**20 November 2024 at 16:32**

**You said:**

Tweak the following code to generate multiple passwords as a single array. Number of passwords to be generated will be given as an input argument by the user Where def PASSRDSEED(dummy=None): is mentioned at the end below (instead of dummy=None) user will give some integer, e.g., 10, 100, etc.from rdrand import RdSeedom import string import xlwings as xw def generate\_password\_rdseed(length1=12): r1 = RdSeedom() char\_set1 = string.ascii\_letters + string.digits + '?@$#^&\*' special\_chars1 = '?@$#^&\*' # Ensure at least one uppercase, one lowercase, and one special character password1 = [ r1.choice(string.ascii\_uppercase), r1.choice(string.ascii\_lowercase), r1.choice(special\_chars1) ] # Fill the rest of the password length with random characters while len(password1) < length1: char1 = r1.choice(char\_set1) # Ensure no more than two special characters if char1 in special\_chars1 and sum(c1 in special\_chars1 for c1 in password1) >= 2: continue password1.append(char1) # Shuffle to avoid predictable patterns using rdrand for i1 in range(len(password1)): j1 = r1.randint(0, len(password1) - 1) password1[i1], password1[j1] = password1[j1], password1[i1] # Ensure the password does not start with a special character while password1[0] in special\_chars1: for i1 in range(len(password1)): j1 = r1.randint(0, len(password1) - 1) password1[i1], password1[j1] = password1[j1], password1[i1] return ''.join(password1) @xw.func def PASSRDSEED(dummy=None): return generate\_password\_rdseed()

**Copilot said:**

Sure, I can help with that! Here's the modified code to generate multiple passwords as a single array. The number of passwords to be generated will be given as an input argument by the user:

from rdrand import RdSeedom

import string

import xlwings as xw

def generate\_password\_rdseed(length1=12):

r1 = RdSeedom()

char\_set1 = string.ascii\_letters + string.digits + '?@$#^&\*'

special\_chars1 = '?@$#^&\*'

# Ensure at least one uppercase, one lowercase, and one special character

password1 = [

r1.choice(string.ascii\_uppercase),

r1.choice(string.ascii\_lowercase),

r1.choice(special\_chars1)

]

# Fill the rest of the password length with random characters

while len(password1) < length1:

char1 = r1.choice(char\_set1)

# Ensure no more than two special characters

if char1 in special\_chars1 and sum(c1 in special\_chars1 for c1 in password1) >= 2:

continue

password1.append(char1)

# Shuffle to avoid predictable patterns using rdrand

for i1 in range(len(password1)):

j1 = r1.randint(0, len(password1) - 1)

password1[i1], password1[j1] = password1[j1], password1[i1]

# Ensure the password does not start with a special character

while password1[0] in special\_chars1:

for i1 in range(len(password1)):

j1 = r1.randint(0, len(password1) - 1)

password1[i1], password1[j1] = password1[j1], password1[i1]

return ''.join(password1)

def generate\_multiple\_passwords(num\_passwords, length1=12):

passwords = []

for \_ in range(num\_passwords):

passwords.append(generate\_password\_rdseed(length1))

return passwords

@xw.func

def PASSRDSEED(num\_passwords):

return generate\_multiple\_passwords(num\_passwords)

This code will generate the specified number of passwords and return them as an array. You can call the PASSRDSEED function with the desired number of passwords as an argument. For example, PASSRDSEED(10) will generate an array of 10 passwords.

Edit in Pages

Copy

1 of 30 responsesAI-generated content may be incorrect

**20 November 2024 at 16:33**

**You said:**

Integrate:def generate\_multiple\_passwords(num\_passwords, length1=12): passwords = [] for \_ in range(num\_passwords): passwords.append(generate\_password\_rdseed(length1)) return passwordsWith:generate\_password\_rdseed. So there will be single function.

**Copilot said:**

Got it! Here's the integrated function that generates multiple passwords within a single function:

from rdrand import RdSeedom

import string

import xlwings as xw

def generate\_passwords(num\_passwords, length1=12):

r1 = RdSeedom()

char\_set1 = string.ascii\_letters + string.digits + '?@$#^&\*'

special\_chars1 = '?@$#^&\*'

def generate\_password():

# Ensure at least one uppercase, one lowercase, and one special character

password1 = [

r1.choice(string.ascii\_uppercase),

r1.choice(string.ascii\_lowercase),

r1.choice(special\_chars1)

]

# Fill the rest of the password length with random characters

while len(password1) < length1:

char1 = r1.choice(char\_set1)

# Ensure no more than two special characters

if char1 in special\_chars1 and sum(c1 in special\_chars1 for c1 in password1) >= 2:

continue

password1.append(char1)

# Shuffle to avoid predictable patterns using rdrand

for i1 in range(len(password1)):

j1 = r1.randint(0, len(password1) - 1)

password1[i1], password1[j1] = password1[j1], password1[i1]

# Ensure the password does not start with a special character

while password1[0] in special\_chars1:

for i1 in range(len(password1)):

j1 = r1.randint(0, len(password1) - 1)

password1[i1], password1[j1] = password1[j1], password1[i1]

return ''.join(password1)

passwords = [generate\_password() for \_ in range(num\_passwords)]

return passwords

@xw.func

def PASSRDSEED(num\_passwords):

return generate\_passwords(num\_passwords)

This function, generate\_passwords, will generate the specified number of passwords and return them as an array. You can call the PASSRDSEED function with the desired number of passwords as an argument. For example, PASSRDSEED(10) will generate an array of 10 passwords.

Edit in Pages

Copy

2 of 30 responsesAI-generated content may be incorrect