Lab 3: Singing a song

Daniel Chamberlin

Table of contents

Introduction

The song "12 Days of Christmas", written around 1780, tells the tale of many gifts a person receives in the days leading up to Christmas (link to lyrics.

Note You can watch a video of the 12 Days of Christmas at the Cambria Christmas Market.

These gifts repeat and compound; on the first day, the narrator receives

A partridge in a pear tree. On the twelfth day, they receive

Twelve Drummers Drumming Eleven Pipers Piping Ten Lords a Leaping Nine Ladies Waiting Eight Maids a Milking Seven Swans a Swimming Six Geese a Laying Five Golden Rings Four Calling Birds Three French Hens Two Turtle Doves And a Partridge in a Pear Tree This week, your task will be to write functions that automatically sing this very repetitive song.

Data set

Run the code provided to load in a data set called xmas that contains the crucial information about the gifts in the song. We will use this data set to test out our functions as we work on them.

```
import pandas as pd
xmas = pd.read_csv("https://www.dropbox.com/scl/fi/qxaslqqp5p08i1650rpc4/xmas.csv?rlkey=erdx
remove_th = {
    1: "one",
    2: "two",
    3: "three",
    4: "four",
    5: "five",
    6: "six",
    7: "seven",
    8: "eight",
    9: "nine",
    10: "ten",
    11: "eleven",
    12: "twelve"
}
xmas['value'] = xmas['Day'].map(remove_th)
xmas
```

	Day	Day.in.Words	Gift.Item	Verb	Adjective	Location	value
0	1	first	partridge	NaN	NaN	in a pear tree	one
1	2	second	dove	NaN	turtle	NaN	two
2	3	third	hen	NaN	french	NaN	three
3	4	fourth	bird	NaN	calling	NaN	four
4	5	fifth	ring	NaN	golden	NaN	five
5	6	sixth	goose	a-laying	NaN	NaN	six
6	7	seventh	swan	a-swimming	NaN	NaN	seven
7	8	eighth	maid	a-milking	NaN	NaN	eight
8	9	ninth	lady	dancing	NaN	NaN	nine

	Day	Day.in.Words	Gift.Item	Verb	Adjective	Location	value
9	10	tenth	lord	a-leaping	NaN	NaN	ten
10	11	eleventh	piper	piping	NaN	NaN	eleven
11	12	twelfth	drummer	drumming	NaN	NaN	twelve

Tip Your functions can - and should! - reference each other. That is, don't duplicate code; use earlier, smaller functions inside your larger functions.

Advice

- If you have some trouble getting started, I recommend writing a function that works in one case, and then trying to generalize. For example, in building my sing_day() function, I might first write a version called sing_third_day() that sings

  ```
  On the third day of Christmas, my true love gave to me:
  three french hens,
  two turtle doves,
  and a patridge in a pear tree.
  ```

- Make smaller versions of the xmas data set (e.g., the first two days). Once you feel confident in your function code, use the smaller version of the data to test the functions you write, before you test them on the full data set.
- Don't sweat the small stuff. There's a lot you can do to polish up the way the song prints. However, the goal of this lab is to practice writing functions and using iteration. Don't get bogged down in details like how the song displays, or small grammar rules (like commas!), until you've finished the main tasks. You will have a chance to do this on the Challenge for this week!

Function 1: pluralize_gift()

The gifts are listed in singular: for example, on day five the narrator receives "five golden rings", but the entry in the data set for the gift on day five simply says "ring".

- Hint 1: The gifts on days six and nine have unusual pluralization. You may assume that in other data sets, there will be no additional special cases besides these types.
- Hint 2: The following small code snippets may be useful to you:

```
obj_1 = "foot"
obj_2 = "baby"
obj_3 = "tree"

obj_1.find("oo")
obj_2[-1]
obj_3.find("oo")

obj_1.replace("oo", "ee")
obj_2.replace("y", "ies")
obj_3 + "s"
```

```
Warning
You should absolutely not "hard-code" anything into this function; this function should work
```

