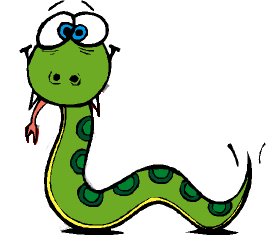
Computing Science

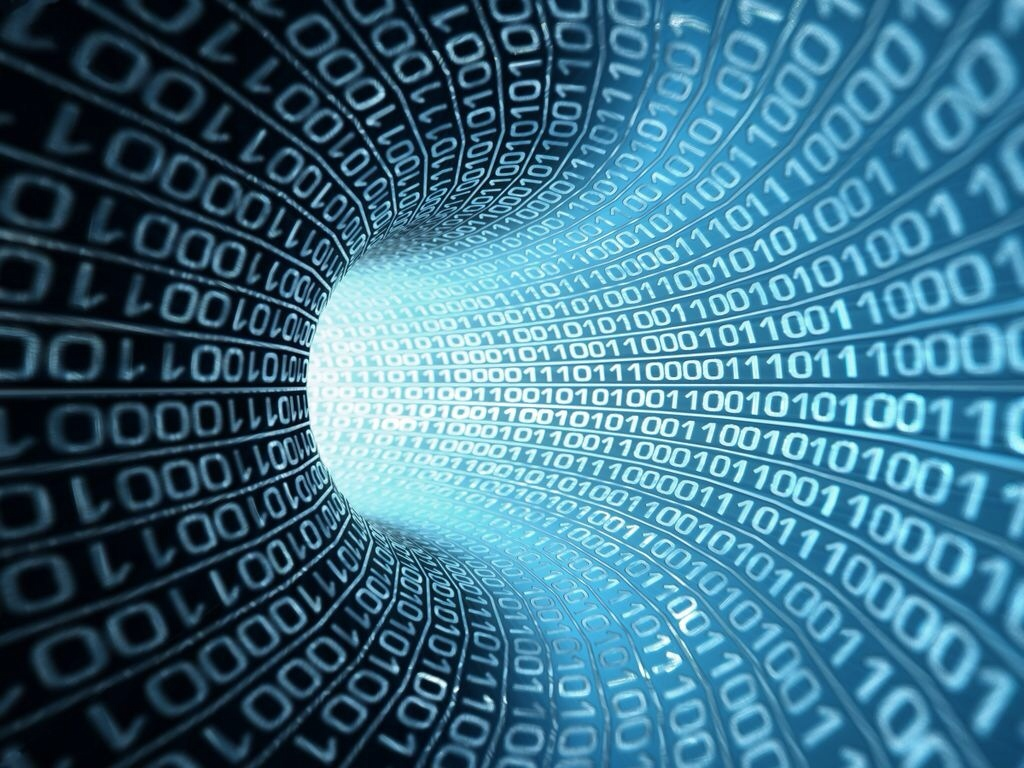
National 4

Software Design & Development

Mr. Stewart







Outcomes and Assessment Standards

## Outcome 1

The candidate will:

**1** **Explain how simple programs work, drawing on understanding of basic concepts in software development by:**

1.1 Reading and explaining code

1.2 Describing the purpose of a range of programming constructs and how they work

1.3 Explaining how data and instructions are stored

The range of programming constructs should includeexpressions, sequence, selection and iteration.

## Outcome 2

The candidate will:

**2** **Develop short programs using a software development environment by:**

2.1 Selecting and using expressions, sequence, selection and iteration

2.2 Selecting and using appropriate simple data types, such as numeric (integer) and string

2.3 Testing digital solutions using supplied test data

2.4 Identifying and rectifying errors in programs

Programs should include at least one construct and one data type.

## Outcome 3

The candidate will:

**3** **Produce a short factual report on a contemporary software-based application by:**

3.1 Describing the application

3.2 Explaining how its features relate to programming constructs and data types

3.3 Describing its impact on the environment or society

Software

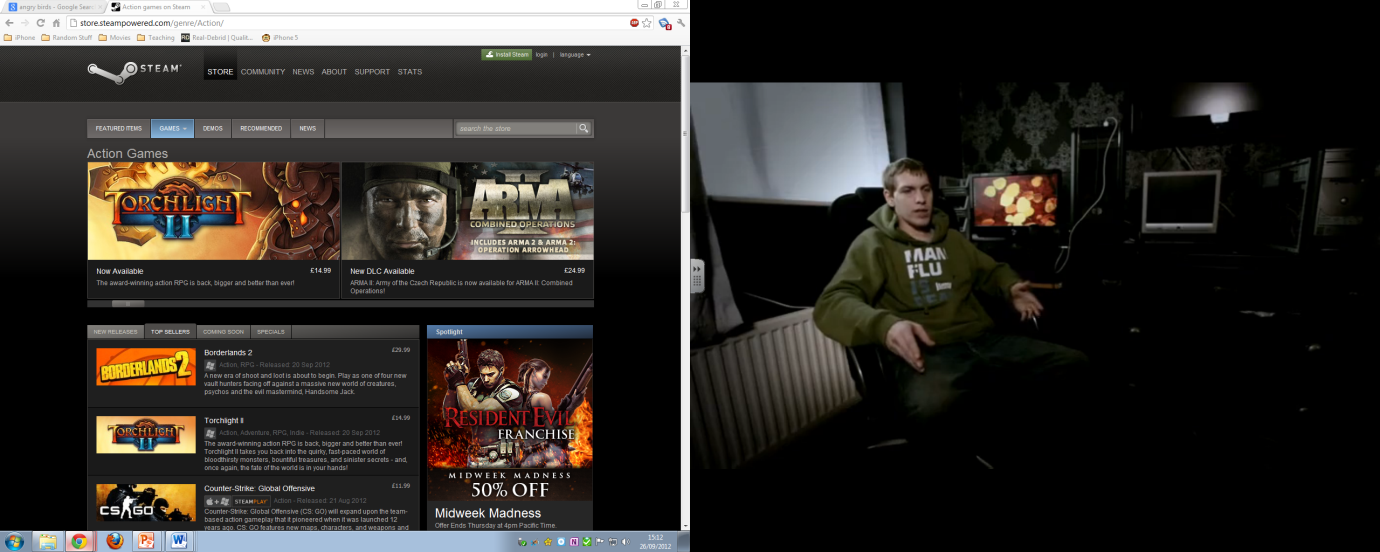
What is software?

You should already know that any computer system is made up of **hardware** and **software**.

The term **hardware** is fairly easy to understand, **because you can see it**. It is all the pieces of equipment that make up the system – the processor, monitor, keyboard, mouse, printer, scanner and so on.

**Software** is not so obvious. It is all the programs, instructions and data that allow the hardware to do something useful and interesting.

A long time ago before fast internet speeds people went to shops like PC World to purchase software in the form of discs. Today with broadband most people download software through the **Internet or App Stores**.



Here is the list of programs that I have used recently.

* **Microsoft Word** (the word processing program that I use – I regularly use three versions of it: Word 2012)
* **Microsoft Excel** (spreadsheets used to keep accounts)
* **Microsoft PowerPoint** (create presentations to explain material to classes)
* **Internet Explorer** (for browsing the web)
* **Safari** (web browser for Mac OS X)
* **Chrome** (web browser made by Google)
* **Spotify** (for listening to music free)
* **iPhoto** (for organising my digital photographs)
* **iMovie** (for editing digital movies)
* **Adobe Photoshop CS6** (for editing digital photographs )
* **Adobe Acrobat** and **Preview** (for viewing PDF files)
* **iTunes** (Organises and stores your music and apps)
* **AVG Anti Virus 2012**
* **Angry Birds** (iOS Game)
* **Temple Run** (iOS Game)
* **Doodle Jump** (iOS Game)



On each computer that I have used, a program (or group of programs) called the **operating system** must have been running. So I must add the following to my list.

* **Windows 7**



* **Linux**
* **iOS**
* **Android**
* **Windows XP**
* **Windows Vista**
* **OS X Mountain Lion**

**Remember.**

Hardware is things you can touch (Monitor, Keyboard, Mouse, etc.)

Software is stuff you cannot touch (PowerPoint, iTunes, Internet Explorer)

Activity 1

Make a list of all the software you have used over the last few days.

