Aufgabenblatt 5

5.1 Einzeiler & Mehrzeiler (9P)

Das beiliegende Skript *oneliners_and_multiliners_2.py* enthält bereits implementierte Funktionen: zwei einzeilige und eine mehrzeilige, z.B. `oneliner_1()`.

(a) Gestalten Sie **für die Einzeiler je eine mehrzeilige Variante** sowie **für den Mehrzeiler eine einzeilige** mit den entsprechenden Namen (z.B. `multiliner_1()`), sodass ein Funktionspaar jeweils das Gleiche tut.

(b) Die mehrzeilige Variante soll möglichst viele Zeilen enthalten, wobei **die komplexen Ausdrücke in einfache zu trennen** sind.

(c) Schreiben Sie **zu jeder Einzeiler-Funktion einen DocString**, der beschreibt, was die Funktion macht. – Achten Sie darauf, dass Sie den Sinn der Funktion beschreiben, nicht einzelne Implementations-Schritte.[1]

Die fehlenden Funktionen wurden bereits vorbereitet – aktuell tun sie nichts, denn sie enthalten ausschließlich den Befehl `pass`. Statt dieser Zeile sollten Sie Ihre Lösung schreiben.

Für den bereits vorbereiteten Main-Block müssen Sie noch ein **Modul namens `checker` importieren**. Dafür können Sie entweder das Modul, das Sie für das vorherige Aufgabenblatt erstellt haben, verwenden, oder das beigefügte Skript `checker_emm.py`. Die Funktion `check_equality()` des Moduls wird aufgerufen und somit sofort geprüft, ob die Funktionspaare tatsächlich das Gleiche zurückgeben.

5.2 Python-Zen (10P)

Wenn man *this.py* aufruft, wird einem das "Python-Zen" ausgegeben. Das Skript ist Teil der Python-Distribution.[2] Den Aufruf kann man folgendermaßen gestalten:

- im Python-Interpreter: `import this`
- von der Kommandozeile: `python -c "import this"` – oder mit Angabe des Pfades zum Skript, z.B.: `python /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/this.py`

[1] Z.B. `text.split()`: "Variable text wird an Leerzeichen aufgesplittet." ist weniger geeignete Formulierung als "Der Text wird in Worte aufgetrennt.", denn die zweite den angewandten Sinn der Funktionalität erfasst.

[2] Auf einem Mac könnte es z.B. unter */Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/this.py* gefunden werden. Man kann die möglichen Pfade dafür mithilfe des Moduls `sys` finden: `sys.path` listet alle gesetzten Pfade für die Umgebungsvariable `PYTHONPATH` auf. In dem einen wird auch *this.py* liegen.

Führen Sie das Skript aus und lesen Sie die Zen-Empfehlungen. Ggf. können Sie diese im Tutorium besprechen.

Wenn man das Skript selbst öffnet, sieht man den ausgegebenen Text nicht direkt, sondern er wird beim Aufruf des Skripts durch einen kleinen Entschlüsselungsalgorithmus aus einem verschlüsselten Text erzeugt.

Die ursprüngliche *this.py*-Datei, sowie eine leicht modifizierte, und mit Fragen und Aufgaben ergänzte Version des Skripts mit Namen *this_prog1.py* wurde dem Aufgabenblatt beigefügt. **Bearbeiten Sie in *this_prog1.py* die dort gestellten Fragen bzw. die Implementierungsaufgabe**.

