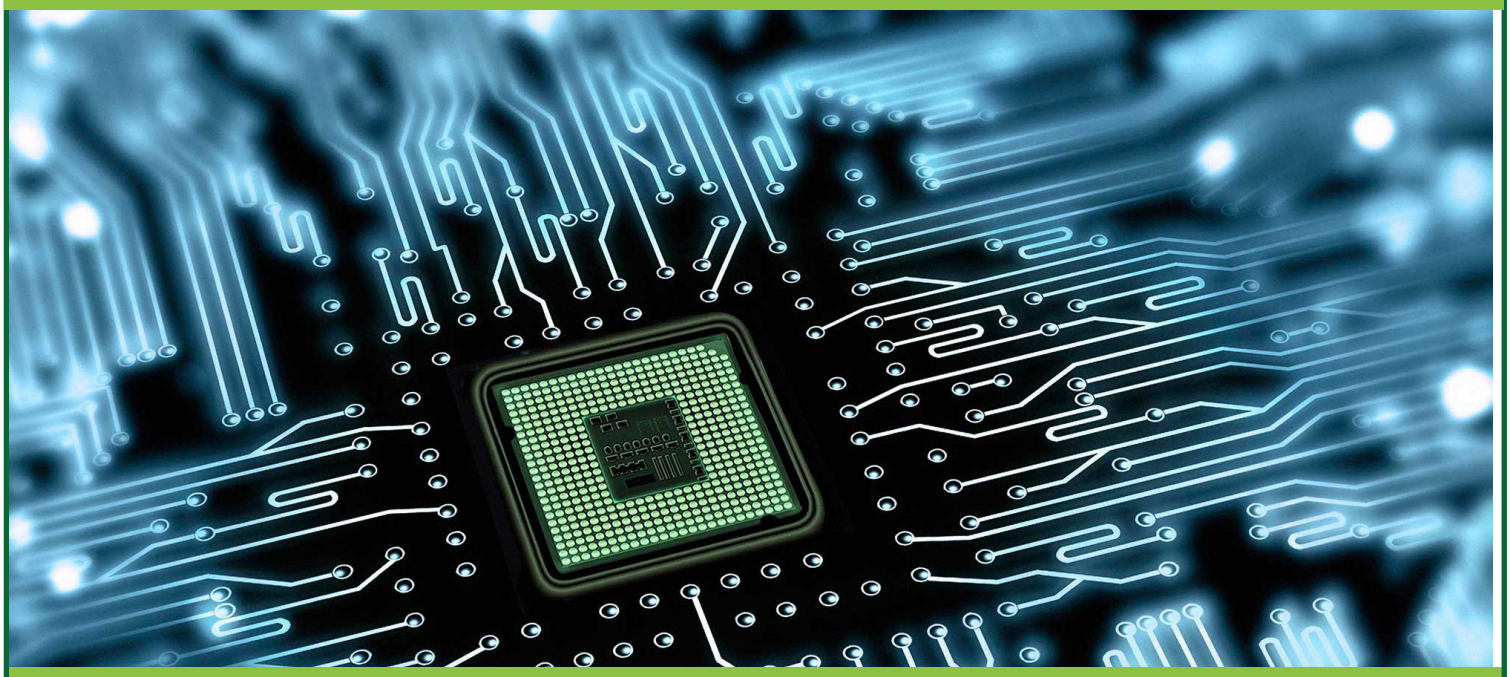
Bentley Wood
High School for Girls

Computer Science Bridging Work

Year 10 into 11 for 2020/21



Name: _____

Tutor Group: _____

Teacher: _____

Year 10

Computer Science

Bridging Work

2020

Pseudo Code Practice

Pseudo code can be broken down into five components.

- Variables:
- Assignment:
- Input/output:
- Selection:
- Repetition:

Variables

A variable has a name, a data type, and a value. There is a location in memory associated with each variable. A variable can be called anything or be given any name. It is considered good practice to use variable names that are relevant to the task at hand.

