# CS10 Online Final Solutions(Spring 2017)

There are two questions, a Snap*!* one and a Python one. Save your code for the Snap*!* question into a Snap*!* file, and name it **`FinalYourfirstnameYourlastname.xml`** (e.g., **`FinalAlanTuring.xml`**). For the Python question, create a new Python file and name it **`FinalYourfirstnameYourlastname.py`** (e.g., **`FinalAlanTuring.py`**). Submit both files on bCourses under the "online" final assignment for your lab section. Both questions are independent, each worth 10 points. Retrieve the Snap*!* starter file by going to: https://bit.ly/2H7NYDl

**Snap*!* Question:**

You devise a game to help you fall asleep faster. First, pick a whole number **N**. Then start naming **N**, 2 × **N**, 3 × **N**, and so on. Whenever you name a number, you think about all of the digits in that number, and keep track of which digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) you have seen at least once so far as part of any number you have named. Once you have seen *each of the ten digits at least once*, you fall asleep. We call this game *Knockoff*, since you're trying to "knock off" all the digits 0-9, in any order.

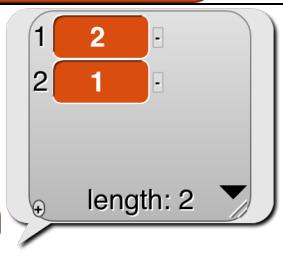
For example, suppose you pick **N** = 1692. You would count as follows:

- **N** = 1692. Now you have seen the digits 1, 2, 6, and 9.
- 2**N** = 3384. Now you have seen the digits 1, 2, 3, 4, 6, 8, and 9.
- 3**N** = 5076. Now you have seen all ten digits, and fall asleep.

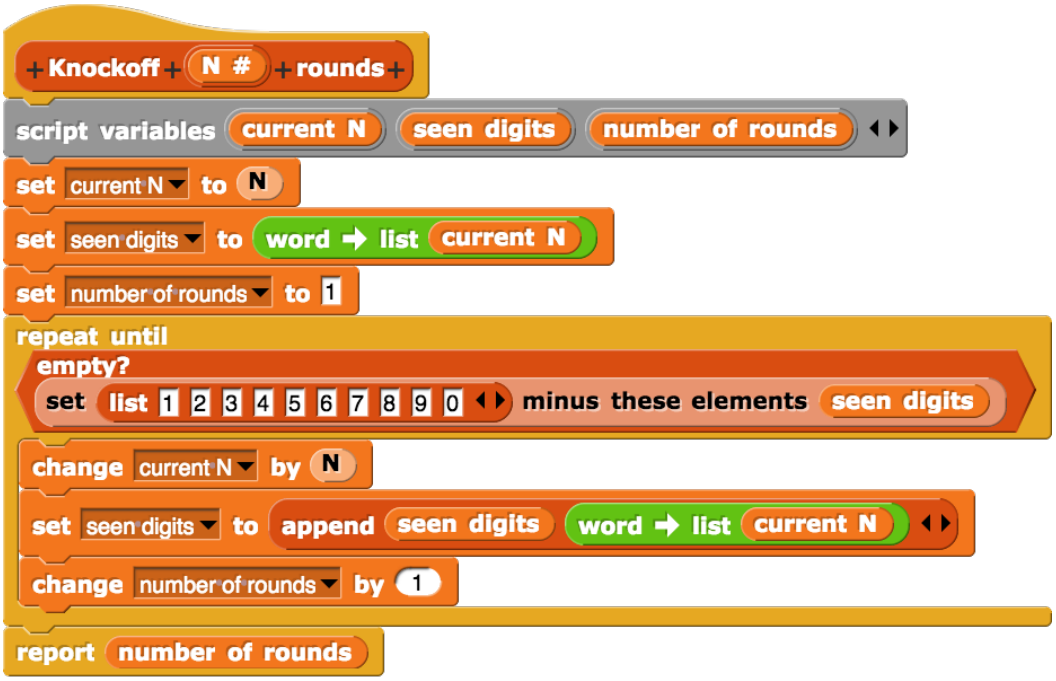
Write a block to report *how many rounds it takes* until you fall asleep (starting with **N**). The chart below shows the return value for several values of **N**; we underline the numbers when they get knocked off. It takes 45 rounds for **N**=2 to finally knock off the 9 with the number 90.

| N | 2N | 3N | 4N | 5N | 6N | 7N | 8N | 9N | 10N | ... | 45N | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1692 | 3384 | 5076 | | | | | | | | | | Knockoff 1692 rounds → 3 |
| 1234567890 | | | | | | | | | | | | Knockoff 1234567890 rounds → 1 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | | Knockoff 1 rounds → 10 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | ... | 90 | Knockoff 2 rounds → 45 |

You may find the following block helpful. It takes a *set* (a list containing no duplicate elements whose elements are not in any particular order) and returns a new set without any of the elements from the list of elements (which *may* contain duplicates). E.g.,:

**Solution:**




**Python Question:**

Write the **`fib`** function to calculate the *n*th element in the Fibonacci series (1, 1, 2, 3, 5, ...). It should work for large *n* (say, 100). You can either try to write it *iteratively*, or modify the recursive version below to remember (and use!) earlier elements in a dictionary so you aren't doing so many redundant calculations.

| | |
|---|---|
| `def fib(n):`<br>`  if n <= 2:`<br>`  return 1`<br>`  else:`<br>`  return fib(n-1) + fib(n-2)` | `>>> fib(1)`<br>`1`<br>`>>> fib(100)`<br>`354224848179261915075` |

**Solution:**

```
def fib(n):
    prev_fib = 1
    current_fib = 1
    while n > 2:
        temp_fib = current_fib
        current_fib += prev_fib
        prev_fib = temp_fib
        n -= 1
    return current_fib
```