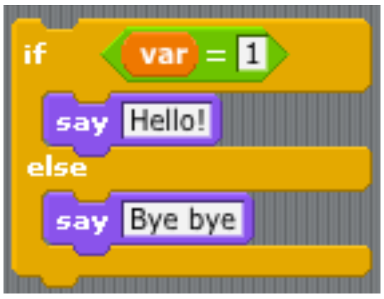
KS2: Python Programming Unit

Jon Chippindall
@drchips_
www.primarycomputing.co.uk

Introduction

This document sets out a scheme of work aimed to introduce upper Key Stage 2 pupils to the Python programming language. The scheme intends to familiarise pupils with the Python programming environment and syntax, and equip pupils with the skills and knowledge to write simple programs.

It is anticipated that pupils will have had prior experience of coding using a visual based programming language, such as Scratch or Kodu, and that this is likely to be the first time they will code using a scripting language. i.e. writing lines of code as opposed to dragging blocks to build algorithms and programs. The example below illustrates the difference between a visual programming language and a scripting language.



```
>>> if var == 1:
        print "Hello!"
else:
        print "Bye bye"
```

An if/else condition block in Scratch and the equivalent coding in Python

Pedagogy

There are four lessons in this scheme of work followed by a final project. Lessons broadly follow a model in which skills and knowledge are taught using worked/modelled examples before pupils tackle a 'Coding Challenge' requiring application of such skills and knowledge. Whilst suggested 'Coding Challenges' have been presented here, I encourage those using this resource to also generate coding challenges for pupils (or encourage pupils to generate their own challenges), which may link into areas of pupils' topic work making the programming more relevant to pupils' wider learning. Similarly, whilst two suggestions have been made for the final project, it is anticipated that teachers using this resources may choose to adapt these for their pupils or encourage pupils to generate their own ideas for the final project programs they wish to code.

Whilst answers to ‘Coding Challenges’ are presented, it should be noted that there will often be different ways to program a successful solution, and pupils should be encouraged to experiment and explore their own methods as opposed to being funnelled towards a predefined solution, since it is the journey of experimentation, trial and error that will facilitate learning.

It is hoped class and school organisation will be such that pupils are given the opportunity to tackle some challenges independently and others cooperatively, helping to develop pupils’ collaborative skills as well as independent perseverance and resilience in problem solving.

Proposed Computing National Curriculum coverage

This scheme aims to cover the following objectives from the proposed Key Stage 2 National Curriculum for Computing. Specific NC objectives appear at the beginning of each lesson.

- design and write programs that accomplish specific goals; solve problems by decomposing them into smaller parts
- use sequence, selection, ~~and repetition*~~ in programs; work with variables and ~~various forms of**~~ input and output; generate appropriate inputs and predicted outputs to test programs
- use logical reasoning to explain how a simple algorithm works and to detect and correct errors in algorithms and programs

* Note this scheme does not cover repetition (loops)
** This scheme only covers one form of input/output (that of the program user entering data via a keyboard)

Lesson Overview

Lesson	Lesson objectives	Vocab
1. Introducing Python	- Navigate Idle (create, save, run programs) - Understand and use mathematical operation and ‘print’ statement	Python, scripting language, visual programming language, syntax, Idle
2. Variables and comments	-Declare a variable -Write comments within Python code -Use mathematical operations and print statement with variables	variables, declare, comment

3. User inputs	- Use raw_input() statement - Use input() statement - Print sentences	Input
4. Selection and inequalities	-Use conditional statements if, else if (elif) and else -Use comparison operators	conditional statement, comparison operator
5. Final project	Reinforcement and application of skills and knowledge covered above	

Vocabulary

A glossary of terms used throughout this SoW is included at the end of this document. Any term appearing in bold in lesson plans appears in the glossary and should be introduced to pupils using the definition provided.

It is suggested that starters including matching words with definitions or code are used to help develop pupils' knowledge of the technical terminology of coding and the syntax of Python.

Code representation and line numbering

Within this document Python code to be written appears in a different font as shown below. Also note that lines of code are numbered for ease of reference within the text however you **do not** include these numbers when coding. Code that would be one long line in Python but which spans several lines when presented here does not start with a new number to indicate this. Finally, please note that Python reads the indentation of code so layout is important (i.e. in the example below 'print' is deliberately indented).

```
1.num1 = input ("Please enter a number")
2.num2 = input ("Please enter a second number")
3.if num1 > 100:
4.   print "Your first number is greater than 100"
5.else:
6.   print "Your first number is less than 100"
```

Pupil resources

At the end of this document there are pupil resources to print to accompany each lesson with the model code and Coding Challenges.

