



Bundesministerium
des Innern
und für Heimat

Bundeskriminalamt

P20
POLIZEI 20/20

GEMEINSAM.
DIGITAL.
VERNETZT.



KI Campus
DER POLIZEI

KI-KOST



ERSTER
**GEN AI
HACKATHON**
DER DEUTSCHEN POLIZEI
16.09. – 17.09.2024
AI Developer

Problem



- Effizienz- und Qualitätsgewinne, welche durch Gen-AI in der Software-Entwicklung bereits heute möglich sind, können im BKA (und weiteren Behörden) derzeit nicht genutzt werden
- Betrifft alle Software-Entwickler und DevOps-Ingenieure
 - allein im BKA: ca. 500-1000
 - indirekt profitieren auch die Anwender

Problem



- Effizienz- und Qualitätsgewinne, welche durch Gen-AI in der Software-Entwicklung bereits heute möglich sind, können im BKA (und weiteren Behörden) derzeit nicht genutzt werden
- Betrifft alle Software-Entwickler und DevOps-Ingenieure
 - allein im BKA: ca. 500-1000
 - indirekt profitieren auch die Anwender
- Offensichtliche Lösungen:
 - ChatGPT
 - Github Copilot

Problem



- Effizienz- und Qualitätsgewinne, welche durch Gen-AI in der Software-Entwicklung bereits heute möglich sind, können im BKA (und weiteren Behörden) derzeit nicht genutzt werden
- Betrifft alle Software-Entwickler und DevOps-Ingenieure
 - allein im BKA: ca. 500-1000
 - indirekt profitieren auch die Anwender
- Offensichtliche Lösungen:
 - ChatGPT
 - Github Copilot

nicht möglich: Sicherheitskonzept



Lösung



- Open Source Code-LLMs
- Open Source Tools / IDEs / Plugins ...

Lösung 1: Autonomer AI Developer: OpenHands

The screenshot displays the OpenHands AI developer interface, which is divided into three main sections: Chat, Code Editor, and Terminal.

Chat Window:

- Message 1: "The server is running successfully on `http://localhost:5001`. Now, let's test the server to ensure it returns the correct HTML content. I'll use `curl` to fetch the content of the server's homepage:"
- Message 2: "The server is working correctly on port 5001, and the content of the server's homepage is:"
- Message 3: "If you have any further requests or need additional modifications, please let me know! If everything is complete, you can indicate that by replying with `^`."
- Message 4: "translate this app to spring boot"
- Message 5: "There was an unexpected error while running the agent"

Code Editor:

The workspace shows a project named `nodejs-app` with a file `server.ts` selected. The code in `server.ts` is as follows:

```
1 import express from 'express';
2
3 const app = express();
4 const port = 5001;
5
6 app.get('/', (req, res) => {
7   res.send(`
8     <html>
9       <head>
10         <title>Simple HTML Page</title>
11       </head>
12       <body>
13         <h1>Welcome to the Simple HTML Page</h1>
14         <p>This is a simple web page served by an Express web server.</p>
15       </body>
16     </html>
17   `);
18 });
19
20 app.listen(port, () => {
21   console.log(`Server is running on http://localhost:${port}`);
22 });
```

