Verschlüsselung

Allgemeines

- Verwenden Sie zur Programmentwicklung als Modulnamen (Dateinamen) für dieses Praktikum den Namen `p01NameVorname.py` und halten sich **strikt** an die in der Aufgabe **vorgegebenen Variablennamen** (Grund: Unit-Test s. u.)

  `Name` ist durch Ihren Nachnamen und `Vorname` durch Ihren Vornamen zu ersetzen. Der Dateiname sollte weiterhin keine Leerzeichen enthalten.

- Bevor Sie Ihre Lösung den Betreuern vorstellen, können Sie mit dem in moodle bereitgestellten Unit-Test `p01unit.py` überprüfen, ob Ihre Lösung die richtigen Ergebnisse liefert.
  - Passen Sie die erste Import-Anweisung im Modul `p01unit.py` gemäß Ihrem Namen an.
  - Im Anaconda-Prompt können Sie den Unit-Test in Ihrem Projektverzeichnis über folgende Anweisung ausführen:
    `python p01unit.py`

- Sollten in der Aufgabenstellung Fragen zu beantworten sein, dann beantworten Sie diese bitte in einem Dokumentationskommentar am Anfang Ihre Python-Datei, z. B.:

```
"""
      Antworten zu den in der Aufgabenstellung gestellten
      Fragen:
      Frage 1: Ihre Antwort
      Frage 2: Ihre Antwort
      ...........
"""
```

- In diesem Versuch sollen nur Comprehensions, Slices, Operatoren (Formeln) und Zuweisungen verwendet werden. **Kontrollstrukturen für Schleifen oder Fallunterscheidungen zur Ablaufsteuerung sind nicht erlaubt.**

1 Das Codewort

Gegeben ist ein Text, der in der Datei 'praktikum1.txt' gespeichert vorliegt. Öffnen Sie einen Python-Befehlsinterpreter und laden Sie den Text mit Hilfe des Befehls

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

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

- Der Code setzt sich zusammen aus dem 3. Zeichen (Index: 2) der 6. Zeile (Index: 5), dem 27.-29. Zeichen der 3. Zeile, den ersten drei Zeichen der zweiten Zeile und den ersten drei Zeichen der letzten Zeile. Speichern Sie das Ergebnis in der Variablen `zwischencode`, nutzen Sie zur Ermittlung auch Slices.
- Invertieren Sie nun die Reihenfolge und hängen Sie die sicher ergebenden Zeichen fünfmal hintereinander. Schneiden Sie die letzten beiden Zeichen ab. Betrachten Sie aus diesem Wort

jeden 8. Buchstaben, beginnend mit dem 9. (Index: 8). Speichern Sie das Ergebnis in der Variablen `code_wort`.
Schaffen Sie die Extraktion des Codeworts aus dem Zwischencode in einer Python-Zeile?

2 Die Entschlüsselung der Botschaft

Das Codewort können wir für die Ver- und Entschlüsselung von Daten nutzen. Eine einfache (und sehr unsichere) Möglichkeit ist es, den Buchstaben Zahlen zuzuordnen. Wir wählen 'A'->0, 'B'->1 etc. Die Verschlüsselung ergibt sich durch ein **xor** zwischen Botschaftszeichen und Codezeichen, dabei werden die Codezeichen durchgewechselt, also z. B. für die Botschaft: 'ABCDE', Code: 'FG':

'A' xor 'F' = 0 xor 5 = 5 = 'F'
'B' xor 'G' = 1 xor 6 = 7 = 'H'
'C' xor 'F' = 2 xor 5 = 7 = 'H' etc.

Das Entschlüsseln kann mit genau demselben Verfahren erfolgen:

'F' xor 'F' = 5 xor 5 = 0 = 'A'
'H' xor 'G' = 7 xor 6 = 1 = 'B'
'H' xor 'F' = 7 xor 5 = 2 = 'C' etc.

Frage 1: Wie lautet die Wertetabelle der **xor**-Verknüpfung für Ein-Bit-Argumente?

Frage 2: Warum kann man dieselbe Verknüpfung zum Entschlüsseln verwenden? Zeigen Sie das anhand einer Erweiterung der o. g. Wertetabelle.

2.1 Einzeichencode

Wir nähern uns der Aufgabe schrittweise und betrachten zunächst einen aus einem Zeichen bestehenden Code. Die zugehörige Nummer eines Buchstabens `ch` erhalten Sie mittels `ord(ch)-65`, den Buchstaben zur Nummer `nb` ergibt sich aus `chr(nb+65)`.[1] Testen Sie z. B. `ord('D')` im Befehlsinterpreter.
Frage 3: Warum wird 65 addiert bzw. subtrahiert?

Entschlüsseln Sie die Botschaft `'RTFVQXSSE'` (Variablenname: `botschaft1`) mit dem Code `'X'` (Variablenname: `code1`). Gehen Sie zur Bearbeitung der Aufgabe schrittweise vor:

- Weisen Sie den Variablen `code1` und `botschaft1` die entsprechenden Strings zu.
- 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 gehörenden Zahlencodes enthalten. Für die Botschaft 'ABC' und den Code 'F' sollte `code_val1` also 5 und `botschaft_val1` [0,1,2] sein.
- Ermitteln Sie die Liste der entschlüsselten Zahlencodes in `result_val1`. Für die Botschaft 'ABC' und den Code 'F' sollte `result_val1` also [5,4,7] sein.
- 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.
  Frage 4 : Wie lautet das Ergebnis für den Ausdruck `'!%$'.join(['A', 'B', 'C'])`

[1] `ord` steht für den Unicode des angegebenen Buchstabens (in UTF-8-Codierung), `chr` für das Zeichen zum angegebenen Unicode. 'A' hat z. B. den UTF-8-Unicode 65.

3 Allgemeiner Code

Entschlüsseln Sie die Botschaft `'SLCVZCILAG'` (Variablenname: `botschaft`) mit dem von Ihnen gefunden Code aus **Teil 1** mit Hilfe einer Zeile Python-Code. Hierzu erweitern Sie das obige Programm um allgemeine Codes.

Gehen Sie wieder schrittweise vor.

- Definieren Sie eine Variable `code_long`, die den Code entsprechend der Länge der Botschaft verlängert.
  Für die Botschaft 'ABCDE' und dem Code 'FG' sollten Sie ein `code_long` mit dem Wert 'FGFGF' erhalten. Die Berechnung soll unabhängig von der Länge der Botschaft und des Codes funktionieren.
- Modifizieren Sie die Berechnung von `code_val1`, so dass Sie `code_long` nutzt und das Ergebnis in `code_val` ablegt.
- Legen Sie die Ganzzahlen der Entschlüsselung in der Variable `result_val` ab. Nutzen Sie dazu die Vorgehensweise aus **Teil 2**.
- Ermitteln Sie den entschlüsselten String, legen diesen in der Variablen `result` ab und geben ihn auf dem Bildschirm aus.
  Tipp: Die `zip`-Funktion hilft!

Frage 5: Wie lautet der Ausdruck um mithilfe der `zip`-Funktion folgende Liste zu erzeugen:
`[('A', 1, 'a'), ('B', 2, 'b'), ('C', 3, 'c')]`