Lesson 1: Introducing Python

L.Os:

1. **Navigate Idle (create, save, run programs)**
2. **Understand and use mathematical operation and 'print' statement**

N.C: design and write programs that accomplish specific goals; use sequence in programs;

Vocabulary: Python, scripting language, visual programming language, syntax, Idle

***Introducing Python and mathematical functions*:** Explain that pupils will be using a language called **Python** to write programs. Recap that pupils will have had prior experience of writing programs using Scratch, which is a **visual programming language** as we built up programs by visually dragging programming blocks. Explain that Python is a **scripting language,** so instead of dragging blocks we have to write code and we have to learn the language of Python's code (called the **syntax**) - just like we may learn languages such as French of Urdu.

Ask pupils to open **Idle** (which is on both Windows and Mac). Explain this is a program we use to write our Python codes. Introduce the following code to pupils:

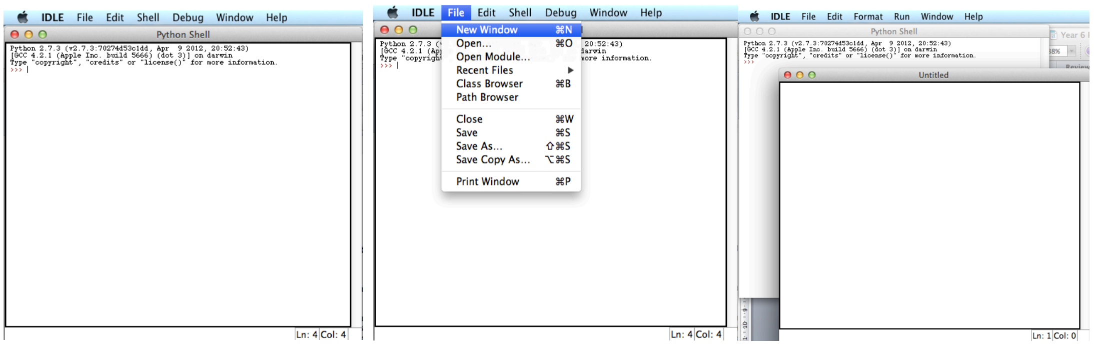
Mathematical operation	**Python code**
Addition	+
Subtraction	-
Multiplication	*
Division	/

Demonstrate that we can use this code to communicate in Python to work out numeracy calculations for us. For example, try writing the following into Idle and press return after each. *Give time for pupils to try using Python to complete calculations.*

```
300+400
987-653
12*9
30/6
```

Hello World program: We are now going to move on to teaching pupils how to write a program to display text within Python. When run, the text displayed will read 'Hello World', as traditionally this is the first program anyone learns to write in a new programming language!

To do this *(and from here onwards when programming in Python)* we are not going to write our code in the 'outer' Idle window we have just been using but rather from within Idle ask pupils to select File > New Window to create a window which we can code into and save. **Show in the screenshots below.** From this new window, pupils should click File > Save and name the file helloworld.py. Explain it is important to include the .py file extension to indicate this is a Python file. Ask pupils to save the file to an appropriate location.



```
Python 2.7.3 (v2.7.3:70274d53c1dd, Apr  9 2012, 20:52:43)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> |
```

3 screenshots showing outer window of Idle (Python shell) then opening a new window to write programs into.

Similarly to introducing the code for mathematical operations above, introduce pupils to the **statement** 'print""'. Explain that when using the statement 'print""' programs will display the text with the quotation marks. So to create a 'Hello World' program ask pupils to write the following code into the window. *Note there is no capital P for the statement print. This is important as Python is a case-sensitive language.*

```
1. print "Hello world!"
```

Once pupils have written this line of code, explain that to run their program they must first save the changes they have made to their file. Once they have done this, they must select 'Run' then 'Run Module'.

The window they have been coding in will close and their program will run in the 'outer' Python window we used earlier. If they have typed the code correctly their program will display the words 'Hello World!'. They've now written their first Python program!

Coding Challenge: *Can pupils write a program which displays more lengthily text on different lines? For example:*

Hello, how are you today?
I hope you are enjoying learning Python.
What shall we code next?

Challenge solution: Pupils need to use the 'print' statement on each new line to display text over several lines. e.g. *Remember not to include the numbers*

```
1.print “Hello, how are you today?”
2.print “I hope you are enjoying learning Python”
3.print “What shall we code next?”
```