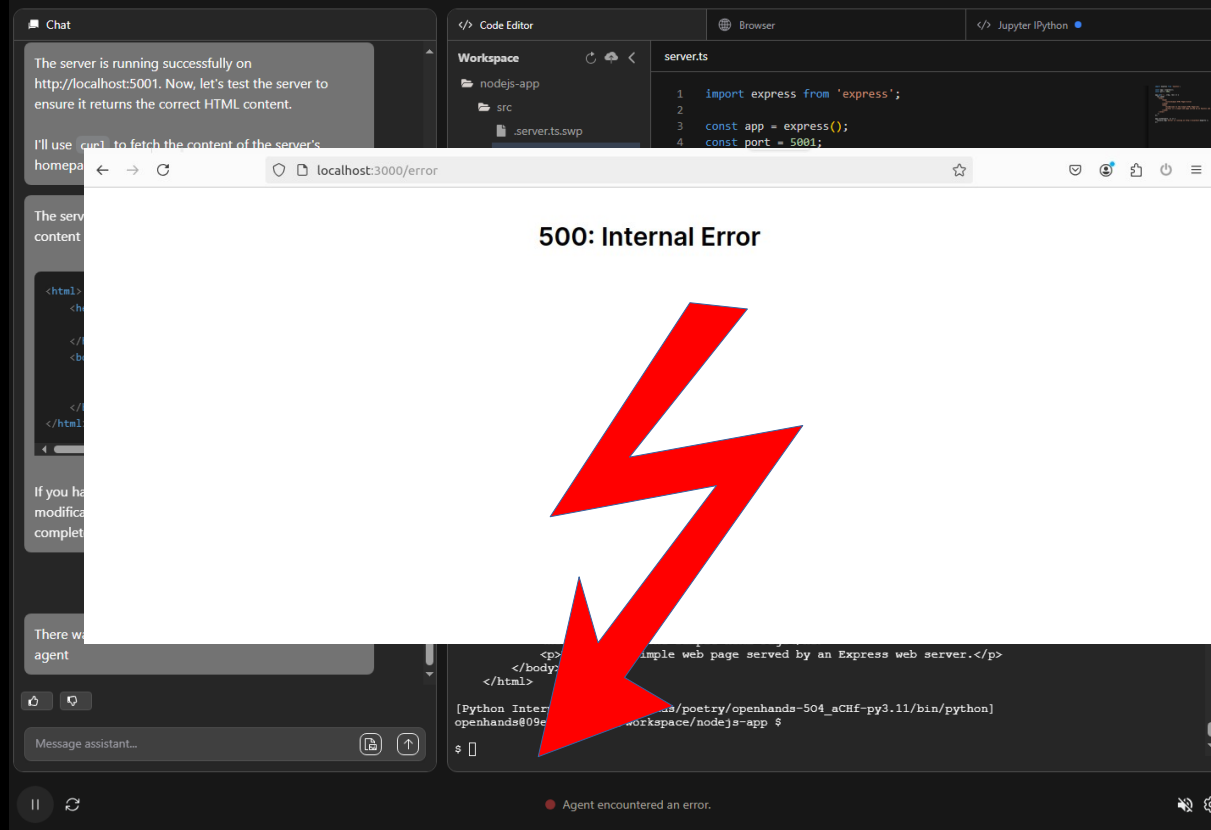
Terminal:

The terminal shows the execution of the `curl` command to test the server:

```
[Python Interpreter: /openhands/poetry/openhands-504_aChf-py3.11/bin/python]
openhands@09eb7d71dd4a: /workspace/nodejs-app $
$ curl http://127.0.0.1:5001
<html>
<head>
<title>Simple HTML Page</title>
</head>
<body>
<h1>Welcome to the Simple HTML Page</h1>
<p>This is a simple web page served by an Express web server.</p>
</body>
</html>
```

At the bottom of the terminal, there is a message: "Agent encountered an error."

