

Programmierung für Naturwissenschaften 1
Wintersemester 2020/2021
Übungen zur Vorlesung: Ausgabe am 10.02.2021

Dieses Übungsblatt besteht aus zwei kleineren und einer größeren Aufgabe von insgesamt 16 Punkten. Aufgabe 12.3 umfaßt zwei Teile, die unabhängig voneinander bearbeitet werden können. Sie sollten sich in jedem Fall an der ersten Aufgabe 12.1 beteiligen, wenn Sie nicht ohnehin zu den Studierenden gehören, die sich darauf vorbereiten müssen. Es wird nicht erwartet, dass Sie die Aufgaben 12.2 und 12.3 vollständig lösen. Sie sollten daher entsprechend Ihrer Interessen die Aufgaben auswählen, die Sie bearbeiten möchten. In die Gesamtwertung gehen nur 10 Punkte ein.

Aufgabe 12.1 (2 Punkte) Aus Zeitgründen können wir in der Vorlesung einen Teil des Themas „Numerische Integration“ (siehe `python-slides.pdf`, Abschnitt *Numerical Integration using midpoints*, Abschnitt 30, frame 29-32) nicht behandeln. Um diesen Abschnitt geht es in dieser Aufgabe. Dazu müssen sich einige Studierende vor der Übung anhand der Vorlesungsfolien auf den Inhalt der genannten Folien vorbereiten und das erworbene Wissen in Kleingruppen in den ersten 15-20 Minuten zu Beginn der Übung an die anderen Studierenden weitergeben.

Hier sehen Sie die Einteilung in Kleingruppen:

- | | |
|----------------------------------------|---------------------------------------------|
| 1. Vehns, Nichvolodina, Tchonet Toukam | 9. Rodrigues Nobre, Koerper, Pohle, Neuhaus |
| 2. Schimansky, Hossfeld, Kourouklis | 10. Schulta, Maecking, Riemann, Nachtschatt |
| 3. Stephan, Mehammed, Nergiz | 11. Trigkas Chatziandreu, Kemnitz, Le |
| 4. Unland, Koester, Wang, Schindler | 12. von der Osten-Sacken, Merl, Sakhire |
| 5. Beiersdorf, Jolic, Zozanyan | 13. Spurny, ASchroeder, Krupp, Kroeger |
| 6. Wacke, Hiep, Reich | 14. Fikani, Altmann, Chow Castro |
| 7. Mrozek, Albetani, Liebsch | 15. Koegel, Coin, Buchstab, Duez |
| 8. Lim, Fuchs, Jakovljevic | |

Die Namen der Studierenden, die sich auf das Thema vorbereiten müssen, sind jeweils als Erstes aufgeführt. Falls jemand von diesen Studierenden nicht zur Übung erscheint, verteilen sich die übrigen Mitglieder der entsprechenden Kleingruppe auf die anderen Kleingruppen.

Nach der Übung dokumentiert jede Kleingruppe in einer E-mail an `pfn1@zbh.uni-hamburg.de` das Vorgehen bei der Erarbeitung des Themas und ggf. noch bestehende Verständnisfragen oder Hinweise zu Unklarheiten in den Folien. Willkommen sind natürlich auch Bemerkungen zur Lehrform selbst. Dabei soll nicht der Inhalt der Folien repliziert werden. Die E-mail soll die folgenden Eigenschaften haben:

- die Betreffzeile lautet `midpointPeer` in genau dieser Schreibweise,
- abgesendet bis zum Abgabetermin der entsprechenden Übung,
- maximal 15 Zeilen mit maximal 80 Zeichen pro Zeile,

- Angabe der Nachnamen aller Mitglieder der Kleingruppe, die teilgenommen haben (alle genannten Personen erhalten die zwei Punkte).

Die E-mail soll von einer/einem Studierenden erstellt werden, die/der sich nicht auf das Thema vorbereitet hat.

