Assignment 3

CSC 4110, DUE FEBRUARY 3RD, 2022

Mr. Jason Myers
WAYNE STATE UNIVERSITY | INFORMATION TECHNOLOGY

Assignment Instructions:

Do problem(s)/ resolve issues, following instructions explicitly (see Rubric). Any suspicion of group work (unless specifically stated) will result in a grade of 0. Problems / issues on final page.

Due:

See upload folder; due date is firm; NO EXCEPTIONS.

Objectives:

Fulfill customer request(s) / resolve customer issues (see Assignment).

Software Allowed/ Required:

Python3.x "script mode", Github (desktop)
Note: Word, Winzip, any text editor, as needed. This does not include any modules, such as *matplotlib, math, or system/sys*.

GitHub:

GitHub Video 1: https://www.youtube.com/watch?v=fJtyf62yAb8

GitHub Video 2: https://www.youtube.com/watch?v=GqNAD4XoZ6k

Reference following article to create repository so you can load this assignment output:

https://docs.github.com/en/desktop/installing-and-configuring-github-desktop/overview/getting-started-with-github-desktop

References: Course videos, supplied resources.

General Requirements:

Add labeling/ comments (name, date, revision #); add in-line requirements where appropriate (such as syntax usage).

#Indicate coding begin and end

Example acceptable code comment:

Revision number BEGIN/ START DATE
Begin John D. Student here (date)

```
>> i =0 #reference variable for while loop
>> while i < 20: #creates a TRUE condition allowing while loop to proceed
..     print("Bring your own lunch!!") # Indent for while loop
..     i+=1 # Increments by one to add a limit
```

Revision number FINAL DATE
End John D. Student here
Group / manager/ lead tech/ project

Adhere to the following coding style (from PEP8):

1. Wrap lines so that they don't exceed 79 characters.
2. Use blank lines to separate functions and classes, and larger blocks of code inside functions
3. When possible, put comments on a line of their own.
4. Name your classes and functions consistently; the convention is to use `UpperCamelCase` for classes and `lowercase_with_underscores` for functions and methods.

Deliverable Instructions:

Upload .py file (not image) to your GitHub (pls change extension to .txt)

Upload link to your .py file in the COMMENTS section of the UPLOAD Folder on CANVAS

See rubric for complete delivery details.

Debriefing Form:

Group/ manager / lead tech/ project #	Omega Group, manger: Ram Valud, lead tech: Michael Walker, project #: project greenwood321
Planning (Hours worked on (billing))	4 hours
Execution (Hours worked on (billing))	6 hours
Host OS	Windows 10
Platform (if any)	Visual Studio Code

