Book ISBN Validator

Have you ever wondered what all those numeric codes on products you buy or use everyday actually mean?

Strings of digits are used in product UPC codes, social insurance numbers, **book ISBN numbers**, even credit card numbers. The last number in each of these sequences is called a **check digit** and is used to verify that the rest of the number is valid.

Verifying that these numbers are valid is a very important application of computer science!

You are to create a Python application that will verify whether 10-digit and 13-digit ISBN (i.e., book) numbers are valid or not.

Program Requirements:

1. Create a module called **checkISBN** that contains the following three functions:

    a. **get_digit(number, position)** This function accepts two integer parameters. The first is a number to get a digit from, and the second is the position (from the right) of the required digit. That is, the one's digit of the number is position 0, the ten's digit is position 1, the hundreds digit is position 2, and so forth. The function returns the digit (as an integer) at the specified position. For example, get_digit(12345**6**789, 3) returns 6.

    Note: If you try to get a digit beyond the left-most digit in the number, then **0** (zero) is returned. For example, get_digit(123456789, 9) returns 0.

    b. **is_ISBN10(number)** that accepts an integer representing an ISBN-10 code, determines if it is a valid code or not by calculating and comparing the check digit, and returns **True** if it is a valid ISBN-10 or **False** otherwise.

    c. **is_ISBN13(number)** that accepts an integer representing an ISBN-13 code, determines if it is a valid code or not by calculating and comparing the check digit, and returns **True** if it is a valid ISBN-13 or **False** otherwise.

2. Create a Python program that uses the functions defined in your **checkISBN** module to allow a user to validate ISBN-10 and ISBN-13 numbers.

a. Your program should include a menu with 3 options:

```
BOOK ISBN VALIDATOR

What type of ISBN code would you like to validate:

1. ISBN-10
2. ISBN-13
Q. Quit Program

Choice:
```

b. If the user selects an invalid option, the user should be told so, and the menu should be displayed again.

c. If the user chooses option 'Q' (or 'q') the program should end.

d. If the user chooses options 1 or 2 they should be asked to enter the code (as an integer) to be validated. **Please note**: Python treats numbers entered with a starting **0 (zero)** as octal, not decimal! If an ISBN starts with a 0, assume that the user will **not** type that leading 0. For example, if the book ISBN is "064200115896", the user will type 64200115896.

e. Calling an appropriate function from your **checkISBN** module, you should tell the user whether the ISBN entered is valid or not.

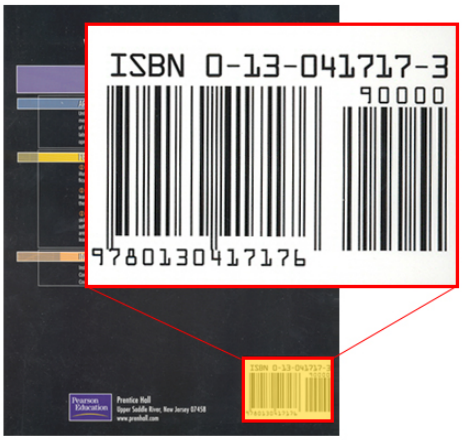
f. The menu should then be displayed again, allowing the user to choose again.

3. Your program should use effective mainline logic. In addition to a **main()** function think about creating other functions to make your code as efficient as possible.

4. Do **NOT** read the ISBN as a string or use string indexing or other operations to solve this problem. Use what you have learned **in class** to solve this problem. Hint: **/** and **%**

5. Your program should demonstrate your mastery of the style rules we have discussed so far this year including docstrings, and code comments, and correct use of modules.

How To Validate a 10-digit ISBN



ISBN stands for International Standard Book Number. The 10 digit version of ISBN was the standard used to identify any published book up to 2006.

The first digit (from the left) is the language (0 means English), the next 2-3 digits represents the publisher, and the next 5-6 digits is the book code (assigned by the publisher). The last digit (on the right) is the **check digit**.

Below is an algorithm to check whether a 10 digit ISBN is valid or not, we'll use **0130417173** as an example UPC code:

1. Excluding the check digit, multiply each digit by a weighted value and take their sum. The weighted values are 1 for the first digit (from the left), 2 for the second digit, 3 for the third digit etc..

   0x1 + 1x2 + 3x3 + 0x4 + 4x5 + 1x6 + 7x7 + 1x8 + 7x9 = **157**

2. Find the remainder of 157 divided by 11, in this case it's **3.** Since the result is the same as the check digit, then the ISBN is correct.

Feel free to find/use another algorithm as long as it produces the same/correct results as the algorithm described above.

How To Validate a 13-digit ISBN

Since January 1, 2007, a 13-digit ISBN has become standard. Most books include both a 13 and 10 digit ISBN number for backward compatibility, but ISBN 10 is being phased out. For example, in the image above, the 13 digit ISBN is under the bar code: 9780130417176.

Validating a 13-digit ISBN is similar to a 10-digit ISBN, but the weight values are different, and the way that the check digit is calculated is a little different.

I will leave it to you to figure out (Google!) the algorithm for this one. Have fun!

Book ISBN Validator

Marking Rubric

Student Name: _____

Criteria	Poor (1)	Fair (2)	Good (3)	Great (4)
Requirements	4+ requirement analysis errors.	2 or 3 requirement analysis errors.	1 requirement analysis error.	All requirements implemented correctly!
Style Rules *(#1 → #9)*	4+ style rules broken or needing improvement.	2 or 3 style rules broken or needing improvement.	1 style rule broken or needing improvement.	Perfect style! All style rules have been followed.
Program Design and Structure	Poor, missing mainline logic and/or code poorly organized.	Fair, need to significantly improve mainline logic and/or code organization.	Good, but some room to improve on mainline logic and/or code organization.	Excellent mainline logic design, code is well organized.
Testing	Program fails 4+ test cases.	Program fails 2 or 3 test cases.	Program fails 1 test case.	Program passes all test cases.
			Overall Mark:	**/16 (T)**