Praktikumsaufgabe 1

Verschlüsselung[1]

Formalia:

1. Zur Programmentwicklung **verwenden Sie als Modul-Namen (Datei-Namen) für das erste Praktikum den Namen** `p01NameVorname.py` **und halten sich strikt an die in der Aufgabe vorgegebenen Namen für die Varaiblen (Grund: Unit-Test s.u.)**.
2. Sie können mit dem in Moodle bereitgestellten Unit-Test `p01unit.py` überprüfen, ob Ihre eigene Lösung die richtigen Ergebnisse liefert:
   a) Im Modul `p01unit.py` passen Sie die 1-te Zeile gemäss Ihrem Namen an.
   b) Im Anaconda-Prompt können Sie den Unit-Test ausführen durch die Anweisung: `python p01unit.py`
3. **Erst wenn Ihre Lösung den Unit-Test besteht, können Sie Ihre Lösung in der Zoom-Sitzung (gleicher Link wie Online-Sprechstunde/Vorlesung) abnehmen lassen**.
4. Falls Sie selber Ihre Fehler im Unit-Test nicht beheben können, haben Sie in der Zoom-Sitzung zum Praktikum die Möglichkeit, die Fehler mit den anderen TeilnehmerInnen im Breakout-Raum zu besprechen und sich helfen zu lassen.
5. Falls Sie in der Breakout-Raum-Gruppe nicht weiter kommen rufen Sie Dozenten-Hilfe. Beachten Sie dass aufgrund der Anzahl der Breakout-Räume es etwas dauern kann, bis die *Dozenten-Hilfe* kommt. **Bringen Sie deshalb Geduld und Verständnis mit in die Praktikums-Zoom-Sitzung**.
6. **Die Abnahme Ihrer Praktikums-Lösung kann nur in der für Ihre Gruppe stattfindenden Zoom-Sitzung (gleicher Link wie Online-Sprechstunde/Vorlesung) erfolgen. Die Teilnahme an allen Praktikums-Zoom-Sitzungen für Ihre Gruppe ist Pflicht für den Erhalt der Bonus-Punkte.**
7. Sollten Sie in der vorgesehenen Zoom-Sitzung die Abnahme nicht schaffen haben Sie **am Ende** der nächsten Zoom-Sitzung für Ihre Gruppe die Möglichkeit eine zurückliegende Praktukums-Aufgabe abnehmen zu lassen.
8. Die Gruppeneinteilung und die Termine für die Zoom-Praktikums-Sitzungen werden über den Moodle-Kurs bekannt gegeben.

Bei der symmetrischen Verschlüsselung (private-key-Verschlüsselung) müssen Sender und Empfänger einer verschlüsselten Nachricht (Botschaft) einen gemeinsamen für die Außenwelt geheimen Schlüssel vereinbaren.
Ein sehr einfaches symmetrisches Verschlüsselungsverfahren beruht darauf, dass als Schlüssel ein Codewort verwendet wird.
Zum Verschlüsseln schreibt man das Codewort wiederholt unter die Nachricht (Botschaft, Klartext) und wendet buchstabenweise eine xor-Verknüpfung an, z.B. für die Nachricht ANGRIFF und das Codewort HUND erhält man:

Nachricht		A	N	G	R	I	F	F
Codewort	xor	H	U	N	D	H	U	N
Geheimtext		H	Z	L	S	P	R	I

[1] In Anlehnung an die Praktikums-Aufgaben der Kollegen Schöttl und Tasin aus dem SoSem 2018.

