Dokumentation zur Projektarbeit: Brute Force MD5 Hash Reversing

Author	Nick Sohl, Stefan Furrer
Date	@March 3, 2025
Fach	Lösungsalgorithmen und Programmieren
Status	Finished
Topic	Programming Python
URL	https://github.com/nick-sohl/teko_brute_force.git

Selbstständigkeitserklärung

Ich (wir) bestätige(n) hiermit, dass ich (wir) die vorstehende Projektarbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und sowohl wörtliche als auch sinngemäss verwendete Textteile, Grafiken oder Bilder kenntlich gemacht habe(n). Diese Arbeit ist in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt worden.

Glattbrugg, 3. März 2025

Initialisierung und Planung

Ausgangssituation

Innerhalb des Unternehmens kam es bei der Erstellung von Schlüsselwörter (Keywords) zu einem Fehler, wobei die Keywords mithilfe des MD5-Hash-Verfahrens verschlüsselt in einer Datenbank abgespeichert wurden. Diese Verschlüsselung ist nicht reversibel, weshalb die Keywords nicht einfach in ihren ursprünglichen Zustand zurück zu führen sind.

> 💡 **Hinweis**
>
> Die Keywords enthalten keine Umlaute. Beispiel: „aavc“ oder „FdUzg“

Auftrag

Der Auftrag der Projektarbeit ist es ein Tool in der Programmiersprache Python zu entwickeln, dass mithilfe der Brute-Fore-Methode die verschlüsselten Keywords von einem MD5-Hash in seinen Ursprungszustand zurück zu wandeln.

Projektarbeit_Brute-Force.pdf

Vorgaben

Benutzerabfrage

Der Nutzer soll im Programm folgende Eingaben machen können:

1. Hash-Wert
2. Anzahl Buchstaben.
3. Beinhaltet das Keyword Gross- und Kleinbuchstaben oder nur Kleinbuchstaben

Keyword-Länge

- Das Programm soll Keywords mit maximal 8 Buchstaben unterstützen

Zusatzinformationen

Für die Berechnung eines MD5-Hashes kann das `hash.py` file verwendet werden.

hash.py

> 💡 **Hinweis**
>
> Das Programm verwendet zur Erstellung der MD5-Hashes die Hashlib-Library.
>
> Die Dokumentation kann unter folgendem Link nachgeschaut werden: https://docs.python.org/3/library/hashlib.html

> 💡 **Wichtiger Hinweis in der Hashlib-Dokumentation zu MD5-Hashes**
>
> **Warning**
> Some algorithms have known hash collision weaknesses (including MD5 and SHA1). Refer to Attacks on cryptographic hash algorithms and the hashlib-seealso section at the end of this document.

Zielformulierung

- Entwicklung eines Programms in Python zur Umkehrung eines MD5-Hashes mittels Brute-Force.
- Implementierung einer Eingabemaske um eine User-Eingabe zu gewährleisten.
- Dokumentation der Vorgehensweise

Projektablauf

1. Verstehen der Aufgabenstellung und Recherche zu Brute-Force-Methoden.
2. Erarbeitung eines algorithmischen Lösungsansatzes.
3. Implementierung und Testphase.
4. Refactoring.
5. Dokumentation und Vorbereitung der Präsentation.

Theorie

Was ist eine Brute-Force-Attacke

> In cryptography, a **brute-force attack** consists of an attacker submitting many passwords or passphrases with the hope of eventually guessing correctly.

Ein Angreifer überprüft systematisch alle möglichen Passwörter bis das korrekte gefunden wurde.

Eine Brute-Force-Attacke wird meistens dann genutzt, wenn es nicht möglich ist, eine andere Schwachstelle in einem System auszunutzen

Erklärung von Wikipedia: Basic concept

Brute-force attacks work by calculating every possible combination that could make up a password and testing it to see if it is the correct password. As the password's length increases, the amount of time, on average, to find the correct password increases exponentially.

Sichere Keys gegen Brute-Force-Attacks

There is a physical argument that a 128-bit symmetric key is computationally secure against brute-force attack.