Lesson 2: Variables and comments

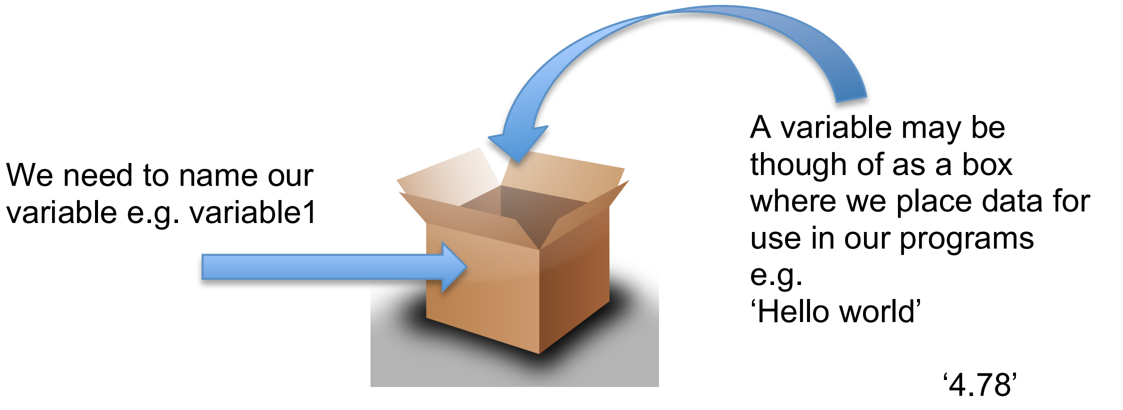
L.Os:

1. **Declare a variable**
2. **Write comments within Python code**
3. **Use mathematical operations and print statement with variables**

N.C: design and write programs that accomplish specific goals; use sequence in programs; work with variables and output; generate appropriate inputs and predicted outputs to test programs; use logical reasoning to explain how a simple algorithm works and to detect and correct errors in algorithms and programs.

Vocabulary: variables, declare, comment

Introducing variables: Explain to pupils that **variables** may be thought of as boxes within our program where we can place data (numbers or text). Explain that we can then use the contents of the boxes within our program and that the values assigned to our variable (the contents of our boxes) may change as our program runs.


We need to name our variable e.g. variable1
A variable may be though of as a box where we place data for use in our programs e.g.
'Hello world'
'4.78'

Explain that when we create a variable and assign data (get a box and put something in it) it is called **declaring** a variable**.** Demonstrate that to declare a variable we use the following code:

```
variable1 = 8
```

Explain that in this example above we have created a variable called 'variable1' and assigned the value 8 to it. Demonstrate that we can create more than one variable and that we can assign text as well as numbers to variables by writing the following code:

```
1.name = sarah
2.laps = 8
```

Can pupils names the 2 variables you have created using the code above and their values? Explain here that naming variables with names that relate to the data they hold (e.g. name & laps), as opposed to using generic terms such as variable1, makes subsequently writing code using these variables easier.

Ask pupils to now create a new Python file: i.e. open Idle, select File > New window then Save as, and call the file *variables.py* and save to an appropriate location.

Ask pupils to create a variable called 'children' with a value of 30 and a variable called 'sweets' with a value of 5. Explain that we will be using these variables to write a program that calculates the total number of sweets required to give a defined number of sweets to a defined number of children.

Solution code:

```
1.children = 30
2.sweets = 5
```

Comments in Python: Explain that a **comment** is a line of code that isn't part of the program but is there to explain to the programmer what parts of the code are doing. We write comments in regular clear English (or whatever language we as programmers may speak) as opposed to the language of Python. Explain that comments are important as when we write longer pieces of code we may forget the function of different parts, or we may work on code collaboratively with others and therefore we need to explain what we are doing.

However, explain that the trouble with writing in regular English within our program is that our computer thinks we are still writing in Python and may try and interpret commands from what we have said. As such, we need to indicate when we are writing a comment and we do that by using '#'. Explain that the computer will ignore any line starting with # as it knows this is a comment to the programmer and not part of the code. *Demonstrate adding comments to the variable we just created to add explanation about what we are coding and ask the children to do the same i.e.*

```
1.children = 30
2.#The number of children in the class
3.sweets = 5
4.#The number of sweets each child will get
```

Using mathematical operations and print statement with variables: Explain we are now going to continue coding to write a program that uses the variables we have just declared as well as the mathematical operations and print statement we covered in the previous lesson.

Recap that the purpose of this program was to work out the total number of sweets required based on the two variables we have declared: the number of children and the number of sweets they get each.

To write the program to calculate this, add the code beneath our variables and comments that appears on line 4&5 below and is highlighted for clarity – and ask pupils to add the same to their program.

```
1.children = 30
2.#The number of children in the class
sweets = 5
3.#The number of sweets each child will get
4. total = children*sweets
5. print total
```

Take a moment to discuss with pupils lines 4&5 of code above. Can children spot a new variable being declared? What is the name of this new variable? Can children spot the mathematical operator? What mathematical operation is it? *(Note the scope here to reinforce numeracy problem solving objectives on choice of operation)* Can pupils see the print statement we used in lesson 1? *Note – we don't need to use quotation marks when requesting to print a variable.*

What do pupils anticipate the output of this program will be? Can pupils interpret into English what line 4 is asking the computer to do? *'Create a new variable called total and assign it the value of* *the value in variable children multiplied by the value in variable sweets'.*

Ask pupils to now save their program and select Run then Run Module. The window they have been coding in will close and their program will run in the 'outer' Python window we used earlier. If they have entered the code correctly their program will display the total number of sweets required for all children i.e. 150.

