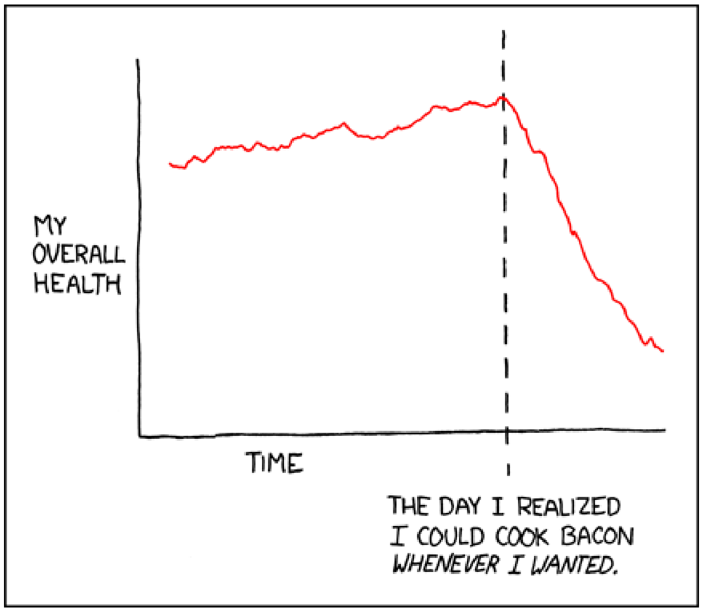
CS30 - Assignment 3

Due: Thursday February 11, at 11:59pm



http://xkcd.com/418/

For this assignment, we're going to be playing with three different problems, the last of which will introduce you to the data that we'll be playing with for the next assignment. Put all of your code into a single file with your first name and last name followed by `assign3.py`. Put comments to separate out the three problems, e.g.

```
# ------------------------------------------------------------
# Problem 1
```

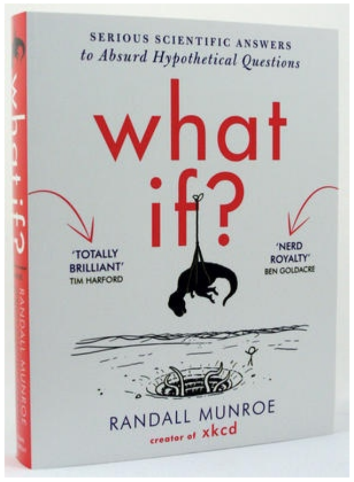
1 ISBN-10 to ISBN-13

An *International Standard Book Number* (ISBN) is a "serial number" assigned to a published book. Until recently, an ISBN consisted of 9 digits together with an additional "check digit." The check digit is a way of ensuring that the ISBN is correct and has not been accidentally changed.

By 2005, the book industry had realized that it was about to run out of the 10-digit ISBNs and started converting to 13-digit ISBNs. Like their predecessors, the new ISBNs include a final check digit. On January 1, 2007, the 13-digit numbers became official.

Recently-published books carry both ISBNs. For example, the book *What If?: Serious Scientific Answers to Absurd Hypothetical Questions*, by the author of the XKCD comics, was published in September, 2014 and has two numbers:

ISBN-10: 0544272994
ISBN-13: 9780544272996


SERIOUS SCIENTIFIC ANSWERS
to Absurd Hypothetical Questions
what if?
'TOTALLY BRILLIANT' TIM HARFORD
'NERD ROYALTY' BEN GOLDACRE
RANDALL MUNROE
creator of xkcd

The check "digit" for a 10-digit ISBN is either a digit or the letter X. All the characters of a 13-digit ISBN are digits. The dashes, which sometimes appear in ISBNs, are not significant.

Read the attached pages from *ISBN-13 for Dummies, Special Edition,* to learn the method for translating a 10-digit ISBN to a 13-digit one (particularly the section on converting ISBNs).

Write a function called `convertISBN` that takes a 10-digit ISBN (as a string) as a parameter and returns the corresponding 13-digit ISBN (as a string).

A few thoughts on how to accomplish this:

- I'm giving you a fair amount of leeway in how you implement this function. That said, it is *critical* that you figure out how to break this problem into a number of smaller functions.
- Before you start programming, take time to plan out the structure of your program, in particular, the functions that you need to write. There are many ways to solve this program, but I recommend converting the string to a list of integers, do your processing on the list and then convert it back to a string when you're all done. *Hint,* some of the examples that we looked at in class will likely be useful.
- One of the key advantages to solving a program like this with multiple functions is that it makes debugging easier. Write your basic functions and then test them individually. Once you're satisfied that they're correct, then (and only then!) move on and utilize them to build up other functions.
- Make sure to test your final function on a number of different examples to make sure that it does the correct things. You can test your function on various ISBNs. Use books that you happen to have with you, or find some in the library catalog or at Amazon. To check your program, you can compare its results with an on-line ISBN converter, located at *http://www.isbn.org/converterpub.asp.*

Working with ISBN-13

Until January 1, 2007, when you order ISBNs from the ISBN agency, you'll be allocated blocks of ISBN-10s. After January 2007, the ISBN agency will allocate only blocks of ISBN-13s.

