



Mache nachfolgenden Code eleganter und beachte dabei die CleanCode Regeln:
passwort = input('Gebe dein Passwort ein:')

```
result = {
    'Zahlen': False,
    'Großbuchstabe': False,
    'Kleinbuchstabe': False,
    'Symbol': False
}

for c in passwort:
    if c.isdigit():
        result['Zahlen'] = True
    if c.isupper():
        result['Großbuchstabe'] = True
    if c.islower():
        result['Kleinbuchstabe'] = True
    if not c.isalnum():
        result['Symbol'] = True

conditions = 0

for key in result:
    if result[key]:
        conditions = conditions + 1

pwLength = len(passwort)

if pwLength <= 25 and conditions <= 2:
    print('Das Passwort ist weniger komplex!')
elif pwLength > 12 and conditions > 2:
    print('Das Passwort ist kurz aber komplex!')

print(result)
```



Eine Möglichkeit, den Code eleganter zu gestalten und Clean Code-Prinzipien zu beachten, ist die Verwendung von Funktionen und die Optimierung von Variablenamen und Kommentaren. Hier ist eine mögliche Lösung:

```
python Copy code

def check_password_complexity(password):
    """
    Checks the complexity of a password and returns a dictionary with the following keys:
    - 'has_numbers': True if password contains at least one digit, False otherwise
    - 'has_uppercase': True if password contains at least one uppercase letter, False otherwise
    - 'has_lowercase': True if password contains at least one lowercase letter, False otherwise
    - 'has_symbols': True if password contains at least one non-alphanumeric character, False otherwise
    """
    complexity = {
        'has_numbers': False,
        'has_uppercase': False,
        'has_lowercase': False,
        'has_symbols': False
    }

    for c in password:
        if c.isdigit():
            complexity['has_numbers'] = True
        elif c.isupper():
            complexity['has_uppercase'] = True
        elif c.islower():
            complexity['has_lowercase'] = True
        elif not c.isalnum():
            complexity['has_symbols'] = True

    return complexity

password = input('Gebe dein Passwort ein: ')
password_complexity = check_password_complexity(password)
conditions_met = sum(password_complexity.values())

if len(password) <= 25 and conditions_met <= 2:
    print('Das Passwort ist weniger komplex!')
elif len(password) > 12 and conditions_met > 2:
    print('Das Passwort ist kurz aber komplex!')

print(password_complexity)
```

In dieser Lösung wurden folgende Änderungen vorgenommen:

- Der Code zur Überprüfung der Passwortkomplexität wurde in eine Funktion `check_password_complexity` extrahiert, um den Code lesbarer und wiederverwendbarer zu machen.
- Die Namen der Variablen wurden aktualisiert, um ihre Bedeutung klarer auszudrücken (z.B. `result` -> `complexity`, `conditions` -> `conditions_met`).
- Ein Kommentar wurde hinzugefügt, um die Funktion `check_password_complexity` zu erklären.
- Der Code zur Überprüfung der Passwortkomplexität wurde aktualisiert, um `elif`-Anweisungen zu verwenden, um unnötige Auswertungen zu vermeiden.
- Die Anzahl der erfüllten Bedingungen wird durch die Funktion `sum` der Werte des Dictionaries `complexity` ermittelt, anstatt durch eine Schleife und eine separate Zählvariable.



Erzeuge mir noch zusätzlich die Inline kommentare.



Natürlich, hier ist die aktualisierte Lösung mit zusätzlichen Inline-Kommentaren:

```
python Copy code

def check_password_complexity(password):
    """
    Checks the complexity of a password and returns a dictionary with the following keys:
    - 'has_numbers': True if password contains at least one digit, False otherwise
    - 'has_uppercase': True if password contains at least one uppercase letter, False otherwise
    - 'has_lowercase': True if password contains at least one lowercase letter, False otherwise
    - 'has_symbols': True if password contains at least one non-alphanumeric character, False otherwise
    """
    complexity = {
        'has_numbers': False,
        'has_uppercase': False,
        'has_lowercase': False,
        'has_symbols': False
    }

    # Check each character in the password and update the complexity dictionary
    for c in password:
        if c.isdigit():
            complexity['has_numbers'] = True
        elif c.isupper():
            complexity['has_uppercase'] = True
        elif c.islower():
            complexity['has_lowercase'] = True
        elif not c.isalnum():
            complexity['has_symbols'] = True

    return complexity

# Prompt user for password input and check its complexity
password = input('Gebe dein Passwort ein: ')
password_complexity = check_password_complexity(password)
conditions_met = sum(password_complexity.values())

# Check if password meets complexity requirements and print appropriate message
if len(password) <= 25 and conditions_met <= 2:
    print('Das Passwort ist weniger komplex!') # Password is not complex enough
elif len(password) > 12 and conditions_met > 2:
    print('Das Passwort ist kurz aber komplex!') # Password is short but complex

# Print the complexity dictionary for debugging purposes
print(password_complexity)
```

In dieser Lösung wurden Inline-Kommentare hinzugefügt, um den Code weiter zu erklären. Jeder Kommentar beschreibt, was die zugehörigen Codezeilen tun oder warum sie wichtig sind.