Written by: Dr. Zachary Piazza for Pickle Robot co.

Documentation for:
number_to_words.py
words_to_number.py
all_wordifications.py

This document provides a bit of information about how to use, and what to expect as results from these three python scripts which contain their corresponding functions: number_to_words(), words_to_number(), and all_wordifications.

number_to_words.py

number_to_words.py contains 1 function: *number_to_words(phone_number_string)*

which takes in a string and returns a string.
The returned string is either a wordified phone number OR 'None', as a string.

Dependencies:

1) phone_to_words_fx.py
2) phone_to_words_dicts.py
3) nltk_eng_words_3to7letters.txt

These 3 files must all be in the same directory as numbers_to_words.py

nltk_eng_words_3to7letters.txt is a trimmed down version of the english wordlist according to: https://www.nltk.org

Examples:

number_to_words("1-800-724-6837") --> "1-800-PAINTER"

number_to_words("760-599-0766") → "760-59907-OO" *(apparently OO is a real word)*

number_to_words("1") --> "None" along with a helpful message printed to the terminal

Some Details:

number_to_words() is only set up for English words. Right now some slang words are not included.

number_to_words() only deals with the last 7 digits of a number. In other words an area code like (617), or common prefix like 1-800 will be left as-is.

number_to_words() will search for the ***longest possible single-word*** wordification. The code in **all_wordifications.py** will return multiple-word wordifications

(Ex: “617-969-8888” can be transformed to “617-WOW-8888”, but number_to_words() returns “617-969-TUTU”, since TUTU is a longer word than WOW, and no 5+ letter single-word wordifications are found)

(Ex: “244-2287” can be transformed to “BIG-BATS”, but number_to_words() returns “2-HICCUP because it is a single-word wordification and HICCUP is longer than either BIG or BATS”

words_to_number.py

words_to_number.py contains 2 functions: *words_to_number(wordified_number_string)*

dash_formatter(phone_number_len, i, d)

dash_formatter() is not very important. it simply helps format the final result.

words_to_number(wordified_number_string) takes a string that should looked like a “wordified” phone number and returns a string.

The returned string is either the “digitified”/”un-wordified” phone number as a string OR “input string may not represent US phone number”

Dependencies:

1) phone_to_words_dicts.py

phone_to_words_dicts.py must be in the same directory as words_to_number.py

Examples:

words_to_number(“1-800-PAINTER”) --> “1-800-724-6837”

words_to_number(“11!1225”) --> “'input string may not represent US phone number”

words_to_number(“2-hiccup”) --> “'224-2287”

all_wordifications.py

all_wordifications.py contains 2 functions: *all_wordifications(phone_number_string)*

return_all_word_combos()

The user need only be concerned with all_wordifications().

all_wordificiations(phone_number_string) takes a phone number as a string and returns a list of strings containing all possible combinations of numbers and English words associated with the original number. In other words, it returns all possible “wordifications” of the input phone number.

In addition, all wordified results will be printed to the terminal.

If the original input can not be transformed into a 11, 10, or 7 digit number, the code will return [‘None’] (in list form) and will print a handy message to the terminal.

If no wordifications are found, the function will print “no wordifications found.” to the terminal, and return only the original input string which should be a valid phone number.

Dependencies:

1) phone_to_words_fx.py
2) phone_to_words_dicts.py
3) nltk_eng_words_3to7letters.txt

These 3 files must all be in the same directory as all_wordifications.py

nltk_eng_words_3to7letters.txt is a trimmed down version of the english wordlist according to: https://www.nltk.org

Some notes:

all_wordifications() is very liberal in some sense about what is a valid woridification. For example, the phone number 222-2222 can return a 7-word wordification:

all_wordifications(“222-2222”) --> [“222-2222”, [“2-BACABA”],…,[“A-A-A-A-A-A-A”]]

I learned today that a Bacaba is a palm of the genus Oenocarpus with drupelike fruits that yield oil used in soap manufacture.

Examples:

```
all_wordifications(“1”) -->  [“None”]

all_wordifications(“!”) -->  [“None”]

all_wordifications(“1-800-724-6837”) -->  [“1-800-724-6837”,
                                           “1-800-PAINTER”,
                                           “1-800-7-CINTER”,
                                           “1-800-72-INTER”,
                                           “1-800-72-HOVER”,
                                            …,
                                            “1-800-7-A-I-NU-ES”]     (428 wordifications)

all_wordifications(“244-2287”) -->         [“244-2287”,
                                            “2-HICCUP”,
                                            …,
                                           “BIG-BATS”,
                                            …,
                                           “A-I-I-A-A-US”]       (1208 wordifications)
```