Questions

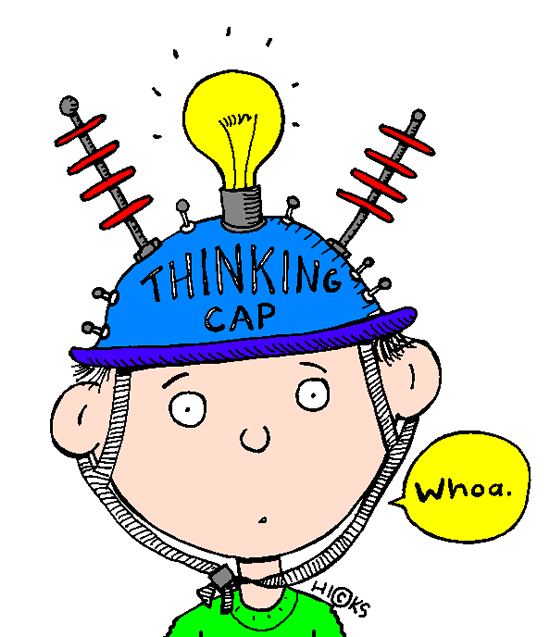
1. What is the meaning of the term hardware?
2. Name three pieces of software from the Microsoft Office Package you have used before.
3. What operating system do the school computers run on?
4. What is your most used piece of software?
5. Name three different web browsers? (These are all examples of software)
6. Identify each of the following as either hardware or software

|  |  |  |
| --- | --- | --- |
| **Item** | **Hardware** | **Software** |
| Printer |  |  |
| Internet Explorer |  |  |
| Windows 7 |  |  |
| Mouse |  |  |
| iPad |  |  |
| Digital Camera |  |  |
| Microsoft Word |  |  |
| OS X |  |  |
| Dell Inspiron Laptop |  |  |
| Adobe Photoshop |  |  |
| iTunes |  |  |
| MacBook Pro |  |  |

Software Design and Development

Software development companies do not just jump straight into coding; they have to spend many hours analysing and designing the software before starting to code. In this unit the examples you do will follow these steps.

* Computational Thinking (Analysis/Design)
* Implementation
* Testing
* Evaluation

Computational Thinking (Analysis/Design)

In this stage of development you will be given a problem and it is then up to you to try and **think about how you are going to solve it**. You will have to explain in English how you are going to approach the problem and **explain what your inputs and outputs will be and detail the processes** that your program will carry out.

A common way in which programs are broken down into smaller understandable chunks is called pseudocode. We will learn more about this later in the unit.

Implementation

When the design is finished a programmer will then be given the task of **writing or implementing the code using a particular programming language**. There are many different programming languages for example Python, Visual Basic, C++, Java etc. The programmer will follow the design that was decided in the previous stage.

Testing

Testing is very important to make sure the customer gets a program that is **error free and works under many different conditions**. Just like a product getting tested in many different forms so does software. It is carefully planned to test a wide range of conditions. There are **three types of testing**.

To show some test data under the headings. Let’s assume an exam was out of 100 marks.

* **Normal Test Data**
  + Making sure the program works when **used normally**. (An example of some test data could be: 21, 30, 76, and 80)
* **Extreme Test Data**
  + Making sure the program works when used that are **on the boundaries** of what could be considered normal. (An example of some test data could be: 0 and 100)
* **Exceptional Data**
  + Making sure the program can handle situations that it has **not been designed to cope with**. (An example of some test data could be: -1, 101, 78.008, 1000000000, abcde)

Evaluation

This is the last stage before the software is released. A report is done to evaluate the three things, fitness for purpose, user interface and the readability.

* **Fitness for purpose:**
  + **Does the software do all the things it is supposed to do**. This involves going back to look at the program specification (produced in the analysis stage) and **checking that all of the features the customer wanted have been implemented**. This also includes is the program error free and usable.
* **User Interface:** 
  + **Is the program easy to use?** Are the menus, buttons and questions asked easy to understand? Could a wide audience such as an old person with little computer knowledge use it or is it for a more confident computer user?
* **Readability:**
  + **Is it possible for someone else to read your program code and understand it?** Are you using **sensible variable names** and including **internal commentary?** Are sub procedures used to structure code logically?

Activity 2

1. Summarise each of the above stages of the development process into your own words.
2. What does readability mean?
3. If I had a program that calculated my exam marks that was out of 30 state some normal, extreme and exceptional test data for this.
4. What is the difference between software and hardware?
5. What does fitness for purpose mean?

Languages and Translators

**Computer Languages**

There are many different computer-programming languages some are better at doing jobs than others.

* PHP: general-purpose server-side scripting language
* C++: general-purpose programming language
* Python: general-purpose interpreted high-level programming language

**These are all examples of high-level languages**.

Low Level Languages

Inside every computer, there is a processor. This is a chip containing digital electronic circuits. These circuits work with tiny pulses of electricity and electronic components. The pulses of electricity can be **represented by the digits 1 and 0**. Every item of data and every instruction for the processor is represented by a group of these binary digits.

**Processors only 'understand' these binary digits**. The only inputs you can make to a processor are groups of binary digits. The only output that a processor can make is a group of binary digits.

Instructions and commands made for processors in this binary digital form are known as machine codes. Here is an example of machine code.

These two programs both print the letter "A" 1000 times on the screen.

**Machine Code**

       169 1 160 0 153 0 128 153 0 129 153 130 153 0 131 200 208 241 96

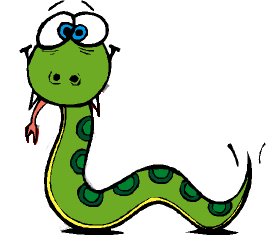
**BASIC (A High Level Language)**

       5 FOR I=1 TO 1000: PRINT "A";: NEXT I

There are several problems with machine code:

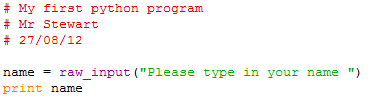
* Machine codes for different processors are different
* They are **very hard for humans to understand and use**
* They take up a lot of space to write down
* It is **difficult to spot errors** in the codes.

**Machine code is an example of a low-level language**. To get around these difficulties computer scientists invented high-level languages.

High Level Languages

High-level languages are **similar to human languages**. Instead of binary codes they **use normal English words**. For example Python uses words like IF, WHILE, PRINT, RAW\_INPUT, INT and so on. So with high level languages using English words it makes the code **easier to understand, easier to spot errors and more readable**.

Below is a very simple program written in Python (A high level language). This simple program is asking the user to type in their name and will output their name on the screen.

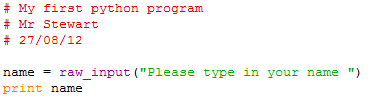


Activity 3

1. Name two high level languages.
2. What type of language (high or low level) is easier to understand?
3. What type would be easier to correct if it had a mistake in it?
4. Explain the difference between high and low level languages.
5. List two advantages of high-level languages.
6. State three words used to code in Python.
7. What is PHP used for?

Translators

High-level languages have some great advantages compared to machine code however there is one major problem – processors don’t understand high-level language at all. To get round this problem computer scientists have developed translator programs that translate high-level languages (written by humans) into machine code (understood by processors).



Translator Program

Compiler or Interpreter

10001001 11001001 1001000 11100001 10001101 1000111

Interpreter

An interpreter takes **each line** of high-level language and **translates it into machine code** and passes it to the processor to carry out that instruction. It then works its way through the program **one line at a time** in this way.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| A program will run even if it is not finished. | **No copy of the machine code is saved.** Meaning the source code has to be translated every time taking longer. |
| **Easy to spot errors during the translation.** | The process of translating the program slows down the running of it. |
| Program will run as soon as the first line is translated. | You will need to have a translator program or you cannot run it. |

Compiler

A compiler takes your high level language and translates the **whole program into machine code once**. The machine code can then be **saved and kept** and does not need to be translated before it is run. This is an example of .exe file. Software that you buy, such as a games or an application, will have been **compiled** into machine code before being distributed and sold.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| The **machine code is saved** so the program only needs to be translated once. | You **have to wait** until the code is complete and the errors have been fixed before the translation can be finished and the machine code is run. |
| The user **does not need a translator program** to run the machine code therefore the program runs quicker. | Each time the program is changed it needs to be **re-translated**. |

Activity 4

1. State two types of translator programs.
2. Describe how an interpreter translates a program.
3. What translator translates a program line by line?
4. What is an advantage of using an interpreter?
5. What is a disadvantage of using an interpreter?
6. What is an advantage of using a compiler?
7. What is a disadvantage of using a compiler?

Python Programming

In this section you will learn how to develop and understand programs using a high level language called Python.

To write your own Python programs you will need a piece of software called an editor. We are going to use the Idle editor to develop and test our Python programs.

Python is FREE

Python Idle editor is a free software development environment that you can get at home. It should run on all PC’s and Macs.

In school we are using version 2.7.3 that can be downloaded from the link below.

<http://www.python.org/getit/>

Python has a very strong community and there are lots of free resources and help available online. If you are looking to do some more programming at home <http://www.codecademy.com/> is a great place to start although this uses Python v3 so some code may be slightly different.