Assignment

Assignment is the physical act of placing a value into a variable. Assignment can be shown using

set = 5;

set = num + set;

The left side is the variable a value is being stored in and the right side is where the variable is being accessed. When a variable is assigned a value, the old value is written over with the new value so the old value is gone. x = 5 does not mean that x is equal to 5; it means set the variable x to have the value 5. Give x the value 5, make x equal to 5.

Input/Output

Input / Output both deal with an outside source (can be a user or another program) receiving or giving information. An example would be assuming a fast food restaurant is a program. A driver (user) would submit their order for a burger and fries (input), they would then drive to the side window and pick up their ordered meal (output.)

- Output – Write / display / print
- Input – Read / get / input

Selection

Selection construct allows for a choice between performing an action and skipping it. It is our conditional statements. Selection statements are written as such:

if (conditional statement)

    statement list

else

    statement list

Repetition

Repetition is a construct that allows instructions to be executed multiple times (IE repeated).

In a repetition problem

– Count is initialized

– Tested

– incremented

Repetition problems are shown as:

while (condition statement)

    Statement list

Activity 1

1: Write pseudo code that reads two numbers and multiplies them together and print out their product.

2: Write pseudo code that tells a user that the number they entered is not a 5 or a 6.

3: Write pseudo code that performs the following: Ask a user to enter a number. If the number is between 0 and 10, write the word blue. If the number is between 10 and 20, write the word red. if the number is between

4: Write pseudo code to print all multiples of 5 between 1 and 100 (including both 1 and 100).

5: Write pseudo code that will count all the even numbers up to a user defined stopping point.

6: Write pseudo code that will perform the following.

a) Read in 5 separate numbers.

b) Calculate the average of the five numbers.

c) Find the smallest (minimum) and largest (maximum) of the five entered numbers.

d) Write out the results found from steps b and c with a message describing what they are

7: Write pseudo code that reads in three numbers and writes them all in sorted order.

8: Write pseudo code that will calculate a running sum. A user will enter numbers that will be added to the sum and when a negative number is encountered, stop adding numbers and write out the final result.

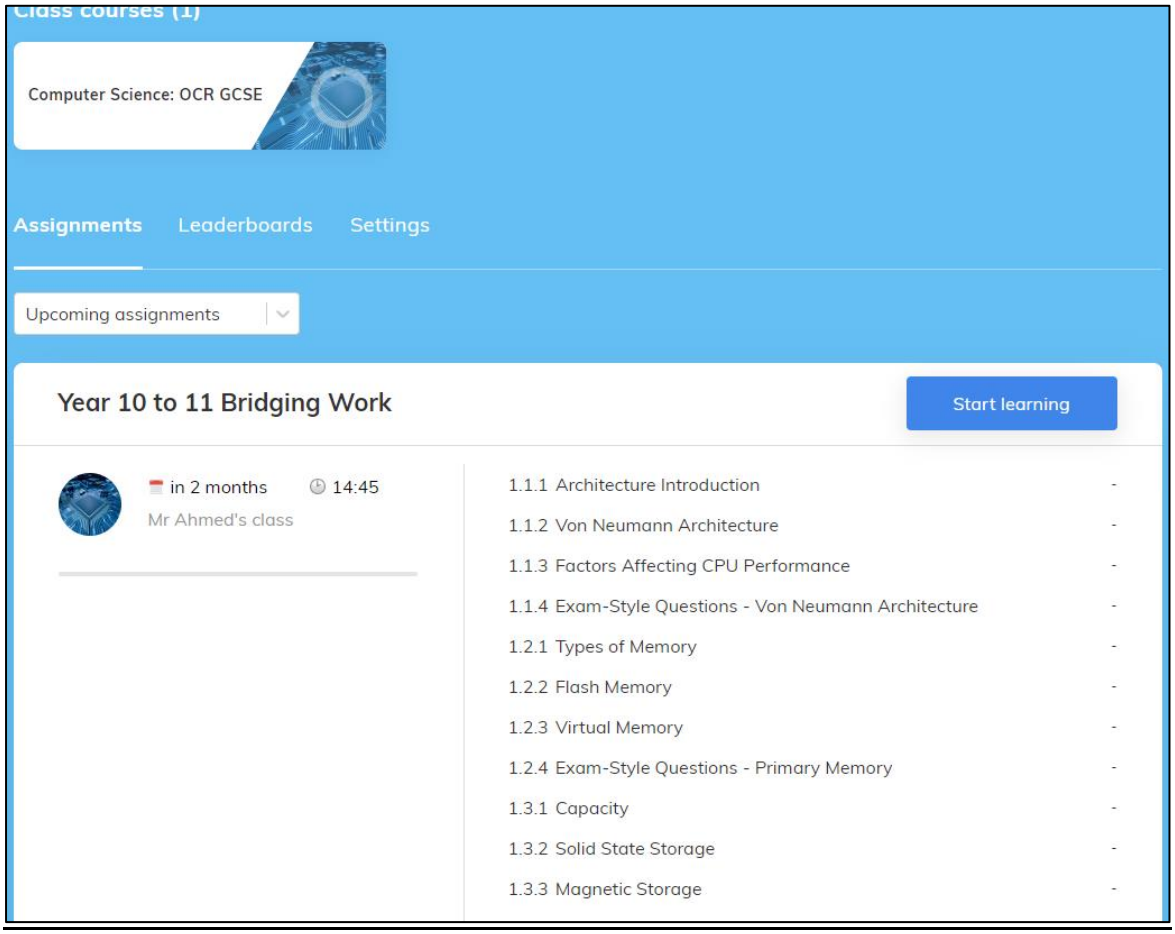
Activity 2 Read / Watch/ Revise - Using Seneca Resources

