Übungsblatt Schleifen

Aufgabe 1: Summe

Berechnen Sie mit Hilfe einer `for`-Schleife die Summe $\sum_{i=3}^{20} i^2$.

Aufgabe 2: Quadrat

Schreiben Sie eine Funktion `quadrat`, die eine naturliche Zahl übergeben bekommt und das Quadrat bis zu der gegebenen Zahl inklusive, als Menge zurückgibt.

Aufgabe 3: Dreieck

Schreiben Sie eine Funktion `triangle`, die ein Dreieck auf der Konsole wie folgt ausgibt. Über das Argument `h` soll die Höhe des Dreieck übergeben werden. Hier ein Beispiel:

```
>>> triangle(7)
*******
******
*****
****
***
**
*
```

Aufgabe 4: Vokale

Schreiben Sie eine Funktion `count_vowels`, die einen String als Argument erhält. Die Funktion zählt wie viele Vokale in dem String vorkommen und gibt die Anzahl an Vokalen zurück.

Hinweis: Zu Vokalen zählen wir 'A', 'E', 'I', 'O', 'U', 'Ä', 'Ö' und 'Ü' als Groß- oder Kleinbuchstaben.

Beispiel:

```
>>> count_vowels('Dieser Satz enthält zehn Vokale.')
10
```

Aufgabe 5: Konsonanten

Schreiben Sie eine Funktion `consonants`, die einen String als Argument erhält. Die Funktion erstellt einen neuen String aus dem übergebenen String, aber ohne Vokale. Der neue String wird zurück gegeben.

Hinweis: Zu Vokalen zählen wir 'A', 'E', 'I', 'O', 'U', 'Ä', 'Ö' und 'Ü' als Groß- oder Kleinbuchstaben. Diese sollen quasi entfernt werden.

Beispiel:
```
>>> consonants('Satz nur mit Konsonanten lesbar?')
'Stz nr mt Knsnntn lsbr?'
```

Aufgabe 6: ISBN

Die Internationale Standardbuchnummer ist eine eindeutige Nummer zur Kennzeichnung von Büchern. Bei der ISBN-10 gibt es 10 Ziffern, wobei die letzte Ziffer eine Prüfziffer ist. Auch gibt es manchmal Striche in der ISBN-10, die allerdings nur der Leserlichkeit dienen soll. Beispielsweise wäre eine gültige ISBN: "3-446-44665-6"
```
>>> isbn = '3-446-44665-6'
```
Schreiben Sie eine Funktion, die `True` zurück gibt, wenn die ISBN gültig ist, also die Prüfziffer korrekt und `False`, falls nicht. Die Prüfziffer d_{10} berechnet sich aus:

$$d_{10} = (1 \cdot d_1 + 2 \cdot d_2 + 3 \cdot d_3 + 4 \cdot d_4 + 5 \cdot d_5 + 6 \cdot d_6 + 7 \cdot d_7 + 8 \cdot d_8 + 9 \cdot d_9) \bmod 11$$

Beispiel:
```
>>> check_isbn('3-446-44665-6')
True
```
Die Funktion soll ohne Striche natürlich dasselbe zurück geben:
```
>>> check_isbn('3446446656')
True
```
Und als Gegenbeispiel, wenn wir nur die Prüfziffer ändern, muss die ISBN natürlich ungültig sein:
```
>>> check_isbn('3446446657')
False
```
Achtung: Als Prüfziffer kann auch eine 10 vorkommen, was als `'X'` geschrieben wird: "3-598-21507-X"

Hinweis: Sie müssen sich in dieser Aufgabe zunächst auf die Vorverarbeitung des Strings und dann um die Berechnung der Prüfzimmer kümmern. Ob die ISBN wirklich genau 10 Ziffern hat oder andere Probleme sind nicht Teil dieser Aufgabenstellung.