Lösung 1: Autonomer AI Developer: OpenHands



Lösung 2: IDE Plugins



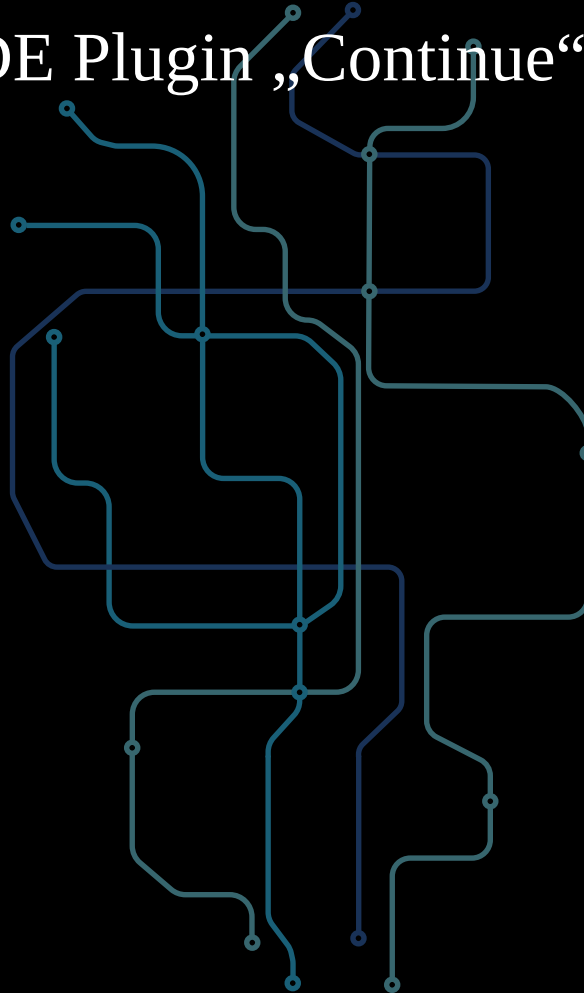
- Github Copilot-ähnliche VS Code Extensions:
 - Continue
 - Twinny
- Open Source LLMs selbst-deployed:
 - deepseek-coder-v2
 - codellama:7b-code
 - codellama:7b-instruct
 - codellama:70b
 - codegemma
 - codestral:22b

Lösung 2: IDE Plugins



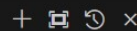
- Github Copilot-ähnliche VS Code Extensions:
 - Continue 🏰
 - Twinny
- Open Source LLMs selbst-deployed:
 - deepseek-coder-v2 (Code Completion) 🏰
 - codellama:7b-code
 - codellama:7b-instruct (Chat) 🏰
 - codellama:70b
 - codegemma
 - codestral:22b

Lösung 2: IDE Plugin „Continue“ + Chat-LLM



config.json

Untitled-1



1 Use Ctrl+I to generate code

Ask anything, '/' for slash commands, '@' to add context

unser codellama Add Context +

Enter

← Last Session



Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text ✓ Continue



1 *Use Ctrl+I to generate code*

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama ▾ Add Context +

Enter

← Last Session

1 Use Ctrl+I to generate code

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

user codellama ▾ Add Context +

^ 📄 Use codebase ⏎ Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length."""
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#%&'*"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

📄 📄 📄
Apply to current file

This script uses the `secrets` module to generate a random integer between 0 and the length of the character set, which is then used as an index into the character set to get the next character in the password. This process is repeated until the desired length for the password is reached.

To add uppercase letters, numbers, and special characters to the password, you can modify the `alphabet` variable to include these characters. For example:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length."""
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#%&'*"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
```

Accept | Reject

```
1 import secrets
2
3 def generate_password(length):
4     """Generate a securely random password of the specified length."""
5     alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama ▾ Add Context +

🗨 Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length."""
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#%&*"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module to generate a random integer between 0 and the length of the character set, which is then used as an index into the character set to get the next character in the password. This process is repeated until the desired length for the password is reached.

To add uppercase letters, numbers, and special characters

C: > Users > Master > projects > ai-developer > generatePW.py > ...

```
1 import secrets
2
3 def generate_password(length):
4     """Generate a securely random password of the specified length."""
5     alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama ▾ Add Context +

⌘ Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length.
    The password will include uppercase and lowercase letters,
    numbers, and special characters.
    """
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module to generate a random integer between 0 and the length of the character set, which is then used as an index into the character set to get the next character in the password. This process is repeated until the desired length for the password is reached.

To add uppercase letters, numbers, and special characters

C: > Users > Master > projects > ai-developer > generatePW.py > ...

