1 Problem with solution 85

1.1 TSNE representation

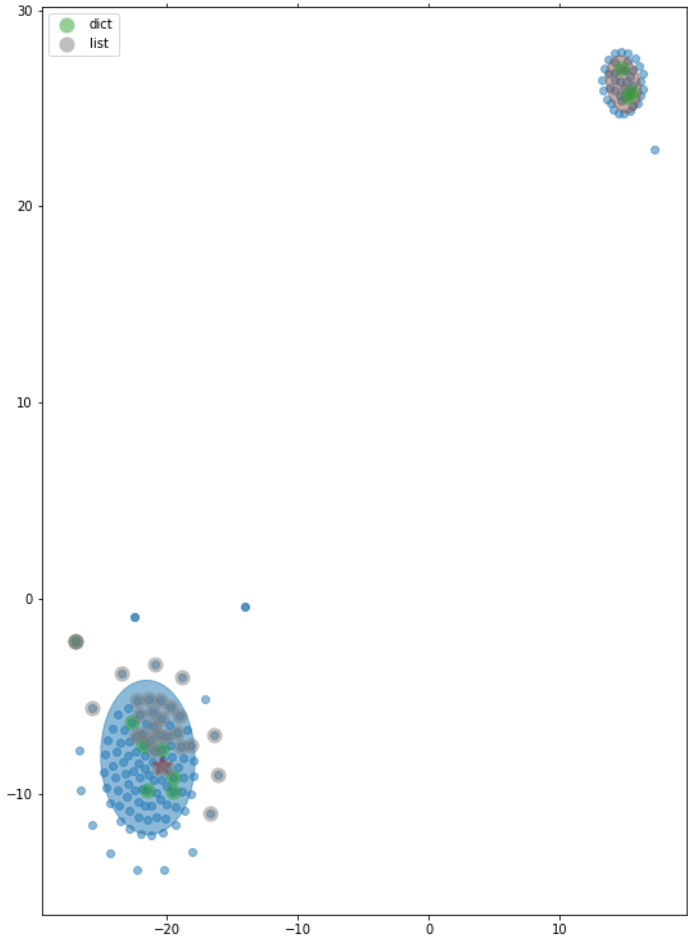

Figure 1: TSNE 2d feature space

1.2 Problem and Solution

The string of a balloon is 120m long and makes an angle of 70 degrees with the horizontal. What is the height of the balloon.

```
import math

 print("Balloon Height")
 print("This program calculates how high a balloon has reached")
 print("when given the number of degrees the string makes with the horizontal")
 print("and the length of the string")
 print()
 degreesFromHorizontal = float(70)
 lengthOfString = float(120)
 #degrees in radians
 radiansFromHorizontal = math.radians(degreesFromHorizontal)
 #sine
 sine = math.sin(radiansFromHorizontal)
 balloonHeight = round(sine * lengthOfString,2)
 print("The balloon is {0} in the air.".format(balloonHeight))
```

2 Problem with solution 121

2.1 TSNE representation

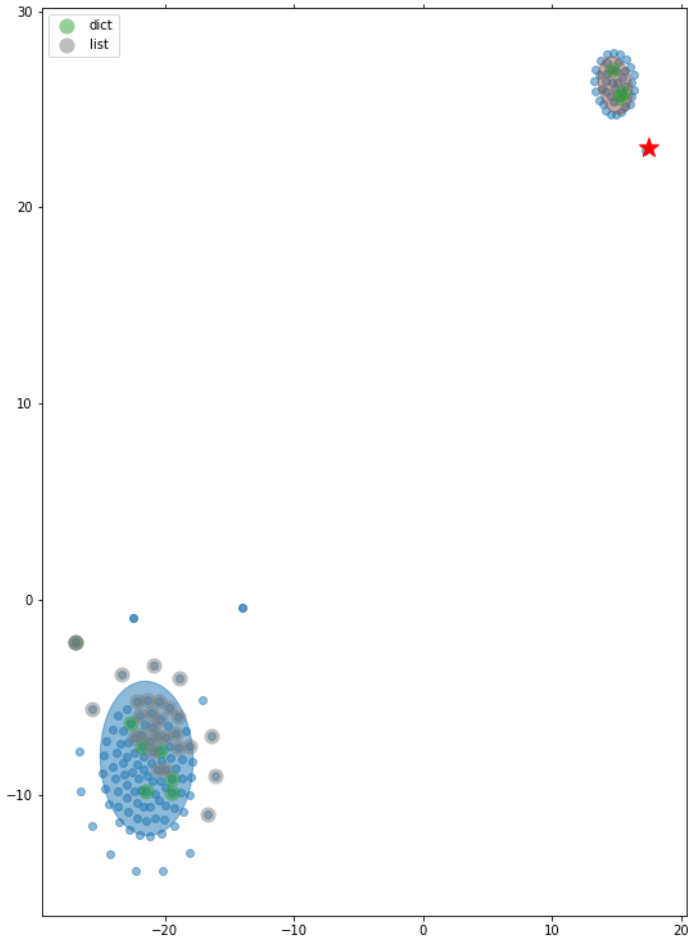

Figure 2: TSNE 2d feature space

2.2 Problem and Solution

Very Challenging Exercise 3 asked you to create a program to convert from hexadecimal to decimal. Refactor this program to use functions.

```
def convertToSegment(hexSeg):
    if hexSeg == 'A':
        hexSeg = 10
    elif hexSeg == 'B':
        hexSeg = 11
    elif hexSeg == 'C':
        hexSeg = 12
    elif hexSeg == 'D':
        hexSeg = 13
    elif hexSeg == 'E':
        hexSeg = 14
    elif hexSeg == 'F':
        hexSeg = 15
    else:
        hexSeg = int(hexSeg)
    return hexSeg

def displayInfo():
    print("Hexadecimal to Denary Conversion")
    print("This program converts a given hexadecimal number into the")
    print("equivalent denary number.")
    print()

def getHexNumber():
    hexNumber = input("Please enter a hex number to convert: ")
    return hexNumber

def convertToDenary(hexNumber):
    denary = 0
    #get lenth of number
    lengthHex = len(hexNumber)
    for element in range(lengthHex):
        #print(element)
        hexSeg = hexNumber[element]
        hexSeg = convertToSegment(hexSeg)
        #work out the place value power of 16
        placePower = 16**(lengthHex-(element+1))
        hexSeg = hexSeg * placePower
        denary = denary + hexSeg
    return denary

def displayDenaryValue(denary):
    print(denary)


def hexToDenary():
    displayInfo()
    hexNum = getHexNumber()
    denNum = convertToDenary(hexNum)
    displayDenaryValue(denNum)
```

3 Problem with solution 1

3.1 TSNE representation

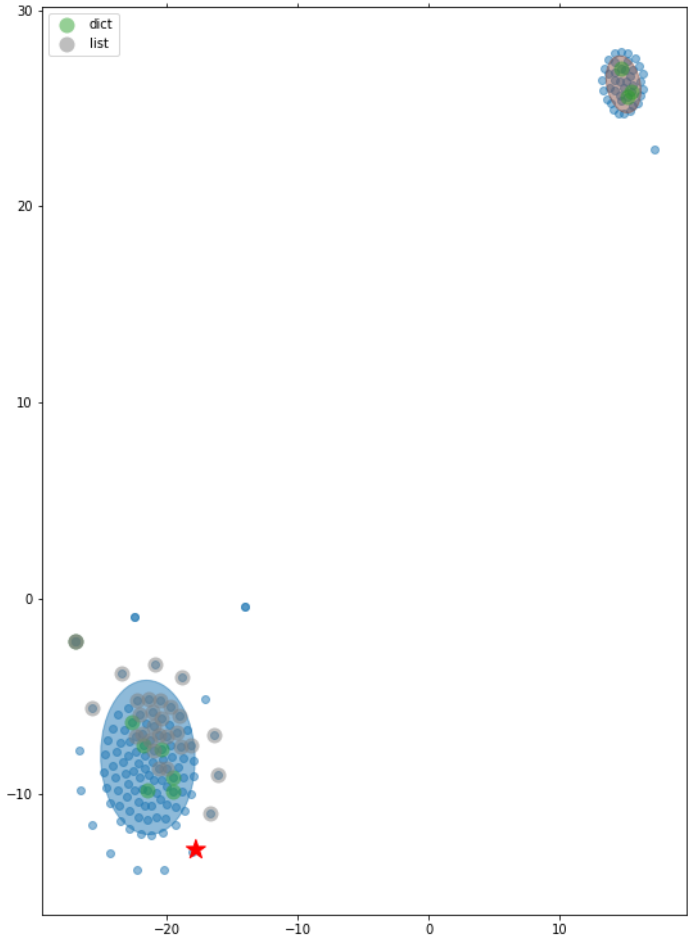

Figure 3: TSNE 2d feature space

3.2 Problem and Solution

Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user. Hint: how does an even / odd number react differently when divided by 2?
Extras:

If the number is a multiple of 4, print out a different message.
Ask the user for two numbers: one number to check (call it num) and one number to divide by (check). If check divides evenly into num, tell that to the user. If not, print a different appropriate message.

```
num = input("Enter a number: ")
mod = num % 2
if mod > 0:
    print("You picked an odd number.")
else:
    print("You picked an even number.")
```