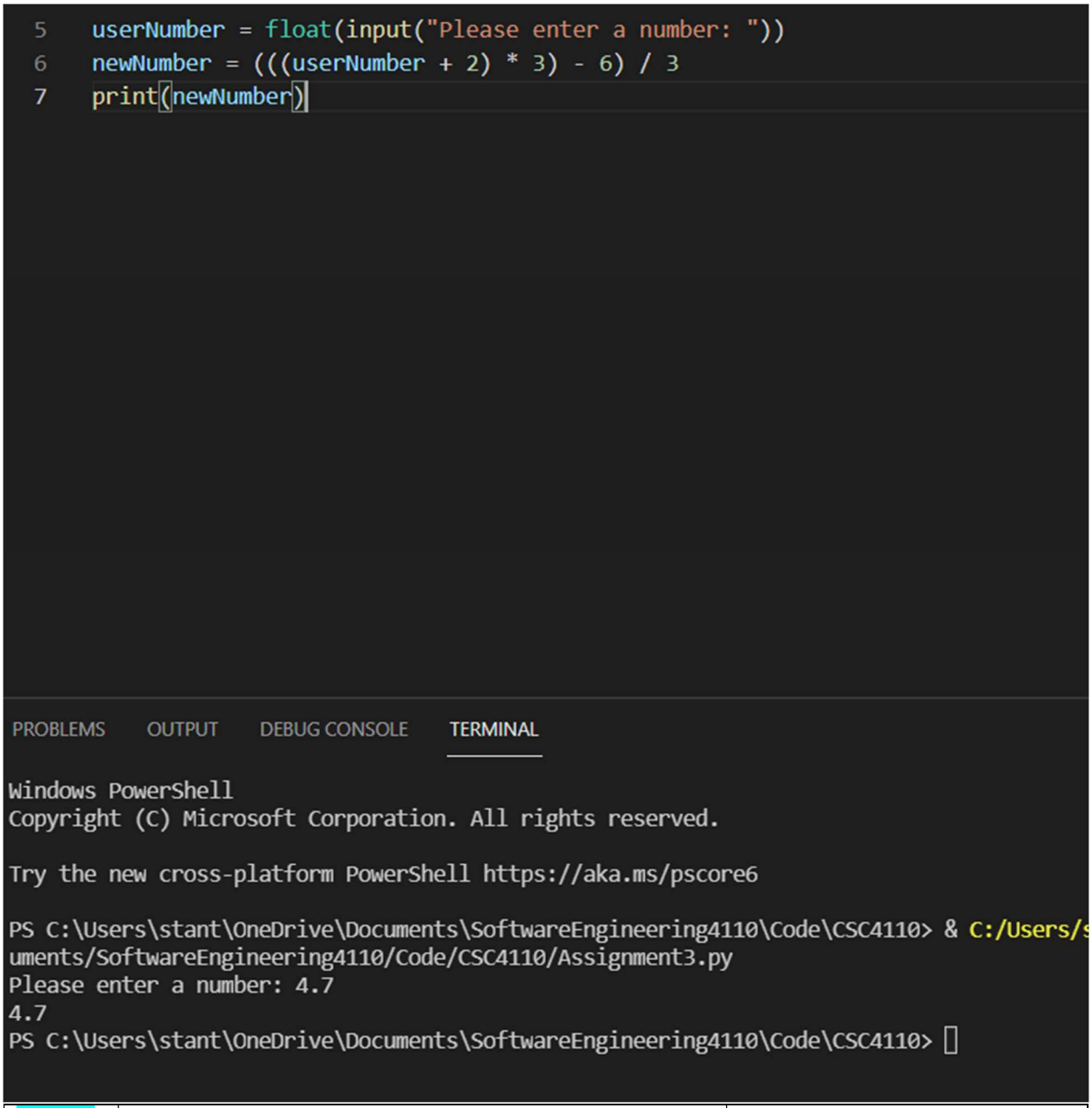
Item	RUBRIC	Pts
1	Customer request fulfilled (all issues completed and output consistent with customer expectation)	60
2	PEP8, lines 1 through 4, as appropriate and where necessary	10
3	.py (as .txt) file uploaded to GitHub repository	10
4	On-time (submitted on due date; not before or after) ←Note: only LATE submissions penalized	10
5	GitHub Link placed in upload comments section	10
6	Code images and OUTPUT images pasted placed in original assignment, uploaded to course shell (with appropriate comments, etc…)	10
7	Comments are appropriate and explanatory; contain all necessary information	10
8	Debriefing form filled out properly and completely	10
9	Only software mentioned is used (i.e. no extra modules imported)	10
10	External packages not imported, unless explicitly allowed	10

TOTAL POINTS: 140

ASSIGNMENT

Customer needs the following issues resolved by stated due date:

Issue 1		
	Customer needs a program written in Python that: prompts for a number, takes that number, adds 2, multiplies by 3, subtracts 6, and divides by 3. • Ensure that you get the number you started with.	This field for customer use only
	Note: no special requirements; just copy/ paste code and output for the customer.	



Issue 2		
	Customer needs to know what happens when you execute the following code groups: Group one: my_var1 = 7.0 my_var2 = 5 print(my_var1 % my_var2) Group two: x = 4	This field for customer use only

	y = 5 print(x//y) Group three: 30−3**2+ 8//3**2*10	
	Note: no special requirements; just copy/ paste code and output	

```
#Issue 2: the client wants to know what the output is of each group
# the output will print along with which group it is
#Group one:
print("Group one: ")
my_var1 = 7.0
my_var2 = 5
print(my_var1 % my_var2)

print("Group two: ")
#Group two:
x = 4
y = 5
print(x//y)

print("Group 3: ")
#Group three:
print(30-3**2+8//3**2*10)

```

```
Group one:
2.0
Group two:
0
Group 3:
21
PS C:\Users\stant\OneDrive\Documents\SoftwareEngineering4110\Code\CSC4110>
```

