AS and A LEVEL

Transition Guide

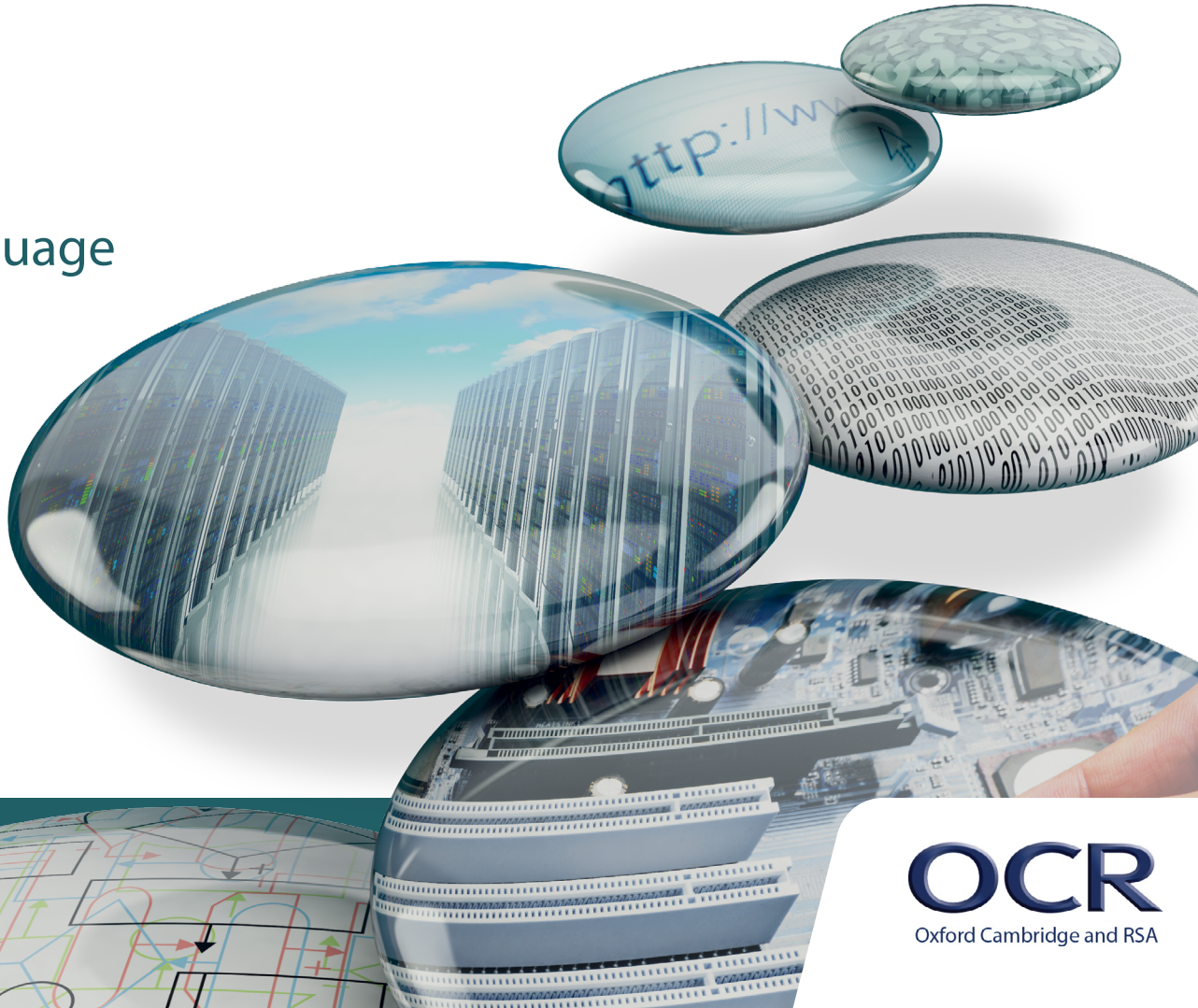
H046, H446

Accredited

COMPUTER SCIENCE

Theme: Types of programming language

September 2015

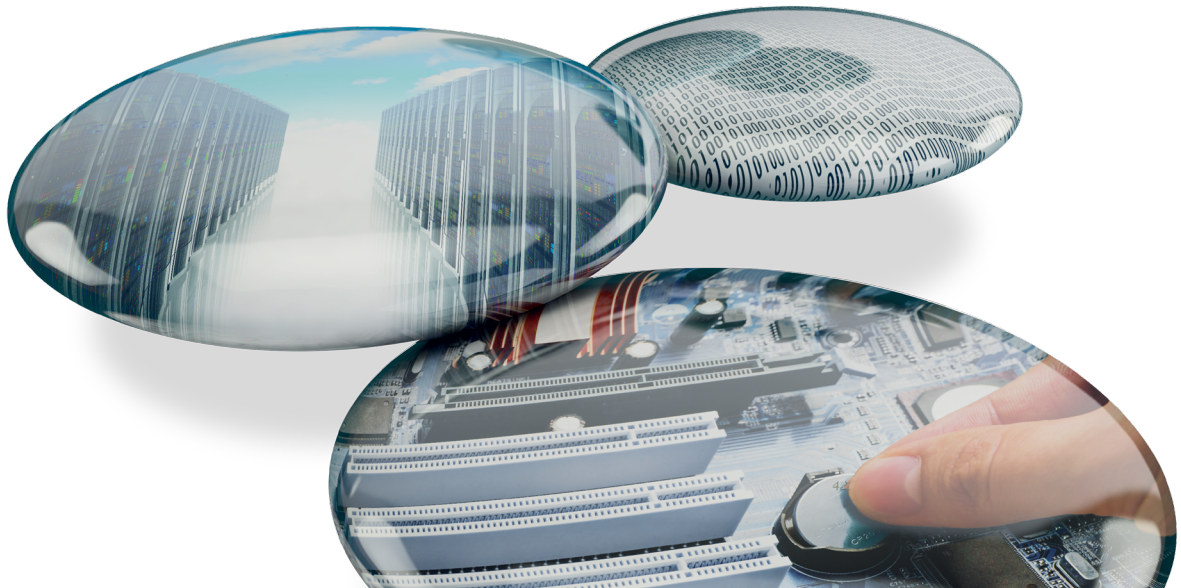


CAMBRIDGE UNIVERSITY PRESS
www.cambridge.org

OCR
Oxford Cambridge and RSA

Mapping KS5 - HE | Possible Teaching Activities (KS5 focus) | Checkpoint Task | Challenge and Extension Tasks | Resources, links and support

Contents



Key Stage Content

1.2.4 Types of Programming Language

a. Procedural languages.

HE Content

The Department of Computer Science, University of Oxford
Imperative Programming I: 2015-2016

Imperative programming constructs, with informal treatment of invariants. Procedures and modules; their use in the design of large programs

http://www.cs.ox.ac.uk/teaching/courses/imperativeprogramming1/

COMMENT

At key stage 5 learners are expected to know about efficient coding, avoiding copy and paste of lines of code with similar functions. They should be able to break down a problem into self-contained units, either as procedures, functions or modules, which are then triggered by the main program. This is even more important in event-driven programming where various parts of the program are triggered independently by mouse clicks and key presses. Learners will need to be aware of the scope of variables and the efficiency of parameters passing between the procedures, as well as returning values from functions. It is desirable that learners learn to pass and return any number of parameters in and out of functions/procedures, including arrays (lists) and objects (e.g. command button pressed).

The use of first order functions is a useful technique that is to be encouraged, as well as clear descriptive identifier names for the procedures.