Below you will find a link to sign up for a Seneca course for your class.

Mr Ahmed's class
You will need to click on the link https://app.senecalearning.com/dashboard/join-class/qsmszidbk0 and use the class code **Qsmszidbk0**

Mr Burnaby's class
You will need to click on the link https://app.senecalearning.com/dashboard/join-class/ikkj7s8lsl and use the class code **ikkj7s8lsl**

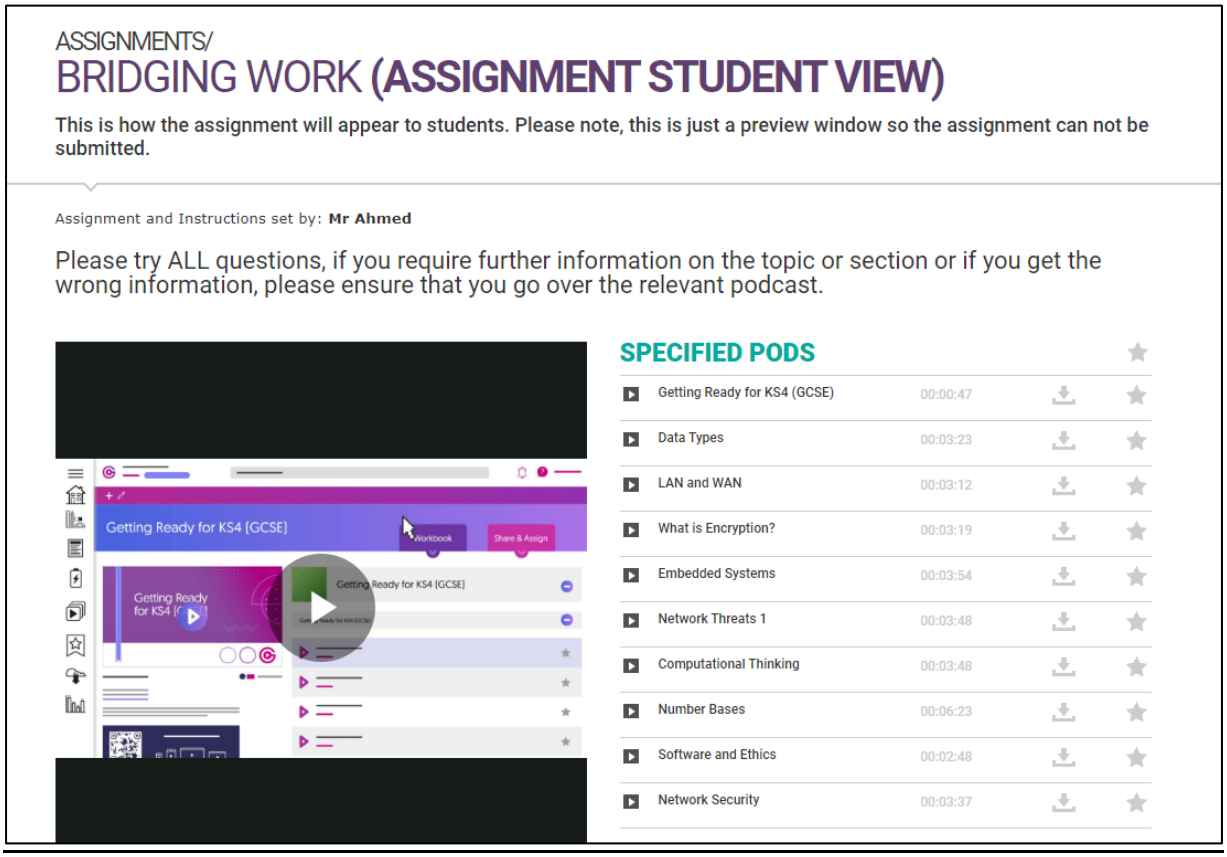
Once you are signed in follow all the course of interactive revision. It's a mixture of some light reading, videos and interactive questions with immediate feedback, great revision resource.



