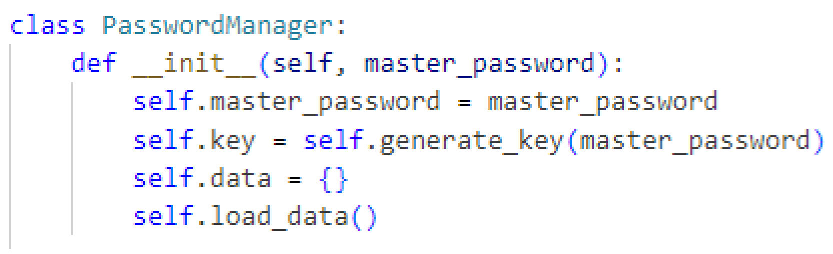
Dokumentation Password-Manager Gruppenarbeit Python

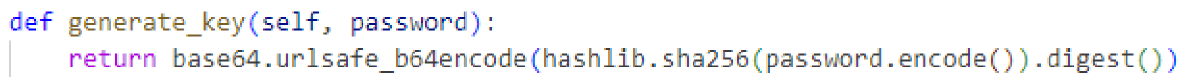
Das Programm Password Manager dient zur sicheren Verwaltung und Speicherung von Passwörtern für verschiedene Websites. Es enthält Funktionen zum Hinzufügen, Anzeigen, Bearbeiten und Löschen von Passworteinträgen.

Utility-Functions in main.py:



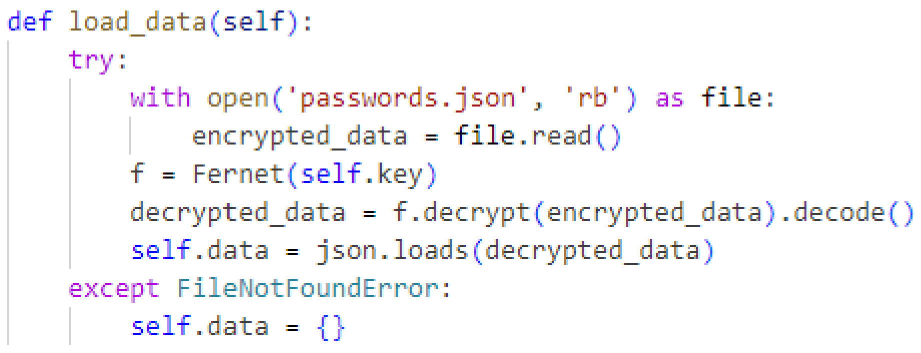
```python
class PasswordManager:
    def __init__(self, master_password):
        self.master_password = master_password
        self.key = self.generate_key(master_password)
        self.data = {}
        self.load_data()
```

- Initialisiert den Password Manager mit einem Master-Passwort. Das Master-Passwort wird verwendet, um Daten aus der verschlüsselten JSON-Datei zu laden.



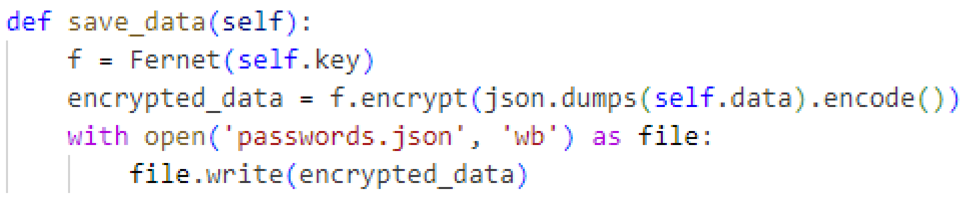
```python
def generate_key(self, password):
    return base64.urlsafe_b64encode(hashlib.sha256(password.encode()).digest())
```

- Generiert einen Key, der aus dem SHA- 256 Hash des angegebenen Passwortes abgeleitet wird