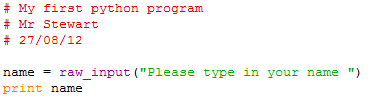
Activity 5

1. If you have not already done so **create a Computing folder** in you’re My Documents and inside that **create a new folder called Python**. This is where all your programs will be stored and kept to help you for the next tasks.
2. Open up IDLE (Python GUI)
3. You will see the Python Shell. You can play about with this.
4. Type in 2+2 and press enter. It should output 4. Try other calculations.
5. Type in name = Stephen and press enter. Now type in print name. Change this to your name.
6. Type in name = John and press enter. Now type in print name. The variable name has now changed its value from Stephen to John.
7. Experiment with it and see what you can do.

Activity 6

We are now going to create our first Python program. **Inside the Python Shell go to File and select New Window.** This stage is very important. By creating your program in a new window it lets you save your code so you can edit it or come back to it at a later date.

Earlier in the languages section you came across this piece of code from Python:



1. Copy the above code into the window.
2. Change 2nd line to your name and the 3rd line to today’s date.
3. When you have copied out the code above go to File then Save. Save it as “My first python program” into your Python folder.
4. For Python to remember your formatting at the end of your file name add .py before clicking save.
5. Then go to the Run tab or press F5 and your program should run in the Python Shell.
6. Congratulations you have now completed your first Python program.

Introducing Python

In your first program you have covered the absolute basics of Python programming. You are now able to use the IDLE Editor and understand how to save and run your programs. There is still a lot more to learn about how Python works.

**Data Types**

**Integers:** **numbers that have no decimal or fractional part in them**, for example -99, 103576, -10000, 107

**Real Numbers:** **numbers with decimal places,** for example 3.7654, 10101.3746, -0.0003, 1.5

**Strings:** any other **combination of characters**, for example John, ABC 123Y etc.

**Boolean:** Stores only two values: **True or False**.

**Variables**

Variables are used in programs so that **data can be assigned to them** for processing. This is useful since we **can run a program over and over and use different data each time**.

Variables **must be one word with no space**. We can get around this by linking words using the underline symbol. This makes programs readable: for example, length\_of\_side, name\_of\_customer.

**Internal Commentary**

It is important that you make internal commentary throughout your program. This will help if you need to go back and change a program (maintenance) at a later date.

In Python **lines beginning with # are internal commentary** and the computer ignores these lines. A line with # at the beginning is to help you to understand your program when you look at it later.

A simple rule at the start of a program is to have the **first few lines as #** to give the program name, date it was written, for example:

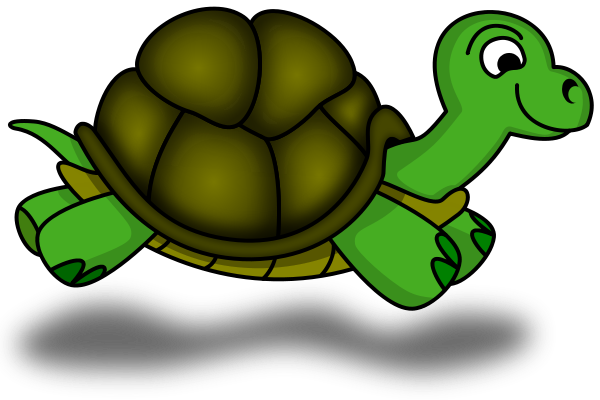
# Wages Calculator

# Mr Stewart

# 17/06/2013

Activity 7

1. State the 3 data types and explain what they store.
2. Why do we use variables in programming?
3. Why do we use internal commentary?

****Timmy Turtle Python Fun

The following Python code is a fun way to learn some of the basics of Python programming.

First Drawing

#Turtle

#Your Name

#Todays Date

import turtle #import the turtle module

window = turtle.Screen() #create a window

timmy = turtle.Turtle() #create a turtle called Timmy

timmy.forward(150)

timmy.right(90)

timmy.forward(100)

timmy.left(90)

timmy.forward(100)

window.exitonclick() #close the window when clicked

This will draw a simple couple of lines. Add to the code to make the turtle move around more.

Using For Loop to Create a Square

#Turtle

#Your Name

#Todays Date

import turtle #import the turtle module

window = turtle.Screen() #create a window

timmy = turtle.Turtle() #create a turtle called Timmy

for loopCounter in range(4): #repeats 4 times

timmy.forward(150)

timmy.right(90)

window.exitonclick() #close the window when clicked

Multiple Shapes

Now that you have drawn a square try creating different shapes.

**Hint: however many sides the shape has divide that by 360 to get the amount of a turn you need.**

* Hexagon
* Pentagon
* Octagon
* Decagon

Drawing Using Procedures

Instead of using a for loop to draw a square we could set up a procedure to do it for us.

#Turtle

#Your Name

#Todays Date

import turtle

def drawASquare(whichTurtle):

for loopCounter in range(4):

whichTurtle.forward(150)

whichTurtle.right(90)

window = turtle.Screen()

timmy = turtle.Turtle()

drawASquare(timmy)

window.exitonclick()

This should simply just draw a square

#Turtle

#Your Name

#Todays Date

import turtle

def drawASquare(whichTurtle):

for loopCounter in range(4):

whichTurtle.forward(150)

whichTurtle.right(90)

window = turtle.Screen()

timmy = turtle.Turtle()

timmy.speed(0)

for loopCounter in range(72): #repeat 72 times

drawASquare(timmy)

timmy.left(5) #Note that 360o ÷ 5 = 72o

window.exitonclick()

**Stage 2: Using Procedures**

In the main program replace: drawASquare(timmy)

Also add in: timmy.speed(0)

Adding Colour

To make the pictures more interesting, we can try changing the colour.

On its own, that might not look too amazing; amend your previous program to this:

colourCounter = 1

for loopCounter in range(72):

drawASquare(timmy)

timmy.left(5)

if colourCounter == 1:

timmy.color(‘blue’)

elif colourCounter == 2:

timmy.color(‘red’)

elif colourCounter == 3:

timmy.color(‘yellow’)

elif colourCounter == 4:

timmy.color(‘green’)

colourCounter = 0

colourCounter = colourCounter + 1

Try using pentagons instead of squares, or triangles, or octagons. Try turning amounts other than 5o.

Lots of Turtles

It is very easy to create lots of turtles. **Make sure you have procedures for a square and a hexagon first,** and then try this:

window = turtle.Screen()

timmy = turtle.Turtle()

tina = turtle.Turtle()

timmy.color(‘blue’)

tina.color(‘pink’)

drawASquare(timmy)

drawAHexagon(tina)

window.exitonclick()

See what other patterns you can come up with using 2, 3 or even more turtles.

Python Log

## Outcome 1

The candidate will:

**1** **Explain how simple programs work, drawing on understanding of basic concepts in software development by:**

1.1 Reading and explaining code

1.2 Describing the purpose of a range of programming constructs and how they work

1.3 Explaining how data and instructions are stored

One of the outcomes of this unit **you will be assessed on the above criteria**. It is important that you are able to write code and also understand it.

For each Python activity you are to fill in the worksheet that your teacher will give you.

The worksheet will ask you various questions. On the “Python Code Understanding” section of the worksheet you will have to explain what a line of code does.

In the examples each one will have certain lines highlighted green like this.

#prompt user for information

name = raw\_input("Please type in your name ")

address = raw\_input("Please type in your address ")

town = raw\_input("Please type in your town ")

phone\_number = raw\_input("Please type in your phone number ")

postcode = raw\_input("Please type in your postcode ")

#print the output on the screen

print "Name: ",name

print "Address: ",address

print "Town: ",town

print "Phone Number: ",phone\_number

print "Postcode: ",postcode

The parts in green are what you will explain on the worksheet.

The worksheet also has space for you to document any notes you think are important and worth remembering for other activities.

Software Development Process

Remember the stages of software development we have looked at before. These stages are what we will work through when creating our programs. From now on all the programs you create will follow these stages.

Activity 8 – Address Card

**Program Specification:** Design, implement and test a program that will prompt the user to enter his or her forename, surname, address, phone number and postcode. The program should then display the details in an address card fashion.

**Computational Thinking (Analysis/Design)**

Every program is designed under the same basic principle of Input, Process and Output or IPO for short.

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Forename  Surname  Address  Phone Number  Postcode |  | Address Card with  Forename  Address  Phone Number  Postcode |

For this example there is no process in the middle as we are simply passing information through.

**User Interface**

What is your forename? John

What is your surname? MacDonald

What is your Address? 3 High Street

What is your Phone Number? 01311 701234

What is your Postcode? PH33 7HQ

Forename: John

Surname: MacDonald

Address: 3 High Street

Phone Number: 01311 701234

Postcode: PH33 7HQ

In industry big companies do not jump straight into typing the code for applications. These stages are important before we do that. The programmer needs to know what the program should look like.

On the right is what the program is going to look like when it is complete.

**Implementation**

This is where you are now going to type up the code for the address card.