Quellen

https://en.wikipedia.org/wiki/Brute-force_attack

Was ist Hashing in der Cybersecurity?

> **Hashing** is a one-way mathematical function that turns data into a string of nondescript text that cannot be reversed or decoded

Komponenten von Hashing

1. Input Key
2. Hash Funktion
3. Hash Table

Input Key	Hash Function	Hash Table

The input data or message that is processed by the hash function and converted into the hash code.	A mathematical function that converts the input data, or key, into a unique hash value.	A data structure that stores data and maps keys to values.

Arten von Hashing

Es gibt eine Vielzahl von verschiedenen Hashing-Methoden.

Vier übliche Hashing-Algorithmen, die im Artikel näher beschrieben werden sind die folgenden:

1. LANMAN
2. NTLM
3. Script
4. Ehtash

Anwendungsfälle von Hashing in der Cybersicherheit

1. Passwort-Speicherung in einer Datenbank
2. Digitale Signaturen
3. Dateien- und Dokument-Management

Passwort-Speicherung

Bezogen auf unsere Projektarbeit wurde Hashing zur Speicherung von Passwörtern in einer Datenbank verwendet. Das ist aus folgendem Grund wichtig:

- Die Speicherung von Passwörter als Text in einer Datenbank ist sehr riskant. Bei einem Angriff auf ein System, wobei die Angreifer die Informationen in der Datenbank erhalten, könnten diese mithilfe der Passwörter weiter in das System eindringen um noch grösseren Schaden anzurichten.

Die Grenzen von Hashing

- Collision
- Performance
- Security Risks

In unserem Fall mit Bezug auf MD5-Hashing ist for allem Collision im Fokus: Eine Kollision liegt vor, wenn zwei oder mehr Eingaben denselben Hash-Wert ergeben.

> *However, MD5 is no longer considered secure against well-funded attackers, and it is susceptible to hash collisions.*

Wichtige Unterscheidung: Hashing vs. encryption

Hashing ist immer als einseitige Umwandlung von Daten gedacht. Der Hash-Wert ist eine eindeutige Textfolge, die nur dann entschlüsselt werden kann, wenn der Angreifer in der Lage ist, die Hash-Funktion zu stehlen oder zu erraten und dann die eingegebenen Daten zurück zu wandeln.

Die Verschlüsselung von Daten hingegen ist ein zweiseitiger Prozess. Bei der Verschlüsselung werden zwar auch kryptografische Algorithmen verwendet, um Text in ein verschlüsseltes Format umzuwandeln, aber es gibt auch einen entsprechenden Entschlüsselungsschlüssel, mit dem ein Benutzer die Daten entschlüsseln kann.

Quellen

https://www.crowdstrike.com/en-us/cybersecurity-101/data-protection/data-hashing/#:~:text=Hashing is a one-way,messages%2C and documents — secure.

https://www.geeksforgeeks.org/md5-hash-python/

MD5-Hashes in Python

Um einen MD5-Hash in Python zu generieren, kann man das `hashlib` Modul verwenden. Das Modul bringt die `md5()` Funktion mit sich, die Bytes entgegen nimmt. Deshalb muss man den `string` vor dem Hashing in Bytes umwandeln.

Beispiel:

```
import hashlib

def get_md5_of_string(input_string):
    return hashlib.md5(input_string.encode()).hexdigest()

# Example usage
print(get_md5_of_string("Hello, World!"))  # Output will be the MD5 hash of the st
```

Collision

Zwei verschiedene `strings` können denselben MD5-Hash haben.

Es wurden Schwachstellen gefunden, wobei zwei verschiedene Eingaben denselben MD5-Hashwert ergeben. Solche Schwachstellen machen MD5 ungeeignet für Funktionen wie SSL-Zertifikate oder Kryptografie, die auf einen eindeutigen Hash-Wert angewiesen sind.

Quellen

https://www.geeksforgeeks.org/md5-hash-python/

Realisierung

Python Modules

Das Import-System

Python-Code in einem Modul erhält Zugang zu einem anderen Modul indem man dieses importiert. Das `import` Statement ist der übliche Weg ein Modul zu importieren, jedoch nicht der einzige. Andere Funktionen wie `importlib.import_module()` oder das built-in `__import__()` können auch verwendet werden.