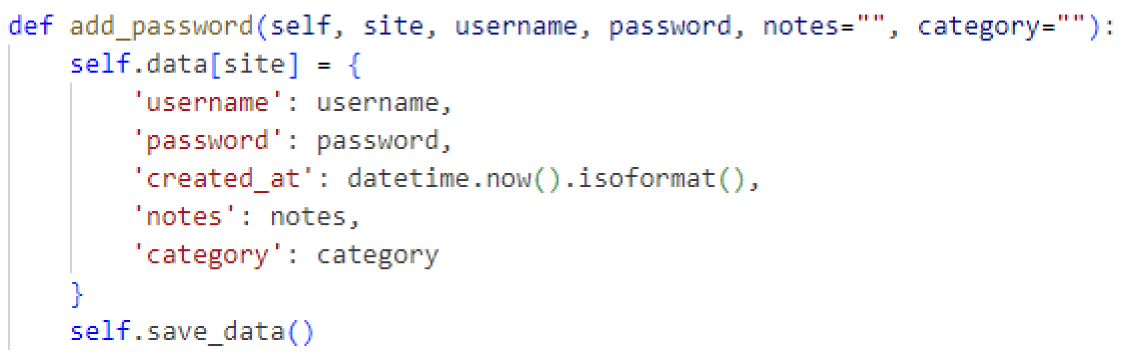
```python
def load_data(self):
    try:
        with open('passwords.json', 'rb') as file:
            encrypted_data = file.read()
        f = Fernet(self.key)
        decrypted_data = f.decrypt(encrypted_data).decode()
        self.data = json.loads(decrypted_data)
    except FileNotFoundError:
        self.data = {}
```

- Lädt die gespeicherten Passwortdaten, entschlüsselt mit dem Master-Passwort, aus der passwords.json



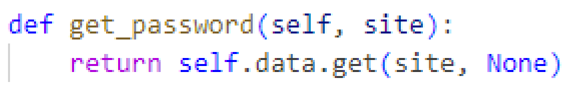
```python
def save_data(self):
    f = Fernet(self.key)
    encrypted_data = f.encrypt(json.dumps(self.data).encode())
    with open('passwords.json', 'wb') as file:
        file.write(encrypted_data)
```

- Verschlüsselt die aktuellen Passwortdaten und speichert sie in passwords.json



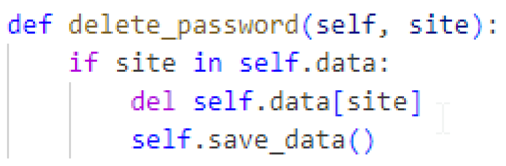
```python
def add_password(self, site, username, password, notes="", category=""):
    self.data[site] = {
        'username': username,
        'password': password,
        'created_at': datetime.now().isoformat(),
        'notes': notes,
        'category': category
    }
    self.save_data()
```

- Fügt neue Passworteinträge für bestimmte Websites hinzu, inklusive der gegebenen Parameter



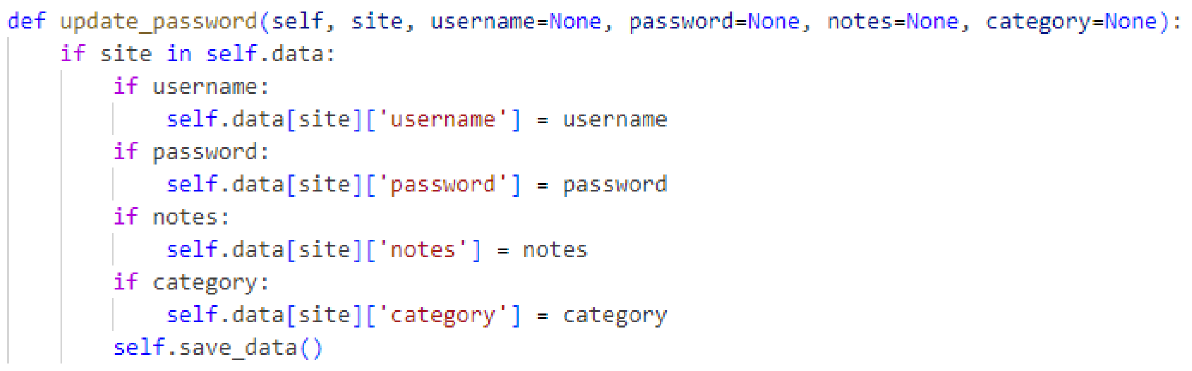
```python
def get_password(self, site):
    return self.data.get(site, None)
```

