Using Python to Create Formative Assessment Questions for Moodle

Simon Burt
Senior Lecturer, UCOL
s.burt@ucol.ac.nz

ABSTRACT

A method for creating numerous formative questions, many with individual feedback, using Python and Moodle is described. Although many question types have a numerical basis, an example of a non-numerical multiple-choice question is given, and a general method of generating non-numerical multiple-choice questions is described. Questions that are integrated into an external application (Cryptool) are presented. Moodle is shown to be an effective tool for delivering formative assessments with feedback whilst not impacting on the workload of the educator beyond an initial investment in writing the code to create the questions. Python's packages and built-in data structures make it an excellent tool for rapidly creating scripts to produce numerous questions.

Keywords: formative assessment; feedback; Moodle; Python; Cryptool.

1. INTRODUCTION

The effectiveness of formative assessment is well supported. Black and Wiliam (2005) conducted an extensive survey of the research literature and concluded that improving formative assessment raised standards overall, but was especially good for "low attainers".

For the educator, while the benefits of formative assessment might be clear and its implementation desirable, introducing formative assessments presents some difficulties, three such difficulties I have encountered are:

- Students may regard formative assessment as optional, and extra work or an extra test. Engagement rates are sometimes very low.
- Introducing formative assessment in a formal classroom environment has resulted in low levels of enthusiasm and students just waiting for the answers. Forcing students to perform "tests" in class does not increase student attendance or participation.
- Formative assessment increases educator workload. Educators must first write the assessment, administer, and mark the assessment and then provide individual feedback to each student.

Irons (2008, p. 90) lists seven reason to use technology to deliver formative assessment, among these seven are "greater flexibility and choice for students", and the rapid delivery of feedback. Race, Brown and Smith (2005, p. 124) suggest that the advantages of computer-delivered feedback include allowing students to work at their own pace, providing instant feedback, and it "legitimises learning by trial and error". However, to implement trial and error learning, there must be more than one trial, or question, and ideally, enough questions so that each student is able to attempt the same assessment many times without getting the same question twice. In some learning environments it may be preferable for each student to receive a different question, alleviating the opportunity for plagiarism. Creating multiple questions without placing a burden on the educator is the hard part.

The purpose of this research is to describe some techniques that I have used to create numerous formative assessment questions using Moodle and Python. A brief description of Moodle question types is followed by a brief description of Python and some tips on its use. Different question types, such as short answer and all-or-nothing multiple choice, with examples, are detailed. Some examples have feedback specific to each question, and other examples use a more general feedback text. Colour is used in some feedback to highlight parts of the feedback text. I also describe how to create multiple exercises for Cryptool, an application designed to help teach students the basics of cryptography. Although my work is primarily in the field of ICT, where numerical questions and answers are common, I suggest techniques that can apply to any field.

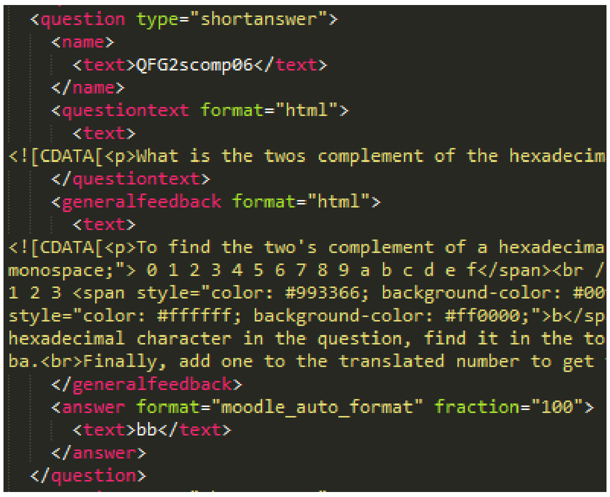
This quality assured paper appeared at the 8th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2017) and the 30th Annual Conference of the National Advisory Committee on Computing Qualifications, Napier, New Zealand, October, 2-4, 2017. Executive Editor: Emre Erturk. Associate Editors: Kathryn MacCallum and David Skelton.

2. MOODLE AND PYTHON

This section includes a short description of Moodle question types, Moodle XML, Moodle quizzes, and Python. Readers familiar with both systems and can safely skip this section.

