Unit 15

Dr. Günter Kolousek

21. Juli 2015

Lege wiederum ein Verzeichnis an. Nennes es `27_unit15`! In diesem Verzeichnis sollen alle Dateien der jeweiligen Einheit abgelegt werden.

1 Schulübungen

In diesem Beispielblock werden geübt:

- Arbeiten mit Listen
- Vertiefung der Verarbeitung der Kommandozeilenparameter
- Sequentielles Lesen und Schreiben von Textdateien

In dieser Einheit werden wir ein Programm `cipher` entwickeln, das eine Datei einliest, diese ent- oder verschlüsselt und wieder abspeichert. Im Vergleich zur schon programmierten Cäsar-Chiffre soll es jetzt allerdings etwas besser und sicherer sein. Die Vigènere-Chiffre muss her!

1. Schreibe ein Modul `cipher`, das eine Funktion `encrypt(data, key)` hat, die einen gegebenen String `data` mittels des übergebenen String `key` (den Schlüssel) verschlüsselt und die mit der Vigènere-Chiffre verschlüsselten Daten zurückliefert.

   Zur Information: Bei der Vigènere-Chiffre werden an sich 26 verschiedene, jeweils um 1 versetzte Alphabet untereinander aufgeschrieben. Dies sieht so aus:

```
Klar | a b c d e f g h i j k l m n o p q r s t u v w x y z
   1 | B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
   2 | C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
   3 | D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
     ...
  23 | W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
  24 | X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
  24 | Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
```

```
25 | Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
26 | A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Zur Verschlüsselung benötigt man einen Key, also ein Wort, das geheim gehalten wird.

Nehmen wir an, wir wollen den Text ”secret for ever” verschlüsseln und wählen dazu den Key ”zauberlehrling”.

Die Vorgangsweise ist folgendermaßen:

a) Für den ersten Buchstabe ”s” wird der erste Buchstabe des Key, nämlich das ”z” herangezogen. Dazu wird das ”s” in der ersten Zeile gesucht und das ”z” in der Spalte. Das ”z” befindet sich in der Zeile 25 und im Schnittpunkt der Zeile und der Spalte kann man das Geheimtextzeichen, nämlich ”R” ablesen.

b) Für den nächsten Buchstaben des Klartextes ”e” wird der nächste Buchstabe des Key genommen und genau so verfahren, wie im vorhergehenden Punkt.

c) Nach dem letzten Buchstaben des Key wird wieder mit dem ersten Buchstaben des Key begonnen.

Und so geht es:

a) Schreibe eine Funktion `init_tab()`, die eine derartige Tabelle anlegt und zurückliefert. D.h. die Tabelle sollte ungefähr folgendermaßen aussehen:

```
[["B", "C", "D", "E", "F",...],
 ["C", "D", "E", "F", "G"...],
 ...]]
```

Allerdings sollte die Tabelle **natürlich nicht** händisch angelegt werden! Wer findet die kürzeste Lösung?

Tipp für Spezialisten: Python kennt das Konzept der ”list comprehension”. Darunter versteht man eine besonders kompakte Form des Anlegens von Listen. Probiere aus:

```
>>> [x ** 2 for x in range(5)]
```

Was passiert da? Offensichtlich wird eine Liste angelegt, die Elemente der Form `x ** 2` haben, wobei `x` alle Werte im Bereich von 0 bis 4 durchläuft. Nicht schlecht, oder?

b) Schreibe eine Funktion `print_tab()`, die die Tabelle ausgibt:

```
Klar | a b c d e f g h i j k l m n o p q r s t u v w x y z
   1 | B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
   2 | C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
   3 | D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
   ...
```

Auch hier gilt: Wer findet eine elegante und kurze Lösung?

c) Rufe die Funktion `init_tab` auf. Das Ergebnis soll als globale Variable abgelegt werden.

Das ist diesmal nicht "Pfui", da die Tabelle nicht verändert wird.

d) Schreibe jetzt die Funktion `encrypt(data, key)`, die den übergebenen String `data` gemäß der Viginère-Chiffre mit dem Key `key` auf Basis der globalen Tabelle verschlüsselt und das Ergebnis zurückliefert!

Der Einfachheit halber schränken wir die Art der Daten etwas ein:

- Alle Buchstaben werden in Großbuchstaben gewandelt.
- Alle Nicht-Buchstaben werden weggeworfen (sind in Tabelle nicht enthalten).

Tipps:

i. Daten und Key zu Großbuchstaben konvertieren.

ii. Alle Zeichen der Daten durchgehen

iii. Index des jeweiligen Daten-Zeichens berechnen (Spalte)

iv. Index des jeweiligen Key-Zeichens berechnen (Zeile)

v. In Tabelle unter den beiden Indizes nachsehen und Zeichen an Ergebnis anhängen.

Damit sollte die Funktion zumindest folgendes Ergebnis zeigen:

```
>>> encrypt("Abc abc", "Zauber")
'ZBWBFT'
```

e) Schreibe jetzt noch die dazupassende Funktion `decrypt(data, key)`, die den übergebenen und verschlüsselten String `data` mit dem Key `key` entschlüsselt und zurückliefert::