4 Problem with solution 138

4.1 TSNE representation

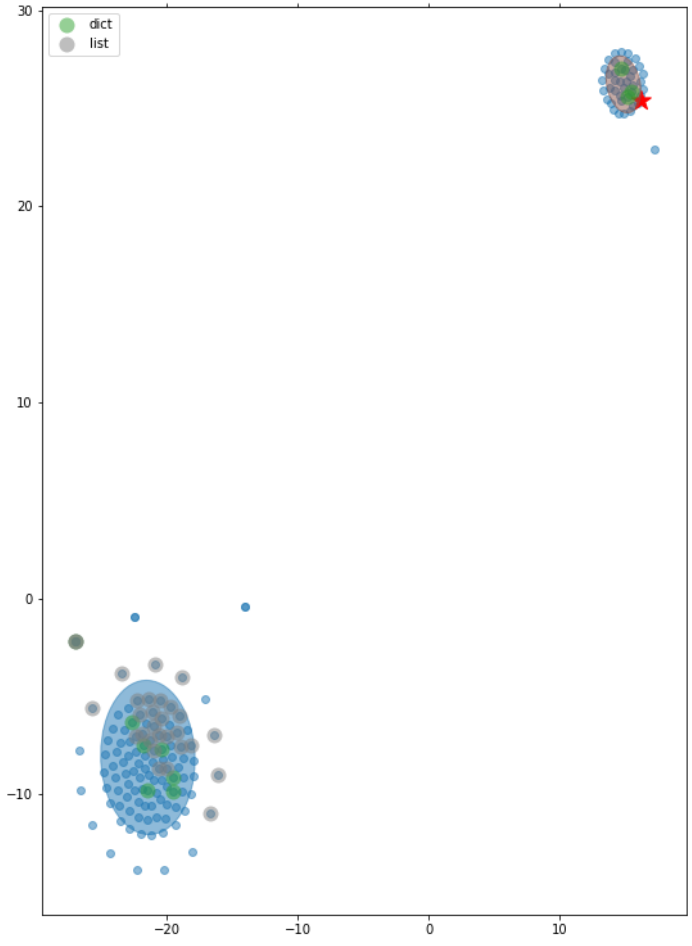

Figure 4: TSNE 2d feature space

4.2 Problem and Solution

Write a program that calculates and prints the value according to the given formula:
Q = Square root of [(2 * C * D)/H]
Following are the fixed values of C and H:
C is 50. H is 30.
D is the variable whose values should be input to your program in a comma-separated sequence.
Example
Let us assume the following comma separated input sequence is given to the program:
100,150,180
The output of the program should be:
18,22,24

```
#!/usr/bin/env python
import math
c=50
h=30
value = []
items=[x for x in raw_input().split(',')]
for d in items:
    value.append(str(int(round(math.sqrt(2*c*float(d)/h)))))

print ','.join(value)
```

5 Problem with solution 133

5.1 TSNE representation

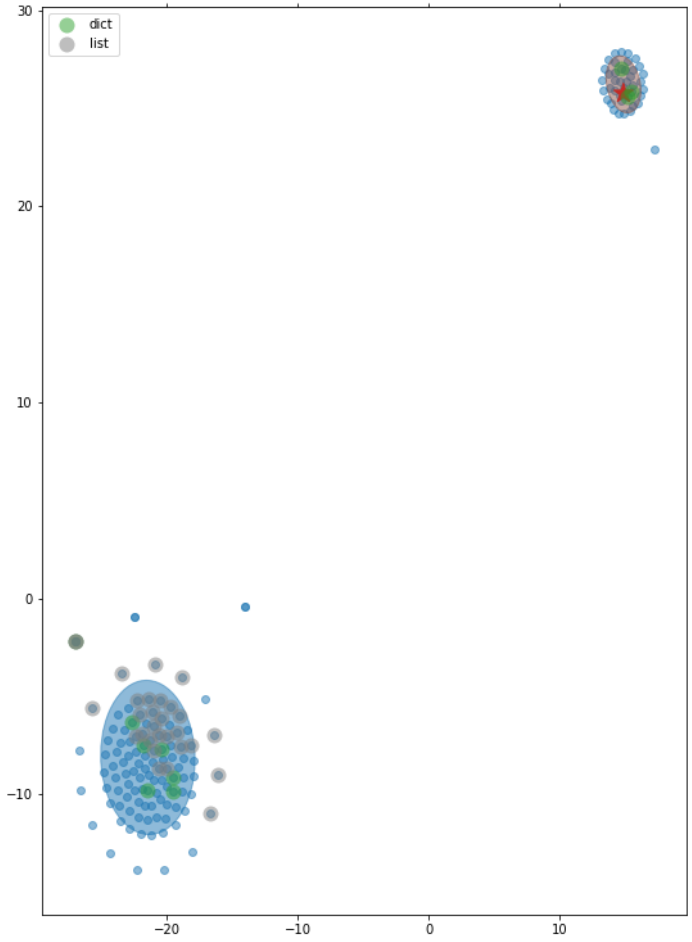

Figure 5: TSNE 2d feature space

5.2 Problem and Solution

Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5,
between 2000 and 3200 (both included).
The numbers obtained should be printed in a comma-separated sequence on a single line.

```
l=[]
for i in range(2000, 3201):
    if (i%7==0) and (i%5!=0):
        l.append(str(i))

print ',',.join(l)
```

6 Problem with solution 2

6.1 TSNE representation

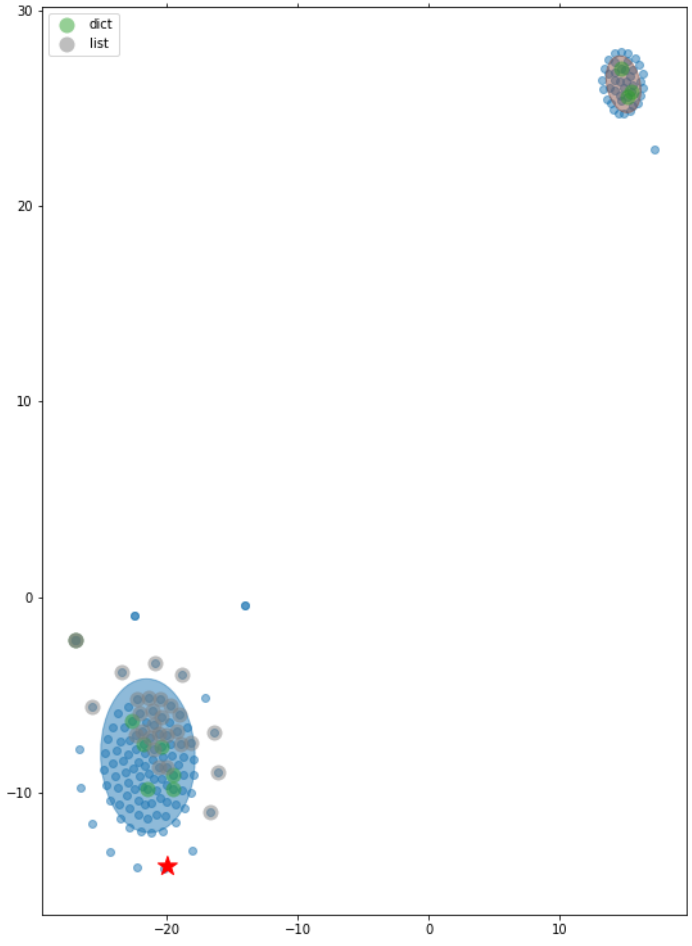

Figure 6: TSNE 2d feature space

6.2 Problem and Solution

Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user. Hint: how does an even / odd number react differently when divided by 2?
Extras:

If the number is a multiple of 4, print out a different message.
Ask the user for two numbers: one number to check (call it num) and one number to divide by (check). If check divides evenly into num, tell that to the user. If not, print a different appropriate message.

```
num = int(input("give me a number to check: "))
check = int(input("give me a number to divide by: "))

if num % 4 == 0:
    print(num, "is a multiple of 4")
elif num % 2 == 0:
    print(num, "is an even number")
else:
    print(num, "is an odd number")

if num % check == 0:
    print(num, "divides evenly by", check)
else:
    print(num, "does not divide evenly by", check)
```

7 Problem with solution 146

7.1 TSNE representation

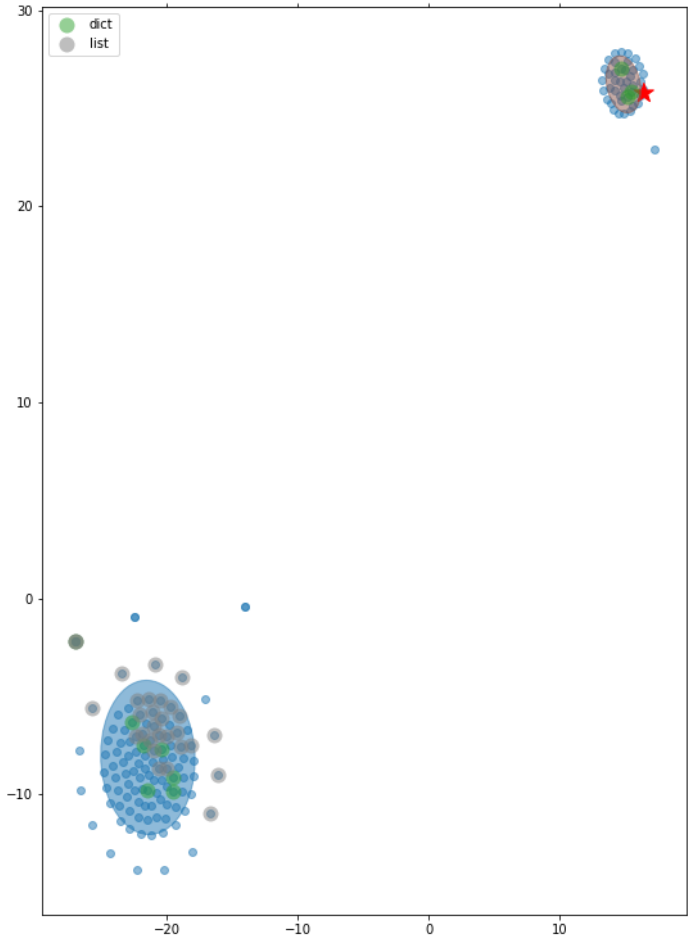

Figure 7: TSNE 2d feature space

7.2 Problem and Solution

Write a program that accepts a sentence and calculate the number of upper case letters and lower case letters.
Suppose the following input is supplied to the program:
Hello world!
Then, the output should be:
UPPER CASE 1
LOWER CASE 9

```
s = raw_input()
d={"UPPER CASE":0, "LOWER CASE":0}
for c in s:
    if c.isupper():
        d["UPPER CASE"]+=1
    elif c.islower():
        d["LOWER CASE"]+=1
    else:
        pass
print "UPPER CASE", d["UPPER CASE"]
print "LOWER CASE", d["LOWER CASE"]
```