2.1 Moodle

Moodle (http://moodle.org) is an open-source learning management system (LMS) at use in many tertiary institutes including UCOL. It allows educators to create quiz questions of different types, including multiple choice, short answer, numerical answer and 'cloze' questions.



```
  <question type="shortanswer">
    <name>
      <text>QFG2scomp06</text>
    </name>
    <questiontext format="html">
      <text>
<![CDATA[<p>What is the twos complement of the hexadecim
    </questiontext>
    <generalfeedback format="html">
      <text>
<![CDATA[<p>To find the two's complement of a hexadecima
monospace;"> 0 1 2 3 4 5 6 7 8 9 a b c d e f</span><br /
1 2 3 <span style="color: #993366; background-color: #00
style="color: #ffffff; background-color: #ff0000;">b</sp
hexadecimal character in the question, find it in the to
ba.<br>Finally, add one to the translated number to get
    </generalfeedback>
    <answer format="moodle_auto_format" fraction="100">
      <text>bb</text>
    </answer>
  </question>
```

Figure 1: An example of XML question text in Moodle format with some feedback containing inline CSS in a CDATA section.

Cloze questions are free text with inline questions of the other types; for example, multiple choice questions appear as dropdown lists. Question text, answer, and feedback can include valid HTML/CSS and hence multiple fonts and colour highlights can be utilised.

Moodle questions can be exported to an XML formatted text file, and questions correctly formatted in XML can be imported into Moodle question categories. A Moodle quiz is composed of multiple questions. Moodle is able to randomly select a question from a question category. If a question category contains only questions of the same type, such as a decimal to binary conversion, randomly selecting one or more questions from the category will present the student with a different question each time he or she attempts the assessment, and if there are enough questions in the category, students are unlikely to be presented with a question they have seen before.

The Open University (http://open.ac.uk) have made, and continue to make, improvements to Moodle questions and quizzes (Butcher, 2008). This work placed particular emphasis on improving the feedback options available for each question type. Depending on the question type, feedback text can be provided for all answers, correct answers, and incorrect answers.

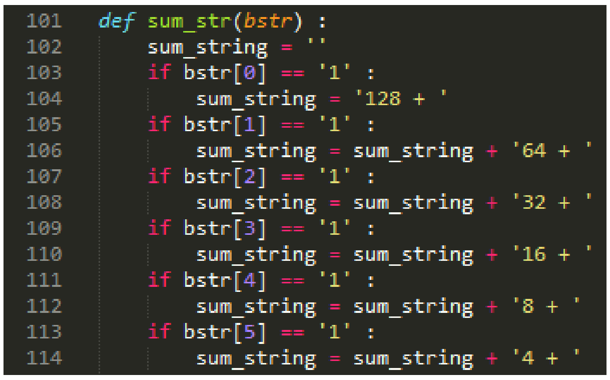
2.2 Python

Python (http://python.org) is an interpreted, high-level programing language. Whilst Python has a standard library of functions, it can be extended using packages. The Python Package Index currently lists over 100,000 packages that are freely available for download and use. These packages cover such a wide range of subject areas that it would not be useful to attempt to describe them here. The work described in this paper required the addition of a few small packages: for writing XML (minidom); for creating zip files (zipfile); for performing cryptographic procedures (rsa, crypto and M2Crypto); and for manipulating IP addresses (ipaddr). The use of publicly available Python packages greatly increases the speed of application development, especially when cryptographic procedures are required. Packages also help to ensure the correctness of results, assuming the package author has performed unit testing, in accordance with the Python documentation.

Python is available in two major versions: 2.7 and 3.6. Python runs on many platforms but packages might not be available for all platforms in all versions. Although writing most Python scripts in version 3.6 seems the most obvious choice, some cryptographic packages are not easily available for version 3.6 under Windows operating systems. Before writing too much code, it is worth expending some effort to investigate package availability for your chosen platform. Linux offers the best platform in terms of package availability.

Python scripts are portable across operating systems provided all of the referenced packages are available. None of the scripts described here uses a graphical user interface, but Python packages are available to achieve this if desired.

