Lab 4.2 (Deadline Friday 11 February 23:59)

- Upload your code to Einstein to have it verified.

Numbers to words

- Write a Python program called *`nums2words_v1_042.py`* that maps lines of numbers to corresponding lines of text. The lines of numbers are read from `stdin`. The number of lines is arbitrary.
- You can assume that everything read from `stdin` is a number and that number is in the range 0-10.
- For example:

```
$ cat nums2words_v1_stdin_00_042.txt
5 1 2 0 6 7 9 10 8 8 3 4
4 3 9 8 2 6 3 0  7 7 1 9
```

```
$ python3 nums2words_v1_042.py < nums2words_v1_stdin_00_042.txt
five one two zero six seven nine ten eight eight three four
four three nine eight two six three zero seven seven one nine
```

Numbers to words (with unknowns)

- Write a Python program called *`nums2words_v2_042.py`* that maps lines of numbers to corresponding lines of text. The lines of numbers are read from `stdin`. The number of lines is arbitrary.
- This time the data read from `stdin` can be anything. Every number in the range 0-10 should be mapped to corresponding text. Everything else however should be mapped to `'unknown'`.
- For example:

```
$ cat nums2words_v2_stdin_00_042.txt
5 1 2 b 6 7 99 10 8 8 3 4
4 3 9 8 2 6 3 0 x 7 1 9 11
```

```
$ python3 nums2words_v2_042.py < nums2words_v2_stdin_00_042.txt
five one two unknown six seven unknown ten eight eight three four
four three nine eight two six three zero unknown seven one nine unknown
```

Numbers to words (with translation)

- Write a Python program called *`nums2words_v3_042.py`* that maps lines of numbers to corresponding lines of text. The lines of numbers are read from `stdin`. The number of lines is arbitrary.

- You can assume that everything read from `stdin` is a number and that number is in the range 0-10.
- The program takes as an argument a file that contains a mapping from English words to their translation. Your program must make appropriate use of this mapping.
- Here is an example:

```
$ cat nums2words_v3_input_00_042.txt
zero naid
one aon
two do
three tri
four ceathar
five cuig
six se
seven seacht
eight ocht
nine naoi
ten deich
```

```
$ cat nums2words_v3_stdin_00_042.txt
5 1 2 0 6 7 9 10 8 8 3 4
4 3 9 8 2 6 3 0  7 7 1 9
```

```
$ python3 nums2words_v3_042.py nums2words_v3_input_00_042.txt < nums2words_v3_stdin_00
cuig aon do naid se seacht naoi deich ocht ocht tri ceathar
ceathar tri naoi ocht do se tri naid seacht seacht aon naoi
```

More numbers to words

- Write a Python program called *`nums2words_v4_042.py`* that maps lines of numbers to corresponding lines of text. The lines of numbers are read from `stdin`. The number of lines is arbitrary.
- You can assume that everything read from `stdin` is a number and that number is in the range 0-100.
- For example:

```
$ cat nums2words_v4_stdin_00_042.txt
5 11 22 0 66 17 99 100 18 68 73 44
4 35 91 83 27 66 30 0 71 17 16 91
```

```
$ python3 nums2words_v4_042.py < nums2words_v4_stdin_00_042.txt
five eleven twenty-two zero sixty-six seventeen ninety-nine one hundred eighteen sixty
four thirty-five ninety-one eighty-three twenty-seven sixty-six thirty zero seventy-or
```

Swapping dictionary keys and values

- Write a Python *module* `swap_v1_042.py` that defines a function called `swap_keys_values()`. The function accepts a single argument: a dictionary `d`.
- The function must return a new dictionary whose *keys* are the *values* in `d` and whose *values* are the corresponding *keys* in `d`.
- For example:

```
#!/usr/bin/env python3

from swap_v1_042 import swap_keys_values

def main():

    my_dict = {'a' : 4, 'b' : 7, 'c' : 10}
    new_dict = swap_keys_values(my_dict)
    print(sorted(new_dict.items()))

if __name__ == '__main__':
    main()
```

```
[(4, 'a'), (7, 'b'), (10, 'c')]
```

- You may assume that all values in `d` are unique and immutable.

Swapping more dictionary keys and values

- Write a new Python module called `swap_v2_042.py` that defines a function called `swap_unique_keys_values()`. The function accepts a single argument: a dictionary `d`.
- The function must return a new dictionary whose *keys* are the *unique values* in `d` and whose *values* are the corresponding *keys* in `d`.
- For example:

```
#!/usr/bin/env python3

from swap_v2_042 import swap_unique_keys_values

def main():
    my_dict = {'a' : 4, 'b' : 7, 'c' : 10, 'd' : 7}
    new_dict = swap_unique_keys_values(my_dict)
    print(sorted(new_dict.items()))

if __name__ == '__main__':
    main()
```

```
[(4, 'a'), (10, 'c')]
```

- You may assume all values in `d` are immutable. There may however be duplicate values. Since values are to become keys this presents a problem: we cannot have duplicate keys. Our solution is to simply ignore duplicate values: they will not be used as keys.