- Gibt die Passwortdaten einer bestimmten Website zurück



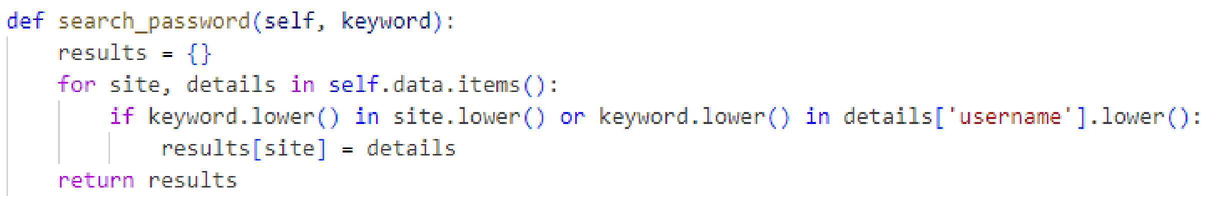
```python
def delete_password(self, site):
    if site in self.data:
        del self.data[site]
        self.save_data()
```

- Löscht die Daten einer bestimmten Website



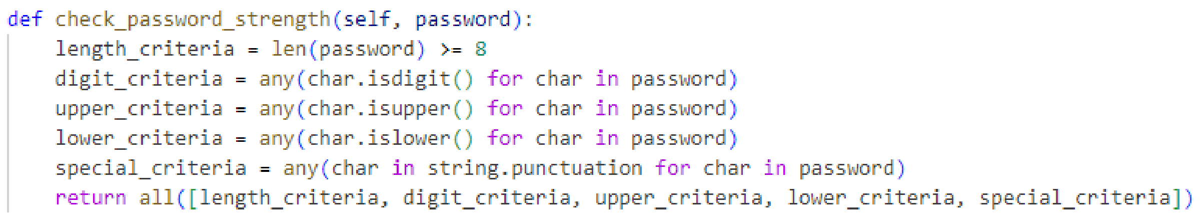
```python
def update_password(self, site, username=None, password=None, notes=None, category=None):
    if site in self.data:
        if username:
            self.data[site]['username'] = username
        if password:
            self.data[site]['password'] = password
        if notes:
            self.data[site]['notes'] = notes
        if category:
            self.data[site]['category'] = category
        self.save_data()
```

- Aktualisiert einen bestimmten Eintrag, jeder Parameter, der auf None gesetzt wird, bleibt unverändert.



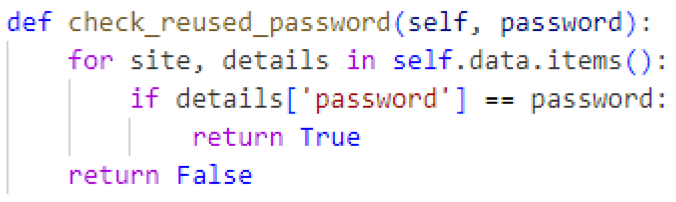
```python
def search_password(self, keyword):
    results = {}
    for site, details in self.data.items():
        if keyword.lower() in site.lower() or keyword.lower() in details['username'].lower():
            results[site] = details
    return results
```

- Sucht nach bestimmten Passworteinträgen anhand des Benutzernamens oder der Website und gibt sie aus