Activity 3 Read / Watch/ Revise - GCSE POD

You have been given an assignment set for you by Mr Ahmed on GCSE POD. There are a series of PODS for you to watch and 39 questions for you to complete. Please ensure this is done in time! Use the link below to go directly to the assignment.

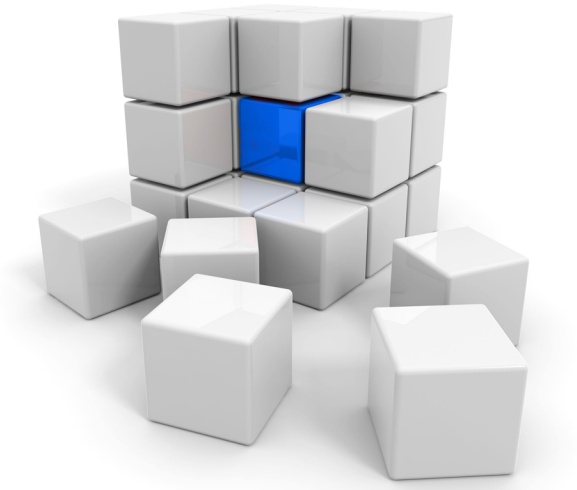
https://members.gcsepod.com/pupils/assignments/assignment/663478



Axsied
ak'seed

pseudocode to python

For OCR (9-1) Computer Science



python™

OCR Pseudocode to Python

Syntax Topic	OCR Pseudocode	Result	Python
Local variables	`x = 10` `playerName = "Sam"`		`x = 10` `playerName = "Sam"`
Global variables	`global currentUserID = 223`	In Python, variables are global if they are declared at the start of the program. If you need to write to the variable in a function/procedure you must use the 'global' keyword before you can use it.	`currentUserID = 0` `def myProcedure():` `  global currentUserID` `  currentUserID = 223`
Casting	`str(3)` `int("3")` `float("3.14")`	returns the string "3" returns the integer 3 returns the floating point number 3.14	`str(3)` `int(3)` `float("3.14")`
Output to screen	`print("hello everyone")`	Outputs "hello everyone"	`print("hello everyone")`
Input from keyboard	`name = input("Enter name")`	Outputs "Enter name" on screen. User types text. The text is stored as a string inside the variable *name*	`name = input("Enter name")`
Iteration – Count Controlled (For loop)	`for i = 0 to 9` `  print("Hi")`	Outputs "Hi" on the screen 10 times (from 0 to 9) – pseudocode includes the last number, Python doesn't	`for i in range(0,10):` `  print("Hi")`

