Kryptoprogramme

mit Python

# Zielsetzung

Nach diesen Übungen sollst du

* verstehen, dass die Unicode-Tabelle jeder natürlichen Zahl ein Zeichen zuordnet und umgekehrt,
* verstehen, wie mit ord() der Index eines Zeichens in der Unicode-Tabelle erhalten werden kann,
* verstehen, wie mit chr() das Zeichen für einen bestimmten Index in der Unicode-Tabelle erhalten werden kann,
* dieses Wissen verwenden, um Verschiebeverschlüsselungen zu knacken,
* dieses Wissen verwenden, um eine monoalphabetische Verschlüsselung zu knacken.

# Unicode

## Unicode

Computer speichern alle Information in Bits – d.h. 0 und 1. Wenn du also ein Textdokument speicherst, wird auf der Festplatte eine Menge an 0 und 1 gespeichert. Jedes Zeichen wird dabei durch eine Binärzahl codiert. Welche Binärzahl welchem Zeichen entspricht, muss standardisiert werden. Die Reihenfolge der Zeichen muss also festgelegt werden (Reihenfolge => engl. *order*). Ein solcher Standard ist Unicode. Die Unicode-Tabelle codiert Symbole zu einer Dezimalzahl bzw. einer Binärzahl.

Zum Beispiel befindet sich das Zeichen «C» an der Position 67. Man spricht auch vom Index 67. Als Binärzahl wird das Zeichen «C» im Computer also mit 1000011 codiert.

In der folgenden Tabelle sind einige Zeichen beispielhaft aufgeführt.

|  |  |  |
| --- | --- | --- |
| Zeichen | Dezimalzahl (Index) | Binärzahl |
| Q | 81 | 01010001 |
| f | 102 | 01100110 |
| ~ | 126 | 01111110 |
| 😑 | 128529 | 11111011000010001 |

Details für Interessierte

Genauer betrachtet sind Python-Strings utf-8 codiert. utf-8 kodiert Zeichen mit variabler Byte-Anzahl. Dabei wird ein Unicode-Zeichen in 1 bis 4 Byte kodiert. Die Zahlen 0 bis 127, die dem ASCII-Zeichensatz entsprechen, werden in einem Byte kodiert, wobei das höchstwertige Bit stets 0 ist. Ein Byte besteht aus 8 Bit. Es können also $2^{7}$ Zeichen codiert werden. Mit Hilfe des achten Bits kann ein längeres Unicode-Zeichen eingeleitet werden, das sich auf 2, 3 oder 4 Byte erstreckt.

## Unicode mit Python

In Python lernen wir zwei Funktionen kennen, welche das Arbeiten mit Unicode vereinfachen.

* Mit *ord()* kann der Index eines Zeichens in der Unicode-Tabelle erhalten werden.
* Mit *chr()* kann das Zeichen beim entsprechenden Index in der Unicode-Tabelle erhalten werden.

## Auftrag: Unicode mit Python

Nach dem Ende der Übung sollten folgende Ideen vorhanden sein:

* Unicode als Zuordnung *int ⬄ Symbol*
* Funktion ord()
* Funktion chr()
* Indizes von a-z
* Indizes von A-Z

Bearbeite das Skript 1\_shift\_cipher\_intro.py und löse die dortigen Programmieraufgaben.

# Verschiebeverschlüsselung

## Verschiebeverschlüsselung

In der vorangegangenen Unterrichtseinheit haben wir gelernt, dass bei der Verschiebeverschlüsselung alle Buchstaben neu geordnet werden. Das Geheimalphabet ist verschoben gegenüber dem Klartextalphabet. Wir verschieben die Unicode-Tabelle.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Klartext | a | b | … | z | A | B | … | Z |
| Geheimtext |  |  |  | a | b | … | … | … |

Der Schlüssel bezeichnet die Anzahl verschobener Stellen in der Unicode-Tabelle (hin zu grösseren Indizes). Die neue Position wird also zu *ord(character) + key*. Durch die Verwendung eines grösseren Schlüsselraums wird das Dechiffrieren ohne Computer zu aufwendig.

## Auftrag: Verschiebeverschlüsselung – Schlüssel bestimmen

Gegeben ist eine Nachricht und deren Verschlüsselung. Gesucht ist der Verschiebeschlüssel. Als Hilfestellung ist die Entschlüsselungsfunktion vorgegeben. Du sollst also

* die encrypt() Funktion nachvollziehen,
* die encrypt () Funktion verwenden, um den Schlüssel zu bestimmen,
* mit Hilfe der encrypt () Funktion eine decrypt() Funktion schreiben,
* mit dem obigen Schlüssel einen Geheimtext entschlüsseln.
1. Bearbeite das Skript 2\_shift\_cipher\_unicode.py und löse die dortigen Programmieraufgaben.
2. Wird das Finden des Schlüssels schwieriger, wenn du mehr Unicode-Symbole d.h. grosse Verschiebungen zulässt?

## Auftrag: Verschiebeverschlüsselung knacken

Gegeben ist eine verschlüsselte Nachricht. Durch Ausprobieren soll der Schlüssel < 1000 ausfindig gemacht werden. Dieses Beispiel zeigt auf, dass die Entschlüsselung trivial ist, auch wenn keine Klartextnachricht bekannt ist. Du darfst den Code von der letzten Übung wiederverwenden.

1. Bearbeite das Skript 3\_shift\_cipher\_decrypt.py und löse die dortigen Programmieraufgaben.
2. Wird das Finden des Klartexts schwieriger, wenn du mehr Unicode-Symbole zulässt d.h. wenn der erlaubte Schlüssel sehr gross sein darf?
3. Wie könntest du das Suchen nach deutschen Wörtern automatisieren?

# Monoalphabetische Verschlüsselung

## Monoalphabetische Verschlüsselung

Bei der monoalphabetischen Verschlüsselung wird jedes Symbol durch ein beliebiges anderes ersetzt. Das Knacken von Hand ist aufwendig. Der Schlüsselraum ist wesentlich grösser als bei der Verschiebeverschlüsselung. Die Unicode-Tabelle beinhaltet im Moment mehr als 100'000 Zeichen (2020) und wächst stetig.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Klartext | a | b | … | z | A | B | … | Z |
| Geheimtext | 🤓 | α | … | 🥳 | % | # | … | / |

Die Verschlüsselung kann geknackt werden, indem man die Symbolhäufigkeit in der Geheimnachricht mit der Häufigkeit der Buchstaben in der deutschen Sprache vergleicht. Dazu findest du vorprogrammierte Hilfsfunktionen vor. Du musst nicht im Detail verstehen, wie diese funktionieren.

## Auftrag: monoalphabetische Verschlüsselung knacken

Gegeben ist ein String, der aus Unicode-Emojis besteht. Jedes Emoji muss also durch einen Buchstaben ersetzt werden. Die Gross- und Kleinschreibung muss beachtet werden. Als Hinweis wird die Tabelle rechts vorgegeben. Man beachte, dass in der Tabelle Gross- und Kleinbuchstaben nicht unterschieden werden, bei den Emojis schon. Die Aufgabe ist trotzdem lösbar, ist aber eine Herausforderung.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

1. Bearbeite das Skript 4\_monoalphabetic\_cipher.py und löse die dortigen Programmieraufgaben.
2. Wird das Finden des Klartexts schwieriger, wenn du mehr Unicode-Symbole zulässt? Wird die Aufgabe schwieriger, wenn du anstatt Emojis chinesische Zeichen vorfindest?