```
1 import secrets
2
3 def generate_password(length):
4     """Generate a securely random password of the specified length."""
5     alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*~"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama ▾ Add Context +

Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length.
    The password will include a mix of uppercase and lowercase letters,
    numbers, and special characters.
    """
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*~"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module to generate a random integer between 0 and the length of the character set, which is then used as an index into the character set to get the next character in the password. This process is repeated until the desired length for the password is reached.

To add uppercase letters, numbers, and special characters,

C: > Users > Master > projects > ai-developer > generatePW.py > ...

```
1 import secrets
2
3 def generate_password(length):
4     """Generate a securely random password of the specified length."""
5     alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*~"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama ▾ Add Context +

Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length using the secrets module.
    The password will include a mix of uppercase and lowercase letters,
    numbers, and special characters.

    Args:
        length (int): The desired length of the password.

    Returns:
        str: A securely generated random password of the specified length.
    """
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*~"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module to generate a random integer between 0 and the length of the character set, which is then used as an index into the character set to get the next character in the password. This process is repeated until the desired length for the password is reached.

To add uppercase letters, numbers, and special characters,

```
PS C:\Users\Master> & C:/Users/Master/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Master
/projects/ai-developer/generatePW.py
Generated Password: jU#Bo2g1EoiIX$eq
PS C:\Users\Master>
```

C: > Users > Master > projects > ai-developer > generatePW.py > ...

```
1 import secrets
2
3 def generate_password(length):
4     """Generate a securely random password of the specified length."""
5     alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*~"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
12
13 def test_generate_password():
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama ▾ Add Context +

Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length."""
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*~"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module to generate a random integer between 0 and the length of the character set, which is then used as an index into the character set to get the next character in the password. This process is repeated until the desired length for the password is reached.

To add uppercase letters, numbers, and special characters

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + ▾

```
PS C:\Users\Master> & C:/Users/Master/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Master
/projects/ai-developer/generatePW.py
Generated Password: jU#Bo2g1EoiIX$eq
PS C:\Users\Master>
```

C: > Users > Master > projects > ai-developer > generatePW.py > ...

```
1 import secrets
2
3 def generate_password(length):
4     """Generate a securely random password of the specified length."""
5     alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
12
13 def test_generate_password():
14     """Test the generate_password function."""
15     length = 16
16     password = generate_password(length)
17     assert len(password) == length, "Password length is incorrect"
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama Add Context +

Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length."""
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module to generate a random integer between 0 and the length of the character set, which is then used as an index into the character set to get the next character in the password. This process is repeated until the desired length for the password is reached.

To add uppercase letters, numbers, and special characters,

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + -

```
PS C:\Users\Master> & C:/Users/Master/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Master
/projects/ai-developer/generatePW.py
Generated Password: jU#Bo2g1EoiIX$eq
PS C:\Users\Master>
```

C: > Users > Master > projects > ai-developer > generatePW.py > ...

```
1 import secrets
2
3 def generate_password(length):
4     """Generate a securely random password of the specified length."""
5     alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*~"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
12
13 def test_generate_password():
14     """Test the generate_password function."""
15     length = 16
16     password = generate_password(length)
17     assert len(password) == length, "Password length is incorrect"
18     assert any(c.islower() for c in password), "Password must contain at least one lowercase letter"
19
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Master> & C:/Users/Master/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Master/projects/ai-developer/gen
• ereatePW.py
  Generated Password: jU#Bo2g1EoiIX$eq
○ PS C:\Users\Master>
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama

Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length."""
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*~"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module

C: > Users > Master > projects > ai-developer > generatePW.py > ...

```
3 def generate_password(length):
4     """Generate a securely random password of the specified length."""
5     alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#%&'*"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
12
13 def test_generate_password():
14     """Test the generate_password function."""
15     length = 16
16     password = generate_password(length)
17     assert len(password) == length, "Password length is incorrect"
18     assert any(c.islower() for c in password), "Password must contain at least one lowercase letter"
19     # print final conclusion after all tests are done
20     print("All tests passed!")
21
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama

Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the specified length."""
    alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#%&'*"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python

```
PS C:\Users\Master> & C:/Users/Master/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Master/projects/ai-developer/generatePW.py
Generated Password: jU#Bo2g1EoiIX$eq
PS C:\Users\Master>
```

C: > Users > Master > projects > ai-developer > generatePW.py > ...

