1) def isAlphabeticalWord(word, wordList):

if (len(word) > 0):

curr = word[0]

for letter in word:

if (curr > letter):

return False

else:

curr = letter

if wordList is None:

return True

return word in wordList

This function checks if a word is alphabetical. A word is alphabetical if letters in the word are in alphabetical order, and is a word from the list of valid words.

For example, if wordList is a list of English words, 'box', 'annoy', and 'chips' are alphabetical words, but 'bat' and 'pack' are not because the letters are not in alphabetical order, and 'abcd', 'aqz' are not because they are not words in the wordList.

Without access to a valid wordList, this function is useless. But, it doesn't have to be that way. Your task is to use *default values* in the function definition so that users of that function may call it without providing a wordList. If no wordList is provided, the function should treat all words as valid words, but should still return True or False depending on whether the word's letters are in alphabetical order.

For example, isAlphabeticalWord('abcd') should return True but isAlphabeticalWord('zoo') should return False.

What should the default value for wordList be, so that the function behaves correctly?

def isAlphabeticalWord(word, wordList = <Code Here> )

What should we put as the default value, in the space <Code Here>?

 None - correct

---------------------------------------------------------------------------

2)

def lotsOfParameters1(a,b,c,d,e):

print a

print b

print c

print d

print e

# ===============

# = Question 1A =

# ===============

# 1. lotsOfParameters1() # => Error

# 2. lotsOfParameters1(1, 2) # => Error

# 3. lotsOfParameters1(1,e=5,d=4,c=3,b=2) # => No Error

# 4. lotsOfParameters1(e=5,a=1,d=4,b=2,c=3) # => No Error

# 5. lotsOfParameters1(a=5,b=1,c=4,d=2,3) # => Error

# ===============

# = Question 1B =

# ===============

# 1. lotsOfParameters1() # => Error

# 2. lotsOfParameters1(1, 2) # => Error

# 3. lotsOfParameters1(1,e=5,d=4,c=3,b=2) # => Produces Same Output

# 4. lotsOfParameters1(e=5,d=4,c=3,b=2,1) # => Error

# 5. lotsOfParameters1(e=5,a=1,d=4,b=2,c=3) # => Produces Same Output

def lotsOfParameters2(a=1,b=2,c=3,d=4,e=5):

print a

print b

print c

print d

print e

# ===============

# = Question 2A =

# ===============

# 1. lotsOfParameters2() # => No Error

# 2. lotsOfParameters2(1, 2) # => No Error

# 3. lotsOfParameters2(1, c=2) # => No Error

# 4. lotsOfParameters2(1, c=2, 3) # => Error

# 5. lotsOfParameters2(1, e=20, b=3) # => No Error

# 6. lotsOfParameters2(1, e=20, b=3, a=10) # => Error

# ===============

# = Question 2B =

# ===============

# 1. lotsOfParameters2() # => Produces Same Output

# 2. lotsOfParameters2(1,2,3,4) # => Produces Same Output

# 3. lotsOfParameters2(5,4,3,2,1) # => Produces Different Output

def lotsOfParameters3(a,b,c=3,d=4,e=5):

print a

print b

print c

print d

print e

# ===============

# = Question 3A =

# ===============

# 1. lotsOfParameters3() # => Error

# 2. lotsOfParameters3(1, 2) # => No Error

# 3. lotsOfParameters3(1, c=2) # => Error

# 4. lotsOfParameters3(1, c=2, 3) # => Error

# 5. lotsOfParameters3(1, c=2, b=3) # => No Error

# ===============

# = Question 3B =

# ===============

# 1. lotsOfParameters3() # => Error

# 2. lotsOfParameters3(1,2) # => Produces Same Output

# 3. lotsOfParameters3(1,e=5,d=4,c=3,b=2) # => Produces Same Output

# 4. lotsOfParameters3(e=5,d=4,c=3,b=2,1) # => Error

# 5. lotsOfParameters3(e=5,a=1,d=4,b=2,c=3) # => Produces Same Output

---------------------------------------------------------------------------

import pylab

pylab.figure(1) #create figure 1

pylab.plot([1,2,3,4], [1,2,3,4]) #draw on figure 1

pylab.figure(2) #create figure 2

pylab.plot([1,4,2,3], [5,6,7,8]) #draw on figure 2

pylab.savefig('Figure-Eric') #save figure 2

pylab.figure(1) #go back to working on figure 1

pylab.plot([5,6,10,3]) #draw again on figure 1

pylab.savefig('Figure-Grimson') #save figure 1

# Second Example (comment the above code before uncommiting this example)

##principal = 10000 #initial investment

##interestRate = 0.05

##years = 20

##values = []

##for i in range(years + 1):

## values.append(principal)

## principal += principal\*interestRate

##pylab.plot(range(years+1), values)

##pylab.title('5% Growth, Compounded Annually')

##pylab.xlabel('Years of Compounding')

##pylab.ylabel('Value of Principal ($)')

##pylab.show()