Listen

- Quadratzahlen korrigieren
  Gegeben sei folgende Liste: `python quadrat_zahlen = [1, 4, 9, 16, 26, 36, 49, 64, 81, 100]`
  - Finden Sie die fehlerhafte Quadratzahl und ersetzen Sie sie mit der korrekten Zahl.
  - Fügen Sie noch eine weitere Quadratzahl hinzu
- Einkaufliste:
  - Erstellen Sie eine Einkaufsliste bestehend aus (maximal) 10 Produkten, sowie eine Liste mit den dazugehörigen Preisen
    (Oder benutzen Sie die hier:

    ```
    einkaufs_liste = ["apfel", "milch", "banane", "joghurt", "kaese"]
    preise = [0.40, 0.59, 0.30, 1.20, 2.30]
    ```

)
  - Iterieren Sie mithilfe einer for-Schleife über alle Elemente in *einkaufs_liste*
  - Benutzen Sie enumerate, um die Elemente in *einkaufs_liste* durchzuzählen
  - Iterieren Sie nun über Gegenstand und Preis, so dass Sie als Ausgabe bekommen:

    ```
    Gegenstand_1, Preis_1
    Gegenstand_2, Preis_2
    ...
    Gegenstand_10, Preis_10
    ```

    (zip!)
  - Lassen Sie sich nun nur jeden zweiten Gegenstand ausgeben
- Erstellen Sie eine Liste der Quadratzahlen von 1 bis 400
- Gegeben ist eine Liste von Strings

  ```
  string_liste = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10"]
  ```

  Machen Sie daraus eine Liste von Integers!
- In https://github.com/gkabbe/Python-Kurs2015/wiki/Woche-2---Datentypen#ein-praktisches-beispiel-1 wurde eine xyz-Datei eingelesen und in eine Atomnamen-Liste und eine Positionsliste gespeichert. Benutzen Sie die beiden Listen um herauszufinden:

  - Wieviel Wasserstoffatome (Buchstabe H) nicht weiter als 2.0 Angström (Längeneinheit, entspricht 10^-10 m) von einem Sauerstoffatom (O) entfernt sind.
  - Wieviele Sauerstoffatome in der Nähe jedes Phosphoratoms (P) sitzen

- Daten einlesen:
  - Schreiben Sie eine Datei "Daten", in die Sie ein paar numerische Werte eintragen (am einfachsten ist es, wenn Sie pro Zeile einen Wert schreiben)
  - Schreiben Sie nun ein Programm, das die Datei "Daten" öffnet und die Zahlen in einer Liste speichert
  - Berechnen Sie die Summe und den Mittelwert der Daten

Mengen

Sie und ein(e) Bekannte(r) waren shoppen. Sie haben gekauft:

```
meine_einkaeufe = {"hose", "jacke", "schuhe", "schokoriegel", "schal", "flatscreen"}
```

Ihr Bekannter hat gekauft:

```
seine_einkaeufe = {"buch", "schuhe", "muetze", "jacke", "softdrink"}
```

Benutzen Sie Mengenoperationen um herauszufinden:

- Was ist die Gesamtmenge, die Sie beide eingekauft haben?
- Was hat Ihr Bekannter gekauft, was Sie auch gekauft haben?
- Was hat Ihr Bekannter gekauft, das Sie nicht gekauft haben?
- Was haben Sie gekauft, das Ihr Bekannter nicht gekauft hat?
- Welche Dinge haben Sie nicht beide gekauft?

Sieb des Eratosthenes

Das Sieb des Eratosthenes ist ein Algorithmus, mit dem man Primzahlen bestimmen kann. Er lässt sich in Python gut mit Mengen lösen.