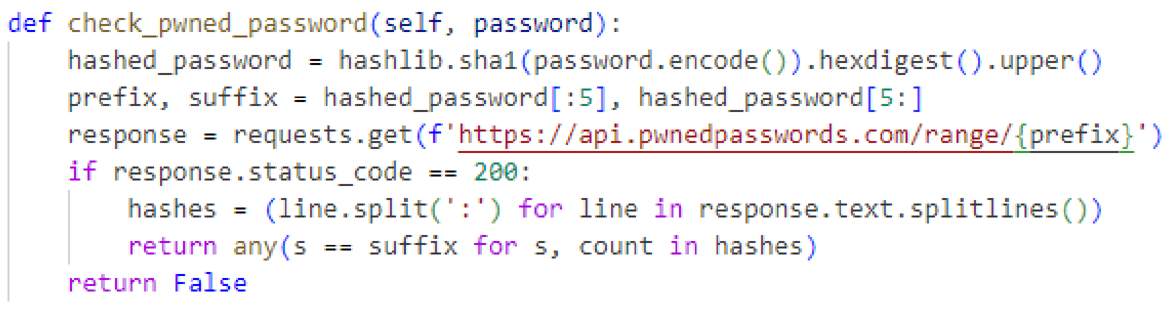
```python
def check_password_strength(self, password):
    length_criteria = len(password) >= 8
    digit_criteria = any(char.isdigit() for char in password)
    upper_criteria = any(char.isupper() for char in password)
    lower_criteria = any(char.islower() for char in password)
    special_criteria = any(char in string.punctuation for char in password)
    return all([length_criteria, digit_criteria, upper_criteria, lower_criteria, special_criteria])
```

- Überprüft das gegebene Passwort anhand von gängigen Kriterien, wie z.b. Lowercase, Uppercase, Zahlen bzw. Sonderzeichen



```python
def check_reused_password(self, password):