Ein Modul erstellen

Um ein Modul zu erstellen, speichert man einfach den gewünschten Code in einer Datei mit der Dateierweiterung `.py`

Ein Modul verwenden

Mit Hilfe dem `import` Statement kann man ein Modul (anderer Python-Code) “importieren und verwenden”

Python Modules in unserem Programm

```
import hashlib
import string
import itertools
import sys
```

Im Programm werden mehrere Module verwendet, um eine Revision eines MD5-Hashes zu erreichen. Die importierten Module gewährleisten die Funktionalität um MD5-Hashes zu berechnen, String-Manipulationen durchzuführen, Kombinationen zu berechnen und um mit dem Python-Interpreter zu interagieren.

Quellen

https://docs.python.org/3/reference/import.html

https://www.w3schools.com/python/python_modules.asp

https://www.digitalocean.com/community/tutorials/python-string-module

https://docs.python.org/3/library/sys.html

Use-Cases im Programm

`String` Module

```
ascii_lowercase_letters = string.ascii_lowercase
ascii_uppercase_letters = string.ascii_uppercase
ascii_all_letters = string.ascii_letters
```

- Das String-Module stellt die `ascii_letters` Konstante bereit. Diese beinhaltet alle upper sowie lower case letters.
- Die Buchstaben werden benötig um später im Code alle möglichen Kombinationen aus eine bestimmten Anzahl von Buchstaben zu generieren.

Quellen

https://www.geeksforgeeks.org/python-string-ascii_letters/

`Hashlib` Module

```
# Generate md5 hash
def calculate_md5(keyword):
    md5_hash = hashlib.md5()
    encoded_keyword = keyword.encode('utf-8')
    md5_hash.update(encoded_keyword)
    return md5_hash.hexdigest()
```

1. Die Funktion `calculate_md5()` nutzt das `hashlib` Modul, um einen MD5-Hash aus einem gegebenen Keyword zu generieren
2. Der import des `hashlib` Modules gibt uns die Möglichkeit die `md5()` Funktion zu verwenden
3. In der Funktion wird als erstes das `hashlib.md5()` Objekt initialisiert und dann einer Variablen zugewiesen. In diesem Fall der `md5_hash` Variable . Das Zuweisen

einer Variable macht den Code leserlicher, übersichtlicher und einfacher anpassbar

4. `Strings` in Python sind standardmässig Unicode. Hashing-Algorithmen benötigen jedoch bytes, daher wird das Keyword in das `utf-8` Format konvertiert. `encode('utf-8')` sorgt dafür, dass das Keyword als `bytes`-Objekt vorliegt.
5. Durch die `update()` Methode kann das zuvor initialisierte `hashlib.md5()` Objekt mit dem UTF-8-Kodierten Keyword aktualisiert werden. `update()` nimmt die Bytes als Eingabe und verarbeitet sie im Hashing-Algorithmus.
6. `hexdigest()` gibt den MD5-Hash als 32-stellige hexadezimale Zeichenkette zurück (z. B. `"5d41402abc4b2a76b9719d911017c592"` für das Wort `"hello"`).

Was ist ein “Digest”?

In der Kryptografie ist ein "Digest" ein kleines Datenstück, das ein grösseres Datenstück darstellt. Er wird mit einer kryptografischen Hash-Funktion erstellt, die eine Eingabe (z. B. eine Datei oder eine Nachricht) verarbeitet und eine Ausgabe fester Grösse (den Digest) erzeugt.

Im Idealfall ist ein Digest schnell zu berechnen, unumkehrbar.