Aufgabe 12.2 (3 Punkte) Einfaches Lennard-Jones-Modell

Das Lennard-Jones-Potential beschreibt die Wechselwirkungsenergie zweier ungeladener Atome oder Moleküle in Abhängigkeit von ihrem Abstand.

Für zwei Moleküle A und B ist das Lennard-Jones-Potential definiert durch:

$$\sum_{a \in A} \sum_{b \in B} 4 \cdot \epsilon_{a,b} \cdot \left(\left(\frac{\sigma_{a,b}}{D(a,b)} \right)^{12} - \left(\frac{\sigma_{a,b}}{D(a,b)} \right)^6 \right) \quad (1)$$

Dabei ist $D(a,b)$ der Abstand zwischen Atom a und Atom b , definiert durch

$$D(a,b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2 + (a_z - b_z)^2}$$

wobei (a_x, a_y, a_z) und (b_x, b_y, b_z) die Koordinaten der beiden Atome a und b sind.

$\epsilon_{a,b}$ und $\sigma_{a,b}$ sind Konstanten, die unabhängig sind von der räumlichen Lage der Atome. Typischerweise verwendet man bekannte Werte $\epsilon_{a,a}$ und $\sigma_{a,a}$ (d.h. für Atome der gleichen Sorte), um $\epsilon_{a,b}$ und $\sigma_{a,b}$ für $a \neq b$ zu schätzen. Für Wasserstoff, Helium, Kohlenstoff, Stickstoff, Sauerstoff, Neon und Argon werden diese Werte durch zwei Methoden der Klasse `LennardJonesData` (siehe `molecule.py` nach Umbenennung) geliefert (`epsilon_get(a)` liefert $\epsilon_{a,a}$ und `sigma_get(a)` liefert $\sigma_{a,a}$). Siehe dazu die Kommentare zum Aufruf dieser Methoden im Programmcode.

Für $a \neq b$ berechnet man $\epsilon_{a,b}$ und $\sigma_{a,b}$ wie folgt:

$$\begin{aligned} \epsilon_{a,b} &= \sqrt{\epsilon_{a,a} \epsilon_{b,b}} \\ \sigma_{a,b} &= \frac{\sigma_{a,a} + \sigma_{b,b}}{2} \end{aligned}$$

Zur Lösung dieser Aufgabe sollen Sie die Klassen `Atom` und `Molecule` aus der Datei `molecule.py`¹ erweitern, so dass gilt:

- Für zwei Instanzen x und y der Klasse `Atom` (die zwei Atome a und b repräsentieren) liefert der Ausdruck $x - y$ den Abstand $D(a,b)$.
- Für zwei Instanzen x und y der Klasse `Molecule` (die Moleküle A bzw. B repräsentieren) liefert der Ausdruck $x - y$ das Lennard-Jones-Potential dieser Moleküle.

Der Subtraktions-Operator wird für beliebige Klassen durch die Methode `__sub__(self, other)` implementiert. In einem Ausdruck $x - y$ für zwei Instanzen x und y der gleichen Klasse ist der Wert dieses Ausdrucks der Rückgabewert von `__sub__(x, y)`.

In den Materialien finden Sie eine Datei mit Unittests. Durch `make test` verifizieren Sie die Korrektheit Ihrer Implementierung für 4 Paare von Molekülen aus den mitgelieferten Mol2-Dateien.

Punkteverteilung:

¹Diese erhalten Sie durch Umbenennung von `molecule.template.py`.

- weitgehend richtige Implementierung von `__sub__` in der Klasse `Atom`: 0.5 Punkte
- weitgehend richtige Implementierung von `__sub__` in der Klasse `Molecule`: 1.5 Punkte
- bestandene Tests: 1 Punkt.