```

3 def generate_password(length):
4     """Generate a secure random password of length length"""
5     alphabet = "abcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*~`-_=+{}|;:~"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
12
13 def test_generate_password():
14     """Test the generate_password function."""
15     length = 16
16     password = generate_password(length)
17     assert len(password) == length, "Password length is incorrect"
18     assert any(c.islower() for c in password), "Password must contain at least one lowercase letter"
19     # print final conclusion after all tests are done
20     print("All tests passed!")
21
22 test_generate_password()
23

```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama v

Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```

import secrets

def generate_password(length):
    """Generate a securely random password of length length"""
    alphabet = "abcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*~`-_=+{}|;:~"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password

```

This script uses the `secrets` module

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + v

```

PS C:\Users\Master> & C:/Users/Master/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Master/projects/ai-developer/gen
eneratePW.py
Generated Password: jU#Bo2g1EoiIX$eq
PS C:\Users\Master>

```



C: > Users > Master > projects > ai-developer > generatePW.py > ...

```
3 def generate_password(length):
4     """Generate a secure random password of the given length"""
5     alphabet = "abcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*~`-_=+{}|;:'<,>[\\]`"
6     password = ""
7     for i in range(length):
8         password += alphabet[secrets.randbelow(len(alphabet))]
9     return password
10
11 print("Generated Password:", generate_password(16))
12
13 def test_generate_password():
14     """Test the generate_password function."""
15     length = 16
16     password = generate_password(length)
17     assert len(password) == length, "Password length is incorrect"
18     assert any(c.islower() for c in password), "Password must contain at least one lowercase letter"
19     # print final conclusion after all tests are done
20     print("All tests passed!")
21
22 test_generate_password()
```

I'm working on a small Python script to generate secure random passwords. I want it to be able to generate passwords that include uppercase and lowercase letters, numbers, and special characters. The user should be able to specify the length of the password. I'm a bit stuck on how to efficiently combine these character sets and ensure the password is truly random and secure. Could you help me write this script?

unser codellama ▾

Enter

Sure, I can help you with that! Here's an example script that uses the `secrets` module in Python to generate securely random passwords:

```
import secrets

def generate_password(length):
    """Generate a securely random password of the given length"""
    alphabet = "abcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*~`-_=+{}|;:'<,>[\\]`"
    password = ""
    for i in range(length):
        password += alphabet[secrets.randbelow(len(alphabet))]
    return password
```

This script uses the `secrets` module



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + ▾ □ ✕ ... ^

```
PS C:\Users\Master> & C:/Users/Master/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Master/projects/ai-developer/generatePW.py
Generated Password: jU#Bo2g1EoiIX$eq
PS C:\Users\Master> & C:/Users/Master/AppData/Local/Programs/Python/Python312/python.exe c:/Users/Master/projects/ai-developer/generatePW.py
Generated Password: 33#!NXkp0vCaX2Pj
All tests passed!
PS C:\Users\Master>
```

Wie geht es im Anschluss an den Hackathon weiter?

- Prüfung der Offline-Fähigkeit von „Continue“
 - Telefoniert evtl. nach Hause
- „Open WebUI“ erproben: ChatGPT-ähnliches Interface für selbst deployte Modelle
- Code-LLMs in echter Umgebung deployen (BKA-PSP)
- Testuser mit dem Plugin ausstatten und Erfahrungen sammeln
- Kontinuierlich neue Code-LLMs evaluieren

AI Developer Team



Christian H.
DevOps



Norman H.
DevOps