**LESSON ACTIVITIES FOR**

**Introduction and Block 1: Problem Solving & Programming**

**Lesson 2 activities**

**Activity 2.7**

Write a program to solve the problem.

The problem: Display your initials 5 characters high on the screen using print commands.

X x x x

Xx x x X

X x x Xxxxxx

X xx x x

X x x x

**Lesson 5 activities**

**Activity 5.5**

**Variable names**

A programmer is trying to decide which would be a valid name for a variable which represents a house number.

Which of the following variable assignments are valid? Why are the others not valid?

|  |  |  |
| --- | --- | --- |
|  | **Valid or invalid variable name?** | **Reason why not valid** |
| 8HouseNumber = 288 | INVALID | Has the number 8 in the name. |
| houseNumber = 288 | VALID |  |
| house Number = 288 | INVALID | Has a space. |
| house\_number = 288 | VALID |  |
| import = 288 | INVALID | Isn’t a good variable. |

What type of error do you get when using an invalid variable name?

**Activity 5.6**

Copy and run the following code and explain the result.

# Programmer Amy Jones 12/8/2013

# adds two numbers

numberOne=15

numberTwo=23

answer=numberOne + numberTwo

print("The answer is “,answer)

Amend the program to add another variable called numberThree. Assign the value 76 to this variable. The answer should add up all three numbers.

**Lesson 6 activities**

**Activity 6.3**

Write a program that asks you to enter a number then displays the number doubled.

**Activity 6.4**

**String formatting**

The string method .format gives you more control over the formatting of your output than printing using space-separated values. The string formatting commands are given in curly brackets {}.

Copy and run the following commands.

>>>foodOne="fish"

>>>foodTwo="chips"

>>>print("{0} and {1}".format(foodOne,foodTwo))

fish and chips

>>>print("{1} and {0}".format(foodOne,foodTwo))

chips and fish

>>>print("{1} {1} {1} and {0} {0} {0}".format(foodOne,foodTwo))

chipschipschips and fishfishfish

Create these variables:

one = “cheese”

two =”onion”

Use the .format command to display the following:

My favourite crisps are cheese and onion. I love them!

cheese and onion and cheese and onion and cheese and onion

cheesecheesecheese and oniononiononion

You guessed it. The best crisps are onion and … cheese.

Try altering the flavours assigned to the variables to your favourite flavour! Enjoy.

**Activity 6.7**

*The following programs should prompt for the input information and display the result appropriately.*

* Write a program that displays the square of a number.
* Write a program that prompts for a number and then displays the cube of a number.
* Write a program to find the perimeter of a square.
* Write a program to find the perimeter of a rectangle.
* Write a program that finds the area of a square.
* Write a program that finds the area of a cube.
* Write a program to convert from pounds to euros.

**Lesson 7 activities**

**Activity 7.1**

**Relational operators**

Password checking is an example of a relational operator. If the password entered is the same as password stored then the condition is true. The operator is “is equal to”.

Brainstorm other examples of condition statements and decide what the operator may be.

|  |  |
| --- | --- |
| **Relation statement** | **Operator** |
|  |  |
|  |  |
|  |  |

**Activity 7.2**

Complete the table of the Python relational operators.

Give an example of each and say whether it will evaluate to true or false. Try out your expression by typing it into the Python interactive shell.

|  |  |  |  |
| --- | --- | --- | --- |
| **Relational operator** | **Operator** | **Example** | **Evaluates to** |
| Equal to |  |  |  |
| Not equal to |  |  |  |
| Greater than |  |  |  |
| Greater than or equal to |  |  |  |
| Less than |  |  |  |
| Less than or equal to |  |  |  |

**Activity 7.3**

**Greater than and less than**

Find a way that works for you of remembering the difference between “less than” < and “greater than” >.

*Hint: As there are only two options you only need to learn one!*

|  |  |  |
| --- | --- | --- |
| **Operator** | > | < |
| **Operator meaning** | Greater than | Less than |
| **How I remember this** |  |  |

**Activity 7.6**

Write a program that asks you to enter the colour of the light at a pedestrian crossing. If the light is green it tells you it is safe to cross otherwise it tells you to stop.

**Activity 7.7**

Write a program that asks you your password. It then asks you to re-enter your password. If they are the same the message “access granted” is displayed. If the passwords are not the same the message “access denied” is displayed.

**Lesson 8 activities**

**Activity 8.2**

**Writing readable code: a style guide**

Program code is read more often that it is written. Here are some guidelines on how to write Python code to improve the readability of the code.

