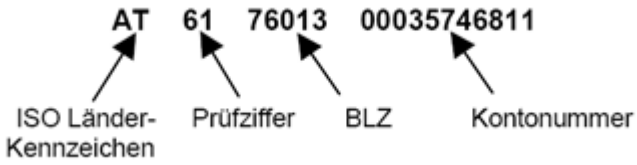
Übung 20

1) Schreiben Sie ein Programm **iban.py** zur Gültigkeitsüberprüfung einer österreichischen IBAN (=“International Bank Account Number“). Diese hat folgenden Aufbau:
   - Sie ist immer 20 Zeichen lang.
   - Die ersten zwei Zeichen sind der Ländercode „AT“.
   - Die nächsten zwei Zeichen sind eine Prüfziffer.
   - Danach kommt eine fünfstellige Bankleitzahl.
   - Abschließend die individuelle Kontonummer des Kunden (11 Stellen)



Um eine IBAN auf ihre Gültigkeit zu überprüfen, kann man folgende Schritte anwenden:

1. Alle Leerzeichen entfernen
2. Die ersten 4 Zeichen ans Ende stellen
3. Die Buchstaben (AT) in Zahlen umwandeln → A = 10, B = 11, C = 12, … , Z = 35
4. Diese große Zahl muss jetzt % 97 gerechnet werden
5. Ist das Ergebnis 1? → IBAN gültig. Ist das Ergebnis nicht 1? → IBAN ungültig.

Ein Beispiel:

IBAN gegeben → `AT61 1904 3002 3457 3201`

IBAN umgestellt → `1904300234573201AT61`

Ländercode in Zahlen umwandeln → `190430023457320110296`1

Diese große Zahl % 97 rechnen → `190430023457320110296`1 `% 97 = 1`, also gültige IBAN

Überprüfen Sie die eingegebene IBAN mit Hilfe einer selbstprogrammierten Funktion `checkIBAN(iban)`, die je nachdem `True` oder `False` zurückliefert. Sollte die IBAN falsch sein, geben Sie zusätzlich eine kurze Meldung aus, wo der Fehler liegt, z.B. bei falscher IBAN-Länge, falschem Ländercode, …

2) Erweitern Sie die vorherige Aufgabe in einer Datei **iban2.py** um die Überprüfung der IBAN aus Deutschland und der Schweiz zu ermöglichen. Somit kann die gesamte DACH-Region auf Gültigkeit überprüft werden.

3) Schreiben Sie ein Programm **text.py** das aus einem Textfile Sätze einliest und eine Textanalyse / Statistik darüber erstellt. Der Name - falls File im aktuellen Verzeichnis - oder der Pfad der Textdatei kann dabei vom User über die Konsole eingegeben werden.

   Ausgegeben wird:

   - Die Anzahl der Zeilen
   - Die Anzahl der Wörter
   - Die Anzahl jedes Buchstaben

   Nehmen wir an, ein Textfile `Test.txt` halt folgenden Inhalt:

```
Mimi im Garten.
Hugo im Haus.
```

   Dann würde das Programm wie folgt funktionieren:

```
Bitte Dateiname oder Pfad eingeben: Test.txt

--- Textanalyse ---

Anzahl der Zeilen: 2

Anzahl der Wörter: 6

Anzahl jedes Buchstabens:
a: 2
e: 1
g: 2
h: 2
i: 4
m: 4
n: 1
o: 1
r: 1
s: 1
t: 1
u: 2
```

4) Schreiben Sie ein Programm **vigenere.py** zum Ver- und Entschlüsseln von Nachrichten mittels Vigenère-Chiffre. Der Einfachheit halber gehen wir nur von Großbuchstaben aus und nehmen ein Vigenère -Quadrat zum Ver- und Entschlüsseln, welches wir selbst im Code erzeugen. Das Programm enthält drei Funktionen:

- `buildSquare()` → baut das Vigenère -Quadrat als **zweidimensionale Liste** auf, die Liste an sich wird global gespeichert
- `encrypt(msg, key)` → verschlüsselt `msg` mit `key`, liefert den Geheimtext als `str` zurück
- `decrypt(msg, key)` → entschlüsselt `msg` mit `key`, liefert den Klartext als `str` zurück

Zur Wiederholung: Die Vigenère-Chiffre ist ein polyalphabetisches Substitutionsverfahren → Dabei werden Buchstaben durch andere ersetzt. Im Gegensatz zur z.B. Cäsar- Chiffre allerdings nicht immer durch dieselben Buchstaben.

Beispiel:

Schlüssel: CODE

Klartext: MEIN FREUND IST HUGO

Geheimtext: OSLR HFHYPR LWV VXKQ

Klartext	M	E	I	N	F	R	E	U	N	D	I	S	T	H	U	G	O
Schlüssel	C	O	D	E	C	O	D	E	C	O	D	E	C	O	D	E	C
Geheimtext	O	S	L	R	H	F	H	Y	P	R	L	W	V	V	X	K	Q

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

5) Erweitern Sie die vorherige Aufgabe in einer Datei **vigenere2.py** um die Funktionen `encryptFile(filename, key)` & `decryptFile(filename, key)`. Mit Hilfe dieser Funktionen sollen nun ganze Textfiles mittels Vigenère-Chiffre ver- und entschlüsselt werden können.

6) Schreiben Sie ein Programm **isbn.py** zur Gültigkeitsüberprüfung einer ISBN-13. Diese ist eine Erweiterung der älteren ISBN-10. Schreiben Sie dazu eine Funktion `checkISBN(isbn)`, die `True` zurückliefert, wenn die ISBN-13 syntaktisch korrekt ist, ansonsten `False`.

Wie funktioniert die Prüfziffer in einer ISBN-13?

Beispiel einer ISBN-13: 9783406548406 →
Die letzte Ziffer ist die Prüfziffer, also in diesem Fall: 6.

1. Die ersten 12 Ziffern werden nun abwechselnd mit 1 und 3 multipliziert und die einzelnen Produkte addiert → 9 x 1 + 7 x 3 + 8 x 1 + 3 x 3 …
   Das ergibt in diesem Beispiel: 104

2. Danach wird der Rest bei der Division durch 10 berechnet → 104 % 10 = 4

3. Dieser Rest wird nun von 10 abgezogen → 10 – 4 = 6

4. Das Resultat dieser Differenz ist eine Prüfziffer → 6