Using the skeleton of the pluralize_gift() function, complete the code so that the function takes a gift and returns the appropriate plural.

```
def pluralize_gift(gift):
  """
  Returns plural of a noun

  Parameters
  ----------
  gift: str
    A noun

  Return
  ------
  str
    Plural version
  """
  # Check to make sure is vectorized for a data frame
  if isinstance(gift, pd.Series):
        return gift.apply(pluralize_gift)

  if gift.find("oo") != -1:
    gift = gift.replace("oo", "ee")
  elif gift[-1] == 'y':
    gift = gift.replace("y", "ies")
  else:
    gift = gift + "s"

  return gift
```

Test Your Function

Try your function out on the smaller and then larger gift data set. Consider: is your function vectorized? If not, how would you run it on all the gifts in the column.

```
# Should work
print(pluralize_gift("goose"))

# Will work if your function is vectorized!
print(pluralize_gift(xmas['Gift.Item']))
```

```
geese
0     partridges
1          doves
2           hens
3          birds
4          rings
5          geese
6          swans
7          maids
8         ladies
9          lords
10        pipers
11      drummers
Name: Gift.Item, dtype: object
```

Function 2: make_phrase()

Write a function called make_phrase() that takes as input the necessary information, and returns a phrase. For example,

```
make_phrase(num_word = "ten",
            item = "lords",
            verb = "a-leaping",
            adjective = "",
            location = "")
```

should return

“ten lords a-leaping”

```
import numpy as np
def make_phrase(num, num_word, item, verb, adjective, location):
  """
    Constructs a phrase based on the input parameters.

    Parameters
    ----------
    num_word: str
        The word representation of the number
    num: int
        The numeric value
    item: str
        The item being described
    verb: str
        The action verb
    adjective: str
        An adjective describing the item
    location: str
        The location for the action

    Returns
    -------
    str
        A complete phrase
  """
  # Turns all floats to strings for concatination at end
  verb = str(verb)
  adjective = str(adjective)
  location = str(location)

  # Will take out all NaN values
  if not isinstance(verb, str):
    verb = verb.fillna("")
  else:
    if verb == "nan":
      verb = ""
  if not isinstance(adjective, str):
    adjective = adjective.fillna("")
  else:
    if adjective == "nan":
      adjective = ""
  if not isinstance(location, str):
    location = location.fillna("")
  else:
    if location == "nan":
      location = ""


  # Step 2: If the day number is larger than 1, the gift items need pluralized!
  ### Hint: call the function you created above!
  if num != 1:
    item = pluralize_gift(item)

  # Step 3: Figure out if a gift item starts with a vowel
  vowel = False
  if item[0] in ("A","E","I","O","U","a","e","i","o","u"):
    vowel = True

  # Step 4: For the first day, if the gift item starts with a vowel, replace the day with "ar
  # a vowel, replace the day with "a" (e.g. a partridge in a pear tree). If it is not the fir
  # (e.g. ten lords a leap)
  if num_word == "first" or num_word == "one":
    if vowel == True:
      num_word = "an"
    else:
      num_word = "a"


  # Step 5: Put all of the pieces together into one string and return!
  phrase = []
  phrase.append(num_word)
  if adjective != '':
    phrase.append(adjective)
  phrase.append(item)
  if verb != '':
    phrase.append(verb)
  if location != '':
    phrase.append(location)

  return ' '.join(phrase)
```

Tip The Day.in.Words variable isn't quite what you want! You want 12 to say "twelve" not "twelfth". Consider using .map() and a dictionary to make a column of the words you need.

If you have trouble with this part, leave the number as is, e.g. 5 golden rings, it will only be a couple points off.

Test Your Function

Make sure to try your function out on small examples and on the xmas data.