Whilst Python provides its own environment for developing programs, I preferred to use Sublime Text 3 (http://sublimetext.com) as it has syntax highlighting and indentation features that are particular to Python. Unlike many other programming languages, Python is unusual in that it uses an indentation-based syntax to delimit blocks of code. Programmers familiar with other languages that use a delimiter (such as curly braces{ and }) to identify a new block might find this confusing at first, but it quickly becomes natural. I highly recommend using a text editor or development environment that is aware of Python's indentation requirements, and which automatically converts a tab to four spaces as suggested by the PEP 8 Python Style Guide ('PEP 8 -- Style Guide for Python Code', n.d.). Using a Python-aware editor means that the programmer can press the tab key for indentation which is much quicker than pressing four spaces, but the editor inserts four spaces instead of a tab. Using the approved style of four spaces improves code portability across platforms.



```python
def sum_str(bstr) :
    sum_string = ''
    if bstr[0] == '1' :
        sum_string = '128 + '
    if bstr[1] == '1' :
        sum_string = sum_string + '64 + '
    if bstr[2] == '1' :
        sum_string = sum_string + '32 + '
    if bstr[3] == '1' :
        sum_string = sum_string + '16 + '
    if bstr[4] == '1' :
        sum_string = sum_string + '8 + '
    if bstr[5] == '1' :
        sum_string = sum_string + '4 + '
```

Figure 2: Python code in Sublime Text 3 has syntax highlighting and assistance with indentation.

Python's built-in data structures include tuples, lists and dictionaries, defined using (), [] and {} respectively. Tuples are a quick way of linking attributes together into a single object, but once created, are immutable. Lists are a collection of objects and are mutable. A single list can include objects of any type, including tuples and other lists. An equivalent to a Python list of tuples in the C programming language is an array of pointers to structs, or perhaps as a linked list of structs.

Python's interpreted nature, along with its built-in data structures and extensive packages, make it an ideal scripting language. All of the scripts created for this research were executed from the command-line, with parameters and question text hard-coded. Once the questions have been generated and successfully imported into Moodle, there is little reason to run the script again.

3. CREATING QUESTIONS

Moodle's dialogue boxes for creating and editing questions use standard HTML forms. Only one question at a time can be edited. In my experience, creating Moodle questions this way is a tedious process, especially if the Moodle server or Internet performance becomes a limiting factor as it sometimes has at UCOL.

Moodle allows questions to be imported using a variety of formats, including formats produced by other LMS systems. There are other tools that can be used to create questions but these tools are merely a different user interface from the Moodle HTML form. For example, questions can be created in a Microsoft Word document as tables, and the Word document is then imported into Moodle. Whilst this is an improvement over the clunky HTML forms, it is no panacea.

For some subjects Moodle questions may be purchased from third parties, typically text book publishers or other specialists such as Respondus (http://respondus.com).

The Moodle XML format is the Moodle recommended format for export and import of Moodle questions. The Moodle documentation describes the XML format in some detail ('Moodle XML format - moodledocs', n.d.) but I found it easier to create an example question, export it, and examine the XML created. I then created scripts that replicated this format. This technique works for all Moodle question types and produces quick results.

4. RESULTS

This section describes some examples of the questions that I have created using Python scripts.

4.1 Short Answer Questions

Short answer questions (and the very similar numerical answer questions) present an empty space where the student is expected to insert the answer. Unlike a multiple choice question, there is a minimal chance of guessing the correct answer.
An example of a simple short answer question that is easy to create using Python is:

Convert the 8-bit binary value 11011111 to decimal.

[Answer: 223]

I wrote a Python script which creates all 256 possible questions of this same type. I placed all such questions in the same Moodle question category and then created a quiz which randomly selected the required number of questions from this category. The Python script also generates this feedback:

To convert from binary to decimal, put the binary value under the correct headings like this:

```
128  64  32  16  8  4  2  1
  1   1   0   1  1  1  1  1
```

Then add the column headings that contain a 1:

128 + 64 + 16 + 8 + 4 + 2 + 1 = 223

The feedback is specific to the question. The reasoning for this is so that the student can check his or her own workings. It would be impractical to create 256 questions with this type of feedback without using some form of automation.

Since Moodle is a web-based application, questions and feedback can be provided in different colours and styles using standard HTML. Any HTML fragment in a Moodle XML question needs to be within a CDATA section.

I used colour to highlight the specific values in the feedback for this question:

What is the twos complement of the hexadecimal value 43 in 8 bit arithmetic? Give your answer in hexadecimal.

[Answer: bd]

The feedback text created was also specific to each question so that the student could check his or her own work:

To find the two's complement of a hexadecimal value start by writing all 16 of the hexadecimal values in order:

0 1 2 3 4 5 6 7 8 9 a b c d e f

Then write the same values underneath, only backwards:

0 1 2 3 4 5 6 7 8 9 a b c d e f

f e d c b a 9 8 7 6 5 4 3 2 1 0

For each hexadecimal character in the question, find it in the top row and replace it with the character in the bottom row.