For example:

```
Procedure GetInput
        myArray=new Array
        WHILE userInput !=-999
                PRINT "Enter a number"
                INPUT userInput
                IF userInput!=-999 THEN
                        APPEND userInput to myArray
                END IF
        RETURN myArray
End Procedure

Procedure FindAverage(paraMyArraY)

End Procedure

Procedure ShowResult
End Procedure

Procedure Main
End Procedure

CALL Main
```

Key concepts:

- Descriptive identifiers
- Self-contained units of code
- Main procedure calling all others
- Use of Functions to return values
- Use of Subroutines to execute multiple lines of code with one call
- Passing multiple parameters and objects
- Lopping procedures
- Variable scope: local, global
- Garbage collection and reclaiming memory
- Goto used in Assembler and non-procedural programming is to be avoided

COMMENT

Learners need to be aware that there is always more than one way to approach computing problems.
Learners need to appreciate that procedural programming is a key concept in imperative programming which is still the dominant teaching and industry paradigm; however, there are applications of other paradigms, such as functional and object-oriented, where procedural programming is not as prominent.

Learners need to know that best procedural programming comes from good planning and reliable pseudocode, where they would plan a solution in big steps which become procedures and then 'zoom in' to implement them. This will support a discussion elsewhere in the course of white box vs black box testing, with black box testing concentrating on integration of procedures and white box testing focusing on the insides of procedures and functions.