Then, use the function to make a new column of the xmas column called Full. Phrase containing the sentences for the new gift on that day.

```python
import unittest

class TestTimesSeven(unittest.TestCase):

    # Test example
    def test_phrase1(self):
        self.assertEqual(make_phrase(num_word = "ten",
                num = 10 ,
                item = "lord",
                verb = "a-leaping",
                adjective = "",
                location = ""), 'ten lords a-leaping')

    # Test differnt plural
    def test_phrase2(self):
        self.assertEqual(make_phrase(num_word = "five",
                num = 5 ,
                item = "goose",
                verb = "a-laying",
                adjective = "",
                location = ""), 'five geese a-laying')
    def test_phrase3(self):
        self.assertEqual(make_phrase(num_word = "nine",
                num = 9 ,
                item = "lady",
                verb = "dancing",
                adjective = "",
                location = ""), 'nine ladies dancing')

    # Test adjective
    def test_phrase4(self):
        self.assertEqual(make_phrase(num_word = "three",
                num = 3 ,
                item = "hen",
                verb = "",
                adjective = "french",
                location = ""), 'three french hens')
    def test_phrase5(self):
        self.assertEqual(make_phrase(num_word = "three",
                num = 3 ,
                item = "hen",
                verb = "flying",
                adjective = "french",
                location = ""), 'three french hens flying')

    # Test location and first
    def test_phrase6(self):
        self.assertEqual(make_phrase(num_word = "one",
                num = 1 ,
                item = "partridge",
                verb = "",
                adjective = "",
                location = "in a pear-tree"), 'a partridge in a pear-tree')

    # Test with xmas df
    # def test_phrase7(self):
        row = xmas.iloc[1]  # Row for day 2
        self.assertEqual(make_phrase(num= row['Day'], num_word= row['value'], item= row['Gif
                                     verb= row['Verb'], adjective= row['Adjective'], locatio
                         'two turtle doves')


if __name__ == '__main__':
    unittest.main(argv=[''], exit=False)
```

```
......
----------------------------------------------------------------------
Ran 6 tests in 0.014s

OK
```

```
# Create Full.Phrase column
xmas['Full.Phrase'] = xmas.apply(lambda row: make_phrase(
    row['Day'],
    row['value'],
    row['Gift.Item'],
    row['Verb'],
    row['Adjective'],
    row['Location']), axis=1)
print(xmas)
```

```
    Day Day.in.Words  Gift.Item        Verb Adjective        Location   value  \
0     1        first  partridge         NaN       NaN  in a pear tree     one
1     2       second       dove         NaN    turtle             NaN     two
2     3        third        hen         NaN    french             NaN   three
3     4       fourth       bird         NaN   calling             NaN    four
4     5        fifth       ring         NaN    golden             NaN    five
5     6        sixth      goose    a-laying       NaN             NaN     six
6     7      seventh       swan  a-swimming       NaN             NaN   seven
7     8       eighth       maid   a-milking       NaN             NaN   eight
8     9        ninth       lady     dancing       NaN             NaN    nine
9    10        tenth       lord   a-leaping       NaN             NaN     ten
10   11     eleventh      piper      piping       NaN             NaN  eleven
11   12      twelfth    drummer    drumming       NaN             NaN  twelve

                   Full.Phrase
0   a partridge in a pear tree
1             two turtle doves
2            three french hens
3           four calling birds
4            five golden rings
5           six geese a-laying
6       seven swans a-swimming
7        eight maids a-milking
8          nine ladies dancing
9          ten lords a-leaping
10        eleven pipers piping
11   twelve drummers drumming
```

Function 3: sing_day()

Write a function called sing_day() that takes as input:

- A dataset (input as a dataframe)
- A number indicating which day to sing about (input as an integer)
- The name of a column in the dataset that contains the phrases for each day (input as an tidy name)

For example,

sing_day(xmas, 2, Full.Phrase) should return

```
On the second day of Christmas, my true love sent to me:
two turtle doves and
a partridge in a pear tree.
```

```python
def sing_day(dataset, num, phrase_col):
  """
    Constructs the lyrics for the specified day of Christmas.

    Parameters
    ----------
    dataset: pd.DataFrame
        A DataFrame containing the phrases for each day.
    num: int
        The day of Christmas to sing about (1-12).
    phrase_col: tidy name
        The name of the column in the DataFrame containing the phrases.

    Returns
    -------
    str
        The complete lyrics for the specified day of Christmas.
  """

  #Step 1: Setup the intro line
  month_order = {
    1: "first",
    2: "second",
    3: "third",
    4: "fourth",
    5: "fifth",
    6: "sixth",
    7: "seventh",
    8: "eighth",
    9: "ninth",
    10: "tenth",
    11: "eleventh",
    12: "twelth"}
  num_order = month_order[num] # convert "1" to "first" etc.
  intro = "On the " + num_order + " day of Christmas, my true love sent to me:"

 #Step 2: Sing the gift phrases
 ### Hint: What order are they gifts sung in each day?
  song = []
  song.append(intro)
  for i in range(num-1, 1, -1):
    song.append(dataset[phrase_col][i]+',')
  song.append(dataset[phrase_col][1]+", and")
  song.append(dataset[phrase_col][0]+'.\n')


  #Step 3: Put it all together and return
  return "\n".join(song)
```