5.3 Stadt-Statistik (11P)

Schreiben sie eine Funktion `read_cities(input_file_name)` → `dict` im Skript *cities_reader.py*, welche **eine Datei mit Informationen über Städte in eine komplexe Datenstruktur einliest**: Ein Dictionary soll zu jeder Stadt ihre Anwohnerzahl, Fläche und das Bundesland, in dem sie liegt, auflisten. Die Abbildung der Angaben zu den einzelnen Städten kann man z.B. als eine einfache Liste speichern, aber auch andere Datenstrukturen sind hierfür möglich.

```
...
Oldenburg    163830   102.99   Niedersachsen
Stuttgart    623738   207.35   Baden-Württemberg
Augsburg     286374   146.84   Bayern
Freiburg     226393   153.06   Baden-Württemberg
...

#eine mögliche Abbildung
city_dict = {...,
    'Stuttgart': [623738, 207.35,  'Baden-Württemberg'],
    'Freiburg': [226393, 153.06, 'Baden-Württemberg'],
    'Augsburg': [286374, 146.84, 'Bayern'],
    'Oldenburg': [163830, 102.99, 'Niedersachsen'],
    ...
}

# Eine alternative Abbildung der Stadt-Angaben: in einem
   eingebetteten Dictionary wie folgt für Stuttgart:
{'land': 'Baden-Württemberg', 'population': 623738, 'area':
   207.35}
```

- Der **Dateiname soll von der Kommandozeile eingelesen** werden. Für die Aufgabe kann die beigefügte cities.tsv-Datei verwendet werden.

- Zeilen mit einer Raute (`"#"`) am Zeilenanfang sind als Kommentarzeilen außer Acht zu lassen.
- Prüfen Sie mindestens drei mögliche Fehlerquellen beim Einlesen bzw. Abbilden der Daten – z.B. mit der Konstruktion `try: ...except` und/oder mit `if`-Abfragen. Behandeln Sie mögliche Fehler angemessen.
  - Einige Fehler sind in die beigefügten "beschädigten" Test-Datei *cities_ corrupted.tsv* eingeschlichen. Ihr Skript soll mindestens mit der Eingabedatei *cities.tsv* ohne unbeabsichtigten Abbruch durchlaufen, es ist aber willkommen, wenn Fehlerquellen in der beschädigten Eingabedatei (, bzw. weitere) ohne Abbruch behandelt werden.

Schreiben Sie eine zusätzliche Funktion `compare_cities(city_dict, city_name_1, city_name_2) -> None`, in der zum gegebenen Stadtnamen ihre Fläche mit einer anderen Stadt im gleichen Bundesland[3] verglichen wird: **Welche Stadt ist größer?** Die Funktion soll das Ergebnis des **Vergleichs bezüglich der Fläche der beiden Städte benutzerfreundlich ausgeben**. Prüfen Sie mindestens eine Fehlerquelle und behandeln Sie sie angemessen. Gestalten Sie Ihre Ausgabe(n) informativ.

Ein **Main-Teil** soll so gestaltet werden, dass nach dem Einlesen der Datei **mindestens ein Vergleich** möglich ist. Die **zu vergleichenden Städte sollten interaktiv vom Benutzer abgefragt** werden. Achten Sie auf mögliche Fehlerquellen und auf benutzerfreundliche Ausgabe(n).

Abgabe:

- Die erstellten oder ergänzten Dateien (*oneliners_ and_ multiliners_ 2.py*, *this_ prog1.py* und *cities_ reader.py*) sowie das verwendete *checker*-Skript müssen in einem komprimierten Ordner als E-Mail-Anhang abgegeben werden. Geben Sie die Eingabedateiein zu Aufgabe 5.3 **nicht** mit ab.
- Alle Skripte müssen einen Header mit einer kurzen Beschreibung des Moduls, den Namen des Autoren und ggf. das Erstellungsdatum enthalten. Die Funktionen sollten angemessenen dokumentiert werden (DocString), ggf. sollten Aufrufe in einem Main-Block gestaltet werden.
- Der Betreff der E-Mail soll lauten: `[Prog1_WS17] Name, Vorname, HA05 (Tt)`
  - Name und Vorname bitte entsprechend eintragen.
  - Für `(Tt)` soll man die Abkürzung des Tags angeben, an dem das Tutorium stattfindet, dem man zugeordnet wurde.

[3] Die Bedingung „im gleichen Bundesland“ ist optional, muss also nicht notwendigerweise geprüft werden.

- Beispiel: `[Prog1_WS17] Mustermann, Max, HA05 (Di)`
  - wenn Herr Mustermann das Dienstagstutorium besucht.