Aufgabe 123 (11 Punkte) In dieser Aufgabe geht es um die Extraktion von CO_2 -Emissionsdaten aus einer HTML-Datei und der Aufbereitung der Daten in Form von Tabellen und Graphiken. Im Material zur Vorlesung finden Sie ein Verzeichnis `CO2Stat` mit einer Datei `CO2ausstoss.html.bz2`. Durch den Aufruf `make data` wird eine dekomprimierte Version als HTML-Datei erzeugt.

Die genannte Datei stammt von Spiegel Online (Datum 9.1.2019). Die Web-Seite enthält Informationen über die Entwicklung der Höhe der CO_2 -Emissionen für 146 Staaten und 14 Gruppen von Staaten im Zeitraum von 1971 bis 2016. Für 114 der 146 Staaten liegen jeweils 11 Werte vor. In der Datei `README` finden Sie die URL. Nach der Umstellung des Layouts von Spiegel Online Anfang Januar 2020 wird die Information auf der Homepage nicht mehr komplett angezeigt. Öffnen Sie daher die genannte HTML-Datei lokal in einem HTML-Browser (z.B. Firefox oder Safari oder Chrome). Wenn man in die Mitte der Seite scrollt, dann sieht man eine Tabelle mit mehreren Teiltabellen, die jeweils die Daten für bestimmte Regionen zeigen.

Diese Aufgabe ist modular aufgebaut, d.h. Sie können die einzelnen Teilaufgaben unabhängig voneinander lösen.

1 Von der HTML-Datei zur tsv-Datei (4 Punkte)

Der erste Schritt der Aufgabe besteht darin, mit Hilfe der Klasse `BeautifulSoup` aus dem Modul `bs4` die CO_2 -Daten aus der Tabelle in der HTML-Datei zu extrahieren.

Die Daten, die extrahiert werden sollen, findet man in der Tabulator-separierten Datei `CO2ausstoss.tsv` (siehe Material), die wie folgt aufgebaut ist:

- Spalte 1: Die Bezeichnung einer geografischen Einheit also eines Staates bzw. einer Gruppe von Staaten.
- Spalte 2: Der Wert 1, falls in Spalte 1 ein Bezeichner für einen Staat steht, sonst 0. Hinweis: Bei der Ansicht der Tabelle im Browser kann man leicht Bezeichner für Staaten von Bezeichnern für Gruppen von Staaten unterscheiden. In der HTML-Datei gibt es einen entsprechenden Tag (d.h. einen Bezeichner in spitzen Klammern). Wenn dieser vorhanden ist, wird eine Gruppe von Staaten bezeichnet, sonst nicht. Diese Eigenschaft nutzt man in der Funktion `co2_stat_table_lines_get` (siehe unten) aus.
- Spalte 3-13: Die CO_2 -Emissionen in Tonnen pro Einwohner für die entsprechende geografische Einheit in den Jahren 1971, 1975, 1980, 1985, 1990, 1995, 2000, 2005, 2010, 2015 und 2016.

Es folgen nun Hinweise, wie Sie bei der Implementierung in der Datei `co2_stat.py` vorgehen können.

- Implementieren Sie eine Funktion `convert_table_headers(h_list)`, die als Parameter die Liste der Strings aus der Kopfzeile erhält und diese konvertiert. Dabei müssen die Abkürzungen der Jahreszahlen wie '75 durch 4-stellige Zahlen ersetzt werden.
- Implementieren Sie eine Funktion `co2_stat_table_lines_get(co2_stat_html)`, die aus einem String `co2_stat_html` in HTML-Notation die CO_2 -Emissionsdaten extrahiert. Sie sollen

hier analog zur Funktion `gb_stat_table_lines_get` aus der Vorlesung (siehe Abschnitt „Extracting and visualizing meta data about Genbank“ und Programmcode in `genbank_size.py`) vorgehen. Ihre Funktion verwendet also die Klasse `BeautifulSoup` und liefert eine Liste von Strings, die die einzelnen Zeilen der Datei `CO2ausstoss.tsv` beinhalten. Dabei sind die einzelnen Werte in einer Zeile durch einen Tabulator getrennt. Natürlich dürfen Sie diese tsv-Datei nicht einlesen, denn Sie sollen ja die genannten Zeilen aus der HTML-Datei selbst erzeugen. Die tsv-Datei dient nur als Referenz zum Testen.