**A style guide for Python code**

* Use the 4 spaces indentation for each indentation level
* Separate functions by 2 blank lines
* Use blank lines to separate different parts of the program
* Use meaningful names for variables using CamelCase or with words separated by underscores
* Put imports at the top of the file
* Include one space around each side of an assignment and other operators
* Comments should be complete sentences with the first word capitalised
* Comments should add clarity to explain what the program does and not just repeat what the code already says
* Function names should be written in lowercase with words separated by underscores
* Use meaningful function names which describe the purpose of the function
* Constants are written in CAPITAL\_LETTERS
* Use meaningful constant names which describe the purpose of the constant

Note: Functions are sub-programs which start with the “deffunction\_name():. Constants are variables which never change. You will be covering functions in a later lessons.

For more detail see <http://www.python.org/dev/peps/pep-0008/>

Implement the style guide for this Python code.

def a(s):

if s<50:

print("You have lost")

else:

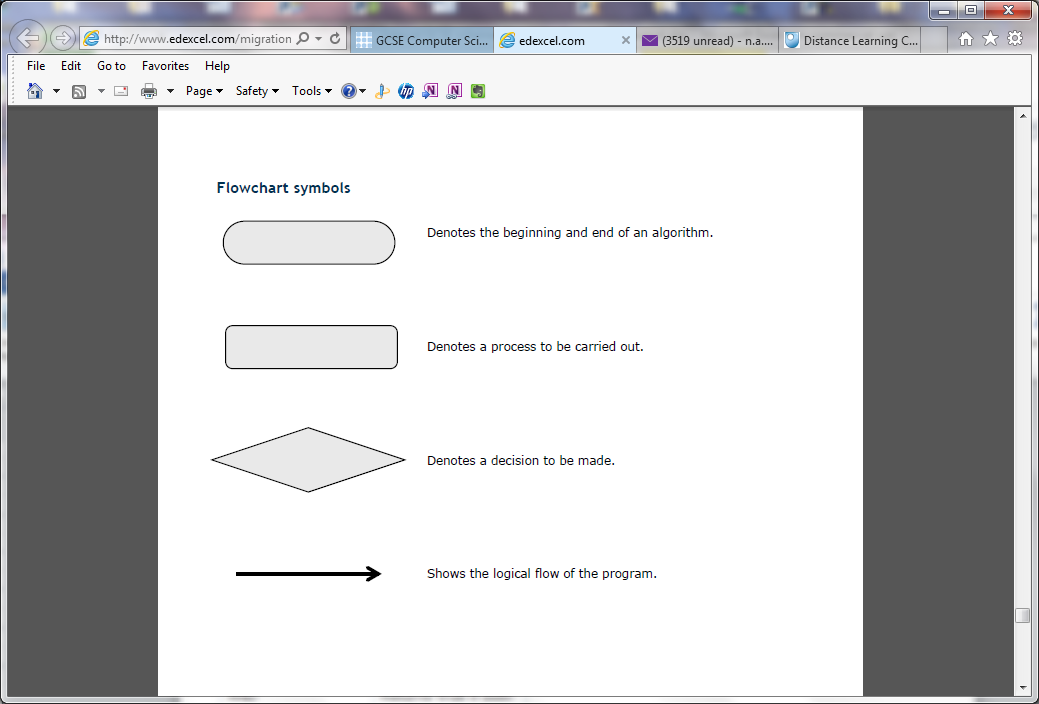
print("You have won")

**Activity 8.3**

Select some of your programs and make them more readable using the style guide.**Lesson 9 activities**

**Activity 9.1**

**Flowcharts**

Flowcharts can be used to represent algorithms. Identify the flowchart symbols.

**Activity 9.5**

**Boolean Operators**

Complete the table using the logical (Boolean) operators AND, OR and NOT.

|  |  |
| --- | --- |
| **Symbol** | **Description** |
|  | Returns true if both conditions are true. |
|  | Returns true if any of the conditions are true. |
|  | Reverses the outcome of the expression; true becomes false, false becomes true. |

**Activity 9.6**

Predict the answer to the conditions and then use the interactive shell to test your answer.

|  |  |  |
| --- | --- | --- |
| **Condition** | **Your prediction (true or false?)** | **Result** |
| (78 = 10) **or** (6 = 7) |  |  |
| (78 == 10) **or** (6 == 6) |  |  |
| (78 == 10) **and** (6 == 6) |  |  |
| (1 < 10) **and** (2 < 10) |  |  |
| (1 < 10) **or** (2 < 10) |  |  |
| not ( 5 ==5) |  |  |
| not (6 < 4) |  |  |

*Hint: Try asking yourself the question:*

*Is condition\_1 true* ***OR*** *condition\_2 true – if YES then the answer is true.*

*Is condition\_1 true AND condition\_2 true – if YES then the answer is true.*

**Activity 9.7**

Make up some boolean operator questions of your own. Check your answers are correct by using the interactive Python shell. Try them out on other students. You must be able to explain the correct answer.