8 Problem with solution 110

8.1 TSNE representation

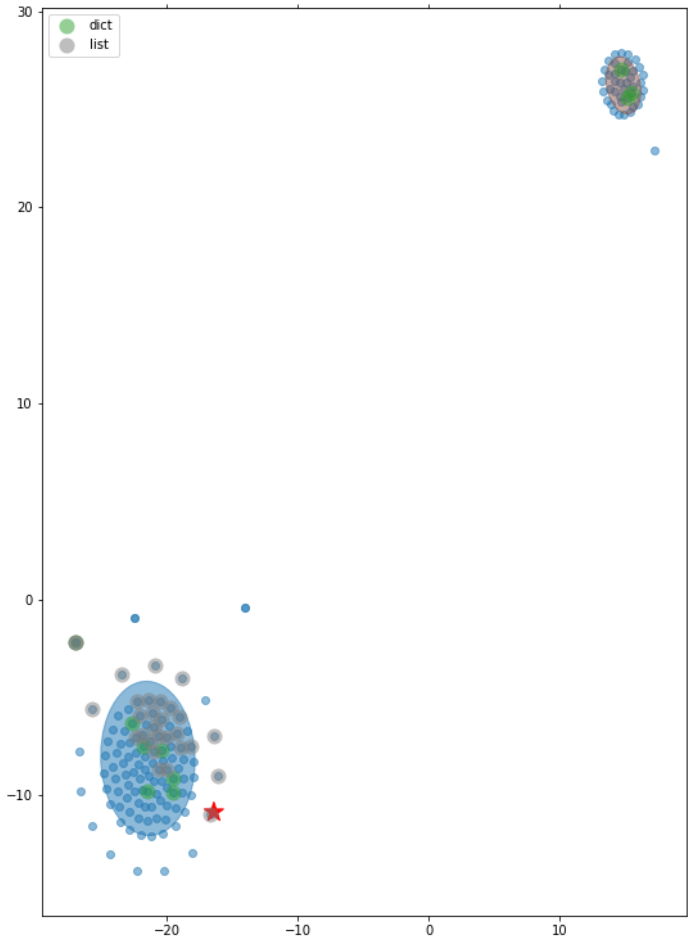

Figure 8: TSNE 2d feature space

8.2 Problem and Solution

Create a program that will keep track of items for a shopping list. The program should keep asking for new items until nothing is entered (no input followed by enter/return key). The program should then display the full shopping list.

```
print("Shopping List")
 print("this program stores your shopping list and then displays it")

 #empty list
 shoppingList = []
 finished = False
 while not finished:
     tempItem = input("Please enter an item for your list: ")
     if len(tempItem) == 0:
         finished = True
     else:
         shoppingList.append(tempItem)

 #display list
 print()
 print("Shopping List")
 print()
 for item in shoppingList:
     print(item)
```

9 Problem with solution 142

9.1 TSNE representation

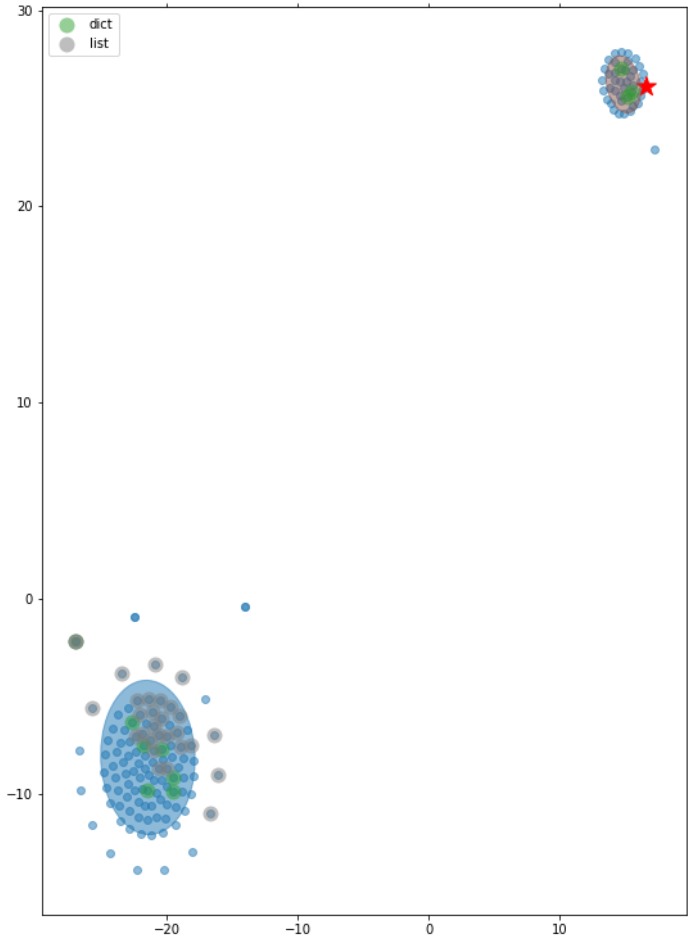

Figure 9: TSNE 2d feature space

9.2 Problem and Solution

Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically.
Suppose the following input is supplied to the program:
hello world and practice makes perfect and hello world again
Then, the output should be:
again and hello makes perfect practice world

```
s = raw_input()
words = [word for word in s.split(" ")]
print " ".join(sorted(list(set(words))))
```

10 Problem with solution 123

10.1 TSNE representation

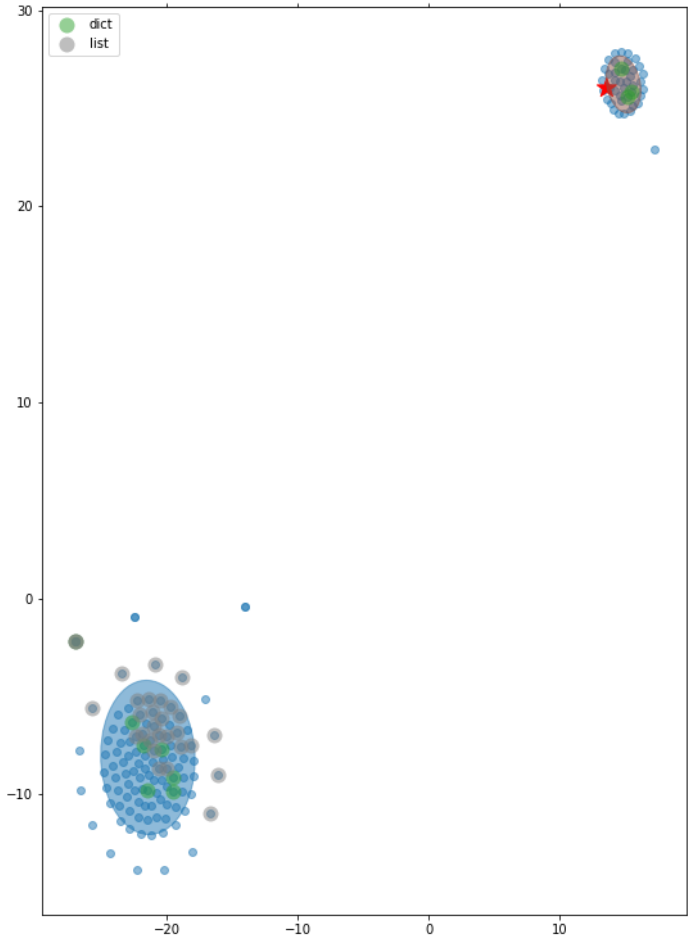

Figure 10: TSNE 2d feature space

10.2 Problem and Solution

1D List Exercise 3 asked you to create a version of Hangman. Refactor this program to use functions.

```
def displayInfo():
    print("Easier Hangman")
    print("This program is a simpler verision of hangman using a list")
    print()

def getWord():
    word = input("Please enter a word for the other player to guess: ")
    return word

def getChances():
    chances = int(input("How many chances should they get to guess the word: "))
    return chances

def constructHangmanList(word):
    #empty list
    hangman = []
    for each in word:
        hangman.append(each)
    return hangman

def constructGuessedCharacterList(hangman):
    #set up list for guesses
    guessedCharacters = []
    for each in hangman:
        guessedCharacters.append("_")
    return guessedCharacters

def setUpGame():
    word = getWord()
    chances = getChances()
    hangman = constructHangmanList(word)
    guessedCharacters = constructGuessedCharacterList(hangman)
    return word,chances,hangman,guessedCharacters


def getGuessCharacter(guessedCharacters,chances):
    print()
    print(guessedCharacters)
    print("you have {0} chances remaining.".format(chances))
    print()
    currentGuess = input("please enter a character to guess: ")
    return currentGuess

def checkGuessCharacter(currentGuess,hangman,guessedCharacters):
    #check guess
    for each in range(len(hangman)):
        if hangman[each] == currentGuess:
            guessedCharacters[each] = currentGuess
def checkWin(guessedCharacters,hangman):
    if guessedCharacters == hangman:
        return True
    else:
        return False

def displayResult(result):
    guessed = result[0]
    word = result[1]
    if guessed:
        print("Well done - you guessed the word!")
    else:
        print("Sorry you didn't guess that the word was {0}.".format(word))

def playGame(setUpValues):
    word = setUpValues[0]
    chances = setUpValues[1]
    hangman = setUpValues[2]
    guessedCharacters = setUpValues[3]
    hangman = constructHangmanList(word)
    guessed = False
    while not guessed and chances > 0:
        currentGuess = getGuessCharacter(guessedCharacters,chances)
        checkGuessCharacter(currentGuess,hangman,guessedCharacters)
        chances = chances - 1
        guessed = checkWin(guessedCharacters,hangman)
    return guessed,word

def hangman():
    setUpValues = setUpGame()
    result = playGame(setUpValues)
    displayResult(result)
```

11 Problem with solution 40

11.1 TSNE representation

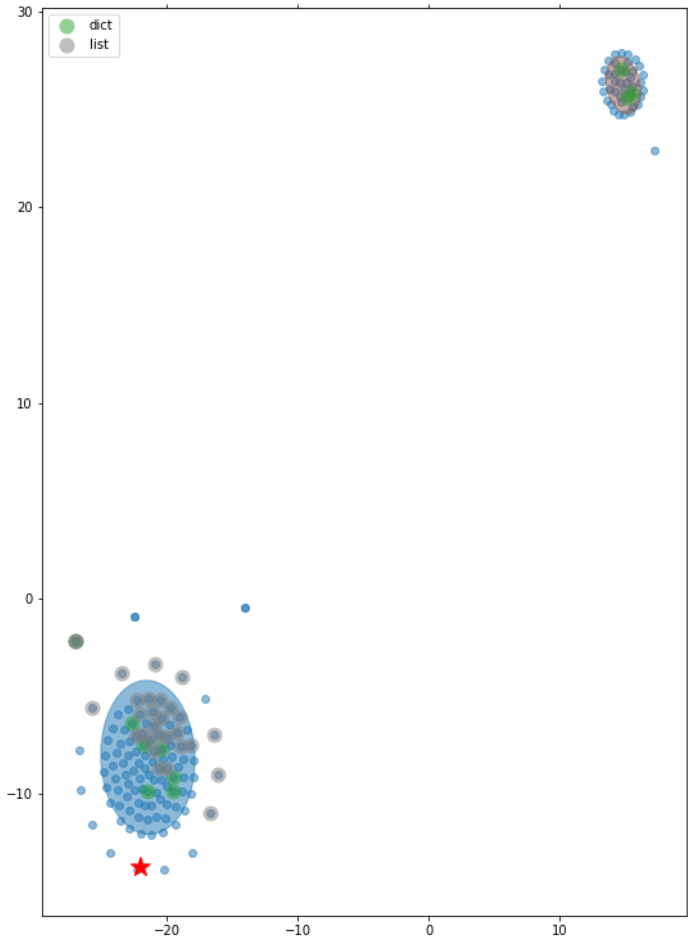

Figure 11: TSNE 2d feature space

11.2 Problem and Solution

Given a .txt file that has a list of a bunch of names, count how many of each name there are in the file, and print out the results to the screen. I have a .txt file for you, if you want to use it!
Extra:

Instead of using the .txt file from above (or instead of, if you want the challenge), take this .txt file, and count how many of each "category" of each image there are. This text file is actually a list of files corresponding to the SUN database scene recognition database, and lists the file directory hierarchy for the images. Once you take a look at the first line or two of the file, it will be clear which part represents the scene category. To do this, you're going to have to remember a bit about string parsing in Python 3. I talked a little bit about it in this post.

```
counter_dict = {}
with open('Training_01.txt') as f:
line = f.readline()
while line:
line = line[3:-26]
if line in counter_dict:
counter_dict[line] += 1
else:
counter_dict[line] = 1
line = f.readline()

print(counter_dict)
```

12 Problem with solution 119

12.1 TSNE representation

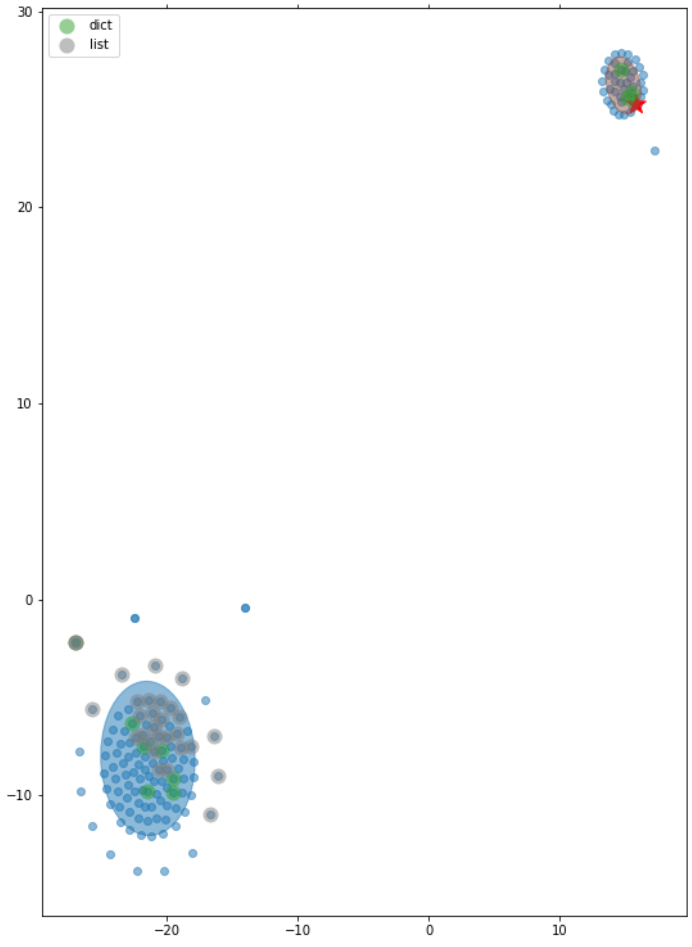

Figure 12: TSNE 2d feature space

12.2 Problem and Solution

Basic Exercise 2 asked you to create a program which will produce times tables. Refactor this program to use functions.

```
def displayInfo():
     print("Times Tables")
     print("This program asks the user for a number and then prints out")
     print("the times table for that number")
     print()

def getTableNumber():
     number = int(input("Please enter a number: "))
     return number

def createTable(number):
     table = []
     for eachNumber in range(1,13):
         entry = (number, eachNumber,eachNumber*number)
         table.append(entry)
     return table

def displayTable(table):
     for entry in table:
         print("{0} x {1:>2} = {2:>3}".format(entry[0],entry[1],entry[2]))


def timesTable():
     displayInfo()
     number = getTableNumber()
     table = createTable(number)
     displayTable(table)
```

13 Problem with solution 48

13.1 TSNE representation

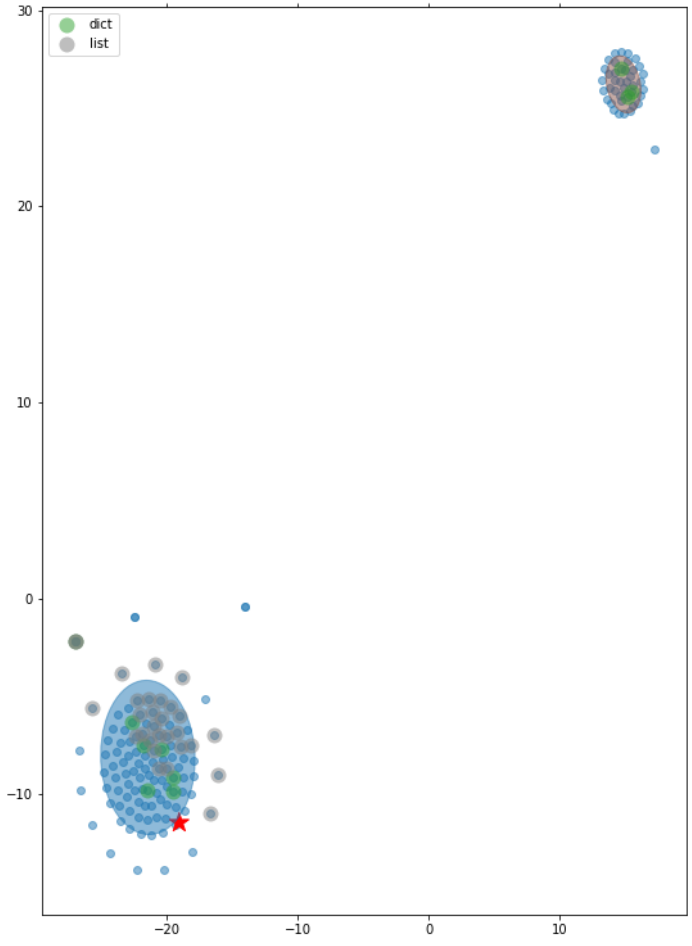

Figure 13: TSNE 2d feature space

13.2 Problem and Solution

This exercise is Part 2 of 4 of the Tic Tac Toe exercise series. The other exercises are: Part 1, Part 3, and Part 4.
As you may have guessed, we are trying to build up to a full tic-tac-toe board. However, this is significantly more than half an hour of coding, so we're doing it in pieces.
Today, we will simply focus on checking whether someone has WON a game of Tic Tac Toe, not worrying about how the moves were made.
If a game of Tic Tac Toe is represented as a list of lists, like so:
game = [[1, 2, 0],
[2, 1, 0],
[2, 1, 1]]
where a 0 means an empty square, a 1 means that player 1 put their token in that space, and a 2 means that player 2 put their token in that space.
Your task this week: given a 3 by 3 list of lists that represents a Tic Tac Toe game board, tell me whether anyone has won, and tell me which player won, if any. A Tic Tac Toe win is 3 in a row - either in a row, a column, or a diagonal. Don't worry about the case where TWO people have won - assume that in every board there will only be one winner.
Here are some more examples to work with:
winner_is_2 = [[2, 2, 0],
[2, 1, 0],
[2, 1, 1]]

winner_is_1 = [[1, 2, 0],
[2, 1, 0],
[2, 1, 1]]

winner_is_also_1 = [[0, 1, 0],
[2, 1, 0],
[2, 1, 1]]

no_winner = [[1, 2, 0],
[2, 1, 0],
[2, 1, 2]]

also_no_winner = [[1, 2, 0],
[2, 1, 0],
[2, 1, 0]]

```
def checkGrid(grid):
# rows
for x in range(0,3):
row = set([grid[x][0],grid[x][1],grid[x][2]])
if len(row) == 1 and grid[x][0] != 0:
return grid[x][0]

# columns
for x in range(0,3):
column = set([grid[0][x],grid[1][x],grid[2][x]])
if len(column) == 1 and grid[0][x] != 0:
return grid[0][x]

# diagonals
diag1 = set([grid[0][0],grid[1][1],grid[2][2]])
diag2 = set([grid[0][2],grid[1][1],grid[2][0]])
if len(diag1) == 1 or len(diag2) == 1 and grid[1][1] != 0:
return grid[1][1]

return 0

winner_is_2 = [[2, 2, 0],
[2, 1, 0],
[2, 1, 1]]
winner_is_1 = [[1, 2, 0],
[2, 1, 0],
[2, 1, 1]]

winner_is_also_1 = [[0, 1, 0],
[2, 1, 0],
[2, 1, 1]]

no_winner = [[1, 2, 0],
[2, 1, 0],
[2, 1, 2]]

also_no_winner = [[1, 2, 0],
[2, 1, 0],
[2, 1, 0]]

print(checkGrid(also_no_winner))
```