    for site, details in self.data.items():
        if details['password'] == password:
            return True
    return False
```

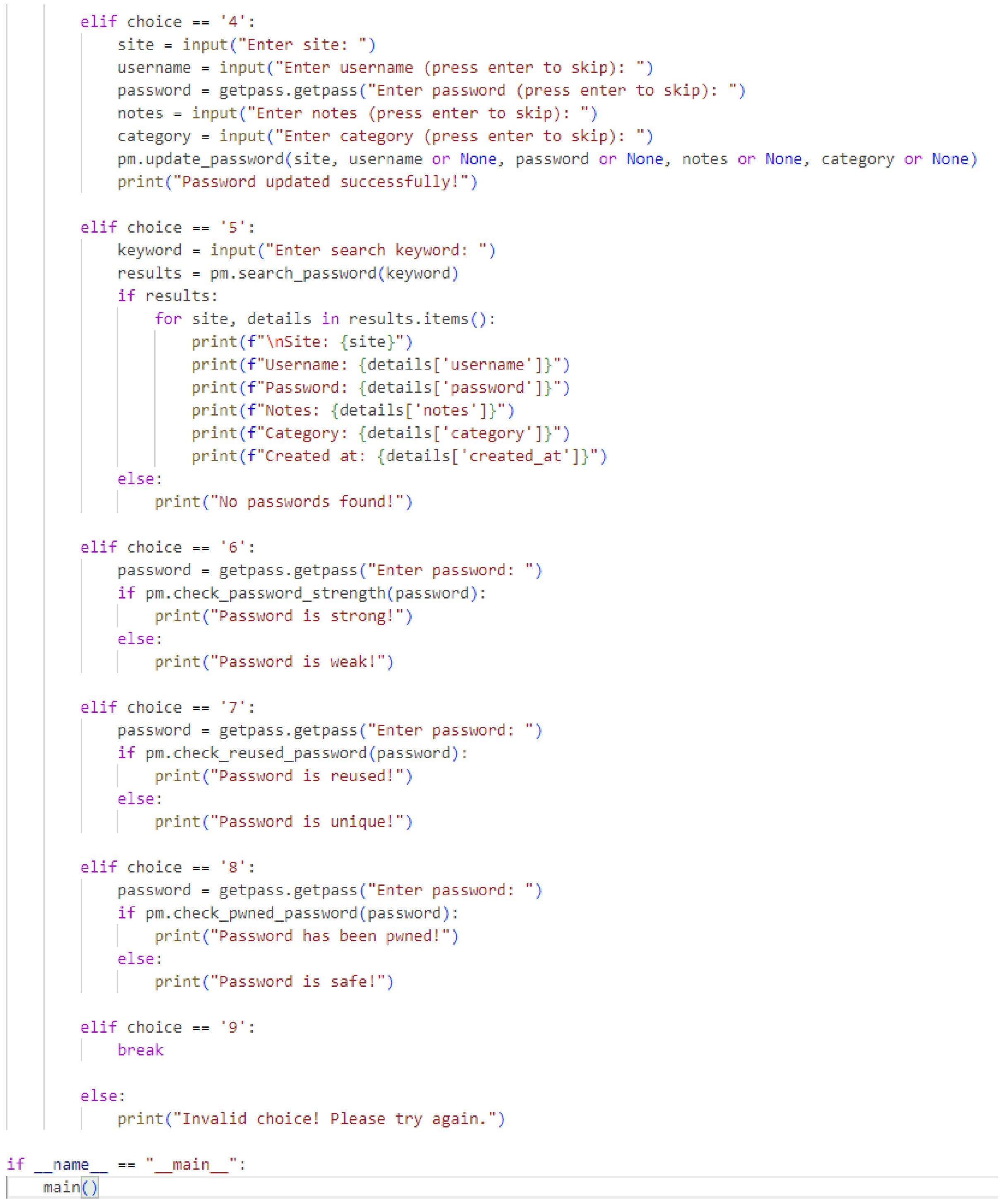
- Überprüft, ob das gegebene Passwort bereits für einen gespeicherten Eintrag verwendet wurde



```python
def check_pwned_password(self, password):
    hashed_password = hashlib.sha1(password.encode()).hexdigest().upper()
    prefix, suffix = hashed_password[:5], hashed_password[5:]
    response = requests.get(f'https://api.pwnedpasswords.com/range/{prefix}')
    if response.status_code == 200:
        hashes = (line.split(':') for line in response.text.splitlines())
        return any(s == suffix for s, count in hashes)
    return False
```