So "4" is replaced by "b" and "3" is replaced by "c" giving bc.

Finally, add one to the translated number to get the twos-complement:

bc + 1 = bd

The text above does not show that the specific columns in the two rows are coloured so as to highlight where the correct answer originates. Questions that used the same hex character for both digits were excluded.

I have also created short answer questions to test other binary, decimal and hexadecimal conversions, such as hexadecimal to binary, and to test IPv4 addressing skills. In each case, the feedback was specific to the question. For example, feedback for the IPv4 addressing questions shows the workings (in binary) for each question. The method described in the feedback is the same as the method used in the lesson that covered this topic.

One problem with short answer questions is that the answers are limited to unambiguous text or numbers. Moodle must be able to distinguish between a correct and incorrect answer, although multiple correct answers can be specified and case can be ignored. A question such as 'Give a regular expression to find all words that begin with a capital letter' is not recommended, since the student will find legitimate regular expressions that the question author has not considered. For example, the regular expression could be enclosed in single or double quotes, might have no quotes, could specify word boundaries with an expression or a command-line option, and could include elements that are not essential but still produce the desired result. Students might spend too much time trying to get an acceptable answer rather than a correct answer in these situations.

4.2 All-or-Nothing Multiple Choice

The Moodle question type all-or-nothing multiple choice presents the student with a list of choices, and a set of (two or more) choices needs to be selected to achieve the grade. Selecting one or more incorrect answers results in a zero grade. I used this question type to create multiple questions such as this:

Which of the following are equivalent specifications of SDRAM?

Choose two.

DDR3-1066
PC4-17000
DDR3-2133
PC2-4200
DDR3-1333
DDR4-2800
PC2-6400
PC3-17000

[Answer: DDR3-2133 and PC3-17000]

A generic feedback text was included:

First match DDR2 to PC2, DDR3 to PC3 and DDR4 to PC4

Then multiply the number after DDRx by 8 to get the number after PCx.

For example: DDR2-800 = PC2-6400.

The procedure for creating this question is as follows:

- Create a tuple containing each standard name and the matching module name eg. ddrtuple = ('DDR2-533', 'PC2-4200'). Each tuple represents a correct answer pair.
- Add each tuple to a list eg. ddrlist.append(ddrtuple).
- For each tuple in the list, select six other tuples - these are the incorrect answers.

The incorrect answers are selected at random, taking care not to duplicate correct answers, or create another valid pair. Note that although there are only 16 pairs of possible correct answers (for DDR2, DDR3, and DDR4) the method used to select six incorrect answers gives a possible 100,100 different questions ($^{15}C_6$ x $^{6}C_3$). As there are eight choices, two of which are correct, the probability of guessing the correct answer is 1/28 or 3.6%. Each run of the script creates 16 questions (one for each correct pair) and I considered 48 questions of this type to be sufficient to provide the required level of variation.

A benefit of using all-or-nothing multiple choice questions is that they can be applied to any matching sets, or categories. In general, the programmer creates one list of matching tuples, and one list of non-matching tuples. The questions are generated by iterating through the matching tuples and randomly selecting options from the non-matching tuples. The variability of the question is improved as more tuples are added to the non-matching list. Unlike short answer questions, there is no need for the answers to fit a format that is easy to mark.

4.3 Integration with External Software

Cryptool (http://cryptool.org) is an open-source Windows application for cryptography and cryptanalysis. Students using Cryptool can experiment with encryption and decryption using ancient and modern algorithms. I have used Cryptool in student exercises for various tasks, including:

- Brute forcing a partial symmetric key to decrypt a ciphertext.
- Factorising a short(ish) RSA modulus to decrypt a ciphertext.
- Using a certificate to digitally sign a plaintext and then verify the signature.

These questions can be varied by choosing different encryption algorithms, different keys, different plaintexts, and varying the verification results. The problem for the educator has been creating example questions for students to practice. With only one or two questions of this type, the correct answer can be quickly remembered, or copied from fellow students.

Using Python to generate different questions for Cryptool exercises required a lot of research into how Cryptool performs certain tasks. Most of this research involved reading the Cryptool source code which is written in C, with comments mostly written in German. Creating a signed plaintext required the knowledge of the private keys used by Cryptool's CA. This also required a lot of digging in the source code. In the process of generating questions that were compatible with Cryptool, many quirks and oddities of Cryptool formats and procedures were discovered, and I reproduced these in Python.