Aus dem String `co2_stat_html` wird in `co2_stat_table_lines_get` eine Instanz der Klasse `BeautifulSoup` erzeugt, aus der man den Kopf der Tabelle und ihren Inhalt mit der Methode `find` extrahieren kann. Durch Anwendung von `find` und `find_all` mit den passenden Tags kann man die Namen der geografischen Einheiten sowie die entsprechenden CO_2 -Daten extrahieren und entscheiden, ob die Einheit ein Staat ist oder nicht. Entsprechend muss der zweite Wert in einer generierten Zeile eine 1 oder eine 0 sein. Gehen Sie bei der Extraktion schrittweise vor, d.h. lassen Sie sich nach der Extraktion eines Unterbaums, z.B. `thead`, diesen durch `print(thead.prettify())` temporär ausgeben, um zu sehen, wo die relevanten Informationen sind. Wie in der Vorlesung gezeigt, kann man den Text zu einem HTML-Knoten `n` durch `n.text` extrahieren. Dabei ist `n.text` ein String, und es ist sinnvoll mit `n.text.strip()` einen neuen String zu erzeugen, der keine Leerzeichen am Anfang und Ende hat. Beachten Sie auch, dass die extrahierten rationalen Zahlen in der HTML-Seite jeweils mit einem Komma geschrieben werden, das durch einen Punkt ersetzt werden muss.

- Implementieren Sie eine Klasse `CO2Stat` mit (u.a.) den folgenden Methoden:
 - Die Methode `__init__(self, co2_stat_table_lines)` erhält als Parameter eine Liste oder einen Stream mit den Zeilen, die die CO_2 -Informationen entsprechend des oben angegebenen Formates enthalten. Wie auch schon beim Abschnitt „Extracting and visualizing meta data about Genbank“ wird aus diesen Zeilen eine Instanz der Klasse `DataMatrix` erzeugt. Dazu müssen Sie das Modul `data_matrix_class.py` importieren und mit `DataMatrix` und den entsprechenden Parametern eine solche Instanz erzeugen. Dabei muss als Spaltennummer für die Schlüssel der Wert 0 angegeben werden, und der Parameter `ordered` muss den Wert `True` haben. In dieser Klasse soll diese Instanz von `DataMatrix` in der Member-Variable `self._dm` gespeichert werden.
 - Die Methode `show_orig(self)` gibt `self._dm` für alle Attribute und alle Schlüssel im Tab-separierten Format aus. Die Attribute und Schlüssel erhalten Sie durch die entsprechenden Methoden der Klasse `DataMatrix`. Die Formatierung müssen Sie nicht selbst vornehmen, sondern können eine entsprechende Methode aus der Klasse `DataMatrix` verwenden.

Die bis hier beschriebenen Methoden werden im Hauptprogramm `co2_stat_mn.py` aufgerufen, wenn man die Option `--tsv` verwendet. Durch `make test-tsv` wird überprüft, ob das Ergebnis korrekt ist. Falls der Test fehlschlägt, ist es sinnvoll die Ausgabe von `./co2_stat_mn.py --tsv CO2ausstoss.html` in eine Datei umzuleiten und diese dann mit Hilfe von `diff` mit der Datei `CO2ausstoss.tsv` zu vergleichen.

Falls Sie Schwierigkeiten bei der Extraktion der Daten aus der HTML-Seite haben, oder Ihr Gruppenpartner/Ihre Gruppenpartnerin mit dem zweiten Teil dieser Aufgabe beginnen möchte, ist das möglich. Sie müssen dann `co2_stat_mn.py`, wie im `Makefile` mit der Datei `CO2ausstoss.tsv` als letztem Argument aufrufen.