Try and do the code yourself but these notes are here to help you if you get stuck.

# Address Card

# Your Name

# Todays Date

#prompt user for information

name = raw\_input("Please type in your name ")

address = raw\_input("Please type in your address ")

town = raw\_input("Please type in your town ")

phone\_number = raw\_input("Please type in your phone number ")

postcode = raw\_input("Please type in your postcode ")

#print the output on the screen

print "Name: ",name

print "Address: ",address

print "Town: ",town

print "Phone Number: ",phone\_number

print "Postcode: ",postcode

Remember the code that is highlighted in green is what you have to explain in your Python Log worksheet that your teacher will give you.

**Testing**

Run some tests to see how your program handles. Run some normal tests by using some other names but also try some extreme testing. For example, what happens if you enter a number instead of a name, or if you enter a double-barrelled name, or if you just press enter without typing anything?

**Evaluation**

The evaluation of your program should answer the following questions.

1. Is the program fit for purpose? (Does it do what is required by the specification?)
2. Is the user interface good to use? Could it be improved?
3. Is the program coding readable? (Have you used sensible variable names and used internal commentary?)

Activity 8 Extra

1. Adapt the program to create a band/singer profile card. This profile card should have information such as band/singer name, nationality, biggest selling hit, your favourite song and their album name.
2. Adapt the program to create a personal birthday message. This should ask for the users name and then display a message in the format. “Happy Birthday [your name], hope you have a great day”. Hint: use the comma

Activity 9

Now that you have done a few programs let’s see if you actually understand the code you are writing. From the Address Card example you done:

1. What is the use of the # at the start?
2. What does the command word “print” do?
3. What does “raw\_input” used for?
4. What is “address” an example of?
5. What is a variable?

Variables

The programs you have completed have all processed words such as your name and address. The address card activity used a phone number but handled this as if it was a string. If a program is using numbers we need to tell it to expect a number instead of a string. The reason for this is that computers store different types of data in different ways.

In this course we look at three data types

* Integers – whole numbers
* Real Numbers – numbers with decimal point
* Strings – anything containing text
* Boolean – can only ever be true or false

Activity 10

Group the different types of data below into the 3 type.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 120 | 1.05 | book | PH33 7BQ | -1000 |
| 5700 | 1.00091 | -1.111112 | 44211 | 1 |
| 998.01 | 234442 | 11 | Lady GaGa | -9999.1 |
| -99 | Mr Stewart | 0 | 9992 | 8.4 |

Assignment

As soon as you assign a value to a variable it takes that value and we can assign values in different ways.

* wage = 200
  + assigns 200 to the variable wage
* name = "John"
  + assigns John to the variable name
* wage =float(raw\_input(“What is your wage”))
  + whatever value is typed in is assigned to wage
* name = raw\_input(“What is your name”)
  + whatever string is typed in is assigned to name

When Python executes these statements during a program, it sets up a storage space of the appropriate type, in the computer's RAM, and labels it with the variable name given.

Storage Boxes

If you are still confused with variables an easy way to remember it is with a storage box analogy. A variable in really simple terms is just like a storage box. You may have packed up old stuff in a box and put it in your loft and put a label on the front so you know what is in it and can find it again. This is kind of the same as a variable.

Let’s look at the variable name. Name is used to store a person’s name. We have a storage box called name. The box stores the contents of whatever has been assigned to it. So if the user has said their name is Jim. Jim is stored inside the storage box.

Name

Jim

So in simple terms, the variable Name is storing Jim. If the program is run by a different person with a different name then the storage box Name will change the value it is storing.

****Activity 11 – B&Q Slab Calculator

**Program Specification:** Design, write and test a program to:

* Input two whole numbers (the number of slabs wide and number of slabs deep)
* Multiply them together (number of slabs needed = number wide \* number deep)
* Input the price of a single slab
* Multiply to get the total price
* Display the results (number of slabs required and total cost). The program should work for any numbers.

**Computational Thinking (Analysis/Design)**

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Number of slabs wide * Number of slabs deep * Cost of a slab |  | * Number of slabs * Total cost |

|  |
| --- |
| **User Interface** |
| Welcome to B&Q  How many slabs wide? 5  How many slabs deep? 10  Cost of one slab (£)? 2  Number of slabs required: 50  Total Cost (£): 100  Thanks for using B&Q |

**REMEMBER!**

A user interface is what the program looks like when you are using it.

**Implementation**

#B&Q Slab Calulator

#Mr Stewart

#11/7/12

#get information from the user

slabsWide = int(raw\_input("How many slabs wide? "))

slabsDeep = int(raw\_input("How many slabs deep? "))

costPerSlab = float(raw\_input("How much does it cost for one slab? "))

# Calculate how many slabs needed

totalSlabs = slabsWide \* slabsDeep

# Calculate how much the total cost will be

totalCost = totalSlabs \* costPerSlab

print

print

print "You require", totalSlabs," slabs"

print "The total cost is £", totalCost

**Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | slabsWide | costPerSlab | Expected Output | Actual Output |
| **Normal** |  |  |  |  |
| **Extreme** |  |  |  |  |
| **Exceptional** |  |  |  |  |

**Evaluation**

The evaluation of your program should answer the following questions.

1. Is the program fit for purpose? (Does it do what is required by the specification?)
2. Is the user interface good to use? Could it be improved?
3. Is the program coding readable? (Have you used sensible variable names and used internal commentary?)

Activity 11 Extra

Adapt the program to create a tile calculator. This will follow the same program specification as the slab example. The amount of tiles will be calculated as horizontal \* vertical. I want the currency to be dollars. Please make the program nice to use and be courteous to the customer. The company name is The Tile Shop (Thanks for shopping with us, Welcome to.)

Arithmetical Expressions

In the B&Q slab calculator you have carried out two simple multiplications.

* totalSlabs = slabsWide \* slabsDeep
* totalCost = totalSlabs \* costPerSlab

Other calculations are carried out in a similar way. Some of the symbols are the same in normal arithmetic however some are different.

* For adding, use +
* Subtraction –
* Multiplication \*
* Division /
* **Raising to a power \*\***

Activity 12

For each of the following tasks you should:

1. Fill in a data flow diagram (IPO)
2. Design a user interface
3. Write up the coding
4. Test the program with normal, extreme and exceptional data
5. Evaluate the program and your performance
6. Design, write and test a program to calculate the average of six test marks.
7. Design, write and test a program to calculate the volume of a cylindrical water tank, using the formula: volume = r2h (r = radius of tank, h = height of tank).
8. Design, write and test a program to calculate the number of points gained by a football team, given the number of wins, draws and lost games, assuming a win is worth 3 points, a draw 1 point, and no points for a lost game.

****Activity 13 – Shinty Team Goal Manager

**Program Specification:** Design, write and test a program to:

* Prompt the user to enter a player's name, team name, shirt number and goals scored in game 1, 2 and 3
* Calculate the player's average score, rounded to the nearest whole number
* Display the player's name, team name, shirt number and average goals scored clearly.

**Computational Thinking (Analysis/Design)**

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Player’s name * Team name * Shirt number * Goals 1 * Goals 2 * Goals 3 |  | * Players name * Team name * Shirt number * Average goals |

|  |
| --- |
| **User Interface** |
| Shinty Team Goal Manager  What is the player’s name?  What team does he/she play for?  What shirt number do they wear?  Game 1: Goals Scored?  Game 2: Goals Scored?  Game 3: Goals Scored?  Players Name:  Team:  Shirt Number:  Average Goals:  Thank You for using the Camanachds Shinty Goal Manager. |

**Implementation**

#Shinty Team Goal Manager

#Your Name

#Todays Date

name = raw\_input("What is your name? ")

team = raw\_input("What team do you play for? ")

number = raw\_input("What is your shirt number? ")

goals1 = int(raw\_input("How many goals did you score in game 1? "))

goals2 = int(raw\_input("How many goals did you score in game 2? "))

goals3 = int(raw\_input("How many goals did you score in game 3? "))

average\_goals = round(((goals1+goals2+goals3)/3.0),0)

print "Name: ",name

print “Team: ”,team

print "Shirt Number: ",number

print "Average goals: ",average\_goals

**Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | goals1 | goals2 | Expected Output | Actual Output |
| Normal |  |  |  |  |
| Extreme |  |  |  |  |
| Exceptional |  |  |  |  |

Checkpoint

|  |  |  |  |
| --- | --- | --- | --- |
| **You have learnt a lot in Python so far. How do you feel about what you have learnt?** | **Red** | **Amber** | **Green** |
| Analyse a problem using a data flow diagram |  |  |  |
| Use string and numeric variables |  |  |  |
| Use raw\_input to get information from the user |  |  |  |
| Use print to output text and variables |  |  |  |
| Write python code for simple calculations. Average Score |  |  |  |
| Test a program using normal, extreme and exceptional data |  |  |  |
| Evaluate a program |  |  |  |

Python Procedures

The programs you have been writing so far do not really have much structure. A good programmer will have good structure and readability in their code. To do this programs are broken down into **smaller chunks known as subroutines**. This **avoids unnecessary duplication of code and makes the design easier to manage and understand**.

