheet to create a basic card game. Using a random number generator and passing it to the **chooseASuit(number)** function will allow the program to generate a suit for the card. The function should return the suit back to the main program. Pupils should then use a similar idea to generate the card’s face value and, finally, print it out.  Give out **Homework 5.** | | PowerPoint Guide Python Next Steps L5 Functions returning values  sample.py  sampleExtended.py  calculator.py  Python Next Steps Worksheet 5 Card Game.docx  cards.py  Python Next Steps Homework 5  Pythone Next Steps Homework 5 Answers |

|  |  |  |
| --- | --- | --- |
| Lesson 6 | Assessment |  |
| Learning Objectives:   * Read through a program * Complete the assessment | | |
| Content | | Resources |
| Pupils should complete each section of the assessment to provide evidence of their understanding of the topics covered.  **Reading code**  This section tests the pupils’ ability to understand a piece of code.  **Data structures**  Pupils should be able to identify that “target” and “guess” are integer values.  **Algorithms**  A simple password program could take one of several forms. An example solution is provided, however any functioning algorithm should be judged as correct. There should be a loop that ends when either the password is correctly guessed or after 3 failed attempts.  A solution that uses a **for** loop (for the 3 attempts) but doesn’t automatically end once the password is guessed would be a working, though inefficient, solution.  **Testing**  Pupils should (at a minimum) test   * 3 incorrect guesses in a row * a correct guess   The third test should be 1 or 2 incorrect guesses and then the correct answer.  The final self assessment will help to review the unit. | | Python Next Steps Assessment Portfolio  hiLo.py  password.py |

Acknowledgements

The authors and publisher would like to thank all contributors for their kind permission to reproduce their photographs or images, screenshots of their websites or other copyright material in the PowerPoint Guides.

Spiderman © Fun Fun Photo, Shutterstock.com

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 contains 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.