14 Problem with solution 122

14.1 TSNE representation

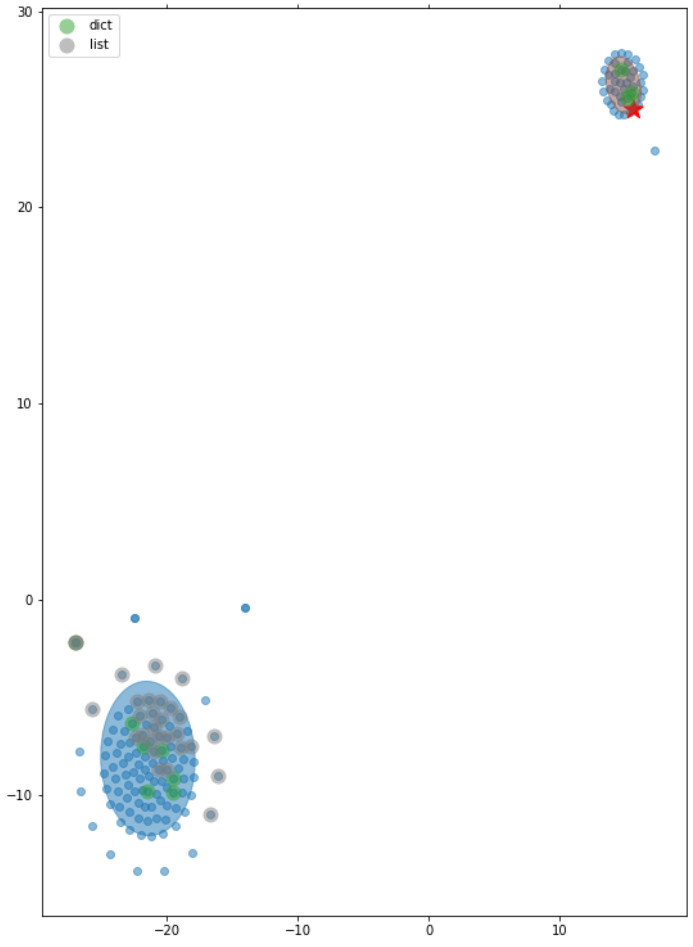

Figure 14: TSNE 2d feature space

14.2 Problem and Solution

1D List Exercise 1 asked to keep track of a shopping list. Refactor this program to use functions.

```
def displayInfo():
     print("Shopping List")
     print("this program stores your shopping list and then displays it")

def getNewItem():
     tempItem = input("Please enter an item for your list: ")
     return tempItem


def addToList(shoppingList,item):
     shoppingList.append(item)
     #notice that there is no return value here for the list...

def displayList(shoppingList):
     #display list
     print()
     print("Shopping List")
     print()
     for item in shoppingList:
         print(item)


def shoppingList():
     #empty list
     shoppingList = []
     finished = False
     while not finished:
         newItem = getNewItem()
         if len(newItem) == 0:
             finished = True
         else:
             addToList(shoppingList,newItem)
     displayList(shoppingList)
```

15 Problem with solution 128

15.1 TSNE representation

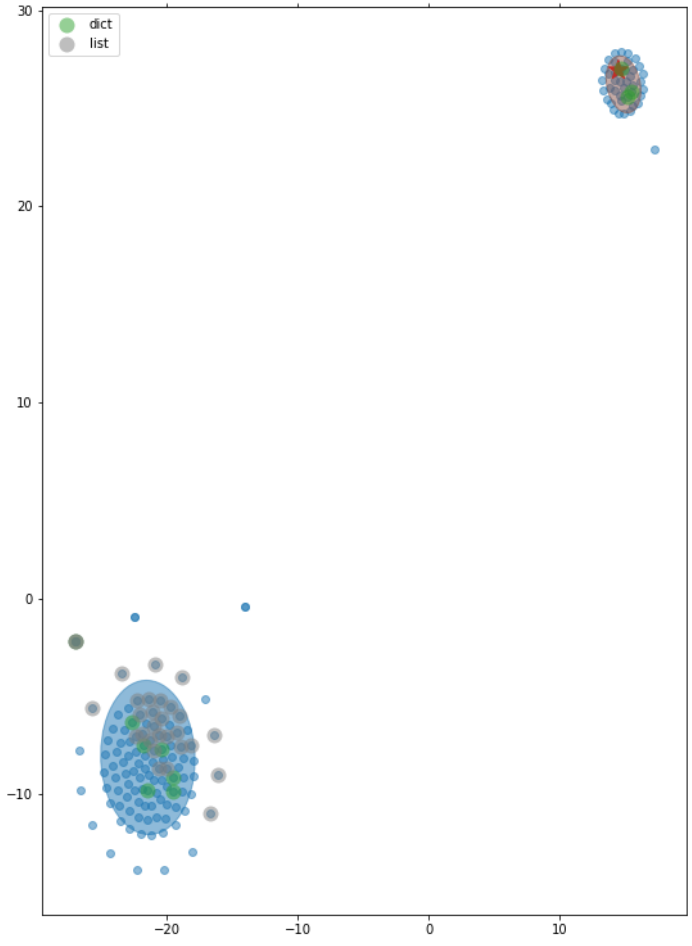

Figure 15: TSNE 2d feature space

15.2 Problem and Solution

Write a shopping list in Notepad and save it as 'shopping.txt'. Write a program to read in the shopping list from the file and display it on the screen.

```
with open("shopping.txt", mode="r",encoding="utf-8") as my_file:
     shopping_list = my_file.read().splitlines()
print("My Shopping List")
print()
for count,item in enumerate(shopping_list):
     print("{0}. {1}".format(count+1,item))
```

16 Problem with solution 46

16.1 TSNE representation

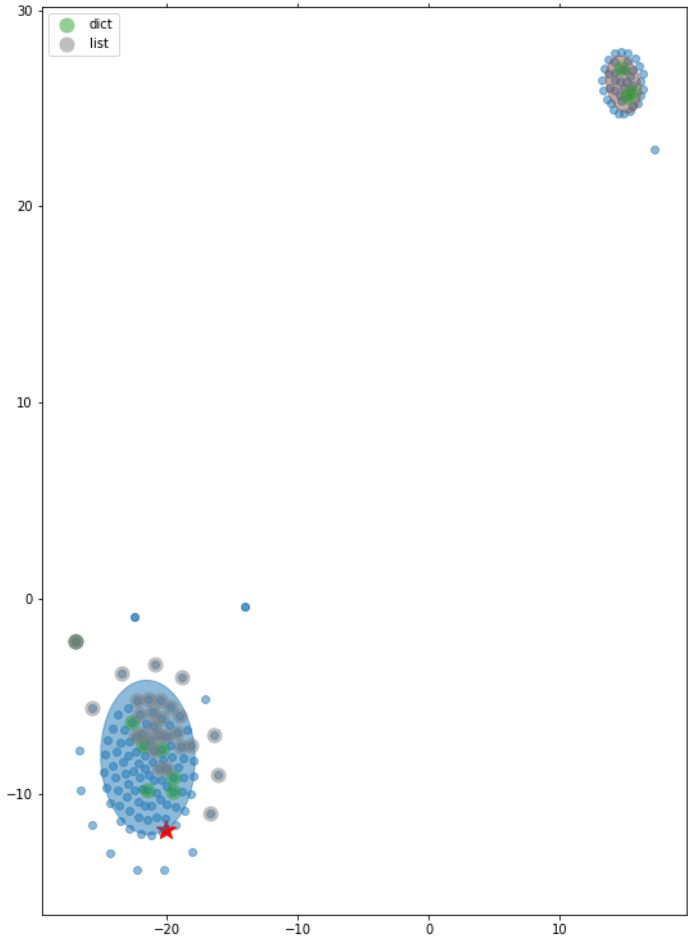

Figure 16: TSNE 2d feature space

16.2 Problem and Solution

In a previous exercise, we've written a program that "knows" a number and asks a user to guess it.
This time, we're going to do exactly the opposite. You, the user, will have in your head a number between 0 and 100. The program will guess a number, and you, the user, will say whether it is too high, too low, or your number.
At the end of this exchange, your program should print out how many guesses it took to get your number.
As the writer of this program, you will have to choose how your program will strategically guess. A naive strategy can be to simply start the guessing at 1, and keep going (2, 3, 4, etc.) until you hit the number. But that's not an optimal guessing strategy. An alternate strategy might be to guess 50 (right in the middle of the range), and then increase / decrease by 1 as needed. After you've written the program, try to find the optimal strategy! (We'll talk about what is the optimal one next week with the solution.)

```
'''
In a previous exercise, we've written a program that 'knows' a number and asks a user to guess it.
This time, we're going to do exactly the opposite.
You, the user, will have in your head a number between 0 and 100.
The program will guess a number, and you, the user, will say whether it is too high, too low, or your number.

At the end of this exchange, your program should print out how many guesses it took to get your number.

As the writer of this program, you will have to choose how your program will arrive at the right guess.
A naive strategy can be to simply start the guessing at 1, and keep going (2, 3, 4, etc.) until you hit the number.
But that's not an optimal guessing strategy.
An alternate strategy might be to guess 50 (right in the middle of the range), and then increase / decrease by 1 as needed.
After you've written the program, try to find the optimal strategy! (We'll talk about what is the optimal one next week with the solution.)
'''

import random

# Awroken

MINIMUM = 0
MAXIMUM = 100
NUMBER = random.randint(MINIMUM, MAXIMUM)
TRY = 0
RUNNING = True
ANSWER = None

while RUNNING:
    print "Is it %s?" % str(NUMBER)
    ANSWER = raw_input()
    if "no" in ANSWER.lower() and "lower" in ANSWER.lower():
        NUMBER -= random.randint(1, 4)
    elif "no" in ANSWER.lower() and "higher" in ANSWER.lower():
        NUMBER += random.randint(1, 4)
    elif ANSWER.lower() == "no":
        print "Higher or lower?"
        ANSWER = raw_input()
        if ANSWER.lower() == "higher":
            NUMBER += random.randint(1, 4)
        elif ANSWER.lower() == "lower":
            NUMBER -= random.randint(1, 4)
    elif ANSWER.lower() == "yes":
        if TRY < 2:
            print "Yes! It only took me %s try!" % str(TRY)
        elif TRY < 10:
            print "Pretty well for a robot, %s tries." % str(TRY)
        else:
            print "That took a while, %s tries!" % str(TRY)
        RUNNING = False
    TRY += 1
print "Thanks for the game!"
```

17 Problem with solution 154

17.1 TSNE representation

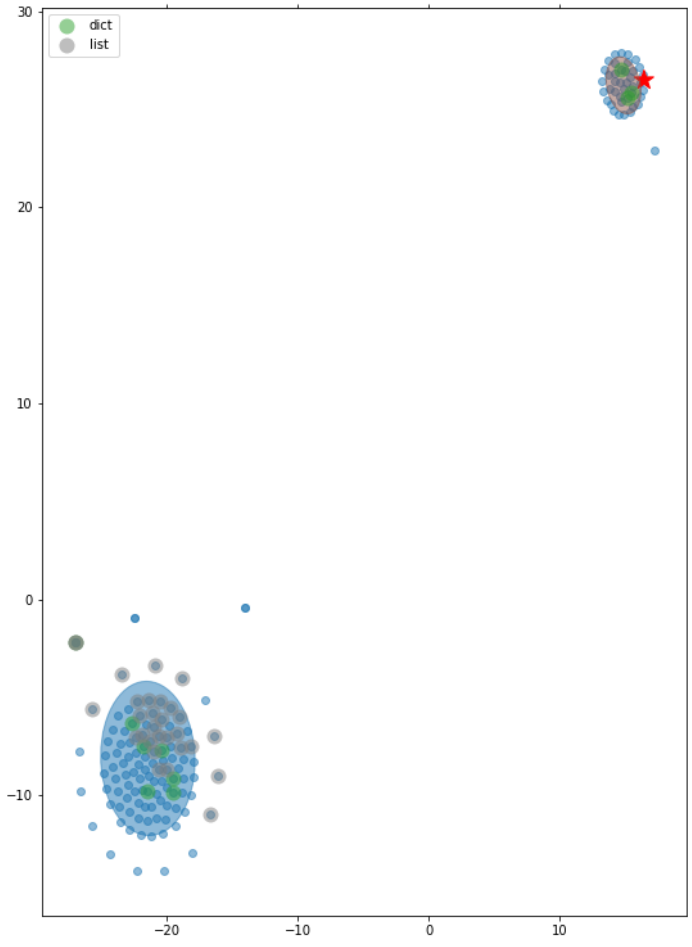

Figure 17: TSNE 2d feature space

17.2 Problem and Solution

Write a program to compute the frequency of the words from the input. The output should output after sorting the key alphanumerically.
Suppose the following input is supplied to the program:
New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3.
Then, the output should be:
2:2
3.:1
3?:1
New:1
Python:5
Read:1
and:1
between:1
choosing:1
or:2
to:1

```
freq = {}   # frequency of words in text
line = raw_input()
for word in line.split():
    freq[word] = freq.get(word,0)+1

words = freq.keys()
words.sort()

for w in words:
    print "%s:%d" % (w,freq[w])
```

18 Problem with solution 64

18.1 TSNE representation

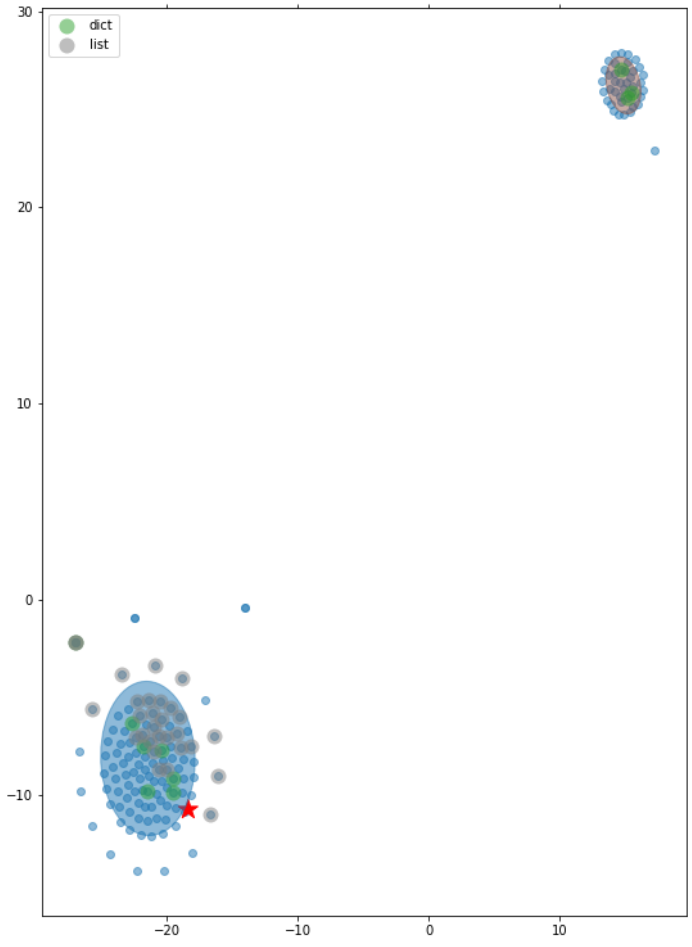

Figure 18: TSNE 2d feature space

18.2 Problem and Solution

This exercise is Part 3 of 4 of the birthday data exercise series. The other exercises are: Part 1, Part 2, and Part 4.
In the previous exercise we saved information about famous scientists' names and birthdays to disk. In this exercise, load that JSON file from disk, extract the months of all the birthdays, and count how many scientists have a birthday in each month.
Your program should output something like:
{
"May": 3,
"November": 2,
"December": 1
}

```
import json
from collections import Counter

month = []

with open("info.json", "r") as f:
    birthdays = json.load(f)

for x in birthdays.values():
    month.append(x.split()[0])

print(Counter(month))
```

19 Problem with solution 145

19.1 TSNE representation

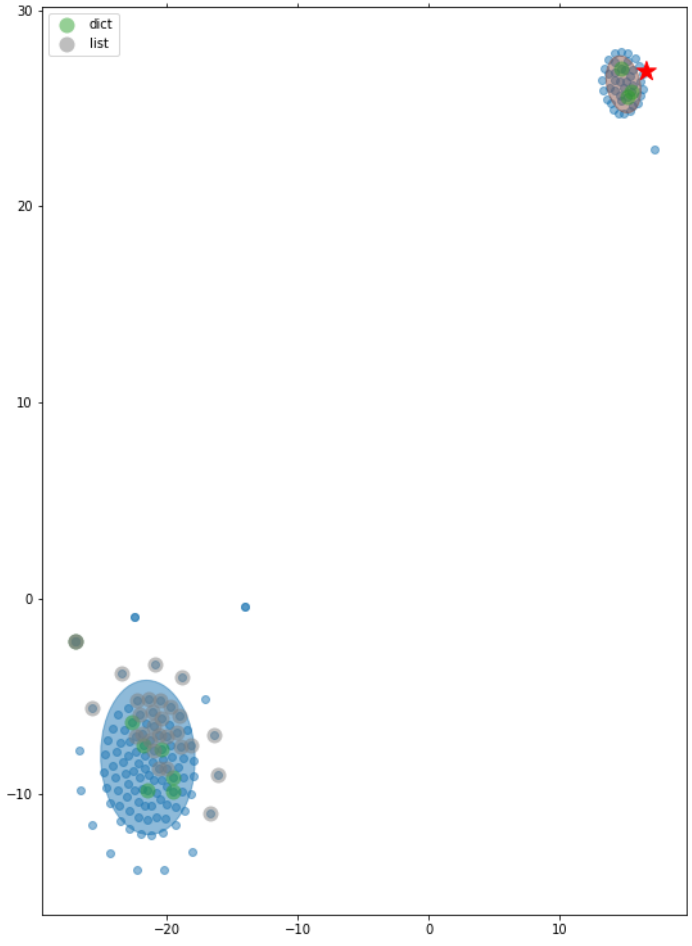

Figure 19: TSNE 2d feature space

19.2 Problem and Solution

Write a program that accepts a sentence and calculate the number of letters and digits.
Suppose the following input is supplied to the program:
hello world! 123
Then, the output should be:
LETTERS 10
DIGITS 3

```
s = raw_input()
d={"DIGITS":0, "LETTERS":0}
for c in s:
    if c.isdigit():
        d["DIGITS"]+=1
    elif c.isalpha():
        d["LETTERS"]+=1
    else:
        pass
print "LETTERS", d["LETTERS"]
print "DIGITS", d["DIGITS"]
```

20 Problem with solution 35

20.1 TSNE representation

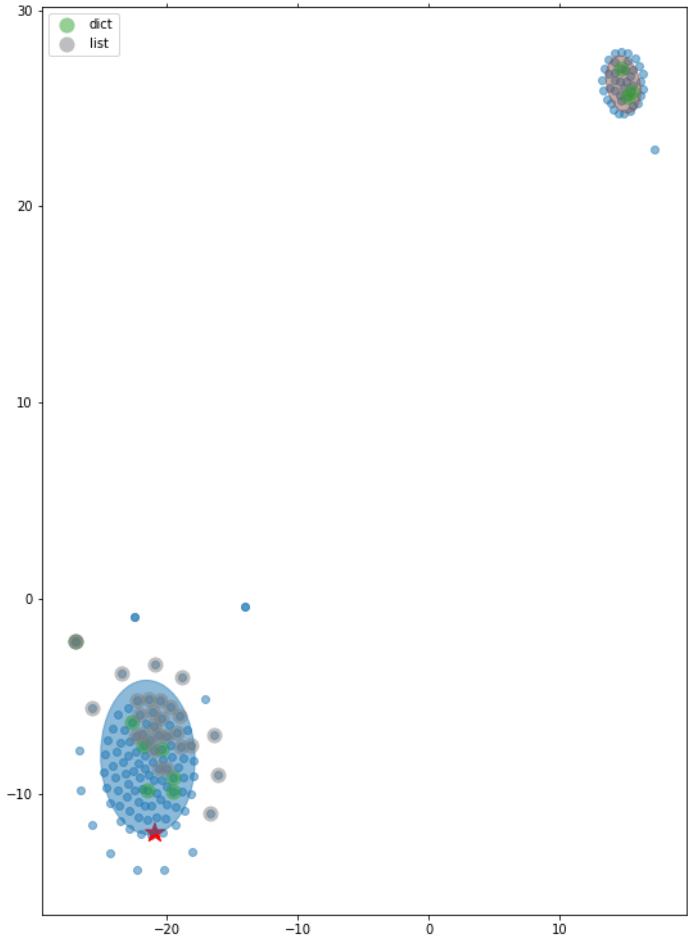

Figure 20: TSNE 2d feature space

20.2 Problem and Solution

Using the requests and BeautifulSoup Python libraries, print to the screen the full text of the article on this website: http://www.vanityfair.com/society/2014/06/monica-lewinsky-humiliation-culture.
The article is long, so it is split up between 4 pages. Your task is to print out the text to the screen so that you can read the full article without having to click any buttons.
(Hint: The post here describes in detail how to use the BeautifulSoup and requests libraries through the solution of the exercise posted here.)
This will just print the full text of the article to the screen. It will not make it easy to read, so next exercise we will learn how to write this text to a .txt file.

```
"""
Using the requests and BeautifulSoup Python libraries, print to the screen the full
article on this website: http://www.vanityfair.com/society/2014/06/monica-lewinsky-h
http://www.practicepython.org/exercise/2014/07/14/19-decode-a-web-page-two.html
"""

import requests
from bs4 import BeautifulSoup

def print_to_text(base_url):
    """
    :param base_url: URL of article to scrape
    :return: naked content to text file
    """
    r = requests.get(base_url)
    soup = BeautifulSoup(r.text)
    with open("work less.txt", "w") as textfile:
        for paragraph in soup.find_all(dir="ltr"):
            textfile.write(paragraph.text.replace("<span>",""))

if __name__ == "__main__":
    #Chose my own article
    base_url = "http://www.theatlantic.com/business/archive/2014/08/to-work-better-w
    base_url2 = "http://www.theatlantic.com/business/archive/2014/08/to-work-better-
    print_to_text(base_url)
    print_to_text(base_url2)
```

21 Problem with solution 136

21.1 TSNE representation

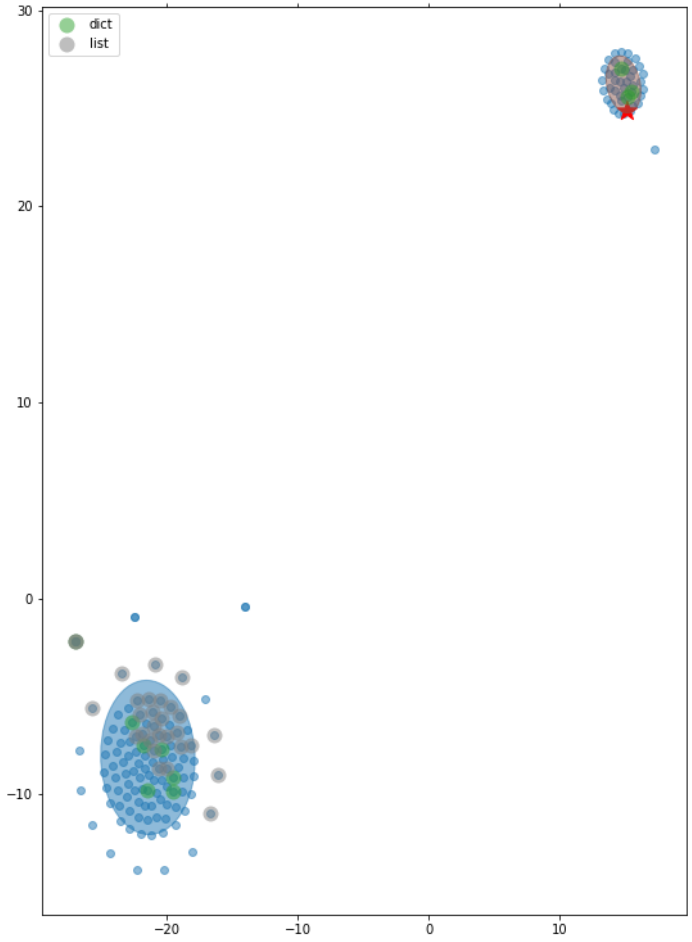

Figure 21: TSNE 2d feature space

21.2 Problem and Solution

Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number.
Suppose the following input is supplied to the program:
34,67,55,33,12,98
Then, the output should be:
['34', '67', '55', '33', '12', '98']
('34', '67', '55', '33', '12', '98')

```
values=raw_input()
l=values.split(",")
t=tuple(l)
print l
print t
```

22 Problem with solution 149

22.1 TSNE representation

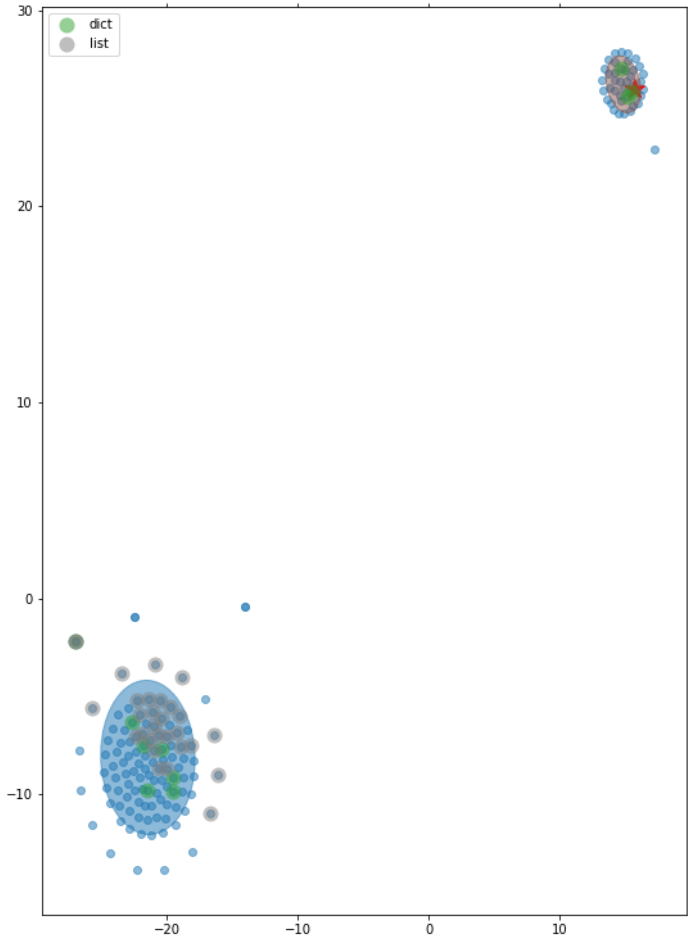

Figure 22: TSNE 2d feature space

22.2 Problem and Solution

Write a program that computes the net amount of a bank account based a transaction log from console input. The transaction log format is shown as following:
D 100
W 200
¡
D means deposit while W means withdrawal.
Suppose the following input is supplied to the program:
D 300
D 300
W 200
D 100
Then, the output should be:
500

```
import sys
netAmount = 0
while True:
    s = raw_input()
    if not s:
        break
    values = s.split(" ")
    operation = values[0]
    amount = int(values[1])
    if operation=="D":
        netAmount+=amount
    elif operation=="W":
        netAmount-=amount
    else:
        pass
print netAmount
```

23 Problem with solution 19

23.1 TSNE representation

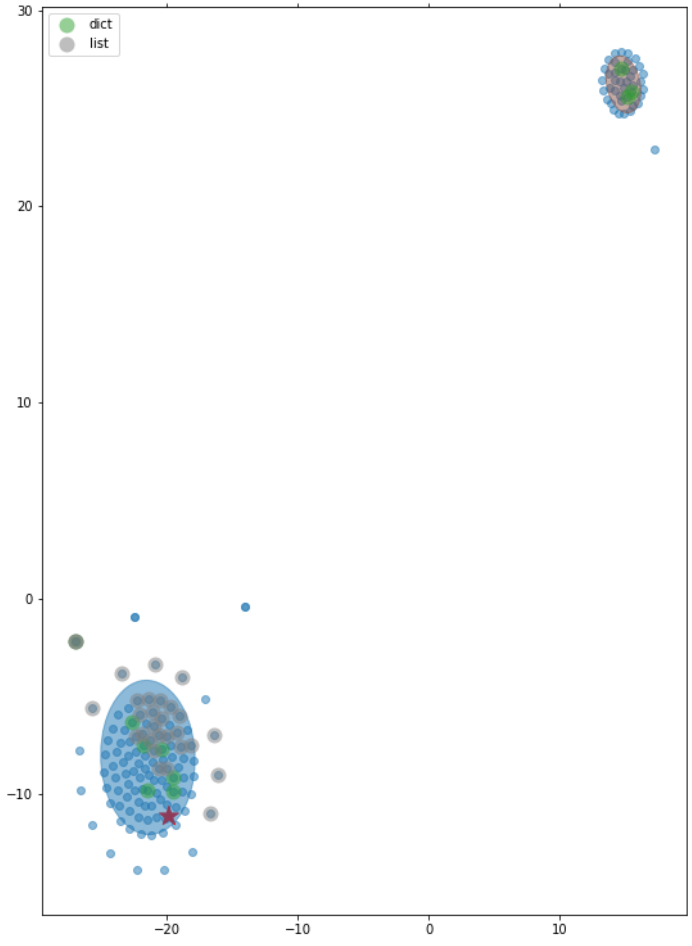

Figure 23: TSNE 2d feature space

23.2 Problem and Solution

Ask the user for a number and determine whether the number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.). You can (and should!) use your answer to Exercise 4 to help you. Take this opportunity to practice using functions, described below.

```
# Assumes that "a" contains an integer > 2 whose primality needs to be verified
# The algorithm builds a list of factors including the number 2 and all odd numbers
# up to the square root of "a", and then checks if any of those numbers divides "a"
# without a remainder - if so then "a" is not prime, else it is
if sum([ True if a%factor == 0 else False for factor in ( [2] + list(range(3,int(mat
  print("Number is composite")
else:
  print("Number is prime")
```

24 Problem with solution 126

24.1 TSNE representation

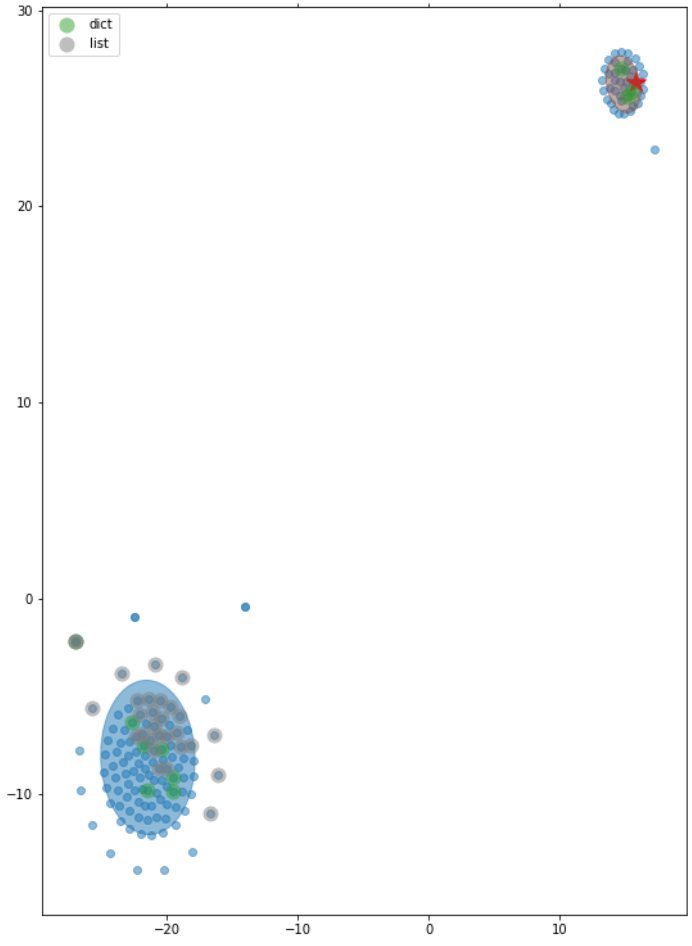

Figure 24: TSNE 2d feature space

24.2 Problem and Solution

Very Challenging Exercise 1 asked you to create a version of Connect 4. Refactor this program to use functions.

```
import random

 def displayInfo():
     print("Connect 4")
     print("This game is Connect 4 for two players")
     print()

 def createBoard():
     #create the board
     #connect 4 board is 7 columns of 6
     board = []
     for column in range(7):
         board.append([])
         for row in range(6):
             board[column].append("O")
     return board

 def getPlayerNames():
     #set-up game
     players = []
     for player in range(2):
         tempPlayer = input("Please enter your name: ")
         players.append(tempPlayer)
     return players

 def getStartPlayer():
     #decide who plays first
     if (random.randint(1,10) % 2) == 0:
         startPlayer = 0
     else:
         startPlayer = 1
     return startPlayer

 def getStartPlayerToken(startPlayer):
     token = input("Whould you like to play as (R)ed or (Y)ellow: ")
     if token == "R" and startPlayer == 0:
         tokens = ["R","Y"]
     elif token == "R" and startPlayer == 1:
         tokens = ["Y","R"]
     elif token == "Y" and startPlayer == 0:
         tokens = ["Y","R"]
     else:
         tokens = ["R","Y"]
     return tokens

 def orderPlayers():
     players = getPlayerNames()
     startPlayer = getStartPlayer()
     print()
     print("{0} you get to start!".format(players[startPlayer]))
     tokens = getStartPlayerToken(startPlayer)
     return players,startPlayer,tokens

 def setUpGame():
     board = createBoard()
     playerSettings = orderPlayers()
     return board, playerSettings

 def displayCurrentBoard(board):
     print()
     print("The board currently looks like this: ")
     print()
     for row in range(6):
         for column in range(7):
             if column < 6:
                 print("{0} ".format(board[column][row]),end=" ")
             else:
                 print("{0}".format(board[column][row]))
     print()
     print("A  B  C  D  E  F  G")
     print()

 def getTurn(players,startPlayer):
     #get go
     print("It is {0}'s turn...".format(players[startPlayer]))
     shot = input("Which column do you want to place a counter in: ")return shot

 def playCounter(shot,board,tokens,startPlayer):
     print("tokens {0}".format(tokens))
     print("startPlayer {0}".format(startPlayer))
     #play counter
     if shot == "A":
         played = False
```

```
    counter = 5
    while not played and counter >= 0:
        if board[0][counter] == "O":
            board[0][counter] = tokens[startPlayer]
            played = True
        else:
            counter = counter - 1
elif shot == "B":
    played = False
    counter = 5
    while not played and counter >= 0:
        if board[1][counter] == "O":
            board[1][counter] = tokens[startPlayer]
            played = True
        else:
            counter = counter - 1
elif shot == "C":
    played = False
    counter = 5
    while not played and counter >= 0:
        if board[2][counter] == "O":
            board[2][counter] = tokens[startPlayer]
            played = True
        else:
            counter = counter - 1
elif shot == "D":
    played = False
    counter = 5
    while not played and counter >= 0:
        if board[3][counter] == "O":
            board[3][counter] = tokens[startPlayer]
            played = True
        else:
            counter = counter - 1
elif shot == "E":
    played = False
    counter = 5
    while not played and counter >= 0:
        if board[4][counter] == "O":
            board[4][counter] = tokens[startPlayer]
            played = True
        else:
            counter = counter - 1
    elif shot == "F":
        played = False
        counter = 5
        while not played and counter >= 0:
            if board[5][counter] == "O":
                board[5][counter] = tokens[startPlayer]
                played = True
            else:
                counter = counter - 1
    elif shot == "G":
        played = False
        counter = 5
        while not played and counter >= 0:
            if board[6][counter] == "O":
                board[6][counter] = tokens[startPlayer]
                played = True
            else:
                counter = counter - 1

def checkVerticalWin(board,startPlayer,tokens):
    gameWon = False
    #check to see if currentplayer has won vertically
    connect = 0
    for each in board:
        for item in each:
            if item == tokens[startPlayer]:
                connect = connect + 1
            else:
                connect = 0
            if connect == 4:
                gameWon = True
    return gameWon

def checkHorizontalWin(board,startPlayer,tokens):
    gameWon = False
    #check to see if currentplayer has won horizontally
    connect = 0
    for row in range(6):
        for column in range(7):
            if board[column][row] == tokens[startPlayer]:
                connect = connect + 1
```

```
            else:
                connect = 0
            if connect == 4:
                gameWon = True
    return gameWon

def checkForWin(board,startPlayer,tokens):
    win = checkVerticalWin(board,startPlayer,tokens)
    if win == False:
        win = checkHorizontalWin(board,startPlayer,tokens)
    return win

def setNextPlayer(startPlayer):
    #switch to next player
    if startPlayer == 0:
        startPlayer = 1
    else:
        startPlayer = 0
    return startPlayer

def playGame(board,playerSettings):
    players = playerSettings[0]
    startPlayer = playerSettings[1]
    print(startPlayer)
    tokens = playerSettings[2]
    gameWon = False
    while not gameWon:
        displayCurrentBoard(board)
        shot = getTurn(players,startPlayer)
        playCounter(shot,board,tokens,startPlayer)
        gameWon = checkForWin(board,startPlayer,tokens)
        if not gameWon:
            startPlayer = setNextPlayer(startPlayer)
    return board,startPlayer,players

def displayWinDetails(gameStats):
    board = gameStats[0]
    startPlayer = gameStats[1]
    players = gameStats[2]
    displayCurrentBoard(board)
    print()
    print("{0} you connected 4 - well done!".format(players[startPlayer]))
```

```
def connectFour():
    setup = setUpGame()
    gameStats = playGame(setup[0],setup[1])
    displayWinDetails(gameStats)
```

25 Problem with solution 109

25.1 TSNE representation

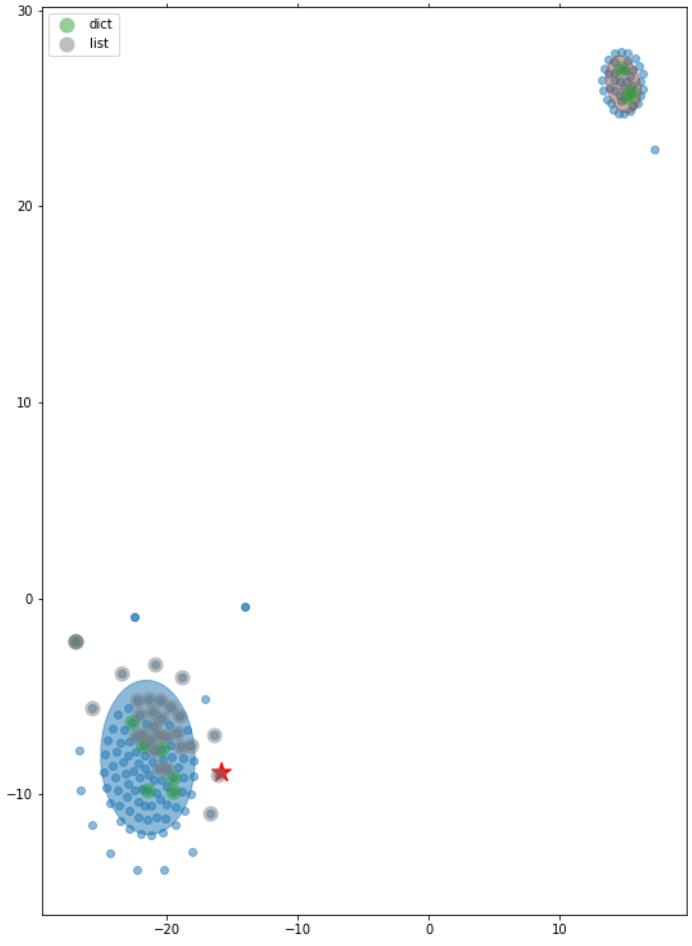

Figure 25: TSNE 2d feature space

25.2 Problem and Solution

Improve the previous exercise so that it is a complete game of hangman for two players which gives a set number of guesses to the user and displays an appropriate message for the winner.

```
import random

print("Hangman")
print("This program plays hangman between two players")
print()
player1 = input("Please enter the name of player one: ")
player2 = input("Please enter the name of player two: ")
#decide which player will be guessing
startPlayer = random.randint(1,2)
if startPlayer == 1:
    print()
    print("{0} you get to set the word to guess.".format(player1))
    print()
    splayer = player1
    gplayer = player2
else:
    print()
    print("{0} you get to set the word to guess.".format(player2))
    print()
    splayer = player2
    gplayer = player1

#get the word from the user
targetStr = input("{0}, please enter a string: ".format(splayer))
guesses = int(input("{0}, how many guesses should {1} get?: ".format(splayer,gplayer)))
print()
print()
character = None
#create a blank string for output
outputStr = ""
#counter for guesses
noOfGuesses = 0
wordFound = False
while noOfGuesses < guesses and wordFound == False:
    noOfGuesses = noOfGuesses + 1
    character = input("{0}, please enter a character to find in the string: ".format(gplayer))
    found = False
    for eachChar in range(len(targetStr)):
        if targetStr[eachChar] == character:
            if outputStr == None:
                outputStr = character
                found = True
            elif len(outputStr) == len(targetStr):
                #get the OutputStr up to the character before the character to add
                #add the character to this string and then add the Output string fr
                #the character after the character we add
                outputStr = outputStr[:eachChar] + character + outputStr[eachChar+1
                found = True
            else:
                outputStr = outputStr + character
                found = True
        else:
            if len(outputStr) < len(targetStr):
                outputStr = outputStr + "_"
    if found:
        print()
        print("The letter {0} was in the string".format(character))
        print("the string is now {0}".format(outputStr))
        print()
    else:
        print()
        print("The letter {0} was not in the string".format(character))
        print("the string is still {0}".format(outputStr))
        print()
    print("You have {0} guesses remaining.".format(guesses-noOfGuesses))
    print()
    print("{0}, the word is {1}".format(splayer,outputStr))
    #decide if the word has been found
    if outputStr == targetStr:
        wordFound = True

if wordFound:
    print("Well done {0}, you have guessed that the word is {1}".format(gplayer,tar
    print("You took {0} guesses".format(noOfGuesses))
else:
    print("Sorry {0}, you didn't get the word {1} in {2} guesses".format(gplayer,ta
```