Lab 3 - Singing a song

AUTHOR
Ben Laufer

▼ Code

```
#libraries
import pandas as pd
import numpy as np

#!pip install num2words
from num2words import num2words
```

▼ Code

```
#data
xmas = pd.read_csv("https://www.dropbox.com/scl/fi/qxaslqqp5p08i1650rpc4/xmas.csv?rlkey=e
```

Function 1: pluralize_gift()

▼ Code

```
#basic tests
#obj_1 = "goose"
#obj_2 = "lady"
#obj_3 = "ring"

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

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

▼ Code

```
#assumes plural of dove is doves and plural of partridge is partridges

def pluralize_gift(gift):

  """
  Returns plural of a noun

  Parameters
  ----------
  gift: str
```

```
    A noun

  Return
  ------
  str
    Plural version
  """

# ex goose -> geese
  if gift.find("oo") == True:
    gift = gift.replace("oo", "ee")

# ex lady -> ladies
  elif gift.endswith("y") == True:
    gift = gift.replace("y", "ies")

# ring -> rings
  else:
    gift = gift + "s"

  return gift
```

Unit Test For Function 1

▶ Code

```
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()

▼ Code

```
#function that converts word to numeric val
def word_to_num(num):

  """
  Returns a word

  Parameters
```

```
    ----------
    num: num

    Return
    ------
    word: str
    """

    map = {
        "zero": 0,
        "one": 1,
        "two": 2,
        "three": 3,
        "four": 4,
        "five": 5,
        "six": 6,
        "seven": 7,
        "eight": 8,
        "nine": 9,
        "ten": 10,
        "eleven": 11,
        "twelve": 12
    }

    word = map[num]

    return(word)
```

▼ Code

```
#unit test of word_to_num function, works
#word_to_num("twelve")
```

▼ Code

```
def make_phrase(num, num_word, item, verb, adjective, location):

    """
    Returns a phrase

    Parameters
    ----------
    num: num
    num_word: str
    item: str
    verb: str
    adjective: str
    location: str
      A noun
```

```
    Return
    ------
    phrase: str
    """

    ## Step 1: Replace NAs with blank strings
    #checks if each item is na, if it is, then sets to blank strings
    if pd.isna(item):
      item = ""
    if pd.isna(verb):
      verb = ""
    if pd.isna(adjective):
      adjective = ""
    if pd.isna(location):
      location = ""

    #places the the hyphon with a space from verb if there is one
    if "-" in verb:
      verb = verb.replace("-", " ")

    #if no num is given
    #convert string num to num
    if pd.isna(num):
      num = word_to_num(num_word)

    #if no num_word is given
    #elif pd.isna(num_word):
      #num_word = num2words(num)

    #converts num_word to the actual number rather than the like "TWELFTH"
    num_word = num2words(num)

    #capitalize num_word
    num_word = num_word.capitalize()

    ## 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
    ## Step 4: For the first day, if the gift item starts with a vowel, replace the day wit
    if num == 1:
      if item.startswith(('a', 'e', 'i', 'o', 'u')):
        num_word = "an"
      else:
        num_word = "a"

    ## Step 5: Put all of the pieces together into one string and return!
```

```
  #if location is na, then the adjective is describing the item NOT the location
  # location would be "" b/c we assigned it that earlier
  if location == "":

    #refrenced chat GPT.
    #stores all the parts into one list
    phrase_parts = [num_word, adjective, item, verb, location]

    #joins each part for all the parts in phrase_part, and joins then by " "
    #and then strip removes any leading/trailing spaces that might occur.
    phrase = " ".join(part for part in phrase_parts if part).strip()

  else:
    phrase_parts = [num_word, item, verb, adjective, location]
    phrase = " ".join(part for part in phrase_parts if part).strip()

  return(phrase)
```

Unit Tests for Function 2

▼ Code

```
#unit test, works
#make_phrase(num = 1, num_word = pd.NA, item = "goose", verb = "in a", adjective = "pear"
```

▼ Code

```
#unit test for xmas dataset, works
xmas['Full.Phrase'] = xmas.apply(lambda x: make_phrase(x['Day'], x['Day.in.Words'], x['Gi
xmas['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
Name: Full.Phrase, dtype: object
```

Function 3: sing_day()

▼ Code

```
def sing_day(dataset, num, phrase_col):

  """
  Returns a phrase

  Parameters
  ----------
  dataset: dat
  num: num
  phrase_col: str

  Return
  ------
  full_phrase: str
  """

  # Step 1: Setup the intro line
  # convert "1" to "first" etc.
  num_word = num2words(num, to='ordinal')
  intro = "On the " + num_word + " 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?

  # use .head so it only stores the desired amount of days
  gifts = dataset[phrase_col].head(num)
  #reverse gifts order
  #refrenced chat GPT
  gifts_reverse = list(reversed(gifts))

  # Step 3: Put it all together and return
  #store song outside
  song = intro
  for i in range(num):
    #checks if its the last line to add the and
    if(i == num - 1):
      song = song + "\n" + "and " + gifts_reverse[i] + "."
    else:
      song = song + "\n" + gifts_reverse[i] + ","

  return(song)
```

Unit Test for Function 3

▼ Code

```
#unit test, works
test1 = sing_day(xmas, 10, "Full.Phrase")

print(test1)
```

```
On the tenth day of Christmas, my true love sent to me:
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.
On the tenth day of Christmas, my true love sent to me:
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.
```

Here is an improved version of the sing_day function that doesn't need the col name and instead using the phrase function to make the song!

▼ Code

```
#here is an improv
def sing_day_improved(dataset, num):

  """
  Returns a phrase

  Parameters
  ----------
  dataset: dat
  num: num

  Return
  ------
  full_phrase: str
  """

  # Step 1: Setup the intro line
  # convert "1" to "first" etc.
  num_word = num2words(num, to='ordinal')
  intro = "On the " + num_word + " 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?
```

```
  #store gifts str
  gifts = ""
  #heres where it makes it so you don't even need the phrase col :)
  #use .head so it only stores the desired amount of days
  gifts = gifts + dataset.head(num).apply(lambda x: make_phrase(x['Day'], x['Day.in.Words

  #reverse gifts order
  #refrenced chat GPT
  gifts_reverse = list(reversed(gifts))

  # Step 3: Put it all together and return
  #store song outside
  song = intro
  for i in range(num):
    #checks if its the last line to add the and
    if(i == num - 1):
      #line for last line of song
      song = song + "\n" + "and " + gifts_reverse[i] + "."
    else:
      #line for every other line except first and last
      song = song + "\n" + gifts_reverse[i] + ","

  return(song)
```

▼ Code

```
#unit test 1, works
test2 = sing_day_improved(xmas, 10)

print(test2)
```

```
On the tenth day of Christmas, my true love sent to me:
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.
On the tenth day of Christmas, my true love sent to me:
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.
```

▼ Code

```
#unit test 2 using xmas2 dataset, works
xmas2 = pd.read_csv("https://www.dropbox.com/scl/fi/p9x9k8xwuzs9rhp582vfy/xmas_2.csv?rlke

test3 = sing_day_improved(xmas2, 10)

print(test3)
```

```
On the tenth day of Christmas, my true love sent to me:
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.
On the tenth day of Christmas, my true love sent to me:
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.
```