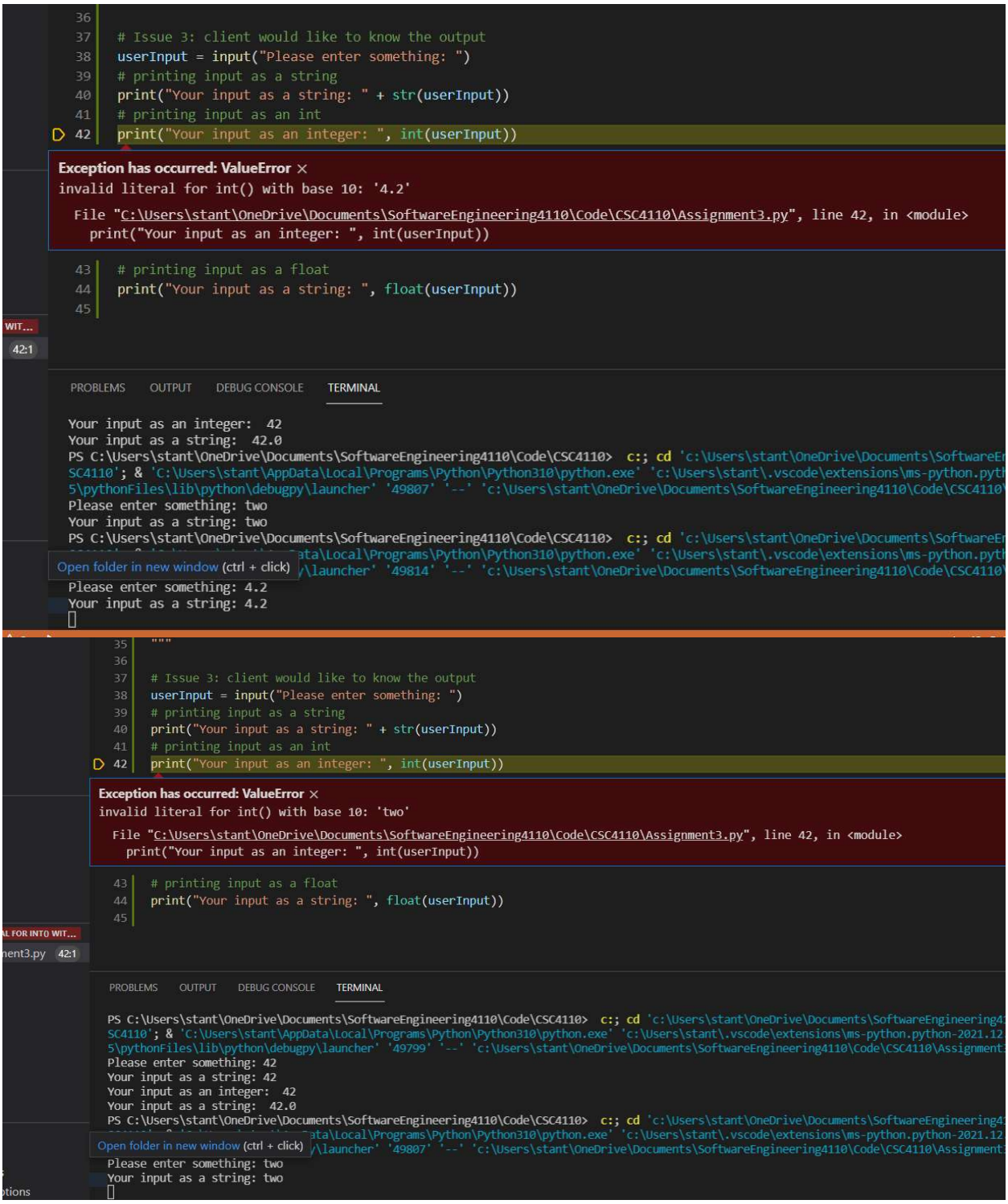
Issue 3		
	Customer needs to know what happens when you: Prompt for input and then print the input as a string, an integer, and a float-point value. What values can you input and print without errors being generated?	This field for customer use only
	Note: no special requirements; just copy/ paste code and output	

```
36
37  # Issue 3: client would like to know the output
38  userInput = input("Please enter something: ")
39  # printing input as a string
40  print("Your input as a string: " + str(userInput))
41  # printing input as an int
42  print("Your input as an integer: ", int(userInput))
43  # printing input as a float
44  print("Your input as a string: ", float(userInput))
45
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

  File "C:\Users\stant\AppData\Local\Programs\Python\Python310\lib\runpy.py", line 86, in _ru
    exec(code, run_globals)
  File "c:\Users\stant\OneDrive\Documents\SoftwareEngineering4110\Code\CSC4110\Assignment3.py
    print("Your input as an integer: " + int(userInput))
TypeError: can only concatenate str (not "int") to str
PS C:\Users\stant\OneDrive\Documents\SoftwareEngineering4110\Code\CSC4110>  c:; cd 'c:\Users\
SC4110'; & 'C:\Users\stant\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\stan
5\pythonFiles\lib\python\debugpy\launcher' '49799' '--' 'c:\Users\stant\OneDrive\Documents\Sc
Please enter something: 42
Your input as a string: 42
Your input as an integer:  42
Your input as a string:  42.0
PS C:\Users\stant\OneDrive\Documents\SoftwareEngineering4110\Code\CSC4110>
```