Activity 14 – Using Procedures

To start putting procedures into our programs we will look back at the Shinty Team Goal Manager example. The data flow diagrams you have been doing are a simple and easy way to help you start using procedures.

Below is the data flow diagram for Shinty Team Goal Manager. Notice we have 3 clear chunks, input, process and output. Inside the process part was where you calculated the average score so we can put that in there.

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Player’s name  Team name  Shirt number  Goals 1  Goals 2  Goals 3 | Calculate average goals scored | Players name  Team name  Shirt number  Average goals |

Now that we have the three blocks / procedures of our code we can start changing the code to implement procedures.

**Stage 1:** First Procedure, getData. Taking the inputs from the user.

name = raw\_input("What is your name? ")

team = raw\_input("What team do you play for? ")

number = raw\_input("What is your shirt number? ")

goals1 = int(raw\_input("How many goals did you score in game 1? "))

goals2 = int(raw\_input("How many goals did you score in game 2? "))

goals3 = int(raw\_input("How many goals did you score in game 3? "))

Before

After

def getData():

name = raw\_input("What is your name? ")

team = raw\_input("What team do you play for? ")

number = raw\_input("What is your shirt number? ")

goals1 = int(raw\_input("How many goals did you score in game 1? "))

goals2 = int(raw\_input("How many goals did you score in game 2? "))

goals3 = int(raw\_input("How many goals did you score in game 3? "))

return name, team, number, goals1, goals2, goals3

So the getData procedure is now used. Notice the brackets after “getData():” these are used to put **parameters** in. We will look at parameter passing soon. The last line of code using return outputs the variables that we will need in another part of the program. We need to return name, team, number because we will use that in our output and goals1, goals2 and goals 3 will be used in another procedure to calculate the average.

**Stage 2:** Second Procedure, performCalculation. Calculating the average goals scored.

Before

average\_goals = round(((goals1+goals2+goals3)/3.0),0)

After

def performCalculation(goals1,goals2,goals3):

average\_goals = round(((goals1+goals2+goals3)/3.0),0)

return average\_goals

Hopefully by now you are starting to see the similarities between getData and performCalculation. In this procedure you see **parameters being passed** into this procedure. This is because **to calculate the average it needs the goals variables so this needs to be passed into it.** The procedure / block does not know about the whole program it only thinks about the calculation part of working out the average so it is our job to give it the correct variables to be able to work that out. You may also note that on the last line of the procedure you **only need to return the average\_goals** as in the program specification you were only asked to display the average.

**Stage 3:** Third Procedure, displayResults. Printing out the required information

Before

print "Name: ",name

print “Team: ”,team

print "Shirt Number: ",number

print "Average goals: ",average\_goals

After

def displayResults(name, team, number, average\_goals):

print "Name: ",name

print “Team: ”,team

print "Shirt Number: ",number

print "Average goals: ",average\_goals

Remember **only pass the required parameters needed** into a certain procedure. Notice there is **no return on the last line**. This is because this procedure does not need to return anything as it is displaying the results.

**Stage 4:** Setting up the main program to run the code.

The last part is to write the code that will **actually run the program** and use the procedures. To do this we create a main program at the bottom. **This is NOT a procedure**.

#Main Program

name,team,number,goals1,goals2,goals3 = getData()

average\_goals = performCalculation(goals1,goals2,goals3)

displayResults(name,team,number,average\_goals)

In this part you are declaring the variables so Python knows what they are and also telling Python in what order to run the code. Make sure your ordering of the parameters going in are the same as the ones in your procedures or it can cause problems.

Activity 15

1. What is a procedure?
2. Why are procedures used?
3. What were your procedure names?
4. What is parameter passing?
5. Why is there no return on the last procedure of displaying the results?
6. State one job the main program at the bottom of your code is doing.

Activity 16

Now that you know how to use procedures re-open the B&Q Slab Calculator example and implement procedures within that program.

Python – Making Choices

|  |  |
| --- | --- |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| = | Equal to |
| != | Not equal to |
| == | Equivalent |

So far, all the programs you have written follow the same list of steps from beginning to end, whatever data you input. This limits the usefulness of the program.

In this section, you will learn how to make programs that do different things depending on the data that is entered. This means that you can **write programs with choices** for the user, and with different options and branches within them.

We will use these symbols in this section



Activity 17: ATM Credit Limit

**Program Specification:** Design, write and test a program to:

* Take a number entered by the user
* Compare it with a credit limit (250)
* Report “over the credit limit” if the number is over 250

**Computational Thinking (Analysis/Design)**

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Any number |  | “over credit limit” message on screen IF appropriate |

|  |
| --- |
| **User Interface** |
| Bank of Scotland ATM  How much do you want to withdraw? 100  Within credit limit – withdrawal allowed  Thank you for banking with us  **OR**  Bank of Scotland ATM  How much do you want to withdraw? 280  Over credit limit – withdrawal denied  Thank you for banking with us |

**Implementation**

# Bank of Scotland ATM

# Your Name

# Todays Date

def atm():

amount = float(raw\_input("How much do you wish to withdraw? "))

if amount > 100:

print "over credit limit - withdrawl denied"

else:

print " Within credit limit - withdrawl allowed"

#Main Program

atm()

**If you are experiencing errors with your code make sure the code is indented in the correct place as Python will not work otherwise.**

**Testing**

|  |  |  |  |
| --- | --- | --- | --- |
|  | amount | Expected Output | Actual Output |
| Normal |  |  |  |
| Extreme |  |  |  |
| Exceptional |  |  |  |



Activity 17 Extra

1. Modify the program so that it asks your age, and gives you the message “You can learn to drive” if you are 17 or over.
2. Modify the program so that it asks your year of birth and gives you the message “You can buy a lottery ticket”. (You have to be 16 to buy a lottery ticket. Hint: > a certain year)
3. ****Modify the program so that it asks for the temperature outside and gives you the message “You should put sun cream on” if the temperature is over 16 degrees Celsius.



Activity 18: Multiple IF - Mr Bonanza’s Price Draw

In the ATM example we only had one IF. You are now going to create a program that there is more than one result. It is a simple competition where you enter a number between 1 – 5 and see what your prize is.

**Program Specification:** Design, write and test a program to:

* Prompt the user to enter a number between 1 and 5
* Store the number
* Output an appropriate message:

Enter a 1 -> "You have won a iPad"

Enter a 2 -> "You have won a holiday to Florida"

..., etc.

(No prize if the number is not between 1 and 5).

**Computational Thinking (Analysis/Design)**

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Any number |  | Appropriate message on screen |

**Design**

|  |
| --- |
| **User Interface** |
| Mr. Bonanza’s Prize Draw  Enter a number between 1 and 5: 1  Congratulations you have won an iPad  **OR**  Mr. Bonanza’s Prize Draw  Enter a number between 1 and 5: 7  Unlucky. That number does not get you a prize. |

**Implementation**

#Mr Bonanza’s Prize Draw

#Your Name

#Today's Date

def choice():

number = int(raw\_input("Enter a number between 1 and 5 "))

return number

def prizeDecider(number):

if number == 1:

print "Number ",number," wins you an iPad"

elif number == 2:

print "Number ",number," wins you a holiday to Florida"

elif number == 3:

print "Number ",number, " wins you a Samsung Smart TV"

elif number == 4:

print "Number ",number," wins you a free dinner at The Hard Rock Cafe"

elif number ==5:

print "Number ",number," wins you a pie from Nevis Bakery"

else:

print "Number ",number," Unlucky, that number does not win you a prize"

#Main Program

number = choice()

prizeDecider(number)

**Testing**

|  |  |  |  |
| --- | --- | --- | --- |
|  | number | Expected Output | Actual Output |
| Normal |  |  |  |
| Extreme |  |  |  |
| Exceptional |  |  |  |

Activity 18 Extra

Adapt the Mr Bonanza’s Prize Draw program to fulfil this new specification.

Design, implement and test a program that asks the user to enter a grade (A, B, C, D or F), and gives you messages like 'A means you got over 70%', 'B means you got between 60% and 70%', 'C means you got between 50% and 60%', 'D means you got between 40% and 50%' and 'F means you failed'. If any other letter is input give a message saying “Grades can only be A, B, C, D or F”.

Activity 19: Using AND

SQA Exam Grader

**Program Specification:** A program is required that could be used to assign grades to exam marks automatically. Over 70% is an A, over 60% is a B, over 50% is a C, over 45% is a D, and less than 45% is a fail.

