TITLE PAGE

TITLE OF REPORT:

RANDOM PASSWORD GENERATOR

NAME:

D.AKHILA

DATE OF SUBMISSION:

31 JULY 2024

TABLE OF CONTENTS

TITLE

1.INTRODUCTION

This report is prepared to give an over view of the project:

RANDOM PASSWORD GENERATOR

The project was developed to provide base concepts in programming and as instances of real applications for learning purposes .The random password generator, generates safe and complex passwords according to the criteria defined by the user.

2.PROJECT OVERVIEW

RANDOM PASSWORD GENERATOR

The Random password Generator will be developed to create a strong password, taking into considerations the requirements given by the user with regard to its length, digits and punctuation marks. It will generate difficult to predict password for a number of applications, thus ensuring security.

2.1 FUNCTIONALITIES AND FEATURES:

The application will generate passwords of a certain length. It contains the digits and punctuation marks specified by the user.

It takes care to have a mixture of letters ,digits, and special characters for its complexity.

2.2 PASSWORD GENERATION CRITERIA:

- Minimum length of 10 characters.
- User set options for digits and punctuation marks
- Randomly selects and shuffles the characters to make it unique

2.3 HOW IT GENERATES PASSWORD:

The password generator uses python’s string and random modules to generate a strong password.

The user will be asked to give an input of enter the length , desired digits, and desired punctuation that a password need to contain according to user. The generator imposes these conditions on the output password.

3.IMPLEMENTATION DETAILS

The code was written by using the lists, loops and exceptional handling and also created two functions

It uses string.digits , string.punctuation , and string.ascii_letter to get possible characters; randomly picks up and shuffles characters to form the final password.

string. ascii_letter takes both uppercase and lower-case letters which is required to build a strong password

And by the function created it also ensures minimum password length is 10

PROBLEMS AND SOLUTIONS:

User-defined digits and punctuation marks should be included in the password while keeping it in the overall length and complexity that is required.

For which I have created two separate lists and added the desired characters of user to them and combined them and used them and made sure every one of desired character is in the password.

4.CODE STRUCTURE:

The code is structured into functions handling the different aspects of password generation and input validation. The function ,exceptions() manage the user’s input and calls the creating_password() function to generate the password

FILE STRUCTURE:

Creating_password(length, desired_digits, desired_punc): this function generates password based upon user’s criteria

expections():this function deals with user input and its validation

CODE

```
import string

import random

def creating_password(length, desired_digits, desired_punc):

   digits = string.digits

   punc = string.punctuation

   letters = string.ascii_letters

   all_characters = letters + digits + punc

   list_1= []

   for i in desired_digits:

      if str(i) in digits:

         list_1.append(str(i))

   list_2 = []

   for i in desired_punc:

      if str(i) in punc:

         list_2.append(str(i))

   password_chars = list_1 + list_2

   if len(password_chars) < length:

      password_chars += random.choices(all_characters, k=length-
len(password_chars))

   random.shuffle(password_chars)
```

```
    password = ' '.join(password_chars)
    return password
def exceptions():
        try:
            length = int(input("Enter the length of the password: "))
            if length < 10:
                raise ValueError("The length must be above 10. Enter a valid length.")
        except Exception as ex:
            print("An unexpected exception occurred: ", ex)
        desired_digits = list(input("Enter the desired digits: "))
        desired_punc = list(input("Enter the desired punctuations: "))
        password = creating_password(length, desired_digits, desired_punc)
        print(password)
exceptions()
```