Quellen

https://de.wikipedia.org/wiki/UTF-8

https://www.computerweekly.com/de/definition/Codierung-und-Decodierung

https://docs.python.org/3/library/hashlib.html#hash-algorithms

https://developer.mozilla.org/en-US/docs/Glossary/Digest

`Itertools` Module

```
combinations = itertools.product(ascii_all_letters, repeat=len_of_keyword) if uppe
```

- **`itertools.product()`** erzeugt alle möglichen Kombinationen einer bestimmten Zeichenmenge.
- Falls `uppercase == True`, werden **Gross- und Kleinbuchstaben** (`ascii_all_letters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"`) verwendet.

- Falls `uppercase == False`, werden **nur Kleinbuchstaben** (`ascii_lowercase_letters = "abcdefghijklmnopqrstuvwxyz"`) genutzt.
- `repeat=len_of_keyword` sorgt dafür, dass nur Kombinationen mit genau dieser Länge erstellt werden.

Quellen

https://www.geeksforgeeks.org/python-itertools/

https://docs.python.org/3/library/itertools.html#itertools.product

`sys` Module

```
for combination in combinations:
    count += 1
    if count > try_limit:
        print("Too many combinations. Please try again with a different keyword.")
        sys.exit()
```

- `sys.exit()` löst die “exception” `SystemExit` aus, beendet den Interpreter und bricht so das Programm ab. In Programm-Code passiert das, sobald eine gewisse condition erreicht ist.

Quellen

https://docs.python.org/3/library/sys.html#sys.exit

https://docs.python.org/3/library/exceptions.html#SystemExit

Funktionen

User Input

Mit einer Brute-Force-Attacke werden alle möglichen Kombinationen getestet, bis die richtige gefunden wurde. Diese Methode ist nur bis zu einer maximalen Anzahl von Zeichen nutzbar, da die Anzahl der möglichen Kombinationen exponentiell mit der Länge des Keywords steigt. Um die Möglichkeiten einzugrenzen, wird der Nutzer nach der Länge des Keywords gefragt und ob dieses Gross- oder Kleinbuchstaben enthält.

```
# User Input
def user_input():
```

```
    md5_hash = input("Enter MD5 hash: ")
    len_of_keyword = int(input("Enter the length of your keyword: "))

    while True:
        uppercase = input("Does your keyword include uppercase letters? Answer v
        if uppercase == 'yes':
            uppercase = True
            break
        elif uppercase == 'no':
            uppercase = False
            break
        else:
            print("Invalid answer! Please enter yes or no.")
            continue

    print(f"Your MD5-Hash is {md5_hash}")
    print(f"Your Keyword contains {len_of_keyword} letters")
    print(f"The number of possible combinations is {52 ** len_of_keyword if upper
    return md5_hash, len_of_keyword, uppercase
```

1. Abfrage des MD5-Hashes und der Länge des Keywords

Zuerst wird die `input()` Method genutzt um den User zur Eingabe des MD5-Hashes sowie der Länge des ursprünglichen Keywords aufzufordern. Den Datatype der Variablen `len_of_keyword` wird durch “Type Casting” in einen `integer` umgewandelt, da dieser Wert als Zahl benötigt wird. Die Eingaben des Users werden in den Variablen `md5_hash` sowie `len_of_keyword` gespeichert.

💡

`input()` method

Die Input-Methode wird genutzt um User-Input entgegenzunehmen. Standardmässig returned die Methode den User-Input als `string` .

Um einen anderen Datentypen vom User entgegenzunehmen, muss man der Input-Methode einen Typen zuweisen.

Type Casting in Python

Das ist eine Methode um den Datentypen einer Variable in einen anderen umzuwandeln.

Beispiel: `int(input("How old are you? "))`

Quellen

https://docs.python.org/3/library/functions.html#input

https://www.geeksforgeeks.org/python-input-function/

https://www.geeksforgeeks.org/type-casting-in-python/

2. Abfrage ob Grossbuchstaben enthalten sind

Im zweiten Schritt wird der User in einer `while` Schleife abgefragt, ob das ursprüngliche Keyword Grossbuchstaben beinhaltet. Diese Eingrenzung ist wichtig, da sich die Anzahl der Möglichkeiten exponentiell steigert, sollte das Keyword Grossbuchstaben enthalten. So ist das Programm effizienter wenn keine Grossbuchstaben enthalten sind, da es weniger Möglichkeiten prüfen muss.