An example question is given below.

>>> answer = 50

>>> (answer <40) or (answer >80)

True or False?

**Activity 9.8**

A truth table lists all the possible combinations of true and false outcomes for each condition.

Complete the truth table for AND and OR operators.

**Truth table showing true and false AND conditions**

|  |  |  |
| --- | --- | --- |
| **Condition 1** | **Condition 2** | **Output** |
| false | false |  |
| true | false |  |
| false | true |  |
| true | true |  |

**Truth table showing true and false OR conditions**

|  |  |  |
| --- | --- | --- |
| **Condition 1** | **Condition 2** | **Output** |
| false | false |  |
| true | false |  |
| false | true |  |
| true | true |  |

**Lesson 10 activities**

**Activity 10.4**

A variable has been created and assigned the string “watch bbcclick today”

myVariable = “watch #bbcclick today”

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| w | a | t | c | h |  | # | b | b | c | c | l | i | c | k |  | t | o | d | a | y |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |

What command would you use to extract the following slices?

#bbcclick

watch

today

Check your answer using the interactive shell.

**Activity 10.5**

**Using string methods**

Any variable with a string assigned is a member of the class called string. String methods can be used to manipulate that string.

Explore string methods by working through the commands in the interactive shell and describing what happens.

>>>myVariable="There's a starman waiting in the sky"

>>>myVariable.upper()

>>>myVariable.replace("a","x")

>>>myVariable.title()

>>>myVariable.swapcase()

**Activity 10.6**

**Review using the string .format method**

Using the string .format method to format output.

There are two ways to output information to the screen. Using the print() function or by giving the expression.

Try these now.

>>>myName="David Bowie"

>>>print(myName)

David Bowie

>>>myName

'David Bowie'

The string .format is used to give more control over formatting.

It allows you to use placeholders and then specify the variable you want to print in the string. The place holders are shown using {}.

>>> first="David"

>>> second="Bowie"

>>>print("The best music is from {0} {1}!".format(first,second))

The best music is from David Bowie!

It also allows you to choose how to format the output.

**Lesson 11 activities**

**Activity 11.1**

Complete the table below to summarise the list commands.

|  |  |
| --- | --- |
| **Things to do with lists** | **Commands** |
| Create a list |  |
| Reference an item in a list |  |
| Delete an item in a list |  |
| Append an item to the end of a list |  |

**Activity 11.2**

**Lists (arrays)**

Lists are a data structure in which you can store things. Lists are assigned to a name using square brackets:

>>>mylist=["apple","oranges","lemon","pear","lime"]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| apples | oranges | lemon | pear | lime |
| 0 | 1 | 2 | 3 | 4 |

Each item in a list is given an index location which starts from position zero (0).

Parts of a list can be referenced by giving the index location as an integer number.

List name[index]

A range of values can be displayed using:

[start index: end index]

* Start index is the index to start at (remember that indexing starts at zero)
* End position is the index AFTER the index required.

Make the list and experiment with the list commands.

>>>mylist=["apple","oranges","lemon","pear","lime"]

What does **mylist[1]** display?

What does **mylist[1:3]** display?

What does **mylist[-1]** display?

What command will display just apple?

What command will display lemon and pear?

Make a new list called myfood containing your 5 favourite foods.

Display the whole list.

Display the item at index position 3

Display the item at index position 0

Display the items at index position 1 to 4.

**Activity 11.3**

**Using lists**

Make the list which contains the class marks for Amy Jones.

Marks = ['Amy', 'Jones', 'English', 67, 'Maths', 76, 'Computer Science', 96]

Answer the following questions:

The English teacher entered the mark incorrectly; it should be 72 not 67. Alter this item in the list.

Add the marks for Physics to the end of the list. “Physics”, 65

The Maths marks are all wrong. Remove the items for “Maths” and the score 76

Write a program to find the average score for the 3 subjects.

**Activity 11.4**

**Using Python docs help**

Use Python docs to find out more about lists.

Use *help/python docs* then select the *Python tutorial* and go to *3.1.4 Lists.*

Read through the discussion of lists and try out the examples. Make a note of three more facts about lists to share in the next class.

Python is avery powerful programming language which is used in Universities and commercial organisations. You do not need to know all the details provided in the Python docs but, with practice, you should be able to find information about Python that can be very useful.

**Lesson 14 activities**

**Activity 14.4**

Write a program that acts like a dice. After each “throw” of the dice it should ask if you wish to continue and stop when you enter “Y”.