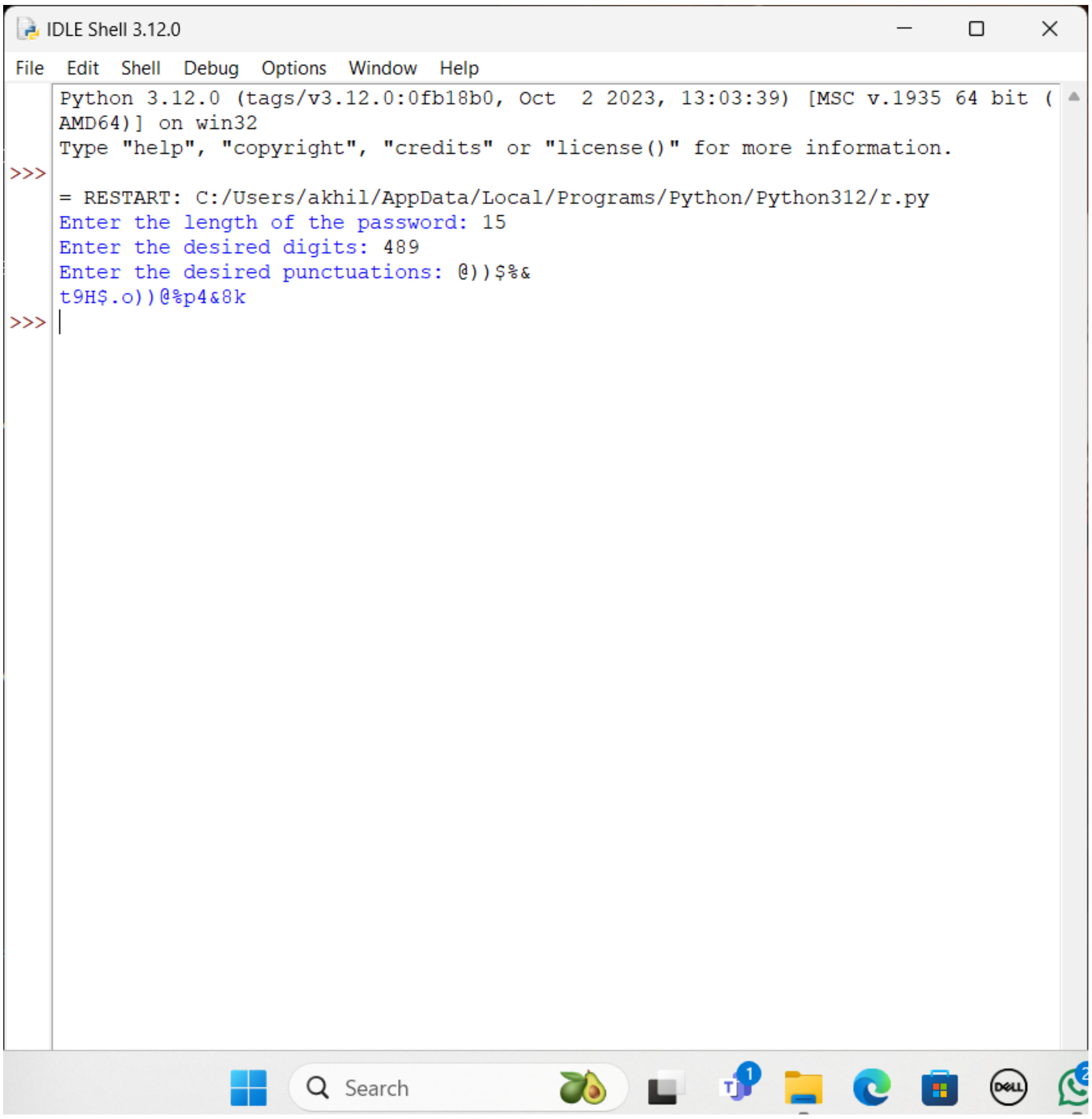
5.TESTING:

TESTING METHODOLOGY:

The project was tested by the Manual Testing Techniques .I used manual testing which was done by running the program a number of times with different combinations to ensue everything works as expected. I tested the user interfaces, error handling , and overall functionality of program

RANDOM PASSWORD GENERATOR

INPUTS AND OUTPUT



```
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/akhil/AppData/Local/Programs/Python/Python312/r.py
Enter the length of the password: 15
Enter the desired digits: 489
Enter the desired punctuations: @))$%&
t9H$.o))@%p4&8k
>>>
```

6. RESULTS AND DISCUSSION

Random password generator

The random password generator efficiently creates secure passwords based on user defined criteria. Generation display of error message are pretty easy from the user experience perspective, making it hassle free to use. Performance is decent; computation time taken to generate one password is minimal

7. CONCLUSION:

The random password generator demonstrates some of the fundamental ideas or concepts of programming and how they are put into practice.

The project meets their objectives by implementing relevant secure password generation and a playable game experience. In the future user interface and feature enhancements for both projects could be considered

8. REFERENCES:

https://www.geeksforgeeks.org/enumerate-in-python/

https://www.w3schools.com/python/gloss_python_for_else.asp

https://www.geeksforgeeks.org/python-string-digits/

https://www.geeksforgeeks.org/python-string-join-method/