Um die Primzahlen von 2 bis *n_max* zu bestimmen, tun Sie folgendes: * Erstellen Sie eine Menge *numbers* der Zahlen von 2 bis *n_max*

- Erstellen Sie eine Variable $next\_prime = 2$
- Solange $next\_prime < n\_max/2$ ist (while-Schleife!), erstellen Sie eine weitere Menge *not_prime*, die alle Vielfachen von *next_prime* enthält. Ziehen Sie diese von *numbers* ab. Erhöhen Sie anschließend *next_prime* um 1

- Wenn die while-Schleife beendet, sollten sich in *numbers* nur noch die Primzahlen zwischen 2 und *n_max* befinden

Dictionary

Gegeben ist ein (zugegebenermaßen sehr knappes) Englisch-Deutsch-Wörterbuch:

```
englisch_deutsch_dict = {"hello": "hallo", "world": "welt", "good bye": "auf wiedersehen"}
```

Machen Sie daraus ein Deutsch-Englisch-Wörterbuch!

Wortzähler

The Zen of Python, by Tim Peters

Beautiful is better than ugly
Explicit is better than implicit
Simple is better than complex
Complex is better than complicated
Flat is better than nested
Sparse is better than dense
Readability counts
Special cases are not special enough to break the rules
Although practicality beats purity
Errors should never pass silently
Unless explicitly silenced
In the face of ambiguity refuse the temptation to guess
There should be one and preferably only one obvious way to do it
Although that way may not be obvious at first unless you are Dutch
Now is better than never
Although never is often better than right now
If the implementation is hard to explain it is a bad idea
If the implementation is easy to explain it may be a good idea
Namespaces are one honking great idea let us do more of those

- Schreiben Sie ein Programm, das die Häufigkeit jedes Wortes im obigen Text zählt.

Hilfestellungen: * Speichern Sie den Text entweder als String mit 3 Anführungszeichen am Anfang und am Ende (dann kann er sich auch über mehrere Zeilen erstrecken), oder lesen Sie ihn aus einer Datei aus, in die Sie ihn vorher geschrieben haben.

- Angenommen, Sie haben den gesamten Text in der Variablen *text* gespeichert. Dann können Sie mit

  `python  wortliste = text.split()` eine Liste der einzelnen Wörter erstellen.

- Um ungewünschte Zeichen aus dem Text zu entfernen, benutzen Sie: `python text.replace(zu_ersetzendes_zeichen, "")`
- Zum Zählen der Wörter ist ein Dictionary sehr hilfreich!
- Schreiben Sie das Programm nun so um, dass es die Häufigkeit der einzelnen Buchstaben zählt.
- Benutzen Sie Sets (Mengen), um herauszufinden, welche Wörter in jeder der ersten 6 Zeilen vorkommen.
- Erstellen Sie mithilfe von Sets eine Liste aller Buchstaben, die in dem Text vorkommen

Galgenmännchen

Schreiben Sie ein Programm, in dem ein String mit dem Namen *unbekanntes_wort* gespeichert ist, sowie eine leere Menge mit dem Namen geratene_woerter. In einer ewigen while-Schleife (while True) soll nun folgendes geschehen:

- Zuerst wird jeder Buchstabe von *unbekanntes_wort* durchlaufen, und, wenn er in *geratene_woerter* enthalten ist, ausgegeben. Ist er nicht enthalten, wird ein "-" ausgegeben.
  Beispiel:
  - Das geheime Wort ist *wochenende*
  - *geratene_woerter* enthält {"n", "e"}
  - Ausgabe ist dann: "- - - - e n e n - e"
- Dann muss der User einen neuen Buchstaben eingeben (Achtung, genau einen!). Dieser wird in *geratene_woerter* gespeichert
- Ergänzen Sie das Programm nun so, dass es mit einer Glückwunschsnachricht abbricht, sobald der User alle Buchstaben geraten hat.
- Erlauben Sie nur maximal 11 Fehlversuche, bevor das Programm mit "Game Over" abbricht
- Bonus: Anstatt das Wort in den Quellcode zu schreiben, lassen Sie es einen Mitspieler am Anfang eingeben. Wenn Sie folgendes zu Beginn schreiben:

`python from getpass import getpass` können Sie mittels

`python unbekanntes_wort = getpass("Bitte zu ratendes Wort eingeben\n")` das geheime Wort eingeben lassen, ohne dass es auf dem Bildschirm zu sehen ist.

- Spielen Sie eine Runde Galgenmännchen mit Ihrem Nachbarn