Learners need to appreciate that not every couple of lines of code need to get their own procedure, only where doing so will avoid repetition and give them fewer lines of code with the same outcome as without procedures.

Learners will need to appreciate that while procedural programming is used in both command line and event-driven GUI, they will be implemented differently.

Depending on the languages used, procedural programming can be implemented slightly differently; for example, in Basic family of languages, functions and procedures are declared with different statements and are clearly different, while in ECMA languages (C, Java) and Python, all procedures are functions, and the functions that don't return any values are like subroutines.

There is a link between procedural programming and recursion which is not possible unless you have functions. Another link is to the object-oriented programming where procedures become methods and another scope of variables is introduced – class and instance variables.

Learners need to distinguish between passing by parameter and by value and the notion that in different languages the default mode of passing can be different; for example, in C++ all parameters are passed by value by default, while in C# they are passed by reference.

Comparing the KS5 and HE requirements, it is clear that imperative and functional programming are treated as similar paradigms but the difference is made absolutely clear. Functional programming, while still procedural, is more mathematical in nature and avoids global variables and modifying original data. Functions receive copies of the original data, process and pass along. This improves security, data integrity and memory usage. Functional programming is also more predictable and, therefore, is recommended for complex programs and simplifies troubleshooting.

Another idea that comes up in HE and is also present in the coursework element of KS5 is unit testing – the ability of a programmer to test each function and procedure out of context, on their own, before being integrated into the main program. This leads to the concept of relocatable code and the notion of a programmer building up a 'recipe book' of functions and units addressing common tasks, for example, file input/output, validation. Once written and tested in one program, these procedures can be put into other programs without having to rewrite them or with minor modifications.

Learners will often encounter problems keeping track of data flows between the procedures – as a variable x is passed to a procedure, it will be referenced inside that procedure as y and if that procedure then passes that data into the third procedure it might be known as z there, hence the emphasis on data flow diagrams on the syllabus.

Teaching Activity (KS5 Focus)

'Hacking' Scratch – programming new blocks
User Sccar3 on Youtube (a learner)
http://tinyurl.com/jw8s9cm

This is a step-by-step guide through creating a new block in Scratch. A lot of learners will have used Scratch before, so this will be a fun activity to revisit this language but with their new knowledge. Creating new blocks reinforces the idea of reusable code and the importance of parameter passing.

Interactive function creation in JavaScript
W3schools
http://tinyurl.com/rvk68

This is an interactive exercise in creating a function and parameter passing.

Python example with functional and procedural features
Codecademy
http://tinyurl.com/nwfdfp7

This example is an interactive exercise in creating and using functions with procedural features (print statement as it runs).

Python procedural programming (2 tasks, b is more complex).
a. Tutorialspoint b. Dr. Andrew N. Harrington
a. http://tinyurl.com/2v6tklj b. http://tinyurl.com/cqybeu5

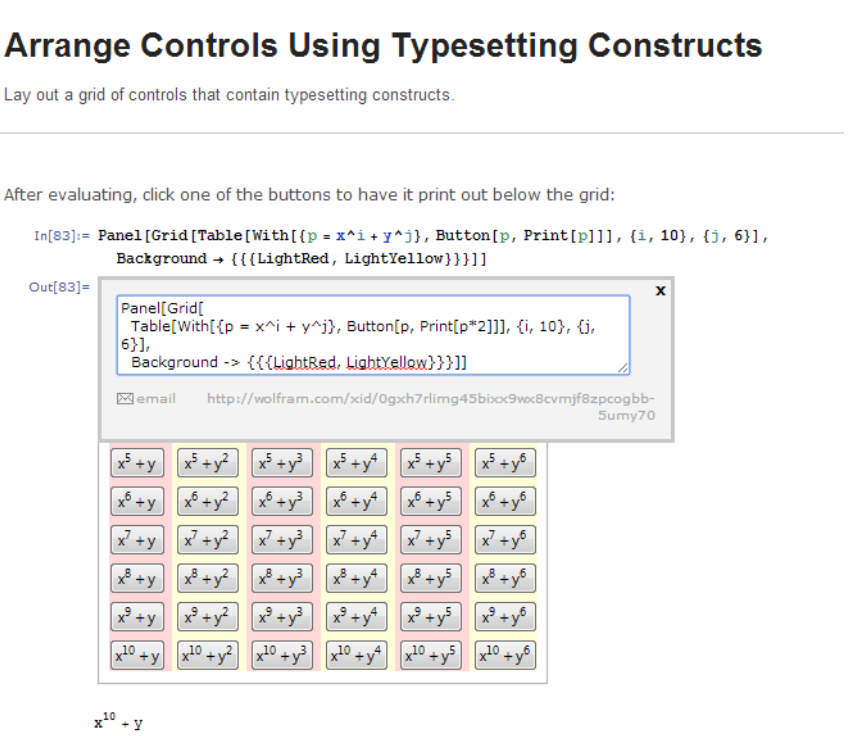
These examples show how to set up procedures in Python and call them from the main procedure.