Design, write and test a program to:

* Prompt the user to enter the highest possible mark for an exam
* Prompt the user to enter a student's name (first name and surname)
* Prompt the user to enter the student's mark
* Calculate the percentage mark
* Display a message displaying the student's initials, percentage and grade.

**Computational Thinking (Analysis/Design)**

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Student first name * Student surname * School * Highest possible mark * Students mark |  | * Students Initials * School * Percentage Mark * Grade |

**Design**

|  |
| --- |
| **User Interface** |
| SQA Exam Grader  Enter student's first name: Bob  Enter student's surname: Simpson  Enter school name: Kinlochleven High School  Enter the top amount of marks available: 125  Enter student's mark: 99  Students Initials: BS  School: Kinlochleven High School  Percentage (%): 79  Grade: A  Thank you for using the SQA Exam Grader |

**Implementation**

This is the code for the SQA Exam Grader. **Put procedures in** to make this more structured. (**If you get really stuck** have a look at the next page)

# Exam Mark Grader

# Your Name

# Todays Date

name = raw\_input("Enter the students first name: ")

surname = raw\_input("Enter the students surname: ")

school = raw\_input("Enter the students school: ")

max\_mark = int(raw\_input("Please input the maximum mark available: "))

mark = int(raw\_input("Please input the students mark: "))

percent = round(mark \* 100 / max\_mark,0)

if percent >=70:

grade = "A"

if percent >=60 and percent <70:

grade = "B"

if percent >=50 and percent <60:

grade = "C"

if percent >=45 and percent <50:

grade = "D"

if percent <45:

grade = "FAIL"

initial1 = name[0]

initial2 = surname[0]

print “-----------------------------“

print "SQA Exam Grader"

print "Student ",initial1,initial2

print "School", school

print "Percentage ",percent,"%"

print "Grade ",grade

**Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | max\_mark | name | Expected Output | Actual Output |
| Normal |  |  |  |  |
| Extreme |  |  |  |  |
| Exceptional |  |  |  |  |

Activity 19 Extra

A new grade called A+ has been introduced for marks of 80% and over. Change the coding to reflect this new grade. Remember to change the condition for an A as well as introducing a new condition for A+.

This is the code for Activity 19 with procedures.

**Only use if you get really stuck.**

# SQA Exam Grader

# Your Name

# Todays Date

def getData():

name = raw\_input("Enter the students first name: ")

surname = raw\_input("Enter the students surname: ")

school = raw\_input("Enter the students school: ")

max\_mark = int(raw\_input("Please input the maximum mark available: "))

student\_mark = int(raw\_input("Please input the students mark: "))

percent = round(student\_mark \* 100 / max\_mark,0)

return name,surname,school,percent

def performCalculation(percent,name,surname):

if percent >=70:

grade = "A"

if percent >=60 and percent <70:

grade = "B"

if percent >=50 and percent <60:

grade = "C"

if percent >=45 and percent <50:

grade = "D"

if percent <45:

grade = "FAIL"

initial1 = name[0]

initial2 = surname[0]

return grade,initial1,initial2

def displayResults(initial1,initial2,school,percent,grade):

print "SQA Exam Grader"

print “-----------------------------------------------“

print "Student ",initial1,initial2

print “School”, school

print "Percentage ",percent,"%"

print "Grade ",grade

#Main Program

name,surname,school,percent = getData()

grade,initial1,initial2 = performCalculation(percent,name,surname)

displayResults(initial1,initial2,school,percent,grade)

Activity 20: Using AND

DVLA Age Checker

**Program Specification:** A program is required that asks the user to enter their age, then displays one of the following messages, as appropriate:

* Sorry, you can't drive (if you are under 16)
* You can only drive a moped (if you are 16)
* You can only drive a car or moped (if you are 17–20)
* You can drive any vehicle (if you are 21–74)
* You need a medical check (if over 75).

**Use your previous example to help you implement AND**.

**(Try and use procedures in this example, if you are struggling then leave them out)**



Activity 21: Using OR

Dulux Colour Chooser

**Program Specification:** Design, write and test a program that asks the user to enter a letter, and prints the word:

* Red if R or D is entered
* Green if G or N is entered
* Blue if B or E is entered
* Yellow if Y or W is entered
* Black if C, A or K is entered.

The program should **respond to both upper case and lower case inputs**.

Here is one line of the code. Can you do the rest on your own?

if colour == "R" or colour == "r" or colour == "D" or colour == "d":

print "The colour you have chosen is red."



Hint: **this is a multiple if** (using if and elif) so have a look back at Mr Bonanza’s Prize Draw if you get stuck.

Activity 22: FOR Loops

So far, every program you have written starts at the beginning, executes each line once, then stops at the end. If you want to repeat the program you have to run it again. It is **often useful in a program to be able to repeat a line or group of lines automatically**.

Before we look at how to use the shortcut to repeat an instruction lets look at how you would print a message ten times.

#Repeat Message (Bad Programming)

#Your Name

#Todays Date

print “Hey how are you?”

print “Hey how are you?”

print “Hey how are you?”

print “Hey how are you?”

print “Hey how are you?”

This code would simply print on the screen “Hey how are you?” 5 times. This is not a good way of repeating instructions. Lets look at a much better way using a FOR loop.

This second version of the program that aims to do the exact same thing of outputting the same message 5 times can be shortened down to this.

#For Loop (Good Programming)

#Your Name

#Todays Date

for counter in range(5):

print “Hey how are you?”

As you can see the amount of coding has significantly dropped. The range number inside the brackets is the amount of times you want the loop to run for. So in this instance five.

1. Do the above examples and get the loop working.
2. Modify the code so that it prints “Hello” 25 times
3. Modify the code so that it prints “I am so intelligent” 100 times
4. Modify the code so that it prints “I must not swing on my chair” 1000 times

Activity 22 Extra

The program would be much more useful if it was possible to make changes to the message and the number of times it was displayed, without having to alter the coding each time. This can be achieved by using variables.

message = raw\_input("What would you like the message to be? ")

how\_many = int(raw\_input("How many times do you want it repeated? "))

for counter in range(how\_many):

print message

Change the coding as follows:

Can you adapt the program even further to print out like this:

|  |  |  |  |
| --- | --- | --- | --- |
| Tick  Tock  Tick  Tock  Tick  Tock  Tick  Tock  Tick  Tock  Tick  Tock | Left  Right  Left  Right  Left  Right  Left  Right  Left  Right  Left  Right | Na na na na na na na na  Batman  Na na na na na na na na  Batman  Na na na na na na na na  Batman  Na na na na na na na na  Batman  Na na na na na na na na  Batman  Na na na na na na na na  Batman | Vengaboys are back in town  We like to party  Vengaboys are back in town  We like to party  Vengaboys are back in town  We like to party  Vengaboys are back in town  We like to party  Vengaboys are back in town  We like to party  Vengaboys are back in town  We like to party |

Hint: You will need to lines of code within the FOR loop.

For Loop

For Loops are examples of **fixed loops**. This is because the programmer fixes the amount of times the action is repeated in advance.

for counter in range(*number*)

*action*

Note that we have called the loop variable 'counter' (because that is what it does), but it can be called anything you like. Also ‘number’ can be any number put in by the programmer.

Activity 23: Counting Program

**Program Specification:** Design, write and test a program to print 1, 2, 3, 4, 5 ... 99, 100.

**Computational Thinking (Analysis/Design)**

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
|  |  | 1, 2, 3, 4, 5 … 99, 100 on screen |

|  |
| --- |
| **User Interface** |
| Counting Program  1,2,3,4,5,6 ……. 99, 100 |

**Implementation**

#Counting program

#Your Name

#Todays Date

for counter in range(100):

print counter

Instead of a message being displayed like before the value of the counter is now being printed.

**Testing**

Does it count from 1 – 100?

Change the numbers in the brackets and fill in the table.

|  |  |
| --- | --- |
| **Code** | **Results** |
| for counter in range (1000) |  |
| for counter in range (1,100) |  |
| for counter in range (1,101) |  |
| for counter in range (2,101,2) |  |
| for counter in range (0,101,10) |  |
| for counter in range (1,1000,100) |  |

The data inside the brackets corresponds to (start number, finish number, step size).