Um die xor-Operation auf Buchstaben anwenden zu können ordnet man den Buchstaben bijektiv eine Zahl zu, naheliegend ist $A \mapsto 0$, $B \mapsto 1$, $\ldots Z \mapsto 25$ und führt Bit-weise die xor-Verknüpfung aus:

Nachricht		0	13	6	17	8	5	5
Codewort	xor	7	20	13	3	7	20	13
Geheimtext		7	25	11	18	15	17	8

binär erhält man (**Frage 1) warum?, geben Sie eine Werte-Tabelle für die xor-Verknüpfung von je 1 Bit als Operanden an** und halten Sie diese als Vorbereitung in der Zoom-Sitzung bereit, z.B. als Dokumentations-Kommentar in Ihrem Code),

Nachricht		0b000	0b01101	0b0110	0b10001	0b1000	0b00101	0b0101
Codewort	xor	0b111	0b10100	0b1101	0b00011	0b0111	0b10100	0b1101
Geheimtext		0b111	0b11001	0b1011	0b10010	0b1111	0b10001	0b1000

Zum Entschlüsseln wendet man auf den Geheimtext ebenfalls eine xor-Verknüpfung mit demselben Codewort an und erhält die Nachricht (Klartext) zurück (**Frage 2) warum?**).

Geheimtext		0b111	0b11001	0b1011	0b10010	0b1111	0b10001	0b1000
Codewort	xor	0b111	0b10100	0b1101	0b00011	0b0111	0b10100	0b1101
Nachricht		0b000	0b01101	0b0110	0b10001	0b1000	0b00101	0b0101

Alle folgenden Aufgaben sind **ohne Kontrollstrukturen (ohne Fallunterscheidungen, ohne Schleifen außerhalb von Comprehensions)** zu lösen, es sollen nur **Slices** und **Listen-Comprehensions** verwendet werden.

1. **Codewort:** Im ersten Schritt wird aus einem Text, der in der Datei `praktikum1.txt` gespeichert ist, ein Codewort konstruiert.

   Laden Sie in Ihrem Python-Programm den Text mit Hilfe des Befehls

   ```
   text = open("praktikum1.txt","r").readlines()
   ```

   bzw., wenn Sie mit der Zeichen-Codierung Probleme haben (z.B. unter IOS oder Linux) verwenden Sie ggf.

   ```
   text = open("praktikum1.txt", "r", encoding="ISO-8859-1").readlines()
   ```

   in den Speicher (die Datei muss hierbei in dem selben Verzeichnis liegen, in dem auch Ihre Python Quell-Code-Datei liegt). Der open-Befehl öffnet eine Text-Datei und stellt ein Datei-Objekt zur Verfügung, die Methode `readlines()` liefert eine Liste aller Textzeilen als Liste von Strings. Man untersuche in der Python-Shell das Objekt auf das die Variable `text` verweist.

   Der Text enthält ein geheimes Codewort. Um es zu extrahieren, gehen Sie wie folgt vor:

   Das Codewort setzt sich zusammen aus dem Zeichen mit Index 2 der Zeile mit Index 5, den Zeichen mit Index 26-28 in der Zeile mit Index 2, den ersten drei Zeichen der Zeile mit Index 1 und den ersten drei Zeichen der letzten Zeile. Speichern Sie das Ergebnis in der Variablen `zwischencode`, nutzen Sie zur Ermittlung auch **Slices** und den Konkatenierungs-Operator `+` für Zeichenketten.

   Invertieren Sie nun die Reihenfolge und hängen Sie die sich ergebenden Zeichen fünfmal hintereinander. Schneiden Sie die letzten beiden Zeichen ab. Betrachten Sie aus diesem Wort jeden 8. Buchstaben, beginnend mit dem Buchstaben mit Index 8. Speichern Sie das Ergebnis in der Variablen `code_wort`. Zunächst kann man in Zwischenschritten vorgehen und sich die Ergebnisse in der Python-Shell ansehen und überprüfen.

   Man kann danach auch zur Übung versuchen, die Extraktion des Codeworts aus dem Zwischencode in einer einzigen Python-Zeile zu formulieren (lesbarer wird Ihr Code dadurch aber bestimmt nicht!).

