1. Übung im Hefter:

```
def istIn(zeichen,text):
     for i in range(len(text)):
         if zeichen==text[i]:
             return True
     return False
```

   Was bewirkt dieses Programm? Ermittle die Ausgabe zu folgenden Anweisungen durch einen Schreibtischtest und verallgemeinere:
   - **`print istIn("A","HALLO")`**
   - **`print istIn("B","AABBCCDEFFF")`**
   - **`print istIn("C","HALLO")`**

2. Übung am Rechner:

```
def erzeugeSchluessel(schluesselwort):
    schluessel=""
    schluesselwort = schluesselwort.upper()
    for i in range(len(schluesselwort)):
        if istIn(schluesselwort[i], schluessel):
            pass
        else:
            schluessel=schluessel+schluesselwort[i]
    for i in range(25,-1,-1):
        zeichen=chr(i+65)
        if istIn(zeichen,schluessel):
            pass
        else:
            schluessel = schluessel + zeichen
    return schluessel
```

   Öffne die Datei **`zyx.py`** und teste sie mit den folgenden Funktionsaufrufen.

   a. Notiere die Ergebnisse.
      - **`print erzeugeSchluessel("Hallo")`**
      - **`print erzeugeSchluessel("AABBCCDEFFF")`**

      Teste das Programm mit mind. zwei eigenen Schlüsselwörtern. Notiere jeweils Schlüsselwort und Ergebnis

   b. Beschreibe die Wirkung des Programms:
      - Wie lang ist jeder erzeugte Schlüssel?
      - Jeder Schlüssel besteht aus zwei Teilen. Kennzeichne diese in deinen Ergebnissen.
      - Ordne jedem der beiden Teile einen Quelltextabschnitt zu.

   c. (für Schnelle) Vervollständige die Funktion **`positionVon`**.
      Diese Funktion ermittelt die Position des Zeichens **`zeichen`** im Text und gibt sie zurück. Kommt das Zeichen nicht vor, soll die Textlänge zurückgegeben werden.

      ```
      def positionVon(zeichen,text):
      ```

3. Der erzeugte Schlüssel kann nun zur Substitution der Klartextzeichen verwendet werden, dazu denkt man sich Alphabet und Schlüssel zeichenweise untereinander geschrieben. Erzeuge zum Schlüsselwort „Platon“ den Schlüssel und verschlüssele die folgenden Klartexte:

   ABCABCABC **DEFDEFDEF** **GHIGHIGHI**

4.
   a. Übernimm die Funktionen aus **zyx.py** in deine **krypto.py**.
   b. Implementiere den im Quelltext dargestellten Algorithmus.

Hinweis:
```
zeichen = chr(ASCII)
ASCII = ord(zeichen)
```

substitution (klartext, schluesselwort)
- schluessel <= erzeugeSchluessel(schluesselwort)
- klartext in Großbuchstaben umwandeln
- leere Zeichenkette geheimtext anlegen
- wiederhole für i=0 solange i < Klartextlänge
  - zeichen <= i-tes Klartextzeichen
  - index <= ASCII von zeichen - 65
  - geheimbuchstabe <= schluessel an der Stelle index
  - Geheimbuchstaben an Geheimtext anhängen
- Rückgabe: geheimtext

   c. Teste dein Programm mit den folgenden aktuellen Parametern. Notiere die Ergebnisse.
   - **print substitution("ABCABCABC","Platon")**

     _____
   - **print substitution("DEFDEFDEF","Platon")**

     _____
   - **print substitution("GHIGHIGHI","Platon")**

     _____
   - **print substitution("ABCDEFGHIJKLMNOPQRSTUVWXYZ","Notalp")**

     _____

   d. Wie verschlüsselt dieses Programm? Beschreibe die Vorgehensweise an selbstgewählten Text- und Schlüsselbeispielen.

5. (für Schnelle)
   Entwirf ein Programm zur Entschlüsselung der verschlüsselten Texte und teste es.

```
def deSubst(geheimtext,schluesselwort):
```