Endlosschleife

Im Programm läuft der `While Loop` so lange, bis der User eine gültige Eingabe tätigt. Eine gültige Eingabe ist im Programm ein `String` ”yes” oder “no”. Der `While Loop` ist dabei endlos und seine Condition immer `True`. Erst durch eine gültige Eingabe wird der Loop durch ein `break` Statement verlassen.

Bedingte Überprüfung:

- Wenn die Eingabe `'yes'` ist, wird `uppercase` auf `True` gesetzt und die Schleife mit `break` verlassen.
- Wenn die Eingabe `'no'` ist, wird `uppercase` auf `False` gesetzt und die Schleife mit `break` verlassen.

- Bei anderen Eingaben wird eine Fehlermeldung angezeigt und die Schleife setzt sich fort.

3. Ausgabe der erhaltenen Informationen:

- **MD5-Hash:** Zeigt den eingegebenen MD5-Hash an.
- **Länge des Schlüsselworts:** Gibt die Anzahl der Buchstaben des Schlüsselworts aus.
- **Anzahl der möglichen Kombinationen:**
  - Wenn `uppercase` `True` ist, werden sowohl Gross- als auch Kleinbuchstaben berücksichtigt (52 Zeichen).
  - Wenn `uppercase` `False` ist, werden nur Kleinbuchstaben berücksichtigt (26 Zeichen).
  - Die Anzahl der Kombinationen wird als `52 ** len_of_keyword` bzw. `26 ** len_of_keyword` berechnet.

4. Rückgabe der gesammelten Daten:

Rückgabewert: Die Funktion gibt ein Tupel mit den Werten `md5_hash`, `len_of_keyword` und `uppercase` zurück, die in späteren Programmteilen verwendet werden können.

Ein Python Tuple ist eine durch Kommata getrennte Sammlung von Objekten.

Quellen

https://www.geeksforgeeks.org/tuples-in-python/

Python While Loop

Ein While Loop wird verwendet um Anweisungen im Code so lange auszuführen bis eine gewisse “Condition” erreicht ist.

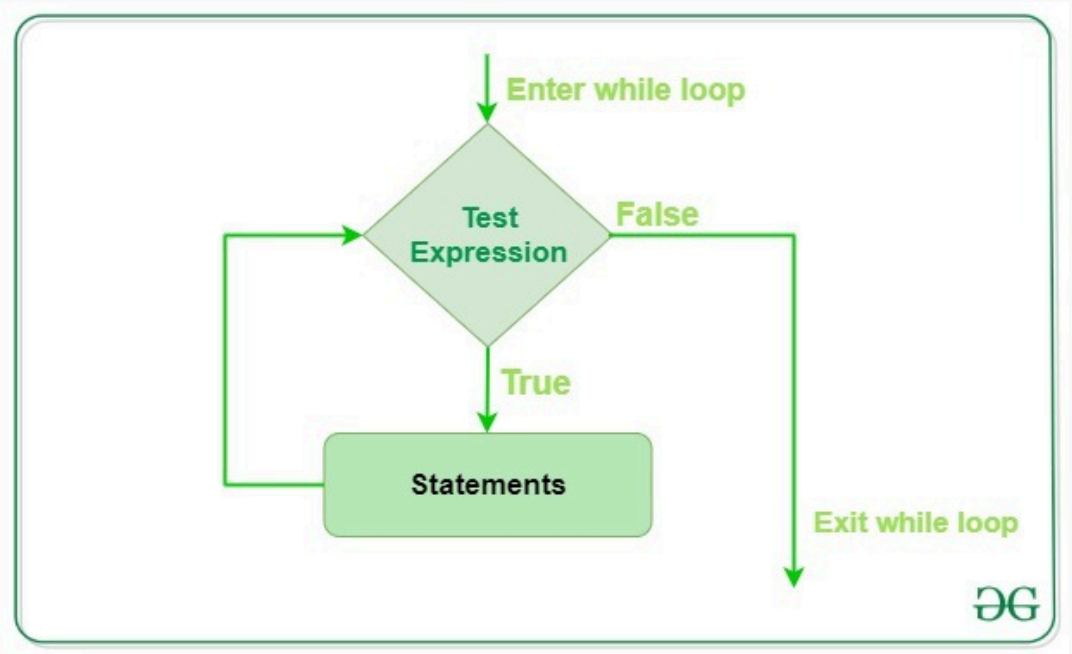
Wenn die Bedingung `False` wird, wird die Zeile unmittelbar nach der Schleife im Programm ausgeführt.