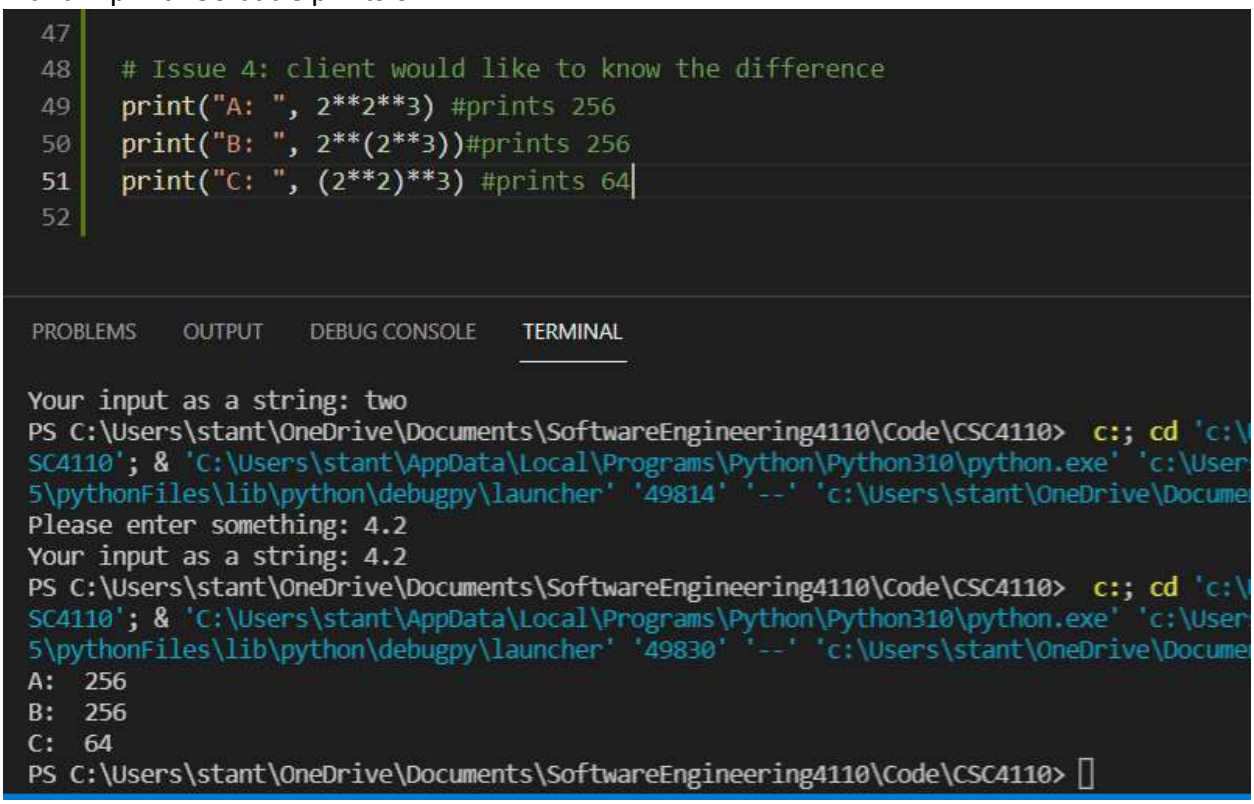
You can only enter integer numeric values without getting an error



Issue 4		
	Customer needs to know if there is a difference in the output of the following expressions, marked a through c :	This field for customer use only

	(a) 2**2**3 (b) 2**(2**3) (c) (2**2)**3	
	Note: no special requirements; just copy/ paste code and output, showing any possible difference.	

