Brute Force Project

Introduction

This brute force program will attempt to break into a user's account by trying multiple passwords over and over and trying all combinations and values until the correct password is inputted.

Basic Task

For the basic task all that was needed was to make the 'generateGuesses' function have to computer generate all possible 1000 guesses for the binary code instead of manually typing out all 1000 numbers. To run the program in the virtual environment you would need to get into the 'Brute-Force' directory and type './src/brute_basic.py'.

```
1.  def generateGuesses():
2.      """ Creates a list of PIN guesses. Needs finishing, testing and documenting...
    """
3.      #Here, the guesses are listed explicitly.  We *could* write out
4.      #1000 guesses, but the point is to use the computer to do the
5.      #brute force work, not our fingers
6.
7.      #For the basic task, the only functionality you need is to replace
8.      #the line below with code that can generate all the potential
9.      #guesses
10.
11.     array=[]
12.     for i in range (0,1000):
13.         numbers = str(i)
14.         fillednum = numbers.zfill(3)
15.         array.append(fillednum)
16.     #this for loop checks each value betwenn 0 and 1000 then the 'str(i)' function
    will turn each of those values into string in order for the rest of the prgram to r
    ead it
17.     #'zfill' puts '00' place holders at the beginning of each number in order for t
    hem all to be the same length and match every binary code.
18.     #each of these values are appended back into the array
19.     return tuple(array)
20.     #tuple makes sure it is listed
```

For the basic task only three-digit numbers in a string format are accepted from the target file nothing else. These three-digit numbers represent pin codes. Once the program is run you are to choose a target file and when the target file is inputted the program will systematically go through every number combination available. Once the program finds a number combination that matches with the target files code it will output it to the user of the program stating the code has been found.

Tests:

Test Number	Test Type	Test Data	Reason	Expected Outcome	Actual Outcome	Pass/Fail?
1	Standard	'0', '1', '2'	Testing if the code does what it is supposed to do; makes sure that it reads the binary code and checks if it matches with any in the list generated.	It should find the correct binary code for each target.	Each targets binary code is correctly found and displayed.	Pass
2	Erroneous	'Hello'	To make sure that the code only accepts the target values and not anything else.	An error message should appear.	Value Error.	Pass

For the basic task, the only algorithm I had to create was an algorithm creating a list of all possible 3-digit pins.

To do this I used a for loop to go through every single number between 0,1000. These numbers would then be filled with "0" 's at the beginning of them until they were 3 characters long with 'Zfill(3)'. This is to make sure they are in pin code format.

Intermediate Task

For the intermediate task I hade to edit the function to make the brute force program check a list of words and other variations of those words. To run the program in the virtual environment you would need to get into the 'Brute-Force' directory and type './src/brute_intermediate.py'.

```
if __name__=="__main__":

    #Load the dictionary
    words=wordsFromFile("dictionaries/base.txt")


    ### YOUR CODE HERE
    ### Currently it passes in the plain words
    ### Change the line "words2=words" so that the list "words2" contains your gues
ses
    ### You need to create a word list that has the dictionary words in PLUS
    ### 1. Each word with all 0-
9 digits appended (so 'swordfish' would be 'swordfish0', 'swordfish1' etc.
    ### 2. Each word turned into "l33t-5p34k"
    ###    Each o becomes 0, each i becomes 1, each a becomes 4, each s becomes 5,
each e becomes 3
    ###    'swordfish' becomes '5w0rdf15h', for example
    ###    You can assume case (upper/lower) will not need to be changed

    words2 = []
    temp=input ("input guess")
    words2.append(temp)
    #this is something that i added which will ask the user to input their own gues
s for the password and it will attempt that along with the rest.

    for w in range (0,len(words)):
       for i in range (0,10):
        word=words[w]
        word=word+str(i)
        words.append(word)
    #the first for loop will go through each word in the dictionary
    #the next imbedded for loop goes through each number between 0-
10 and adds each indiviudual number onto each word in the dictionary and appends th
ose as a new set of words back into he original dictionary.


    for i in range (0,len(words)):
        wordnum = words[i]
        wordnum = wordnum.replace("o","0")
        wordnum = wordnum.replace("i","1")
        wordnum = wordnum.replace("a","4")
        wordnum = wordnum.replace("s","5")
        wordnum = wordnum.replace("e","3")
        words.append(wordnum)

    #this for loop is exaclty the same as the other one but instead of adding numbe
rs at the end this one uses .replace function to replace certain letters with numbe
rs.

    words2 += words
```

Only files in list format are supported as the program compares the target file to a list containing strings. It cannot correctly read integers or other formats. Once the program is run it will ask for a personal guess from the user to add to the dictionary in the list. Then, it will go through each word in the list and check whether it is correct or not. Once the correct password is found it will display a message letting you know. If not it will just show all the passwords with ‘’password incorrect’’ next to them.

There are 2 inputs required from the user for this code hence why in the test data there will be 2 separate inputs for test data.

Test Number	Test Type	Test Data	Reason	Expected Outcome	Actual Outcome	Pass/Fail?
1	Standard	Input guess: Mayyhin Input 2: 0	First input is to make sure the users' input is added to the word dictionary and also checked to see if it matched the target password. The second input is to make sure the brute force program checks the password in target 1 with the ones in the dictionary of the program.	It should add the check the word inputted by the user as incorrect. It should go through all the words and should find the correct password being 'c04t3d'.	'mayyhin' – password incorrect '0' – The guess 'c04t3d' appears to be correct.	Pass
2	Erroneous	Input guess : fjskd5849 Input 2 : hello	First input does not affect the program as it will take it as a string. Second input is to test what would happen if something other than the target file is inputted.	It should just take the input as a string An error message should show up.	Error message saying there is a value error as hello has no value as it is not a target file.	Pass

For the intermediate task I made 3 algorithms. One algorithm for getting a user input of the guess and having that put back in the word dictionary. The second algorithm would take the words from the dictionary and duplicate them and add numbers 0-9 at the end of them and put them back into the dictionary. The final algorithm in this was to create another duplicate of the original words and change them into `l33t-5p34k` and have them put back into 'words'.

Inputting own guess:

For the first algorithm all I had to do was make a variable to take in a user input and then append it back into ‘words’.

Adding numbers to the end of the words:

The second algorithm creates a for loop going through each word in the dictionary 10 times. After each time it goes through the word it adds a number to the end of it corresponding to the iteration of the loop. So if it was the first time it went through that word it would add 0, then 1, then 2 so on and so forth. All these new words are then appended back into the dictionary.

Changing dictionary words into l33t-sp34k:

I created a for loop to go through each word in the dictionary and whenever it iterates through a word it will go through each letter and replace the certain letters with its corresponding number in l33t-sp34k with the function .replace. These newly created words are appended back into words.

https://github.coventry.ac.uk/abdussamam/Brute-Force