- Überprüft anhand der Pwned-API, ob das gegebene Passwort kompromittiert wurde

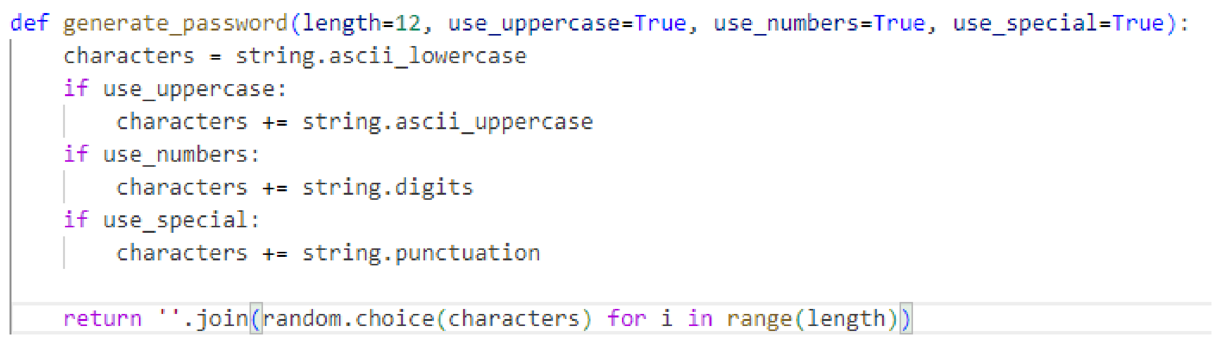
```python
def main():
    master_password = getpass.getpass('Enter your master password: ')
    pm = PasswordManager(master_password)

    while True:
        print("\nPassword Manager")
        print("1. Add Password")
        print("2. Get Password")
        print("3. Delete Password")
        print("4. Update Password")
        print("5. Search Password")
        print("6. Check Password Strength")
        print("7. Check Reused Password")
        print("8. Check Pwned Password")
        print("9. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            site = input("Enter site: ")
            username = input("Enter username: ")
            password = getpass.getpass("Enter password: ")
            notes = input("Enter notes (optional): ")
            category = input("Enter category (optional): ")
            pm.add_password(site, username, password, notes, category)
            print("Password added successfully!")

        elif choice == '2':
            site = input("Enter site: ")
            password = pm.get_password(site)
            if password:
                print(f"Username: {password['username']}")
                print(f"Password: {password['password']}")
                print(f"Notes: {password['notes']}")
                print(f"Category: {password['category']}")
                print(f"Created at: {password['created_at']}")
            else:
                print("Password not found!")

        elif choice == '3':
            site = input("Enter site: ")
            pm.delete_password(site)
            print("Password deleted successfully!")

        elif choice == '4':
            site = input("Enter site: ")
            username = input("Enter username (press enter to skip): ")
            password = getpass.getpass("Enter password (press enter to skip): ")
            notes = input("Enter notes (press enter to skip): ")
            category = input("Enter category (press enter to skip): ")
            pm.update_password(site, username or None, password or None, notes or None, category or None)
            print("Password updated successfully!")
```



```python
        elif choice == '4':
            site = input("Enter site: ")
            username = input("Enter username (press enter to skip): ")
            password = getpass.getpass("Enter password (press enter to skip): ")
            notes = input("Enter notes (press enter to skip): ")
            category = input("Enter category (press enter to skip): ")
            pm.update_password(site, username or None, password or None, notes or None, category or None)
            print("Password updated successfully!")

        elif choice == '5':
            keyword = input("Enter search keyword: ")
            results = pm.search_password(keyword)
            if results:
                for site, details in results.items():
                    print(f"\nSite: {site}")
                    print(f"Username: {details['username']}")
                    print(f"Password: {details['password']}")
                    print(f"Notes: {details['notes']}")
                    print(f"Category: {details['category']}")
                    print(f"Created at: {details['created_at']}")
            else:
                print("No passwords found!")

        elif choice == '6':
            password = getpass.getpass("Enter password: ")
            if pm.check_password_strength(password):
                print("Password is strong!")
            else:
                print("Password is weak!")

        elif choice == '7':
            password = getpass.getpass("Enter password: ")
            if pm.check_reused_password(password):
                print("Password is reused!")
            else:
                print("Password is unique!")

        elif choice == '8':
            password = getpass.getpass("Enter password: ")
            if pm.check_pwned_password(password):
                print("Password has been pwned!")
            else:
                print("Password is safe!")

        elif choice == '9':
            break

        else:
            print("Invalid choice! Please try again.")

if __name__ == "__main__":
    main()
```

- Eigentliche Main.py, die auf die davor erklärten Functions verweist bzw. diese ausführt und den User durch das Programm nutzen lässt

