**TASK 3**

import random

import time

# Characters to create random passwords

characters = ["a", "c", "e", "h","k","n", "p", "s", "t", "u", "v" ,"0", "1","2", "3" \

"A", "C", "E", "H","K","N", "P", "S", "T", "U", "V" ]

# The password to guess

target\_password = ""

# Set the length of the passwords to generate and guess for bulk runs

password\_length = 6

# Number of passwords to crack

reps = [0, 1]

# Flag to stop guessing when attempt is sucessfull

status = "ongoing"

# Number of guesses for all passwords

all\_guesses = 0

# Indexes for each character in a password up to eight characters

digits = {1:0, 2:0, 3:0, 4:0, 5:0, 6:0, 7:0, 8:0}

# Generate a password guess with values from digits array as index in the characters array

def generate():

global password\_length, digits, status

# Number of characters filled so far

char\_count = 0

current\_guess = ""

# The current key in the digits array

index\_counter = password\_length

# Create the guess according to current digits array

while char\_count < password\_length:

char\_count += 1

current\_guess += characters[digits[char\_count]]

# Incerement the relative key in the digits array for next run

digits\_modded = "no"

while digits\_modded != "yes":

# Increment end character value if not equal to length of character array

if digits[index\_counter] < (len(characters) - 1):

digits[index\_counter] += 1

digits\_modded = "yes"

# Otherwise move focus left if not already on first key in digits array

else:

if index\_counter > 1:

# Reset value fo existing key

digits[index\_counter] = 0

# Move left to next key

index\_counter -= 1

# Increment the value for the new key

if digits[index\_counter] < (len(characters) - 1):

digits[index\_counter] += 1

digits\_modded = "yes"

else:

# All combinations tried, give up

digits\_modded = "yes"

status = "not\_found"

print(status, current\_guess)

return current\_guess

# Get the starting time to compare to end time for bulk runs

total\_time = 0.0

# Until the quota of passwords have been cracked

while reps[0] < reps[1]:

status = "ongoing"

# Reset the digits array to default

for pos in digits:

digits[pos] = 0

# If multiple runs, create random target\_password for this run

if reps[1] > 1:

# Create a string from randomly chosen characters

target\_password = ""

while len(target\_password) < password\_length:

# Append a random index from the characters list to target password

target\_password += random.choice(characters)

reps[0] += 1

# For single runs, prompt for target password each time instead

else:

target\_password = str(input("Enter a password to test\n"))

password\_length = len(target\_password)

# Leaving empty will exit main loop

if target\_password == "":

reps[0] += 1

status = "stopped"

# Get the starting time for this run

this\_time = time.time()

guesses = 0

# Start guessing until correct

while status == "ongoing":

# Generate a new guess

guesses += 1

if generate() == target\_password:

elapsed = time.time() - this\_time

status = "Cracked"

print(status + ": " + target\_password)

print(str(guesses) + " guesses, " + str(elapsed) + " seconds.\n\_\_\_\_\_\_\_\_\_\_\_\n")

all\_guesses += guesses

total\_time += elapsed

# Calculate and display overall stats for bulk runs

if reps[1] > 1:

print(str(all\_guesses / reps[1]) + " guesses per password")

print(str(total\_time / reps[1]) + " seconds per password")

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

this code first of all make wordlist which makes easy for guessing. Then matching one by one wordlist with password . If it found to be exactly match it returns the result password.

Enter a password to test

UETP

Cracked: UETP

369946 guesses, 0.6938815116882324 seconds.

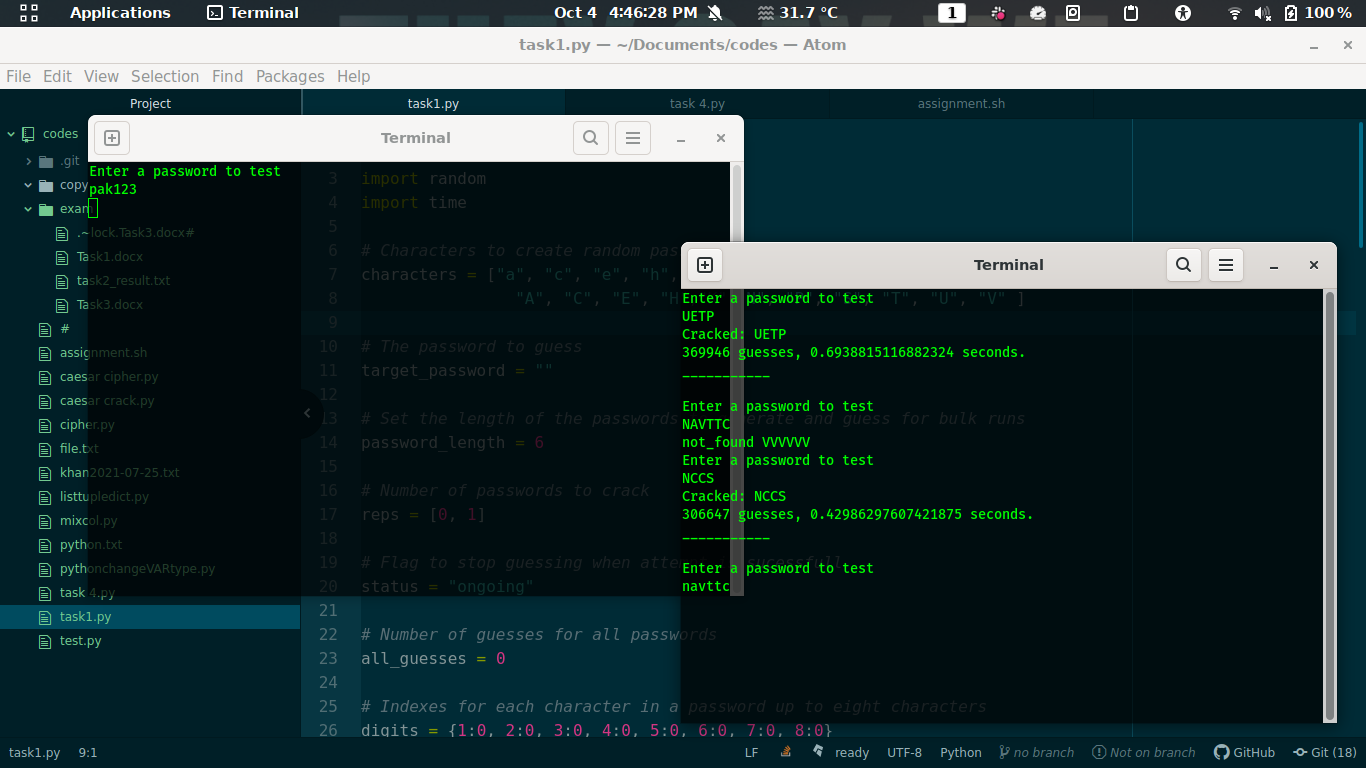
-----

Enter a password to test

NCCS

Cracked: NCCS

306647 guesses, 0.40169310569763184 seconds.

As you can see for six digit password its still bruteforcing maybe take for more than 20 min