**Can you get Python to output a counter for the 3 times table up to 36?**

Activity 24: Premier Inn Booking System

**Program Specification:** Design, write and test a program to:

* Prompt the user to the category of room they would like
* Prompt the user for the cost per night
* Calculate the cost for up to 14 nights
* If category of the room is a “single” the user gets a discount

**Computational Thinking (Analysis/Design)**

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Category of room * Cost per night |  | * Cost per night * Whether or not the user gets a discount |

|  |
| --- |
| **User Interface** |
| Welcome to the Premier Inn  What category of room would you like? Double  Cost per night (£)? 2  Number of Nights Cost (£)  1 2  2 4  3 6  …  Thank You for staying with Premier Inn |

**Implementation**

def getDetails():

print "Welcome to Premier Inn"

roomType = raw\_input("Category of room ")

costPerNight = float(raw\_input("Cost per night "))

return roomType, costPerNight

def outputInformation(roomType, costPerNight):

print "Number of nights", "Cost £"

for counter in range(1,15):

print counter," ",counter \* costPerNight

if roomType =="single":

print "You are entitled to a discount voucher"

#Main program

roomType, costPerNight = getDetails()

outputInformation(roomType,costPerNight)

Activity 25: While Loops

Now that you have learnt how to code For loops we will now go on to look at a different kind of loop. For loops are used when you know how many times something needs to be repeated. While loops are used when you do not know the amount of repetitions known in advance.

For example, a quiz program might give the user repeated chances to get the answer correct. The programmer doesn't know in advance whether the user will get the question right first time, or take 2, 3, 4 or more attempts.

Here is a simple example of how we code a While loop.

def question():

userAnswer = int(raw\_input("What is 2+2? "))

return userAnswer

def process(userAnswer):

correctAnswer = 4

while userAnswer != correctAnswer:

userAnswer = int(raw\_input("Sorry. Wrong Answer. Try Again "))

print "Well Done"

#Main Program

userAnswer = question()

process(userAnswer)

**Mini Quiz Using While Loops**

1. Now you can create your own while loop using the code above to help.
2. Adapt the program to ask a different question. What is 100 – 50? Does it still work?
3. Now try a general knowledge question? Who won the X-Factor last year? How high is Ben Nevis?

Activity 25 Extra

You are now going to add a counter into your program to tell you how many times it took you to get the correct answer. To do this we add a few extra lines of code.

def question():

userAnswer = int(raw\_input("What is 2+2? "))

return userAnswer

def process(userAnswer):

counter = 0

correctAnswer = 4

while userAnswer != correctAnswer:

counter = counter + 1

userAnswer = int(raw\_input("Sorry. Wrong Answer. Try Again "))

print "Well Done you took ",counter," tries to get that right"

#Main Program

userAnswer = question()

process(userAnswer)

After you have finished your mini quiz with a new question ask a friend to attempt to answer it. Make sure they don’t see your code though or else they may see the answer.



Activity 26: Bank of Scotland PIN Protection

**Program Specification:** Design, write and test a program for a bank cash machine.

* The program should prompt the user to enter their PIN.
* If the PIN is correct, it should display "Welcome to the Bank of Scotland" (message 1).
* If not, it should notify the user that their PIN was entered wrongly (message 2), and let them try again, but only allowing three tries.
* If the user enters their PIN wrongly three times, they should be warned that their card is being kept (message 3).

**Computational Thinking (Analysis/Design)**

Data Flow Diagram

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| PIN entered |  | Appropriate message |

|  |
| --- |
| **User Interface** |
| Bank of Scotland  Please enter your PIN: 1234  Welcome to Bank of Scotland  **OR**  Bank of Scotland  Please enter your PIN: 1112  Sorry your PIN is incorrect. Please try again. |

**Implementation**

#Bank of Scotland

#Your Name

#Todays Date

def getPin():

pin= int(raw\_input("Please Now Enter You Pin"))

return pin

def ATM(pin):

counter=0

correct\_pin= 1234

while pin != correct\_pin and counter < 2:

counter = counter + 1

pin = int(raw\_input("Incorrect Pin, Please Try Again"))

if counter == 2 and pin != correct\_pin:

print "Your PIN has been entered incorrect too many times. We have retained your card."

if pin == correct\_pin:

print "Correct Pin!"

print "Welcome To The Bank Of Scotland"

#Main program

pin=getPin()

ATM(pin)

Activity 26 Extra

Amend the Bank of Scotland program to now let you be able to have 5 tries with your PIN. Also change the bank name to something of your own choice. Also change the PIN to something of your own choice.

Checkpoint

|  |  |  |  |
| --- | --- | --- | --- |
| **You have learnt a lot in Python so far. How do you feel about what you have learnt?** | **Red** | **Amber** | **Green** |
| Analyse a problem using a data flow diagram |  |  |  |
| Identify string and numeric variables |  |  |  |
| Use Input and Print statements |  |  |  |
| Test a program using normal, extreme and exceptional data |  |  |  |
| Evaluate a program |  |  |  |
| Use conditional statements involving If, Else, Elif |  |  |  |
| Use simple and complex conditions involving comparison operators, AND, OR |  |  |  |
| Create fixed loops using For |  |  |  |
| Create conditional loops using While |  |  |  |
| Make use of the loop counter within a loop |  |  |  |

Outcome 1: Practice Assignment

Questions Set 1: Python

**Q1:** Read the following carefully:

|  |  |
| --- | --- |
| Line 1 | ***price\_per\_night = 10*** |
| Line 2 | ***number\_of\_nights = int(raw\_input("Please enter the number of nights you would like to stay "))*** |
|  |  |
| Line 3 | ***if number\_of\_nights > 14:*** |
| Line 4 | ***discount = 50*** |
| Line 5 | ***else:*** |
| Line 6 | ***discount = 0*** |
|  |  |
| Line 7 | ***total\_cost = (number\_of\_nights \* price\_per\_night) - discount*** |
|  |  |
| Line 8 | ***hotel = "Travel Lodge"*** |
| Line 9 | ***print "The total cost of your stay at", hotel ,"in Edinburgh = £", total\_cost*** |

**Q1a:** Explain in the boxes below what happens when each of these lines of code is run.

|  |  |
| --- | --- |
| **Line 1:** |  |
| **Line 2:** |  |
| **Line 8:** |  |

**Q1b:** Explain what the *If* construct in lines 3 and 4 does.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q1c**: Explain (i) how the total cost in line 7 is calculated and (ii) how the *If* construct in lines 3 and 4 affects the calculation.

(i) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ii) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Outcome 1

Questions Set 2: Python

**Q2:** Read the following code carefully:

|  |  |
| --- | --- |
| Line 1 | ***total\_cost = 0*** |
|  |  |
| Line 2 | ***number\_of\_cars = int(raw\_input("Please enter number of cars "))*** |
|  |  |
| Line 3 | ***for counter in range(number\_of\_cars):*** |
| Line 4 | ***name\_of\_car = raw\_input("Please enter the name of the car ")*** |
| Line 5 | ***cost\_of\_car = float(raw\_input("Please enter the cost of the car"))*** |
| Line 6 | ***total\_cost = total\_cost + cost\_of\_car*** |
|  |  |
| Line 7 | ***print "The total cost for all the cars is £",total\_cost*** |

**Q2a:** Explain what the variable ***Number\_of\_cars*** is used for in line 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q2b:** Explain what the ***for loop*** construct in lines 3 – 6 does.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q2c:** Explain how the variable ***total\_cost*** is (i) calculated and (ii) stored in the computer.

(i) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ii) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Outcome 1

Questions Set 3: Python

**Q3:** Read the following code carefully:

|  |  |
| --- | --- |
| Line 1 | ***wages = float(raw\_input("Please enter the amount of wages in ££'s "))*** |
|  |  |
| Line 2 | ***while wages > 50:*** |
| Line 3 | ***bill = raw\_input("Please enter the name of a bill you have to pay ")*** |
| Line 4 | ***cost\_of\_bill = float(raw\_input("Please enter the cost of the bill £"))*** |
| Line 5 | ***wages = wages - cost\_of\_bill*** |
|  |  |
| Line 6 | ***print "You do not have much wages left use them wisely"*** |

**Q3a**: Explain what the **while loop** in lines 2–5 does

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q3b**: Explain how the **wages** in line 5 is **(i)** calculated and **(ii)** stored.