Handling the full ISBN-13

Because many of the new ISBN-13s will eventually begin with 979 instead of 978, your systems must be able to accommodate the 13-digit numbers *in their entirety.* Be aware that sometimes designers of both internal and external computer systems will take shortcuts by storing a common prefix separately from the core number. In the case of the ISBN, for example, system designers could choose to store a common 978 prefix separately from a constantly changing 10-digit core number; This approach will *not* work for ISBN-13s because, as previously noted, the prefix for these numbers could be *either* 978 or 979: a common prefix simply won't apply.

REMEMBER

You will need to modify any paper form or computer system that uses the ISBN-10 so that it will accept the full ISBN-13.

WARNING!

Your computer systems will have to accept ISBNs with prefixes of both 978 and 979. To be safe, however, retailers and distributors should build their systems to accept any valid 13-digit EAN, because these organizations frequently handle non-book product (such as note paper and greeting cards) as well as books.

Banking your ISBN-10s

You may have some unassigned ISBN-10s remaining after January 2007. Don't worry! You won't need to throw out or trade in your bank of ISBN-10s after the transition occurs: you can convert them into ISBN-13s yourself. To find out how, see the steps in the following section, "Converting your ISBN-10s to ISBN-13s" — and hold on to your ISBN-10s for future use.

Converting your ISBN-10s to ISBN-13s

To change an ISBN-10 to an ISBN-13, follow these three basic steps:

From Zoë Wykes, editor, *ISBN-13 for Dummies, Special Edition,* Wiley, 2005. Available at *http://www.bisg.org/isbn-13/for.dummies.html.*

1. **Drop the check digit (the last digit) from your existing ISBN-10.**

   For example, your ISBN-10 is 0-940016-73-7. By dropping the check digit (7), you get a 9-digit number, 0-940016-73.

2. **Add the prefix "978" to the beginning of your 9-digit number.**

   Your 9-digit 0-940016-73 now becomes 12 digits, 978-0-940016-73.

3. **Recalculate your check digit using the modulus 10 check digit routine.**

   Note: The modulus 10 check digit routine is the current routine used to calculate the check digit for the Bookland EAN.

   Here's how, using the calculations shown in Table 1-1.

Table 1-1 The ISBN-10/ISBN-13 Conversion Chart

ISBN =	9	7	8	0	9	4	0	0	1	6	7	3
Weighting Factors	1	3	1	3	1	3	1	3	1	3	1	3
Values (product)	9 +	21 +	8 +	0 +	9 +	12 +	0 +	0 +	1 +	18 +	7 +	9 = 94

   a. **Using the 12-digit number from Step 2, shown in Table 1-1, multiply each digit by the weighting factor shown beneath it in the table.**

      In this example, you have (9x1) + (7x3) + (8x1) + (0x3) . . . and so on.

   b. **Add the resulting values together.**

      The sum of the values equals 94.

   c. **Divide the sum by the modulus (which is 10).**

      Divide 94 by 10. Your result is 9, with a remainder of 4.

   d. **Using the standard modulus (10), subtract the remainder from 10 to get the check digit (last digit).**

      In this example, 10 minus 4 equals the check digit of 6. (10 – 4 = 6).

      Note: This formula does have one exception: Whenever the remainder is zero (0), the check digit is always zero (0) as well.

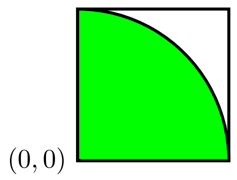
   e. **Add the check digit to the end of the 12-digit number created in Step 2. The conversion from an ISBN-10 to an ISBN-13 is complete.**

      The ISBN-13 becomes 978-0-940016-73-6.

2 Playing with higher order functions

Suppose we generate two random numbers, x and y. We can think of the pair/tuple (x, y) as specifying a point in a square. Because the coordinates are random, we get a different point every time. It is as if we are throwing darts at the square.

Consider a quarter circle inscribed in this square:



Some of the points will lie inside the shaded quarter circle (and some won't :). The probability of landing in the circle is the area of the circle relative to the square.

$$\text{probability of landing in quarter circle} = \frac{\text{area of quarter circle}}{\text{area of square}}$$

If we knew this probability, then we could calculate the area of the quarter circle as

$$\text{area of quarter circle} = \text{probability of landing in quarter circle} * \text{area of square}$$

In this case, we know exactly what the area of the quarter circle is, $\pi/4$, but if we didn't we could approximate it (and also, then, the value of π) by trying to estimate the probability of the quarter circle. This can be done by randomly picking points in the box and then measuring what proportion of the points fall inside the circle. For a quarter circle, it is easy to tell when a point (x, y) is inside the circle: It happens when $x^2 + y^2 < 1.0$.

We can actually measure the area of any arbitrary shape inside this square using a similar technique as long as we can decide if a point is inside the shape or not. This is a technique called *Monte Carlo sampling* and is common in applied mathematics.