2 Plotten der Daten

Für diesen Teil der Aufgabe müssen Sie die Klasse `CO2Stat` um einige Member-Variablen und Methoden zum Gruppieren und Plotten von Daten erweitern. Entsprechende Techniken haben Sie in der Vorlesung zu Zeitreihen kennengelernt. Im Material zu dieser Übung finden Sie ein Verzeichnis `TimeSeries` mit dem entsprechenden Programm und den Daten aus der Vorlesung.

Bei der Implementierung der Klasse `CO2Stat` in der Datei `co2_stat.py` beachten Sie bitte folgende Hinweise:

- Initialisieren Sie in der Methode `__init__` die Member-Variable `self._year_list`. Diese enthält die Liste der Jahreszahlen (als Strings) aus der Attributliste. Diese können Sie durch einen entsprechenden regulären Ausdruck selektieren.

$\frac{1}{2}$ Punkt

- Initialisieren Sie in der Methode `__init__` die Member-Variable `self._all_countries`. Diese enthält die Liste aller geografischen Einheiten mit CO_2 -Daten, die Länder sind, d.h. für die das Attribut `is_country` den Wert '1' hat. Diese Liste kann aus der Liste der Schlüssel von `self._dm` unter Berücksichtigung des genannten Attributs extrahiert werden.

$\frac{1}{2}$ Punkt

- Implementieren Sie die Methode

`plot_for_country_list(self, country_list, color_list, groupname, prefix)`. Diese erzeugt mit Hilfe von `matplotlib.pyplot` (für das Sie den Aliasnamen `plt` einführen) einen kontinuierlichen Plot (line plot) der CO_2 -Emissionen für die Liste der Staaten aus `country_list`. Sie müssen für jeden dieser Staaten `ax.plot` aufrufen. Die Farben der Linien im Plot sind in der Liste `color_list` spezifiziert. Diese Liste ist mindestens genau so lang wie `country_list`. Die Farbe `color_list[i]` wird für `country_list[i]` verwendet. `groupname` wird in der Überschrift des Plots verwendet, siehe z.B. die Aufrufe dieser Funktion in `co2_stat_mn.py` und den Plot in der Datei `References/big.pdf`. Zur Spezifikation der Platzierung und der Grösse der Legende verwenden Sie bitte

```
ax.legend(loc='best', fontsize='small')
```

Der Plot soll in der PDF-Datei mit den Präfix `prefix` gespeichert werden. Durch den Aufruf `./co2_stat_mn.py --prefix CO2ausstoss.tsv`

für `prefix ∈ {big, worst, developing, middle-european}`

erzeugen Sie also jeweils eine PDF-Datei `prefix.pdf`. Sie können diese Funktion durch `make test-country-list` testen.

2 Punkte

- Um für einen Staat `country` und ein Jahr `y` (jeweils durch einen String repräsentiert) den Emissionswert (als String) aus `self._dm` zu extrahieren, verwenden Sie die Notation `self._dm[country][y]`. Dokumentieren Sie bei der ersten Verwendung dieser Notation in Ihrem Programmcode, welche Methode aus der Klasse `DataMatrix` dabei aufgerufen wird.

1 Punkt

- Beachten Sie, dass die Jahreszahlen und Emissionswerte in `DataMatrix` jeweils durch einen String repräsentiert werden. Die Listen der Werte auf der X- und Y-Achse, die an `ax.plot` übergeben wird, muss aber aus ganzen bzw. rationalen Zahlen bestehen. Entsprechendes gilt für `ax.hist` und `ax.boxplot`. Daher müssen entsprechende Konvertierungen vorgenommen werden (mit der Methode `int` für die Jahreszahlen und der Methode `float` für Emissionswerte). Um zu berücksichtigen, dass fehlende Emissionswerte durch leere Strings repräsentiert werden und die Konvertierung durch `float()` bei fehlerhaften Eingaben fehlschlagen kann, sollten Sie die Methode `float_or_None` aus dem Modul `float_or_None.py` verwenden und nur Werte

aus `self._dm[country][y]` berücksichtigen, für die diese Funktion nicht `None` liefert.