Syntax Topic	OCR Pseudocode	Result	Python
Iteration – Condition Controlled (While loop and Do-While loop)	`password = ""` `while password != "CdRtp45@"` `  password = input("Password?")` `endwhile`	Asks the user for their *Password?* Checks if the password is equal to *CdRtp45@* Repeats if *Password* isn't equal	`password = ""` `while password != "CdRtp45@":` `  password = input("Password?")`
	`do` `  password = input("Password?")` `until password == "CdRtp45@"`	This gives the same result as the previous code. Python doesn't have a do-until structure, but this shows how the question can be asked first regardless of the condition. *[Note: You could implement a do-until loop in Python by using a "while True" loop and break. This is bad programming practice. The sample code also uses two password = input("Password?") statements – this is bad practice and should instead make use of a procedure]*	`password = input("Password?")` `while not (password == "CdRtp45@"):` `  password = input("Password?")`

Syntax Topic	OCR Pseudocode	Result	Python
Logical and Comparison Operators	Example: `while x <= 10 AND x >= 5`	Loops if x is between 5 and 10 inclusive AND is the logical operator in this example	`Example:` `while x <= 10 and x >= 5`
	Logical Operators: `AND` `OR` `NOT`	Operator meaning: And Or Not	Logical Operators: `and` `or` `not`
	Comparison Operators: `==` `!=` `<` `<=` `>` `>=`	 Equal to Not equal to Less than Less than or equal to Greater than Greater than or equal to	Comparison Operators: `==` `!=` `<` `<=` `>` `>=`
Arithmetic Operators	`+` `–` `*` `/` `MOD` `DIV` `^`	Addition Subtraction Multiplication Division Modulus (remainder) Quotient / Integer division / Floor division Exponentiation / Power of	`+` `–` `*` `/` `%` `//` `**`
String Operator	`+`	Concatenation (combine two strings together)	`+`

Syntax Topic	OCR Pseudocode	Result	Python
Selection	`if choice == "heads" then` `  print("You chose heads")` `elseif choice == "tails" then` `  print("You chose tails")` `else` `  print("Invalid choice")` `endif`		`if choice == "heads":` `  print("You chose heads")` `elif choice == "tails":` `  print("You chose tails")` `else:` `  print("Invalid choice")`
Switch/Case	`switch coinResult:` `  case "heads":` `  print("You got heads")` `  case "tails":` `  print("You got tails")` `  default:` `  print("Not valid coin")` `endswitch`	The switch statement will select one of the many different cases based on the value stored in coinResult. If none of the cases match then the default case will be executed Python doesn't have a switch/case structure. A dictionary structure in Python may be useful in providing similar functionality	
String length	`stringname.length` `Example:` `myName = "Sophie"` `print(myName.length)`	Outputs 6 on the screen	`len(stringName)` `myName = "Sophie"` `print(len(myName))`
Substring	`stringname.subString(startingPosition` `  ,numberOfCharacters)` `Example:` `myName = "Sophie"` `print(myName.substring(2,2))`	Outputs *ph* on the screen Note that the character position starts at the 0th character which is *S* in this example	`stringname[start:end]` `myName = "Sophie"` `print(myName[2:4])`

Syntax Topic	OCR Pseudocode	Result	Python
Subroutines - Functions	`function add(firstNum, secondNum)` `  total = firstNum + secondNum` `  return total` `endfunction` `totalScore = add(5,3)`	This function takes two arguments (inputs) which are firstNum and secondNum and returns the total of the two added together This calls the function and assigns the returned value to the variable *totalScore*	`def add(firstNum, secondNum):` `  total = firstNum + secondNum` `  return total` `totalScore = add(5,3)`
Subroutines - Procedures	`procedure showNames(firstName,` `  lastName)` `  print("Your name is: ")` `  print(firstName + lastName)` `endprocedure` `showNames("Sam", "Green")`	Your name is: Sam Green This procedure has two arguments (inputs) which are firstName and lastName. When the procedure is called the names are concatenated together and output to the screen	`def showName(firstName, lastName):` `  print("Your name is: ")` `  print(firstName + lastName)` `showNames("Sam", "Green")`
Pass by Reference **Pass by Value**	`a = 0` `b = 0` `procedure move(x: byVal, y: byRef)` `  x = x + 1` `  y = y + 1` `endprocedure` `move(a, b)` `print(a)` `print(b)`	Output: 0 1 In the pseudocode, the variable x will be passed to the procedure by copying the value. Changing x in the procedure **won't** change the original value. The variable y will be passed to the procedure by pointing to the original value. Changing y in the procedure **will** change the original value.	Python doesn't work in the same way as the pseudocode and there is no option to pass by reference. For most cases you can think of Python as passing by value.