A and B print 256 but C prints 64



Issue 5	
	Customer needs a 'game of chance' simulation to do the following: (a) build and populate treasure chest with as many items customer requires (b) create a bank / loot stash

	(c) wagers to be placed per “spin” or treasure chest “grab” (d) customer “plays” until bank account reaches 0 or below.
	Note: the name of the simulation shall be “pirate” related; copy/ paste code and output, showing different outcomes; client requests “random” module to be imported.

```
53  # Issue 5: create a game of chance
54  import random
55
56  # ask user how many items to add to the chest
57  print("Welcome to Pirates Treasure! How many items can you collect before the money runs out?!\n")
58  numberOfChestItems = int(input("How many items would you like inside the treasure chest? "))
59  treasures = ["Gold", "Silver", "Copper"]
60  treasureChest = []
61  while numberOfChestItems > 0:
62      treasureChest.append(random.choice(treasures))
63      numberOfChestItems-= 1
64  print("The treasure chest is filled with: \n", treasureChest)
65  bank = 50.00
66  while bank > 0:
67      wager = random.randint(1,50) # randomly selects wager value
68      while wager > bank:
69          wager = random.randint(1,50) # randomly selects wager again if wager si greater than bank
70      print("Your wager is: ", wager)
71      bank -= wager
72      print("You have $%.2f" %bank + " left in the bank for your next wager and spin!")
73  print("You have nothing left in the bank! Thanks for playing!")
74
```

```
Welcome to Pirates Treasure! How many items can you collect before the money runs out?!

How many items would you like inside the treasure chest? 7
The treasure chest is filled with:
 ['Copper', 'Copper', 'Copper', 'Gold', 'Gold', 'Gold', 'Gold']
Your wager is:  29
You have $21.00 left in the bank for your next wager and spin!
Your wager is:  1
You have $20.00 left in the bank for your next wager and spin!
Your wager is:  12
You have $8.00 left in the bank for your next wager and spin!
Your wager is:  4
You have $4.00 left in the bank for your next wager and spin!
Your wager is:  3
You have $1.00 left in the bank for your next wager and spin!
Your wager is:  1
You have $0.00 left in the bank for your next wager and spin!
You have nothing left in the bank! Thanks for playing!
```

Issue 6	
	Customer needs a password simulator to do the following: (a) create random passwords in perpetuity (b) if the password is “acceptable,” it gets archived (c) “unaccepted” passwords get deleted (d) no less than 40 iterations
	Customer rules of ‘accepted passwords’ include: “special symbols,” and password cannot be a word in a dictionary list; client requests “random” module to be imported.

**** ISSUE 6: create your own dictionary list, with an appropriate amount of entries.
Feel free to reference the “rockyou.txt” list online.

```python
# Issue 6: password simulator
import  random

acceptablePasswords = []
dictionaryList = []
# open file with list of passwords alredy in use
with open('rockYouPartialList.txt', 'r') as f:
    dictionaryList = f.readlines()

specialSymbols = ["!", "@", "#" , "$", "%", "^", "&", "*", ".", "?"]
characterList = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&&*.?"
count = 1
while count < 45:
    randomCharacters = [random.choice(characterList) for i in range(3,9)] # get up to 9 random characters
    randomPassword = "".join(randomCharacters) # creates a string out of the cahracters
    print(str(count) + ": Random Password Generated: " + randomPassword)

    if any(char in randomPassword for char in specialSymbols): #if the random password contains a special symbol
        if not any(randomPassword for randomPassword in dictionaryList): #check if the password is already in the dictionay
            print(randomPassword + " is unaccepted")
        else: # if not in dictionary but has a special character then password is accepted
            print(randomPassword + " is accepted and will be archived")
            acceptablePasswords.append(randomPassword)
    else:
        print(randomPassword + " is unaccepted")
    count+=1
    print()
```

```
30: Random Password Generated: LnoKbg
LnoKbg is unaccepted

31: Random Password Generated: NLmFih
NLmFih is unaccepted

32: Random Password Generated: IHGf$2
IHGf$2 is accepted and will be archived

33: Random Password Generated: TsA*pw
TsA*pw is accepted and will be archived

34: Random Password Generated: K^zR14
K^zR14 is accepted and will be archived

35: Random Password Generated: &V$Iyv
&V$Iyv is accepted and will be archived

36: Random Password Generated: t$uvrP
t$uvrP is accepted and will be archived

37: Random Password Generated: *eQ1nI
*eQ1nI is accepted and will be archived

38: Random Password Generated: Gz6kFg
Gz6kFg is unaccepted

39: Random Password Generated: f&Pfnl
f&Pfnl is accepted and will be archived

40: Random Password Generated: 1wtT^n
1wtT^n is accepted and will be archived

41: Random Password Generated: 3Y!8Eo
3Y!8Eo is accepted and will be archived

42: Random Password Generated: QkfBKF
QkfBKF is unaccepted

43: Random Password Generated: !lL&XY
```