- Implementieren Sie die Methode `histogram_all_emissions(self, prefix)`. Diese erzeugt ein Histogramm für die Emissionswerte aller Länder. Verwenden Sie dabei `750` bins und beschränken Sie mit `ax.set_xlim(0, 20)` den Wertebereich der Emissionen von `0` bis `20`. Es soll eine Abbildung wie in `histogram.pdf` (siehe References) erzeugt werden. Der entsprechende Test erfolgt durch `make test-histogram`. 1 Punkt
- Implementieren Sie die Methode `groupby(self, select_func)`. Diese gruppiert die Emissionswerte aus `self._dm` für jeden Staat `c` aus `self._all_countries` entsprechend des Wertes `select_func(c)`. Wenn z.B. `select_func` die Funktion ist, die für jeden Staat seinen Kontinent liefert, dann werden alle Emissionswerte des Staates seinem Kontinent zugeordnet. Falls `select_func(c)` den Rückgabewert `None` liefert, dann erfolgt keine Gruppierung der Werte für den Staat `c`. Die Methode soll ein Objekt der Klasse `OrderedDict` zurückliefern, dessen Schlüssel die Funktionswerte sind, die `select_func` zurückliefert. Die mit den Schlüsseln assoziierten Werte sind Listen von Emissionswerten in Form von Fließkommazahlen. Dazu müssen Sie die Anweisung `from collections import OrderedDict` in der Datei `co2_stat.py` einfügen. 1 Punkt
- Implementieren Sie die Methode `boxplot_by_unit(self, for_continent, prefix)`, die einen Box-Whisker-Plot mit einer Gruppierung nach Kontinenten (falls `for_continent` den Wert `True` hat), andernfalls nach Regionen. Dazu verwenden Sie die Klasse `Country2Region`, d.h. Sie erzeugen eine Instanz `cr` dieser Klasse. Die Methoden `cr.continent()` und `cr.region()` liefern für einen Staat (als Argument angegeben) den Bezeichner des Kontinents bzw. der Region, falls vorhanden. Die Zuordnung erfolgt auf Basis der Datei `country2region.tsv`. Für Gruppen von Staaten bzw. Staaten, die keiner Region zugeordnet sind, wird der Wert `None` zurückgeliefert. Es sollen Abbildungen wie in `References/boxplot_continent.pdf` bzw. `References/boxplot_region.pdf` erzeugt werden. Um die Markierungen der Gruppen auf der X-Achse zur Verbesserung der Lesbarkeit zu rotieren, verwenden Sie `ax.set_xticklabels(groups, rotation=25, ha='right')`, wobei `groups` die Instanz der Klasse `OrderedDict` ist, die die Gruppierung repräsentiert. Der entsprechende Test erfolgt durch `make test-boxplot`. 1 Punkt

In dieser Beschreibung ist `ax` das `axes`-Objekt (siehe `python-slides.pdf`, Abschnitt „Plotting Data using matplotlib“, frame 8), in dem der Plot, das Histogramm oder der Boxplot erzeugt wird.

Durch `make test` verifizieren Sie, dass alle Tests erfolgreich sind. Leider ist es nicht sinnvoll, die entstehenden PDF-Dateien mit einer Referenz zu vergleichen. Daher wird jeweils nur getestet, dass das Programm nicht abbricht und eine Datei mit dem erwarteten Namen entsteht. Sie sollten sich daher den Inhalt der entstehenden PDF-Dateien ansehen.

Bitte die Lösungen zu diesen Aufgaben bis zum 15.02.2021 um 18:00 Uhr an `pfn1@zbh.uni-hamburg.de` schicken.