**Activity 14.5**

Write a program that asks you to guess a number between 1 and 10. It then compares your answer with a randomly generated number and stops the program when the number generated matches the number guessed.

**Activity 15.4**

**Summary sheet for Python and two-dimensional arrays**

|  |  |
| --- | --- |
| **How to use two-dimensional arrays in Python (nested lists which start from zero)** | |
| **Task** | **Example** |
| How to initialise a two-dimensional array |  |
| How to address an array element |  |
| How to assign values in a two-dimensional array. |  |
| How to print a two-dimensional array |  |

**Lesson 16 activities**

**Activity 16.3**

**Validation: presence check**

Write a program that asks the user to enter a name and uses a presence check to make sure that an answer has been entered. If nothing has been entered the user is asked to re-enter.

Extension: Give an appropriate message when the user fails to enter any characters.

**Activity 16.4**

**Validation: type check**

Write a program that asks the user to enter their age and checks that they have entered an integer. It should display a message asking them to enter a number if they have not done so.

*Hint: Use the “try except else” command*

**Activity 16.5**

**Validation**

Write a program that asks the user to enter an email address and then checks the string entered to make sure it contains an “@”. If it does not the user is asked to enter the email address again.

**Activity 16.6**

**Validation: range check**

Write a program that asks the user to enter a % of charge left in their mobile phone. Use a range check to make sure the value is less 100% or more than 0%. The user is asked to re-enter the value if it is outside the range.

Extension: Alter the program so it only allows integer numbers to be entered.

**Activity 16.7**

**Try command: divide by zero error check**

Write a program that asks for two numbers and then divides the numbers and displays the answer. If the program generates a divide by zero error display a message to explain they enter a zero as the second number.

**Activity 16.8**

Write a program that asks the user to enter their name, age and email address using validation to ensure the data contains reasonable values. It should then display the data and ask the user if it is correct. Use validation to make sure the user can reply Y, y or Yes and N,n, or No. The user should be allowed to re-enter the data if it is incorrect.

|  |  |  |
| --- | --- | --- |
| **Variables and arrays** |  |  |
| **Syntax** | **Description** | **Example** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Selection** | | |
| **Syntax** | **Description** | **Example** |
| try :  <commands1>  except:  <commands2>  else:  <commands2> | If <commands1> cause an error then <commands2> will execute. If the commands1 execute successfully then <commands3> will execute. | try:  ans=numOne/numTwo  except ZeroDivisionError:  print("Second number cannot be zero!")  else:  print("The answer is: ", ans) |

|  |  |  |
| --- | --- | --- |
| **Repetition** | | |
| **Syntax** | **Description** | **Example** |
| for variable in <expression>:  <command1>  <command2> | Executes <commands> for a fixed number of times given by <expression> | MyList=["cat","dog","cow","donkey","rabbit","canary"]  for next in myList:  print(next) |
| while <condition> :  <command1>  <command 2> | Executes the commands whilst <condition> is true. This is a pre-condition loop. | answer="N"  counter=0  while answer != "Y" :  print("are you hungry? You have been asked {0} times".format(counter))  answer = input("Please respond Y or N :")  counter=counter + 1  print("Please get something to eat!") |

|  |  |  |
| --- | --- | --- |
| **Input/output** |  |  |
| **Syntax** | **Description** | **Example** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **File handling** | | |
| **Syntax** | **Description** | **Example** |
| variable = open (“filename”, “character showing way file is to be used” | Opens a file for reading, r, or writing,w. This creates a file object which is assigned to a variable. | myFile=open(“file.txt,r)  myFile=open(file.txt,w) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| **Escape sequence** | **Effect** |
| \t | tab |
| \n | new line |
| \\ | displays \ |
| \’ | displays ‘ |
| \” | displays “ |

|  |  |
| --- | --- |
| **Mathematical operator symbol** | **Operation** |
| / | divide |
| \* | multiple |
| \*\* | exponential |
| + | add |
| - | subtract |
| // | integer division |
| % | modulus (remainder after the division) |

|  |
| --- |
| **Precedence** |
| The parentheses control the order in which the numbers are calculated. Anything in parentheses is evaluated first.  The precedence order is: parenthesis (round brackets), exponential, division and multiplication, subtract and add  B E D M A S |

|  |  |
| --- | --- |
| **Type of Python errors** | **Description** |
| TypeError | When an operation is attempt that is invalid for that type of data |
| RuntimeError | An error occurs when the program is running |
| NameError | When a name is used that is not known about (often a misspelt variable name |
| ZeroDivisionError | Dividing a number by zero |
| KeyBoardInterrupt | When a program is interrupted from the keyboard by pressing control+c |