Pupils have now written a program which, when the number of children and sweets per child has been entered, can calculate the total number of sweets required. Pupils should have some time to edit their program by changing the values for the two variables. To modify the program pupils should return to the window it was written in and make changes before selecting Save the Run then Run Module.

Coding Challenge 1: Can you write a program to calculate the number of cakes needed for a children's party? Choose your own number of children and how many cakes they should each get!

Solution 1:

```
#Variable for number of children
children = 27
#Variable for number of cakes they should get
cakes = 4
#Declare variable for total number of cakes
total = children*cakes
print total
```

Coding challenge 2: Write a program to calculate the total number of house points 4 classes got in a school. *Choose your own numbers for how many house points each class got!*

Solution 2:

```
#Declare variables for each house
class1 = 345
class2 = 265
class3 = 265

class4 = 189
```

```
6. #Declare variable for total number of house points
7. total = class1+class2+class3+class4
8. print total
```

Coding challenge 3: In an enterprise activity, 3 students make £620 by selling products they have made. Their total costs were £400. Write a program to calculate how much money they make each after they have paid their costs.

Solution 3:

```
1. #Declare variables for income, costs and number of students
2. income = 620
3. costs = 500
4. students = 3
5. #Calculate total profit and profit each by declaring two new variables
6. profit = income – costs
7. each = profit/students
8. #print solution
9. print each
```

Lesson 3: User inputs

L.Os:

1. **Use raw_input() statement**
2. **Use input() statement**
3. **print sentences**

NC: design and write programs that accomplish specific goals; solve problems by decomposing them into smaller parts; use sequence; work with variables and input and output; generate appropriate inputs and predicted outputs to test programs; use logical reasoning to explain how a simple algorithm works and to detect and correct errors in algorithms and programs

Vocabulary: input

Using the input function: Explain that Python has statements, which can be used to accept an input of data from the user. The type of input statement to use depends on the data type to be entered (either text or numerical). The table below shows the statement to use for the two data types:

Data type	**Input statement for coding**
Text (Crumpsall Lane… etc)	raw_input()
Numerical (5, 345….. etc)	input()

The data which is entered can be used to declare variables, so rather than us defining the values assigned to variables when writing a program, we can ask the user to enter the data. Write in the following code into a new window within Idle to demonstrate this:

```
1. name = raw_input (“What is your name?”)
2. print name, ”is a lovely name.”
```

What do pupils think this program will do when it is run? Run the program and demonstrate that it asks for a users name and then output the sentence: ….. is a lovely name, with the user’s name starting the sentence.

Talk through the following points with the pupils about using this piece of input code:

Line 1: Firstly, if we work through what line 1 is doing we can see that we are declaring a variable called ‘name’ but instead of assigning it a value we use the `raw_input()`statement which allows us to write a question to the user inside the brackets – note the text must also be inside quotation marks. When the user responds to this question, whatever response they give will be assigned to the variable ‘name’.

Line 2: The second line uses the print command to print the variable ‘name’ and then the text “is a lovely name.”. Importantly the variable and text are separated by a comma and the text is in quotation marks - as pupils learnt to do in the ‘Hello World’ lesson 1.

Inputs and mathematical operations: Explain to pupils that we are now going to return to problems similar to those that we tackled last week, but make our programs more complex, so the user can input data.

Explain that together you are going to write a program which will: *Allow the user to enter the number of children in a class and the number of exercise books they require each and output the total number of books required.*

Ask pupils how they would tackle this coding problem based on the demonstration of using the input statement above (remember we are using numerical data now so we will use a slightly different input statement).

Working jointly with pupils via questioning and prompting construct the following code (obviously comment lines may vary)

```
1. #write input code to declare variables for number of children and number of books
2. children = raw_input(“How many children are there?”)
3. books = raw_input (“How many books does each child need?”)
4. # declare ‘total’ variable from children and books
5. total = children*books
6. #print total along with text
7. print “You will need”, total, “books”
```

Run the program to demonstrate that the user is asked for the number of children and the number of books each child requires before it outputs the total number of books required.

Coding challenge 1: Write a program which asks the user to enter how many children are in a class and how many children are in each group. The program should then output how many whole groups there will be in a sentence reading: *The will be …. whole groups.*

Solution:

```
# Declare two variables based on user input
children = input("How many children are in the class?")
number = input("How many children are in each group?")
# Use mathematical operator to define solution and print this with text explanation
groups = children/number
print "There will be", groups, "whole groups."
```

Please note, and explain to pupils, that this program will only return the number of whole groups as within Python we need to change numbers to 'floats' to handle decimals. We will cover this in a future lesson

Coding challenge 2: Write a program which asks the user their name and how many days until their next birthday. The program should then output a personalised sentence with a rough approximation of the number of seconds until their next birthday.