A Survey of Programming Techniques
MIT
http://tinyurl.com/kuokz6c

This is a comprehensive discussion of various programming paradigms that relates procedural programming with iteration and object-oriented paradigm. It uses C++ style pseudocode.

Procedural examples in Wolfram Mathematica language
Wolfram
http://tinyurl.com/mrva7yw

This is a stretch activity that allows learners to solve simulation problems using procedures and functions. It is interactive and allows instant code modification.



A collection of documents and video tutorials on the subject
WizIQ Education Online
http://tinyurl.com/m8peuuy

This is a collection of tutorials in different formats indexed by this virtual learning website WizIQ.

Checkpoint task

Average of an array activity.

Teacher Instructions

http://www.ocr.org.uk/Images/253516-types-of-programming-language-checkpoint-task-teacher-instructions.pdf

Learner Activity

http://www.ocr.org.uk/Images/253515-types-of-programming-language-checkpoint-task-activity.doc

Challenge and Extension Tasks

Fun with words – doesn't have to be in Python!

Exercises

1. Write a program to sort words by length. Define a helper function cmp_len which uses the cmp comparison function on word lengths.

2. Write a function novel10 (text) that prints any word that appeared in the last 10% of a text that had not been encountered earlier.

(Exercises 1 and 2 retrieved from http://www.nltk.org/book/ch04.html)

Various tasks using Python

A selection of other tasks that has a collection of simple Python exercises downloadable using the link:
http://www.ling.gu.se/~lager/python_exercises.html
Most of these exercises involve characters, words and phrases. Rather than numbers, and are therefore suitable for learners. Some of these tasks are listed below:
Represent a small bilingual lexicon as a Python dictionary in the following fashion `{"merry":"god", "christmas":"jul", "and":"och", "happy":gott", "new":"nytt", "year":"år"}` and use it to translate your Christmas cards from English into Swedish. That is, write a function `translate()` that takes a list of English words and returns a list of Swedish words.
Define a function `max()` that takes two numbers as arguments and returns the largest of them. Use the if-then-else construct available in Python. (It is true that Python has the `max()` function built in, but writing it yourself is nevertheless a good exercise.)

Drawing fractals with Python Turtle

Familiarise yourself with the code here: http://tinyurl.com/mz8ftrb
In Python, a function is a named sequence of statement that belong together. Their primary purpose is to help us organise programs into chunks that match how we think about the problem.
The syntax for a function definition is:

```
def NAME( PARAMETERS ):
    STATEMENTS
```

Task1:
Write a function that will accept a number of sides and draw a polygon with that number of sides.

The small print

We'd like to know your view on the resources we produce. By clicking on the 'Like' or 'Dislike' button you can help us to ensure that our resources work for you. When the email template pops up please add additional comments if you wish and then just click 'Send'. Thank you.

If you do not currently offer this OCR qualification but would like to do so, please complete the Expression of Interest Form which can be found here: www.ocr.org.uk/expression-of-interest

OCR Resources: *the small print*
OCR's resources are provided to support the teaching of OCR specifications, but in no way constitute an endorsed teaching method that is required by the Board and the decision to use them lies with the individual teacher. Whilst every effort is made to ensure the accuracy of the content, OCR cannot be held responsible for any errors or omissions within these resources. We update our resources on a regular basis, so please check the OCR website to ensure you have the most up to date version.

Please get in touch if you want to discuss the accessibility of resources we offer to support delivery of our qualifications: resources.feedback@ocr.org.uk

We will inform centres about any changes to the specification. We will also publish changes on our website. The latest version of our specification will always be the one on our website (www.ocr.org.uk) and this may differ from printed versions.

Copyright
OCR retains the copyright on all its publications, including the specifications. However, registered centres for OCR are permitted to copy material from this specification booklet for their own internal use.

ocr.org.uk/alevelreform
OCR customer contact centre

General qualifications
Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk