Assignment 8 1 - Program

```
from sys import exit #'exit' causes the program as a whole to stop

#function for encryption/decryption itself; arguments are the mode (either 'enc' or 'dec'), the text to process
def encdec(mode,text,key):
  #print short message indicating operating mode; or, if somehow invalid mode, return here.
  #NOTE: This is *not* proper exception handling! There are formal and more elaborate ways to do it properly!
  if (mode == 'enc'):
    print 'Processing encryption'
  elif (mode == 'dec'):
    print 'Processing decryption'
  else:
    print 'ERROR'
    return None

  outtext = '' #initialise the text
  #change the two 1D, N-length character strings into N 1D 2-length character pairs; iterate through each of these pairs at a time
  for tchar,kchar in zip(text,key):
    t = ord(tchar) #'ord' returns the ASCII code (actually more extensive than ASCII)
    k = ord(kchar)
    if (t != 10): #due to processing of character string in python, the 'newline' character has been read in as any other would be; this tells the program to ignore it completely.
      #two different processes from here, for encryption or decryption
      if (mode == 'enc'):
        o = t + (k-32)
        if (o > 126):
          o -= 95
      else:
        o = t - (k-32)
        if (o < 32):
          o += 95
      ochar = chr(o) #convert back to character from ascii value
      outtext += ochar #extend the existing output string by this character
  return outtext #function passes back completed encrypted/decrypted text

#function for obtaining keyphrase from user
def get_key(rep_len):
  start_key = raw_input('Please enter keyphrase: ')
  end_key = start_key * int((rep_len/len(start_key))+1) #repeat at least as long as necessary
  end_key = end_key[:rep_len] #trim to exactly as long as necessary
  return end_key


###################################
# Non-function code begins here. #
###################################


#list of possible matches for input text
enc_synonyms = ['e','enc','encrypt','encryption']
dec_synonyms = ['d','dec','decrypt','decryption']
quit_synonyms = ['q','quit','exit','leave','get me out of here']

```

```
valid = False

#keep looping until a sensible input is provided
while (not valid):
  select_mode = raw_input('Specify whether to encrypt or decrypt (‘quit‘ to exit): ').lower() #read in operation to perform, the ‘.lower()’ converts to lower case, so we only have to compare with lower case possible matches, rather than all combinations of upper and lower.

  #this works a bit like a ‘SELECT CASE’ statement, it automatically checks all of the elements in, for example, ‘enc_synonyms’ against our variable ‘select_mode’, in one operation.
  if (select_mode in enc_synonyms):
    select_mode = 'enc'
    valid = True
  elif (select_mode in dec_synonyms):
    select_mode = 'dec'
    valid = True
  elif (select_mode in quit_synonyms):
    print 'Quitting process'
    exit()
  else:
    print 'Did not understand input'

#open relevant input and output files depending on mode of operation
if (select_mode == 'enc'):
  fin = open('plaintext.txt','r')
  fout = open('assign_8_encrypt.out','w')
elif (select_mode == 'dec'):
  fin = open('assign_8_encrypt.out','r')
  fout = open('assign_8_decrypt.out','w')


#################################################################
# This bit following is arguably the actual program itself #
#################################################################

intext = fin.read() #get the plaintext/encrypted text
fin.close()
key = get_key(len(intext)) #get the keytext
outtext = encdec(select_mode,intext,key) #find the encrypted text/decrypted text using function

fout.write(outtext) #provide output to file
fout.close()

#################################################################
# Quite short, isn't it? Really, the checks earlier on     #
# should have functions, and be similarly compact in       #
# the main program section.                                #
#################################################################
```

Assignment 8 1 - assign_8_encrypt.out

```
HIbjrM]o*<33#syu4gbg]hTz 4y&:#2q&4;Y‘Zfa[i}u1(w{$=4>XXcr+um2%$t #uu40^rVr8^g"~1tx%|!w2oj^gJna2}r"/x#!"F2XiXNVe($v?/s~u4(Rbjgaco2tzx/"v1u:‘[nkKPt{  3x!0uy,‘rhcCRe2%yt$2Y1’,Q_alafi&x1\6v0}}:dXcXFnt"0){p’0~.F]bi[Ga ’ }w/ u1,/Uat<afa&0+#%!w?6FLOtt9Wy>0){p’0u}+of]Xace~|1-~(O34#Lrv<aSo!7&3z! )@F9rY\F]’’0}|#’u BH
```

Assignment 8 1 - assign_8_decrypt.out

```
1 "You know," said Arthur, "it's at times like this, when I'm trapped in a
     Vogon airlock with a man from Betelgeuse, and about to die of asphyxiation
      in deep space that I really wish I'd listened to what my mother told me
     when I was young." \\ "Why, what did she tell you?" \\ "I don't know, I
     didn't listen."
```