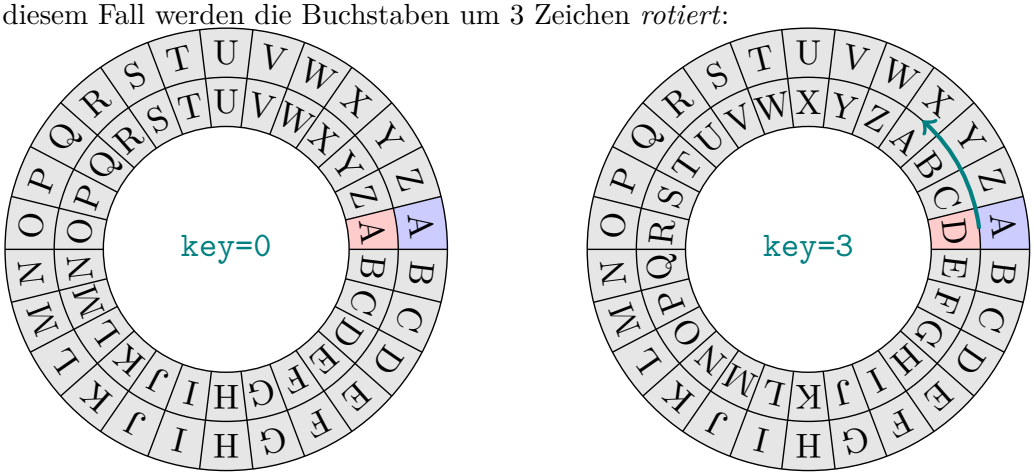
Aufgabe 7: Caesar

Schreiben Sie eine Funktion `caesar_code` zum Verschlüsseln eines Textes. Die Funktion erhält drei Argumente:

- `text` ist der Text, der kodiert werden soll.
- `key` ist die Zahl mit der die Buchstaben verschoben werden sollen.
- `alphabet` ist das Alphabet, das die Reihenfolge definiert. Als Default enthält es einfach die 26 Buchstaben `"abcdefghijklmnopqrstuvwxyz"`.

Der Benutzer soll einen beliebigen Klartext in das Argument `text` übergeben. Wandeln Sie den Klartext zunächst in Kleinbuchstaben um. Die Buchstaben sollen um `key` Zeichen

im Alphabet periodisch verschoben werden. Als Beispiel, betrachten wir den Fall `key=3`. In diesem Fall werden die Buchstaben um 3 Zeichen *rotiert*:



- a → d
- b → e
- c → f
- ...
- w → z
- x → a
- y → b
- z → c

Erstellen Sie so Zeichen für Zeichen einen Chiffretext (engl. cipher text). Wenn ein Zeichen nicht im Alphabet ist, soll es einfach unkodiert hinzugefügt werden. Das bewahrt Leerzeichen, Satzzeichen und Sonderzeichen. Geben Sie am Ende den Chiffretext aus der Funktion zurück geben.

Hier Ihr Start-Code:

```
def caesar_code(text, key, alphabet="abcdefghijklmnopqrstuvwxyz"):
    coded_text = ''


    # TODO: Hier Caeser-Kodierung implementieren


    return coded_text
```

Überlegen Sie auch was mit einer negativen Zahl für `key` passiert!

Beispiele:

```
>>> caesar_code('AKIS', 3)
'dnlv'
>>> caesar_code('Dies ist ein Satz.', 1)
'ejft jtu fjo tbua.'
>>> caesar_code('python', 6)
'veznut'
```

```
>>> caesar_code('veznut', -6)
'python'
```

Bonus: Bewahren Sie Groß- und Kleinschreibung!

Aufgabe 8: Ziffern

Schreiben Sie eine Funktion `num_digits`, die die Anzahl Ziffern einer natürlichen Zahl zählt und zurück gibt. Zum Beispiel hat die Zahl 36793 5 Ziffern.

Hinweis: Überlegen Sie was passiert, wenn Sie die Zahl mehrmals durch 10 teilen.

Aufgabe 9: Klammern

Schreiben Sie eine Funktion `parenthesis_content`, die einen String als Argument erhält und den Inhalt des ersten äußersten Klammernpaares zurück gibt. Hier ein Beispiel:

```
>>> text = 'Ein erster (einfacher) Test!'
>>> parenthesis_content(text)
'einfacher'
>>> text = 'Ein zweiter (auch einfacher) Test (weil nur das erste Paar zählt)!
    '
>>> parenthesis_content(text)
'auch einfacher'
>>> text = 'Ein dritter (komplizierterer (weil verschachtelter)) Test!'
>>> parenthesis_content(text)
'komplizierterer (weil verschachtelter)'
```

Hinweis: Sie können bei einer '(' einen Zähler inkrementieren und bei einer ')' dekrementieren. Bei welchem Zählerstand können Sie die Schleife abbrechen?

Bonus: Geben Sie mit `print` eine einfache Warnung aus, wenn mehr offene als geschlossene Klammern oder umgekehrt vorhanden sind.