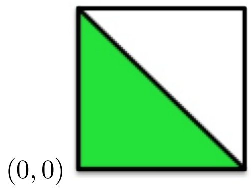
Write a Python function, `montecarlo(trials, test_function)`, which takes two parameters, the number of trials and a *function* which is related to the shape under consideration (remember we can pass functions as parameters too!). The function should take a point, represented as a tuple containing the two coordinates of a point, and returns `True` or `False` according to whether the point is in the area under consideration.

For example, we could approximate π as follows:

```
def in_circle(pt):
    (x,y) = pt
    return x*x + y*y < 1

print "pi is approximately " + str(4 * montecarlo(10000000, in_circle))
```

As another example, consider the following "shape" that just represents the lower-left half of the square:



We can write a function to determine if a point is in the triangle as:

```
def in_triangle(pt):
    (x,y) = pt
    return x + y < 1
```

and then could approximate its area (which should be 1/2) as:

```
montecarlo(10000000, in_triangle)
```

Your submission for this problem should just be a single function, `montecarlo`. *Note:* we will test your function on shapes other than the two above.

3 An introduction to the movies database

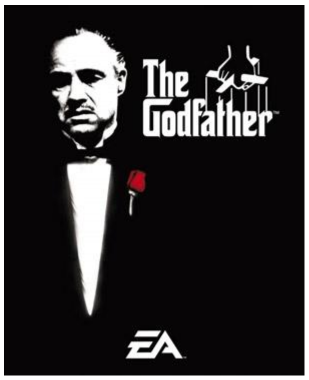
In this exercise we will start playing with the basics of a movie database. Later, we will refine it and construct a natural language query system for it.

A database is simply a collection of *records.* For our purposes, a record contains the following information about a particular movie—represented as a tuple:

(title, director, year, list-of-actors)

The title and director are strings, the year is an integer, and the list of actors is a list of strings. For example, *The Godfather* would have the record

```
('the godfather',
 'francis ford coppola',
 1972,
 ['marlon brando',
  'al pacino',
  'james caan',
  'robert duval',
  'diane keaton'])
```



We will use all lower-case letters to avoid worrying about capitalization when matching strings. (The picture was just added for your enjoyment. No pictures are actually in the database.)

The movies file can be found online. To download this file:

- In Wing, create a new file and save it as `movies.py` in your `cs30` directory.
- Go to:
  `http://www.cs.pomona.edu/~dkauchak/classes/cs30/assignments/assign3/movies.txt`
- Select all the text in this file and copy and paste it into the `movies.py` file in Wing and save.

If you want to use this database you can `import` it by adding `from movies import movieDB`. If you type this into the python shell (or include it at the top of your file) you will then be able to access the `moviesDB`. Note, the `movies.py` file *must* be in the same directory as any other .py file that imports it (like this assignment).

Now that we have access to our basic movie database, I would like you to write some statements that do the following based on the movies database.

a. Write a statement that print out how many movie records are in the database, something like "The movies database contains X records" where X is some number.

b. Use the `map` function (see Section 5.1.3 of the "official Python tutorial" linked at the bottom of the course web page or the course notes) to create a list of titles in `movieDB`, and then print that list with one title on a line. Include a preliminary `print` statements that says something like "Movies in the database:". Note, this will involve you writing your own function to then use as an argument to `map`.

c. Write a statement to insert into `movieDB` a new movie of your own choosing. Note that nothing will be displayed.

d. Use the `filter` function (again, see Section 5.1.3 of the "official Python tutorial" or the course notes) to create a list of all the records in `movieDB` for movies made in the 1970's. Then, print all of the titles in this list, with one title to a line. Include a header before these, something like "Movies in the database from the 1970s:".

e. *0.5 points extra credit:* Use the `map` function to create a list of the number of actors involved with each movie. Then, using this list, calculate the average number of actors for the movies in the database.

When you're done

Make sure that your program is properly commented:

- You should have comments at the very beginning of the file stating your name, course, assignment number and the date.

- You should have comments delimiting the three problems.
- Each function should have an appropriate docstring.
- Include other miscellaneous comments to make things clear.

In addition, make sure that you've used good *style*. This includes:

- Following naming conventions, e.g. all variables and functions should be lowercase.
- Using good variable names.
- Good use of booleans. You should NOT have anything like:)

  ```
  if boolean_expression == True:
  ```

  or

  ```
  if boolean_expression == False:
  ```

  instead use:

  ```
  if boolean_expression:
  ```

  or

  ```
  if not (boolean_expression): # or some other way of negating the expression
  ```

- Proper use of whitespace, including indenting and use of blank lines to separate chunks of code that belong together.
- Make sure that none of the lines are too long, i.e. cross the red line in Wing.

Submit your .py file online using the courses submission mechanism.

Grading

		points
Problem 1		
	correctness	6
	proper decomposition into functions	2
Problem 2		5
Problem 3		
	a.	1
	b.	2
	c.	1
	d.	2
Comments, style		3
extra credit		0.5
total		22 (+0.5)