Syntax Topic	OCR Pseudocode	Result	Python
Arrays	`array testScores[4]` `testScores[0] = 90` `testScores[1] = 100` `testScores[2] = 50` `testScores[3] = 75` `print(testScores[2])`	Outputs 50 Array referencing starts at 0, so to access the first element in the array, use: testScores[0] Python uses lists not arrays. These are different to array as mixed data types can be stored inside them – for example, you can store integers and strings inside the same list – in an array you would need to use two lists, one for each data type Python lists can also grow or shrink. Arrays have a fixed length. For this reason, you do not need to initialise a list with a length before assigning, but you can access/assign individual elements in the same way as the pseudocode.	`testScores = [90, 100, 50, 75]` `print(testScores[2])` `Alternatively:` `testScores = [None]*4` `testScores[0] = 90` `testScores[1] = 100` `testScores[2] = 50` `testScores[3] = 75` `print(testScores[2])`

Syntax Topic	OCR Pseudocode	Result	Python
2D Arrays	`array board[8, 8]` `board[0,0] = "castle"`	The first line of code creates a 2D array – this is an array that contains other arrays; think of it as a grid or table. You would expect that the following Python code would initialise the array: board = [[None]*8]*8 This will not work as each row in the array will be a reference to the first one. So when you change board[0][0] you would change the entire first column. The second example shows a less complicated way in which an array can be built in Python (but obviously this requires more lines of code).	`board = [[None]*8 for _ in range(8)]` `board[0][0] = "castle"` `#alternative way to build the array` `board = []` `for i in range(8):` `  row = []` `  for j in range(8):` `  row.append(None)` `  board.append(row)` `board[0][0] = "castle"`
Opening and Reading from Files	`myFile = openRead("example.txt")` `line = myFile.readLine()` `print(line)` `myFile.close()`	myFile is a file handler. It stores the "link" to the file that has been opened. The 'r' character shows that the file is open in read mode in Python. Readline is a method which will read the next line in the file; this is then assigned to the line variable It is important to close the file once it has been used to that other applications can use the file	`myFile = open("example.txt", 'r')` `line = myFile.readline()` `print(line)` `myFile.close()`

Syntax Topic	OCR Pseudocode	Result	Python
Reading to the End of File	`myFile = openRead("example.txt")` `while NOT myFile.endOfFile()` `  print(myFile.readLine())` `endwhile` `myFile.close()`	To read all the lines in a file, first open the file, then use a loop to read each of the lines, then close the file. In pseudocode (and many other languages) we read the line until we reach end of file (EOF). In Python we use a for loop to do this. \n is a newline character. We use this in a string to mean a new line (return). When you read a file in Python, the new line character is also read. When printed this gives an extra line. The rstrip() method removes the new line character.	`myFile = open("example.txt", 'r')` `for line in myFile:` `  print(line)` `myFile.close()` `#remove the new line character when` `#reading from a file` `myFile = open("example.txt")` `for line in myFile:` `  print(line.rstrip())` `myFile.close()`
Writing to a File	`myFile = openWrite("example.txt")` `myFile.writeLine("Some text")` `myFile.close()`	The 'w' character shows that the file is open in write mode in Python. This code will replace the contents of a file with whatever write() or writeLine() statements are written; append mode will "add" to the contents of a file	`myFile = open("example.txt",'w')` `myFile.write("Some text")` `myFile.close()`
Comments	`print("hello")  //Comments go here`	The // symbols in pseudocode and the # symbol in Python are used for comments. Any text (on the same line) after the comment symbol will be ignored by the program.	`print("hello")  #Comments go here`