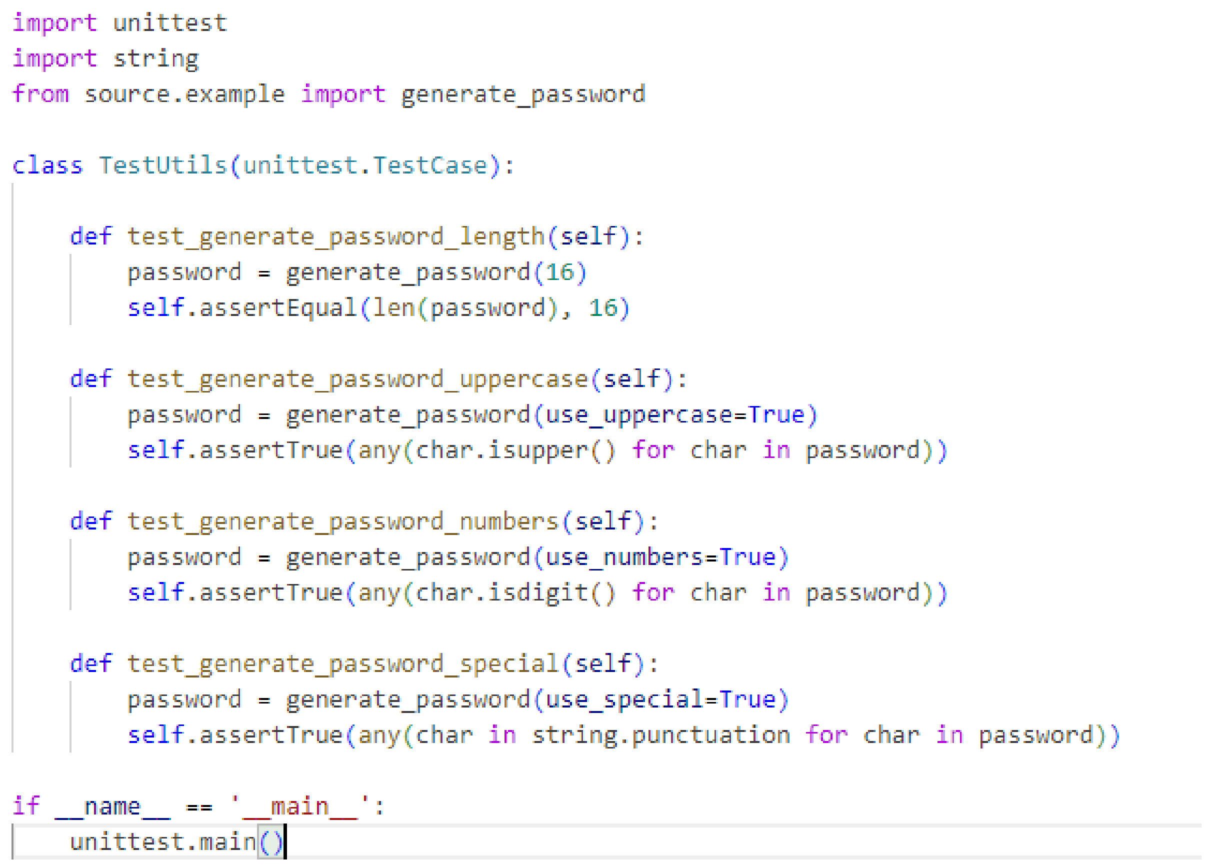
Utility-Functions in example.py:



```python
def generate_password(length=12, use_uppercase=True, use_numbers=True, use_special=True):
    characters = string.ascii_lowercase
    if use_uppercase:
        characters += string.ascii_uppercase
    if use_numbers:
        characters += string.digits
    if use_special:
        characters += string.punctuation

    return ''.join(random.choice(characters) for i in range(length))
```

- Erzeugt ein zufälliges Beispielpasswort

Utility-Functions in test_example.py:



```python
import unittest
import string
from source.example import generate_password

class TestUtils(unittest.TestCase):

    def test_generate_password_length(self):
        password = generate_password(16)
        self.assertEqual(len(password), 16)

    def test_generate_password_uppercase(self):
        password = generate_password(use_uppercase=True)
        self.assertTrue(any(char.isupper() for char in password))

    def test_generate_password_numbers(self):
        password = generate_password(use_numbers=True)
        self.assertTrue(any(char.isdigit() for char in password))

    def test_generate_password_special(self):
        password = generate_password(use_special=True)
        self.assertTrue(any(char in string.punctuation for char in password))

if __name__ == '__main__':
    unittest.main()
```

- Stellt sicher, dass die function generate_password richtig funktioniert – Überprüft ob die Anforderungen an Länge, Großbuchstaben, Ziffern und Sonderzeichen erfüllt

Verwendete Bibliotheken:

pylint=3.2.3

coverage=7.5.3

mypy=1.10.0

cryptography=42.0.0