Here are some example questions created using Python scripts:

Given a partial DES-CBC key of: 50 8E 71 D4 F1

and a ciphertext of: 7C 4F 30 64 5D CD 6A F4 BE 03 1D DD D1 AD 30 E8 D9 19 45 EC B4 2B 46 7A C7 8F FC 35 2E EB B5 43 A9 DE 69 53 E4 03 48 36

What is the magic word? [Answer: redistribution]

The question text is displayed in a manner such that it can be easily pasted into Cryptool. When a single question requires multiple answers, the Moodle cloze question type is used:

A plaintext has been encrypted using the RC4 algorithm with a 64 bit key.

The first 40 bits of the key are: AA 7E 72 F3 82

The ciphertext created is: 6F 0E 17 24 3B B6 F7 BA 6E 1D FD E0 5D 00 6F 3E BE 96 70 D2 12 07 FE AC 0A 37 04 D3 0A 69 94 03 32 05

What is the magic word? [Answer: rapprochements]

What are the last 3 bytes of the key? [Answer: EC B7 DF]

Cryptool has an asymmetric encryption demonstration feature that allows the student to experiment with public and private key encryption and digital signatures. An example of an asymmetric encryption question is:

Given the RSA public modulus N: 1615833784006546902179133803377697315072229603853 0244557

The public key e: 65537

The ciphertext: 6266547067994897443022136128536717839268540777939 907450

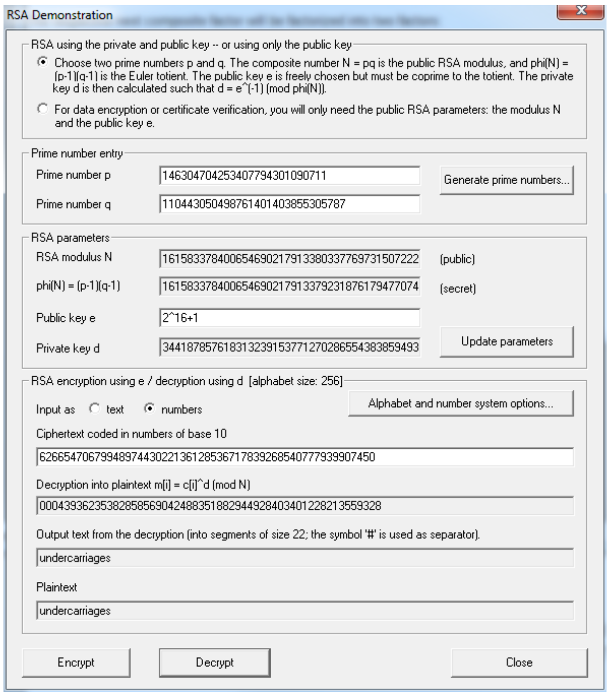
What is th private key d? [Answer 3441878576183132391537712702865543838594938985483 298273]

What is the plaintext? [Answer: undercarriages]

The public modulus is [Answer: 184] bits long.

The [Answer: Quadratic sieve] method was used to factor the modulus.

Note that the question shown above uses numerical answer, short answer, and multiple choice in the same cloze question.


RSA Demonstration
Generate prime numbers...
Update parameters
Alphabet and number system options...
Encrypt
Decrypt
Close

RSA using the private and public key -- or using only the public key

(●) Choose two prime numbers p and q. The composite number N = pq is the public RSA modulus, and phi(N) = (p-1)(q-1) is the Euler totient. The public key e is freely chosen but must be coprime to the totient. The private key d is then calculated such that d = e^(-1) (mod phi(N)).

( ) For data encryption or certificate verification, you will only need the public RSA parameters: the modulus N and the public key e.

Prime number entry

Prime number p: 146304704253407794301090711

Prime number q: 110443050498761401403855305787

RSA parameters

RSA modulus N: 161583378400654690217913380337769731507222 (public)

phi(N) = (p-1)(q-1): 161583378400654690217913379231876179477074 (secret)

Public key e: 2^16+1

Private key d: 344187857618313239153771270286554383859493

RSA encryption using e / decryption using d [alphabet size: 256]

Input as ( ) text (●) numbers

Ciphertext coded in numbers of base 10

6266547067994897443022136128536717839268540777939907450

Decryption into plaintext m[i] = c[i]^d (mod N)

00043936235382858569042488351882944928403401228213559328

Output text from the decryption (into segments of size 22; the symbol '#' is used as separator).

undercarriages

Plaintext

undercarriages

Figure 3: RSA Demonstration window in Cryptool showing factorised public modulus and correct decryption of ciphertext (undercarriages).