Solution:

```
# declare variable of name and days till birthday from user input
name = raw_input ("What is your name?")
days = input ("How many days until your next birthday?")
# calculate seconds using mathematical operations
seconds = days*24*60*60
# print result
print "Hi", name, "there are approximately", seconds, " seconds till your next birthday!"
```

Lesson 4: Selection

L.Os:

1. **Use conditional statements if, else if (elif) and else**
2. **Use comparison operators**

N.C: design and write programs that accomplish specific goals; solve problems by decomposing them into smaller parts; use sequence and selection in programs; work with variables and input and output; generate appropriate inputs and predicted outputs to test programs; use logical reasoning to explain how a simple algorithm works and to detect and correct errors in algorithms and programs

Vocabulary: conditional statement, comparison operator

Introducing conditional statements: Recap that in the previous lesson we wrote code that allowed users to interact with our programs by entering data for variables. However, the program was quite simple in that it always performed the same operation with the data that had been entered. Explain that to extend this further we are now going to learn how to write code that responds differently depending on what data the user enters and offers the user options - making our programs more interactive. The concepts covered in this lesson underpin programming computer games as well as simple forms of artificial intelligence.

Explain that the statements we are going to use to achieve this are called **conditional statements**. i.e. what the code does depends on a condition being met or not met. We will go through an example shortly to make this clearer but first here are the new statements we will be using in this lesson – we are developing our knowledge of Python's syntax!

Conditional statements	**Comparison operators**	
If	==	Check is equal to
else	!=	Checks if not equal to
elif (else if)	>	Checks if greater than
	<	Checks if less than
	>=	Checks if greater than or equal to
	<=	Checks if less than or equal to

Explain in the table above we have also introduced the syntax for **comparison operators** as we will require these to work with our conditional statements. Pupils may be reasonably familiar with the notation here since they will have encountered inequalities within numeracy.

Explain that we are going to write a short piece of code using conditional statements and comparison operators.

Use the following code as an example:

```
#Declare name variable using input from user
name = raw_input (“Hi, What is your name?”)
print “It is lovely to meet you”, name, “.”
# Declare feeling variable using number input from user
feeling = input (“How are you feeling today? Excited = 1, Happy = 2, Miserable = 3, Nervous = 4”)
# Use conditional statement with ‘equal to’ comparison to determine which response is given
if feeling == 1:
    print “Fantastic. What are you feeling excited about?”
elif feeling == 2:
    print “I am happy too. Yay”
elif feeling == 3:
    print “I am sorry to hear that you are miserable. How can I cheer you up?”
elif feeling == 4:
    print “What are you nervous about?”
```

Discussing how this code works. Work through the code above with your pupils to explain the following points.

1. Firstly we have used a raw input to declare the variable name and used print to return a sentence which includes the user’s name. This portion of the code is similar to that which we have covered in previous lessons.
2. We have then declared a second variable called feeling. The user is asked to enter a number which corresponds to a list of feelings i.e. we have provided a menu for the user
3. We have then used our new syntax, the ‘if’ statement, along with a comparison operator to say (in English): *if the value of*

the variable feeling is equal to 1 then print 'Fantastic. What are you feeling excited about?' **Note the use of the colon and that print is indented. Also note that the code to check if feeling is equal to 1 uses a double equals sign (==) as a single equals sign (=) is used to declare variables.**

4. Since we have several options in our menu of feelings, we then use another conditional statement called 'else if' (written in code as elif) along with the comparison operator == to provide a different response for the other feelings that the user may select.

Programming a calculator: Here is a second example which uses a range of comparison operators. In this example the user's inputs also cause the program to perform mathematical operations on the variables declared, like a calculator

```
num1 = input ("Please enter a number")
num2 = input ("Please enter a second number")
if num1 > 100:
    print "Your first number is greater than 100"
else:
    print "Your first number is less than 100"
if num2 > 100:
    print "Your second number is greater than 100"
else:
    print "Your second number is less than 100"
op = input ("Select an operation: 1. Add 2. Subtract 3. Multiply")
if op == 1:
    add = num1 + num2
    print add
elif op == 2:
    sub = num1 – num2
    print sub
elif op == 3:
    mul = num1*num2
    print mul
```

Coding Challenge: Can you program a French dictionary that provides the user with a range of English words to chose from and returns the translation in French?