Beispiel

In diesem Beispiel ist die Condition des Loops so lange `True` solange die Variable `count` weniger als 5 ist.

```
# Beispiel für einen while loop
count = 0
while (count < 5):
    count = count + 1
    print("Hello")
```

While Loop Flowchart



Quellen

https://www.geeksforgeeks.org/python-while-loop/

Reverse MD5-Hash

Die Funktion `reverse_md5_hash()` versucht mit Hilfe der Brute-Force-Methode das ursprüngliche Keyword wieder herzustellen. Im Grundsatz generiert die Funktion alle möglichen Kombinationen von Buchstaben der gegebenen Länge, generiert aus jeder Kombination deren MD5-Hash und vergleicht diesen mit dem vom User bereitgestellten Hash.

1. Definition der Funktion

```
def reverse_md5_hash(hash, len_of_keyword, uppercase):
```

In der Definition hat die Funktion 3 Parameter:

- `hash`: Der zu entschlüsselnde MD5-Hash.
- `len_of_keyword`: Die Länge des ursprünglichen Schlüsselworts.
- `uppercase`: Ein boolescher Wert, der angibt, ob das Schlüsselwort Grossbuchstaben enthält.

2. Generieren aller möglicher Buchstabenkombinationen

```
combinations = itertools.product(ascii_all_letters, repeat=len_of_keyword) if uppe
```

`itertools.product()`

Erzeugt das kartesische Produkt der Eingabeiterablen. Das bedeutet einfach gesagt alle möglichen Buchstabenkombinationen aus den gegebenen Buchstaben mit der Länge `len_of_keyword` .

Bedingung `if uppercase`

- Wenn `uppercase` wahr ist, werden sowohl Klein- als auch Grossbuchstaben (`ascii_all_letters`) verwendet.
- Andernfalls werden nur Kleinbuchstaben (`ascii_lowercase_letters`) genutzt.

Quellen

https://de.serlo.org/mathe/2117/kartesisches-produkt

https://www.pythonlikeyoumeanit.com/Module2_EssentialsOfPython/Iterables.html

3. Initialisierung des Counters und Start des Brute-Forcings

```
count = 0
print("Brute forcing...")
```

- `count`: Zählt die Anzahl der getesteten Kombinationen.
- **Ausgabe**: Informiert den Benutzer über den Beginn des Brute-Force-Vorgangs.

4. Iteration über alle möglichen Kombinationen

```
for combination in combinations:
    count += 1
```

- **Schleife**: Durchläuft jede mögliche Buchstabenkombi.
- `count += 1`: Erhöht den Zähler bei jedem Versuch.

5. Limitation der Versuche

```
if count > try_limit:
    print("Too many combinations. Please try again with a different keyword.")
    sys.exit()
```

- `try_limit`: Eine zuvor definierte Obergrenze für die Anzahl der Versuche, um übermässig lange Laufzeiten zu vermeiden.
- `sys.exit()`: Beendet das Programm, wenn die Obergrenze erreicht ist.

7. Kombination zu einem `String` zusammenfügen

```
keyword = ''.join(combination)
```

- `''.join(combination)`: Verbindet die Tupel von Zeichen zu einem vollständigen String

Beisipiel:

```
>>> ",".join(["a", "b", "c"])
'a,b,c'
```

Quellen

https://stackoverflow.com/questions/1876191/what-exactly-does-the-join-method-do

8. Berechnung und Vergleich des MD5-Hashes

```
if calculate_md5(keyword) == hash:
    print(f"Your Keyword is: {keyword}")
    print(f"Tries to get: {count}")
    break
```

- `calculate_md5(keyword)`: Berechnet den MD5-Hash des aktuellen Schlüsselworts
- **Vergleich**: Wenn der berechnete Hash mit dem gegebenen Hash übereinstimmt, wird das Schlüsselwort gefunden.
- **Ausgabe**: Gibt das gefundene Schlüsselwort und die Anzahl der Versuche aus.
- `break`: Beendet die Schleife, da das Ziel erreicht wurde.