Creating digital signature questions proved to be a challenge, as the format of the certificates and digitally signed document files needs to be exactly correct, otherwise Cryptool will not accept them.

This is an example digital signature question:

Get this file 101questionsq01.zip from the L: drive.

Extract the contents of the file to your desktop. There should be four files: c01.bin, mesg1, mesg2 and mesg3

The file c01.bin has been encrypted using RC4 with a 64 bit key. The key is 14A4FEFD4EC6DBF4

Decrypt this file and save the plaintext as cert.p12 on your desktop. This file is a certificate.

Use this certificate to verify the signatures on the three message files.

Which signatures are valid, and which are not?

As can be seen from the question text, each question refers to a specific zip file containing the required certificate, and the message files. Each question has a random combination of valid and invalid signatures, reducing the chances of guessing the correct answer.

No feedback text was provided with the Moodle questions for Cryptool. Students undertaking a Cryptool exercise “know” when they have the correct answer because Cryptool itself is providing feedback as each stage of the exercise is completed.

5. CONCLUSIONS AND FURTHER WORK

Moodle has a range of question types and feedback options which means that it is a good choice for delivering assessments. Freely available Python packages ease the burden of writing code, and allow fast prototyping and rapid application development. Creating multiple formative questions using Python has required an initial investment in time and effort, but it has been rewarded by the creation of a plethora of assessment questions. Had I undertaken this work using the C programming language, I’d still be writing functions to iterate through a linked list of structs.

Students are able to undertake the formative assessments whenever and wherever they wish. Feedback is instant, and for many questions, explanatory. Plagiarism is reduced because each student receives a different question. Involvement can be improved by linking the formative assessment to the summative assessment - by including formative questions in summative assessments. I have found that these formative questions then act as a “marker” indicating whether or not the student attempted the formative assessment.

Whilst questions with a random numerical component are the easiest to create, and most obvious to use, Python and Moodle can be used to create non-numerical questions. When authoring multiple choice questions, creating wrong answers requires the most effort. This is true whether or not the question is written manually or using a Python script.

The basic technique of automatically creating multiple choice questions relies on creating a pool of correct answers and a pool of incorrect answers. For each correct answer, the required number of incorrect answers are randomly selected from the incorrect pool. Questions and answers need not have a numerical component. Whilst I primarily teach computer networking, these techniques could be easily extended to create questions in other fields.

Questions are not restricted to text - Python can be used to randomly select images; the image data is base64 encoded and embedded into a CDATA section. Techniques such as these will increase the range of subject matter that can be automatically generated. For example:

Which of the following images shows an RJ45 connector?

Which HTML produced the output shown in the image?

Which of the following is a zygote?

Which picture shows a dovetail joint?

Python has packages which allow image editing, so multiple different images can be created, perhaps with different embedded text applied to a base image.

UCOL uses Cisco Networking Academy (http://www.netacad.com) material for some programmes. Cisco has developed a network simulator application called Cisco Packet Tracer. Cisco have made available an API for interfacing with this application, and in future work I hope to demonstrate how Python scripts can generate networking exercises in Cisco Packet Tracer.

6. REFERENCES

Black, P., & Wiliam, D. (2005). Inside the black box: raising standards through classroom assessment. Granada Learning.

Butcher, P. (2008). Online assessment at the Open University using open source software: Moodle, OpenMark and more. Retrieved from https://dspace.lboro.ac.uk/dspace/handle/2134/4656

Irons, A. (2008). Enhancing learning through formative assessment. London; New York: Routledge.

Moodle XML format - moodledocs. (n.d.). Retrieved 3 August 2017, from https://docs.moodle.org/23/en/Moodle_XML_format

PEP 8 -- Style guide for Python code. (n.d.). Retrieved from https://www.python.org/dev/peps/pep-0008/

Race, P., Brown, S. A., & Smith, B. (2005). 500 tips on assessment. Abingdon; New York: RoutledgeFalmer. Retrieved from http://site.ebrary.com/id/10094619