Solution:

```
print “Welcome to this English – French translator for animals”
word = input (“Please select a number: 1 – Cat, 2 – Dog, 3 – Fish, 4 – Hamster, 5 – Rabbit)
If word == 1:
  print “In French cat is le chat.”
elif word == 2:
  print “In French dog is le chien.”
elif word == 3:
  print: “In French fish is le poisson.”
elif word == 4:
  print: “In French hamster is le hamster.”
```

Lesson 5&6: Final project

Once pupils have completed the scheme of 4 lessons above, including the coding challenges, it is intended that they have the opportunity to program a final project to help reinforce the skills and knowledge covered. As mentioned in the introduction, whilst two suggestions are presented below for final projects, it is envisaged those using this resource may also generate ideas relating to topic work and thus linking programming with pupils' wider learning. In addition, pupils may be encouraged to think of their own ideas for programs they wish to code.

Final project 1: Artificial Intelligence

Project question: 'Can you write a program to bring a computer to life?'

Introducing the project: The first idea is an extension of the example used in lesson 4 which is to write a program that appears to transform our computer into an 'intelligent robot' which may have a conversation with you.

The videos below can be used to introduce the project. The first is a short clip of a Horizon documentary on artificial intelligence whereby robots are seen learning from each other, and the second is a compilation of different robots. When watching the second clip explain that all robots (including those featured) are made up of both their physical construction and the code they are programmed to operate to – such as that we have been learning to write.

Horizon clip on Artificial Intelligence:
http://www.youtube.com/watch?v=ImoXByLkK14

Robots are Awesome:
http://www.youtube.com/watch?v=rjernKRnLRo

It is my understanding that 'actual' artificial intelligence is the ability for a system to learn and evolve which is a little beyond the scope of the content of this introduction to Python! However, explain to pupils that through the extended use of inputs and selection algorithms we can write code that appears to turn our computer into a robot able to communicate with us!

An example of code that a pupil may write for this final project appears below. Pupils should start by recapping the code in lesson 4 in which the program asks and responds to the user about how they are feeling. Pupils can build up more lengthily code from this starting point.

Notes on the code: in the example below inputs have been used within conditional statement so 'if' the user responds yes to the question about going to school (line 5) the next line of code is an input asking about the lessons they are doing (line 6) – this has been highlighted below in yellow. What's more, a second condition statement follows (line 8) which has been **nested** (meaning embedded within) within the first condition.

Nesting is achieved by indenting the 'if' statement as can be seen on line 8 – this has been highlighted blue. Nesting the condition statements like this means that if the user responds to say they are going to school (line 5) the code asks a question about studying numeracy and responds to this (lines 8-11) but if the user says they are not going to school the code skips to line 12 and misses any questions about what they might be studying. ***Note the lesson on selection did not extend to nesting conditions in this manner but I have included an example as once shown I am sure some pupils will be able to understand and apply this concept to enhance their coding.***

For free Primary Computing resources please visit…

PRIMARY COMPUTING.co.uk

Example ‘AI’ code:

```
print "Hello, my name is Compu"
name = raw_input("What is your name?")
print "Well it is nice to meet you", name
day = raw_input (" What day is it today?”)
print "Thanks", name, "now I remember it is", day, "."
today = input("So are you going to school today? 1 - yes 2 - no")
if today == 1:
    lesson = input("Are you studying maths today? 1 = yes 2 = no")
    if lesson == 1:
        print "Great I like maths, I’ll help!"
    else:
        print "That’s a shame as I like maths"
elif today == 2:
    print "Oh sorry I forgot you don't go to school on", day
print "Lets play a game", name,"."
number=input("I'm thinking of a number between 1&10 and you've got to guess it. What's your first guess?")
if number == 7:
    print "Yay you guessed it!"
elif number < 7:
    print "Higher"
elif number > 7:
    print "Lower"
print "I'll give you one more guess"
number1 = input("Have another go?")
if number1 == 7:
    print "Yay you guessed it!"
elif number1 < 7:
    print "That's too low it was 7"
elif number1 > 7:
    print "That's too high it was 7"
```

Final project 2: Text Adventure Game

Project question: Can you create and code a section of a text adventure game?

Introducing the project: An adventure game is a video game in which the player assumes the role of a protagonist in an interactive story. The screenshot above is from the Dr Who adventure game for the PC.

Whilst modern day adventure games are often graphics based, original adventure games were text based. These were some of the earliest computer games and represented the cutting edge of technology when they were released.

Adventure games are based on providing the user with a story to engage with and make decisions about. The decisions they make influence the direction of the story and the path their character (often written in the first person such that the player is the character) takes through the adventure.