Main Funktion

1. Definition der Funktion `main()`

```
def main():
    hash, len_of_combination, uppercase = user_input()
    reverse_md5_hash(hash, len_of_combination, uppercase)
```

Die `main` Funktion ist der zentrale Einstiegspunkt des Programms

- Innerhalb dieser Funktion werden drei Variablen (`hash`, `len_of_combination`, `uppercase`) durch den Aufruf der Funktion `user_input()` zugewiesen.
- Anschliessend wird die Funktion `reverse_md5_hash()` mit diesen Variablen als Argumente aufgerufen.

Quellen

https://www.geeksforgeeks.org/deep-dive-into-parameters-and-arguments-in-python/

2. „if name == „main“

```
if __name__ == "__main__":
```

```
    main()
```

`if __name__ == "__main__"` wird verwendet um zu Überprüfen ob ein Skript direkt aufgerufen wird. Wenn das der Fall ist, wird eine Funktion aufgerufen.

Wenn das Programm, also das Skript direkt aufgerufen wird, definiert der Code dass die Funktion `main()` aufgerufen werden soll.

Warum ist das nützlich?

Wird das Skript als Modul in ein anderes Skript importiert, wird dieser Codeblock nicht automatisch ausgeführt. Dies ermöglicht es, Funktionen oder Klassen in einem Skript zu definieren und sie bei Bedarf in anderen Skripten zu verwenden, ohne dass der Code im `if __name__ == "__main__":` Block beim Import ausgeführt wird.

Beispiel

```
def main():
    print('Hello World')

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

Hier wird die Funktion `main()` nur aufgerufen, wenn das Skript direkt ausgeführt wird. Bei einem Import des Skripts in ein anderes Modul wird `main()` nicht automatisch aufgerufen.

Quellen

https://christoph-schmalfuss.de/blog/2022/04/30/python-if-__name__-__main__-einfach-erklaert-python-tutorial-fuer-anfaenger-deutsch/

Schlusswort

Im Rahmen unseres Projekts haben wir uns intensiv mit der Implementierung und Analyse von Hashing-Algorithmen beschäftigt. Durch die Entwicklung eines Programms zur Berechnung und Umkehrung von MD5-Hashes konnten wir unser Verständnis grundlegender Python-Konzepte sowie allgemeiner Programmierprinzipien vertiefen.

Während der Projektarbeit standen wir vor der Herausforderung das Programm effizienter zu machen. Zu Beginn wollten wir das im Projekt-Auftrag vorgegebene Ziel, ein gehashtes Keyword mit 8 Gross- sowie Kleinbuchstaben, entschlüsseln

können. Um so mehr wir uns mit dem Thema auseinander gesetzt haben, umso ferner wurde dieses Ziel. Unsere Recherche zeigte uns auf, wie ineffizient die Brute-Force-Methode bei langen Schlüsselwörtern ist. Dies zeigte uns die Wichtigkeit von langen Passwörtern auf. Dabei lernten wir auch, welche Algorithmen es gibt um Passwörter zu verschlüsseln, welche Algorithmen veraltet sind und welche modernen Ansätze es gibt, um Passwörter sicher zu hashen oder zu verschlüsseln.

Abschliessend möchten wir betonen, dass dieses Projekt nicht nur unser technisches Wissen erweitert hat, sondern uns auch die Bedeutung von Teamarbeit und effektiver Kommunikation vor Augen führte. Die gesammelten Erfahrungen werden uns in zukünftigen Projekten und beruflichen Herausforderungen – vor allem als Applikationsentwickler – von Nutzen sein.