Test your function

Use this code to show the function works:

```
print(sing_day(xmas, 3, "Full.Phrase"))

# Full song
print(sing_day(xmas, 12, "Full.Phrase"))
```

```
On the third day of Christmas, my true love sent to me:
three french hens,
two turtle doves, and
a partridge in a pear tree.

On the twelth day of Christmas, my true love sent to me:
twelve drummers drumming,
eleven pipers piping,
ten lords a-leaping,
nine ladies dancing,
eight maids a-milking,
seven swans a-swimming,
six geese a-laying,
five golden rings,
four calling birds,
three french hens,
two turtle doves, and
a partridge in a pear tree.
```

Use Your Functions!

Run appropriate code to output the lyrics for the entire 12 Days of Christmas song.

Then, load the following dataset, and run your code again on this dataset instead to get a surprise song! (The column names and formats of xmas2 are the same as those for xmas.)

```python
xmas2 = pd.read_csv("https://www.dropbox.com/scl/fi/p9x9k8xwuzs9rhp582vfy/xmas_2.csv?rlkey=k
remove_th = {
    1: "one",
    2: "two",
    3: "three",
    4: "four",
    5: "five",
    6: "six",
    7: "seven",
    8: "eight",
    9: "nine",
    10: "ten",
    11: "eleven",
    12: "twelve"
}
xmas2['value'] = xmas2['Day'].map(remove_th)

xmas2['Full.Phrase'] = xmas2.apply(lambda row: make_phrase(
    row['Day'],
    row['value'],
    row['Gift.Item'],
    row['Verb'],
    row['Adjective'],
    row['Location']), axis=1)

print(sing_day(xmas2, 12, "Full.Phrase"))
```

```
On the twelth day of Christmas, my true love sent to me:
twelve hours sleeping,
eleven friends goodbye-ing,
ten loads of laundry,
nine parties bumping,
eight moms a-calling,
seven seniors stressing,
six graders grading,
five practice exams,
four course reviews,
three lost pens,
two meal points, and
an email from Cal Poly.
```

Warning In this lab, you will get automatic deductions for: Functions that do not work the way they are intended "Hard-Coding" in functions; e.g. writing "rings". Using loops instead of .map() to iterate the function you just wrote. Not singing the full song. Remember, each day, you get the gift for that day and all the prior days. ## Make it nice It would be nice if your song output in a visually appealing way, with proper spacing and grammar.

This section of the lab is only worth the points indicated below; that is, you can achieve an A-range grade of 95/100 without any of these add-ons!

Whitespace (+1)

Remove any additional spaces between words, such that there should only be one whitespace character between words and no whitespace at the beginning or end of each phrase.

New lines (+1)

Ensure each phase (e.g., "two turtle doves") of your song is printed on its own line. For example, your function should output the following for sing_line(xmas, num = 2, phrase_col = Full.Phrase):

```
    On the second day of Christmas, my true love gave to me:
    two turtle doves and
    a partridge in a pear tree
```

Separating lines (+1)

Ensure there are blank spaces between the different lines of the song. For example, when iterating your function, your output should look like the following:

```
On the second day of Christmas, my true love gave to me:
two turtle doves and
a partridge in a pear tree

On the first day of Christmas, my true love gave to me:
a partridge in a pear tree
```

Grammar (+2)

Ensure the lines of your song are grammatically correct. There are three components you should address in making your lines grammatically correct:

1. Use of Commas – each line should end in a comma except for the last line
2. Use of And – there should be an "and" included either at the end of the second to last line or at the beginning of the final line
3. Use of Period – there should be a period at the end of the final line For example, your function should output the following for sing_line(xmas, num = 3, phrase_col = "Full.Phrase"):

```
On the third day of Christmas, my true love gave to me:
three french hens,
two turtle doves, and
a partridge in a pear tree.
```