The programming concepts which underpin the coding of an adventure game are similar to the artificial intelligence project above, namely inputs and conditional statements. The code for the beginnings of an example adventure text game appears below. This code could be shared as a starting point and developed further. Explanations of various parts of the code appears beneath.

```
1. print “Welcome to the Dragon’s Cave Adventure”
2.print "You wake in a cave and can see a dragon!"
3. print “Luckily it appears to be sleeping.”
4.print "You look around for a way out. What should you do? Maybe you could climb over the dragon's tail"
5.choice1 = input(" Should you 1 - Explore the cave more? 2 - Climb over the dragon's tail?")
6.if choice1 == 1:
7.    print "You look into the darkness of the cave but it is so pitch black you can't see a thing!"
8.elif choice1 == 2:
9.    print "You try gently climbing over the dragon's tail but it lets out a large huff, smokes a little and you decide climbing his tail isn't a very good idea!"
10.print "The dragon shuffles and now you can get past and out the cave but just then you hear a noise coming from the darkness to your right. It sounds like someone calling your name."
11.choice2 = input("Do you 1 - Go to investigate the noise 2 - Try and sneak past the dragon?")
12.if choice2 == 1:
13.    choice3 = input("You go to investigate and the noise stops but you can see a light. Do you continue 1 - Yes 2 - No")
14.     if choice3 == 1:
15.         print "You get deeper into the cave and see the light is actually a passage to the outside where your friend is calling your name"
16.     elif choice3 == 2:
17.         print "You turn around but as you do you are met by the dragon... GAME OVER"
18.elif choice2 == 2:
19.     print "Slowly you edge past the dragon desperate not to make a sound!"
20.print "You emerge from the cave and see your friend. What a lucky escape!"
```

Notes on this code: The code is built on selection statements and inputs as covered in previous lessons. The interactively of the adventure text starts on line 5 when an input is used along with a condition statement when the user is asked whether they should explore the cave more or climb over the dragon (highlighted in yellow). Depending on their choice the user will receive different text progressing the story.

However, you'll see that whilst the user receives a different response depending on their choice, no matter what choice the user makes they will always arrive at line 10 when the text continues with "The dragon shuffles…" (also highlighted in yellow).

As such, to make the programming more complex and truly vary the outcome of this adventure text based on user decisions, the section highlighted in blue uses a nested condition. **Please refer to the artificial intelligence project above for an explanation of what 'nesting' is and how to program nested conditional statements.**

Glossary:

Algorithm: A process or set of rules to be followed in calculations or other problem-solving operations often by a computer.

Comment: Text which is not part of an algorithm but explains details of the program to those working on it

Comparison operator: These are used to test relationships between entities i.e. is a == 1 or is b > 5

Conditional statement: Programing commands that test for a condition and action subsequent code depending on whether the condition is met or otherwise.

Declare: When a variable is created and assigned a value

Idle: A environment for programming Python.

Input: Data that is accepted into a program

Output: Data that is produced from a program

Python: A programming language developed in 1989 which is ideal for beginners to learn as the syntax for many commands are close to the English language that describes their function *e.g. print ""*

Scripting language: A programming language that requires users to learn the syntax of the language and write lines of code

Syntax: The set of rules, structure and commands that make up a programming language.

Variables: A parameter which can be created to store a value which may be retrieved, changed and used in programs at any point.

Visual programming language: A coding language that uses a graphical method of constructing algorithms such as Scratch.

Lesson 1 pupil sheet: Introducing Python

Mathematical operation	Python code
Addition	+
Subtraction	-
Multiplication	*
Division	/

Hello World program code:

```
1. print “Hello world!”
```

Coding Challenge:

Can you write a Hello world program which displays more lengthily text on different lines? For example:

Hello, how are you today?
I hope you are enjoying learning Python.
What shall we code next?

Lesson 2 pupil sheet: Variable and comments

Declaring variables:

```
name = sarah
laps = 8
```

Adding comments in Python:

```
children = 30
#The number of children in the class
sweets = 5
#The number of sweets each child will get
```

Using mathematical operations and print statement with variables:

```
children = 30
#The number of children in the class
sweets = 5
#The number of sweets each child will get
total = children*sweets
print total
```

Coding Challenge 1: Write a program to calculate the total number of cakes required for a class party if we know the number of children attending the party and the number of cakes each child should get. *Choose your own number of children and how many cakes they should get!*

Coding challenge 2: Write a program to calculate the total number of house points 4 classes got in a school. *Choose your own numbers for how many house points each class got!*

Coding challenge 3: In an enterprise activity, 3 students make £620 by selling products they have made. Their total costs were £400. Write a program to calculate how much money they make each after they have paid their costs.

Lesson 3 pupil sheet: User inputs

Input statements:

Data type	**Input statement for coding**
Text (Crumpsall Lane… etc)	raw_input()
Numerical (5, 345….. etc)	input()

Using input statements:

```
name = raw_input (“What is your name?”)
print name, ”is a lovely name.”
```

Inputs and mathematical operations:

```
#write input code to declare variables for number of children and number of books
children = raw_input(“How many children are there?”)
books = raw_input (“How many books does each child need?”)
# declare ‘total’ variable from children and books
total = children*books
#print total along with text
print “You will need”, total, “books”
```

Coding challenge 1: Write a program which asks the user to enter how many children are in a class and how many children are in each group. The program should then output how many whole groups there will be in a sentence reading: *The will be …. whole groups.*

Coding challenge 2: Write a program which asks the user their name and how many days until their next birthday. The program should then output a personalised sentence with a rough approximation of the number of seconds until their next birthday.