2. **Einzeichencode:** Wir nähern uns der Aufgabe schrittweise und betrachten zunächst ein aus einem Zeichen bestehendes Codewort. Die zugehörige Nummer eines Buchstabens `ch` erhalten Sie mittels `ord(ch)-65`, den Buchstaben zur Nummer `nb` ergibt sich aus `chr(nb+65)` **Frage 3) Warum muss man ausgerechnet 65 subtrahieren bzw. addieren?**

   Können Sie in der Python-Shell sich die Hilfe zu den internen Funktionen `chr()` und `ord()` anzeigen lassen? **Frage 4) Formulieren Sie die Anweisungen die hierfür erforderlich sind.**

   Testen Sie z. B. `ord("D")` in der Python-Shell.

   Entschlüsseln Sie die Botschaft `botschaft1 = "RTFVQXSSE"` mit dem Code-Wort `code1 = "X"`.

   Gehen Sie zur Bearbeitung der Aufgabe schrittweise vor:

   a) Weisen Sie den Variablen `code1` und `botschaft1` die entsprechenden Strings zu.

   b) Ermitteln Sie nun die Variable `code_val1` und `botschaft_val1`. Die Variable `code_val1` soll den zum Codezeichen gehörigen Zahlencode, `botschaft_val1` eine Liste der zu den Buchstaben der Botschaft gehörenden Zahlencodes enthalten.
   Z.B.: Für die Botschaft `"ABC"` und das Codewort `"F"` sollte `code_val1` also `5` und `botschaft_val1` auf die Liste `[0,1,2]` verweisen.

   c) Ermitteln Sie die Liste der entschlüsselten Zahlencodes in `result_val1`. In Python ist der Operator `^` das Bit-weise xor.
   Z. B.: Für die Botschaft `"ABC"` und den Code `"F"` sollte `result_val1` also `[5,4,7]` sein. (**Frage 5) Können Sie diese Berechnung der Bit-weisen xor-Verknüpfung auf Papier (bzw. im Dokumentations-Kommentar) nachprüfen?**).

   d) Weisen Sie der Variablen `result1` den sich ergebenden String zu.
   Tipp: Durch `"".join(["A","B","C"])` kann eine Liste von Strings in einen String `"ABC"` kombiniert werden. Wie kann man sich mit der Hilfe in der Python-Shell die genaue Dokumentation zu `join()` ansehen? (**Frage 6) Erklären sie welches Ergebnis**
   `"!%$".join(["A", "B", "C"])`
   liefert.)

3. **Allgemeines Codewort (bestehend aus mehreren Buchstaben):**
   Entschlüsseln Sie die Botschaft `botschaft = "SLCVZCILAG"` mit dem von Ihnen gefundenen Codewort `code_wort` aus dem Aufgaben-Teil 1.). Erweitern Sie hierzu das obige Programm um ein allgemeines Code-Wort.

   Gehen Sie wieder schrittweise vor.

   a) Definieren Sie eine Variable `code_long`, die das Codewort entsprechend der Länge der Botschaft verlängert.

   b) Z. B. für die Botschaft `"ABCDE"` und dem Code-Wort `"FG"` sollten Sie ein `code_long` mit dem Wert `"FGFGF"` erhalten. Die Berechnung soll unabhängig von der Länge der Botschaft und des Codewortes funktionieren.

   c) Modifizieren Sie die Berechnung von `code_val1`, so dass sie nun `code_long` nutzt und speichern Sie nun das Ergebnis in einer Variablen `code_val`.

   d) Modifizieren Sie die Berechnung der Variablen `result_val1` entsprechend, so dass nun `botschaft_val` und `code_val` verwendet werden und speichern Sie das Ergebnis in einer Variablen `result_val`.
   Tipp:
   Um die Paarungen für die xor-Verknüpfung zu bilden hilft die interne `zip`-Funktion.

Frage 7) Wie läßt sich mit Hilfe von `zip` die Liste
`[("A", 1, "a"), ("B", 2, "b"), ("C", 3, "c")]`
aus den Listen

```
l1 = ["A", "B", "C"]
l2 = [1, 2, 3]
l3 = ["a", "b", "c"]
```

erstellen?

e) Anstelle `result1` speichern Sie nun das Endergebnis in der Variablen `result`.