(i) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ii) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Q3c**: Explain what the condition ***while wages > 50*** is used for in line 2.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Outcome 2:

Practice Task 1: Flight Total Cost

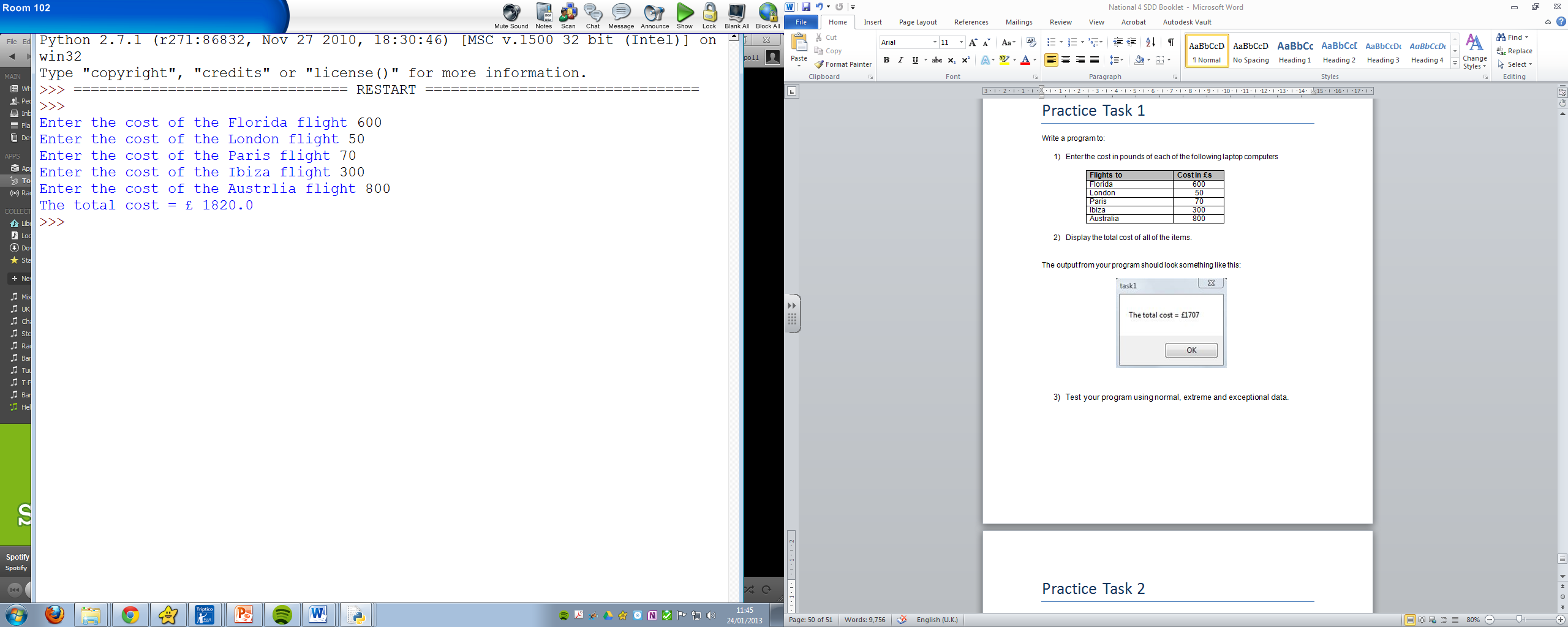
Write a program to:

1. Enter the cost in pounds of each of the following laptop computers

|  |  |
| --- | --- |
| **Flights to** | **Cost in £s** |
| Florida | 600 |
| London | 50 |
| Paris | 70 |
| Ibiza | 300 |
| Australia | 800 |

1. Display the total cost of all of the items.

The output from your program should look something like this:



1. Test your program using normal, extreme and exceptional data.

Practice Task 2: Average Speed

Write a program that:

Takes in a drivers speed for 7 trips

Calculates and displays the drivers average speed

Award a safety certificate if the average is equal to or below 30.

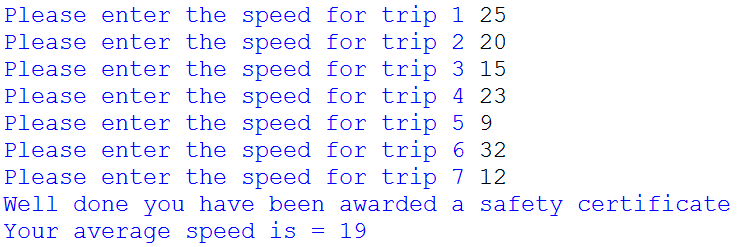
Test your program using these two sets of marks:

|  |  |
| --- | --- |
| **Set 2** | |
| Trip 1 | 42 |
| Trip 2 | 12 |
| Trip 3 | 33 |
| Trip 4 | 35 |
| Trip 5 | 62 |
| Trip 6 | 29 |
| Trip 7 | 38 |

|  |  |
| --- | --- |
| **Set 1** | |
| Trip 1 | 25 |
| Trip 2 | 20 |
| Trip 3 | 15 |
| Trip 4 | 23 |
| Trip 5 | 9 |
| Trip 6 | 32 |
| Trip 7 | 12 |

The output from your program should look something like this:

Test your program using normal, extreme and exceptional data.



Task 3: Season Ticket

Write a program that calculates the cost of a number of football season tickets. The program should check the clubs password before letting them order their season tickets.

The program should:

Store the cost of a season ticket as £200

Check the clubs password to make sure it is correct

When the correct password has been entered it should:

Ask the user to enter the number of season tickets for the year the club wants to buy

Calculate and display the cost of the tickets

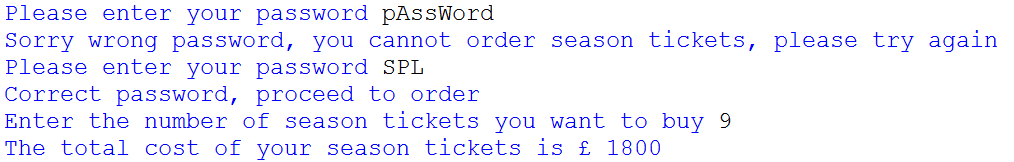
**Test data:** test your program using the following data:

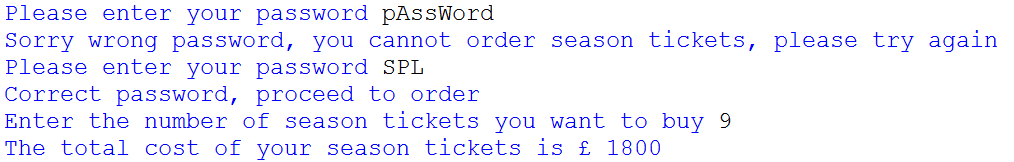
The correct password = SPL

**Password test data Ticket order test data**

|  |  |  |
| --- | --- | --- |
| pAssWord |  | 9 |
| #open |
| SPL |

The output from your program should look something like this:





Outcome 3 – Report

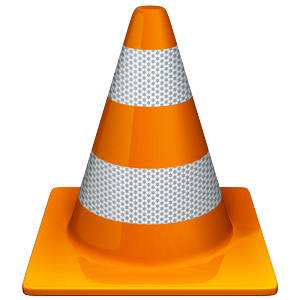
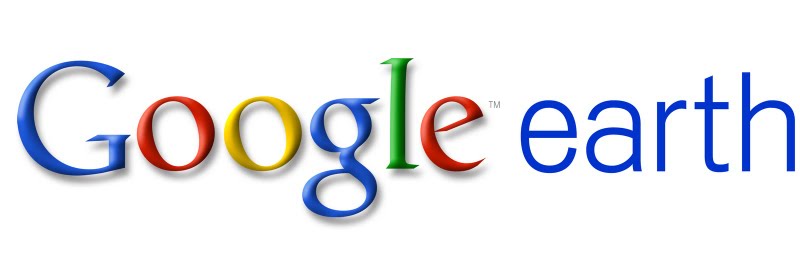
In this outcome you will produce a short factual report on a contemporary software based application by:

* Describing the application
* Explaining how its features relate to programming constructs and data types
* Describing its impact on the environment or society

**This report can be delivered in any appropriate format such as a text document, a visual presentation, a video clip, a website, a blog, or a podcast.**

Step 1: Plan

Think of an application you have **used before** and know how to use it. Below are a few of examples if you can’t think.



iTunes, Microsoft Word, Google Chrome, Microsoft PowerPoint, VLC Media Player, Google Earth, Photoshop.

Discuss the application that you want to research with your teacher. You should **record the outcome of the discussion**.

Decide how you are going to do your research. You could use a combination of: books, magazines and the internet.

Step 2: Producing the Report

a) You have to **describe the software application**.

This will mean you need to describe the purpose of the application, its **key features and also its interface**.

b) Next you have to think about how some of the **programming constructs and variables** that you have learned about are used in your chosen application.

**You could, for example, think about how the application uses integer and string variables, or how it uses *if..end if*, or how it uses loops.**

c) Then you have to consider the **effect the application has on society or the environment.**

You could think about how the application might help reduce or increase energy use or resources such as paper. You might think about how the application might affect people’s privacy, data security or online crime.

d) Finally you have to hand in your report. Don't forget to include a page at the end where you **state all the sources** of your information.

Here is a table outlining the key steps that you can follow to help you complete all the sections of your report.

|  |
| --- |
| **Step 1: A description of the chosen application, its purpose, key features and interface**  **Purpose:**  **Key features:**  **Interface:** |
| **Step 2: A description of how the chosen application makes use of software development constructs and data types**  **Use of variables:**  **Use of constructs:** |
| **Step 3: The impact of the chosen application on the environment or society** |
| **Sources:** |