Lesson 4 pupil sheet 1: Selection and comparison operators

Conditional statements	Comparison operators	
If	==	Check is equal to
else	!=	Checks if not equal to
elif (else if)	>	Checks if greater than
	<	Checks if less than
	>=	Checks if greater than or equal to
	<=	Checks if less than or equal to

```
#Declare name variable using input from user
name = raw_input ("Hi, What is your name?")
print "It is lovely to meet you", name, "."
# Declare feeling variable using number input from user
feeling = input ("How are you feeling today? Excited = 1, Happy = 2, Miserable = 3, Nervous = 4")
# Use conditional statement with 'equal to' comparison to determine which response is given
if feeling == 1:
 print "Fantastic. What are you feeling excited about?"
elif feeling == 2:
 print "I am happy too. Yay"
elif feeling == 3:
 print "I am sorry to hear that you are miserable. How can I cheer you up?"
elif feeling == 4:
 print "What are you nervous about?"
```

Lesson 4 pupil sheet 2: Selection and comparison operators

Programming a calculator:

```
num1 = input (“Please enter a number”)
num2 = input (“Please enter a second number”)
if num1 > 100:
    print “Your first number is greater than 100”
else:
    print “Your first number is less than 100”
if num2 > 100:
    print “Your second number is greater than 100”
else:
    print “Your second number is less than 100”
op = input (“Select an operation: 1. Add 2. Subtract 3. Multiply”)
if op == 1:
    add = num1 + num2
    print add
elif op == 2:
    sub = num1 – num2
    print sub
elif op == 3:
    mul = num1*num2
    print mul
```

Coding Challenge: Can you program a French dictionary that provides the user with a range of English words to chose from and returns the translation in French?

Final project 1: Artificial Intelligence

Project question: ‘Can you write a program to bring a computer to life?’

Example ‘AI’ code:

```
print "Hello, my name is Compu"
name = raw_input("What is your name?")
print "Well it is nice to meet you", name
day = raw_input (" What day is it today?”)
print "Thanks", name, "now I remember it is", day, "."
today = input("So are you going to school today? 1 - yes 2 - no")
if today == 1:
    lesson = input("Are you studying maths today? 1 = yes 2 = no")
    if lesson == 1:
        print "Great I like maths, I’ll help!"
    else:
        print "That’s a shame as I like maths"
elif today == 2:
    print "Oh sorry I forgot you don't go to school on", day
print "Lets play a game", name,"."
number=input("I'm thinking of a number between 1&10 and you've got to guess it. What's your first guess?")
if number == 7:
    print "Yay you guessed it!"
elif number < 7:
    print "Higher"
elif number > 7:
    print "Lower"
print "I'll give you one more guess"
number1 = input("Have another go?")
if number1 == 7:
    print "Yay you guessed it!"
elif number1 < 7:
    print "That's too low it was 7"
elif number1 > 7:
    print "That's too high it was 7"
```

Final project 2: Text Adventure Game

Project question: Can you create and code a section of a text adventure game?

```
print “Welcome to the Dragon’s Cave Adventure”
print "You wake in a cave and can see a dragon!"
print “Luckily it appears to be sleeping.”
print "You look around for a way out. What should you do? Maybe you could climb over the dragon's tail"
choice1 = input(" Should you 1 - Explore the cave more? 2 - Climb over the dragon's tail?")
if choice1 == 1:
    print "You look into the darkness of the cave but it is so pitch black you can't see a thing!"
elif choice1 == 2:
    print "You try gently climbing over the dragon's tail but it lets out a large huff, smokes a little and you decide climbing his tail isn't a very good idea!"
print "The dragon shuffles and now you can get past and out the cave but just then you hear a noise coming from the darkness to your right. It sounds like someone calling your name."
choice2 = input("Do you 1 - Go to investigate the noise 2 - Try and sneak past the dragon?")
if choice2 == 1:
    choice3 = input("You go to investigate and the noise stops but you can see a light. Do you continue 1 - Yes 2 - No")
     if choice3 == 1:
         print "You get deeper into the cave and see the light is actually a passage to the outside where your friend is calling your name"
     elif choice3 == 2:
         print "You turn around but as you do you are met by the dragon... GAME OVER"
elif choice2 == 2:
     print "Slowly you edge past the dragon desperate not to make a sound!"
print "You emerge from the cave and see your friend. What a lucky escape!"
```