```
>>> decrypt("ZBWBFT", "Zauber")
'ABCABC'
```

Funktioniert doch soweit ganz gut, wenn es auch nicht die sicherste Methode ist und auch die Anwendbarkeit auf Grund des kleinem Zeichenvorrates (Alphabet) nur eingeschränkt sinnvoll ist.

2. So, allerdings ist das was wir bis jetzt gemacht haben ziemlich umständlich. Es geht bedeutend einfacher!

   Schauen wir uns dazu nochmals die Tabelle genauer an: Nehmen wir einmal an, wir wollen das Klartextzeichen "H" mit dem Key-Zeichen "B" verschlüsseln. Schauen wir uns in die Tabelle: "I" ist das verschlüsselte Zeichen. Also genau das auf "H" folgende Zeichen. Wird an Stelle des Key-Zeichen "B" das Key-Zeichen "C" verwendet, dann wird das übernächste Zeichen, nämlich das "J" als Geheimzeichen verwendet.

   D.h. es wird immer dasjenige Zeichen genommen, dass genau so viele Zeichen hinter dem Zeichen steht, wie die Zeilennummer des Geheimtextzeichen in der Tabelle steht.

   Damit kann man sich die ganze Tabelle sparen und einfach eine Berechnung durchführen: `chr(ord(c) + ord(k) - 65)` (c = Klartextzeichen, k = Key-Zeichen).

   Super einfach, nicht wahr, aber was ist z.B. wenn das Klartextzeichen "H" mit dem dem Key-Zeichen "T" verknüpft werden soll: `ord("H") + ord("T") - 65` ergibt 91, wobei der letzte Großbuchstaben den Wert 90 hat. `chr(91)` ist das Zeichen "[". Hmm, aber wenn wir einfach 26 abziehen, wenn der Wert größer als 90 ist, dann könnte es gehen.

   Kommentiere die bisherigen Funktionen aus und ersetze diese durch die kürzeren Varianten!

3. Schreibe ein Hauptprogramm `cipher`, das die Verarbeitung folgender Kommandozeile durchführt:

   ```
   python3 cipher.py [--keep|-k] FILE
   ```

   Das bedeutet folgendes:

   - Angaben in eckigen Klammern sind optional. D.h. können aber müssen nicht vorkommen.
   - Die Teile, die mit einem Bindestrich `-` beginnen, werden als Optionen bezeichnet. Optionen steuern den Ablauf und die Funktion des aufgerufenen Programmes.

     Es gibt diese in der Kurzform, dann hat diese nur einen Bindestrich mit genau einem Buchstaben oder in einer Langform, dann hat diese zwei Bindestriche und mehrere Buchstaben (wie im obigen Fall).
   - Sind Teile mit einem `|` getrennt bedeutet dies, dass entweder der eine oder der andere Teil vorkommen darf.
   - Teile, die überhaupt nicht markiert sind, sind verpflichtend anzugeben und werden Parameter bzw. Argumente genannt.

Für uns bedeutet das:

- `FILE` muss angegeben werden.
  - Hat der angegebene Dateiname eine Erweiterung von `.cpt`, dann wird angenommen, dass diese verschlüsselt ist und wird daher entschlüsselt.

    Die entschlüsselten Daten werden in einer Datei abgelegt, die den gleichen Namen hat wie die verschlüsselte Datei und die verschlüsselte Datei wird gelöscht!
  - Hat der angegebene Dateiname keine Erweiterung von `.cpt` dann wird angenommen, dass diese zu verschlüsseln ist.

    Die verschlüsselten Daten werden in einer Datei abgelegt, die den gleichen Namen hat wie die Klartext-Datei, aber es wird die Erweiterung `.cpt` hinten am Namen angehängt. Die ursprüngliche Datei wird gelöscht!
- Die Option `--keep` oder `-k` kann, muss aber nicht angegeben werden. Ist diese Option vorhanden, dann soll die verschlüsselte Datei (bei einer Entschlüsselung) bzw. die entschlüsselte Datei (bei einer Verschlüsselung) nicht gelöscht werden.

Damit lassen wir es vorerst bewenden und programmieren die Benutzerschnittstelle. D.h.: Überprüfe, ob Kommandozeile korrekt ist und gib bei fehlerhafter Kommandozeile eine möglichst hilfreiche Fehlermeldung und eine "usage"-Meldung aus. Die eigentliche Funktionalität soll noch nicht programmiert werden!

4. Implementiere nun die Funktionalität.

   a) Zuerst ist die Kommandozeile zu überprüfen und eine "usage" Meldung auszugeben, wenn die Kommandozeile nicht korrekt ist.

   b) Dann wird der Schlüssel abgefragt. Verwende wieder das Modul `getpass`!

   c) Im Anschluss wird der Inhalt der angegebenen Datei eingelesen und die Datei wieder geschlossen.

   d) Danach ist in Abhängigkeit des Dateinamens der Inhalt entweder zu verschlüsseln und zu entschlüsseln.

   e) Jetzt kann der modifizierte Inhalt wieder in eine Datei geschrieben werden. Der Name hängt davon ab, ob ein entschlüsselter Inhalt oder ein verschlüsselter Inhalt gespeichert werden muss.

      Auch dazu muss die Datei geöffnet, der Inhalt geschrieben und die Datei wieder geschlossen werden.

   f) Ist kein Fehler aufgetreten und weder die Option `--keep` noch `-k` angegeben worden, dann ist die ursprüngliche Datei zu löschen.

Hinweise:

- `open(name, mode)` öffnet eine Datei `name` im Modus `mode` und liefert ein `file`-Objekt zurück.

  `mode`: `"r"` bedeutet lesen und `"w"` bedeutet schreiben.

- Die Methode `read` eines `file`-Objektes liest die gesamte Datei ein und liefert einen String zurück.
- Die Methode `close` eines `file`-Objektes schließt die Datei wieder.
- Die Methode `write(content)` eines `file`-Objektes schreibt den String `content` in die Datei.
- Die Funktion `remove(name)` im Modul `os` löscht die angegebene Datei mit dem Namen `name`.