Informatik I: Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Dr. Daniel Büscher, Hannes Saffrich
Wintersemester 2019

Universität Freiburg
Institut für Informatik

Übungsblatt 10 – Lösungen

Abgabe: Montag, 13.01.2020, 9:00 Uhr morgens

Hinweis: Um ein python-Skript von Hand zu testen, möchte man häufiger Code außerhalb von Funktionen schreiben, z.B.

```
def f(x):
    ...
print(f(2))
print(f(4))
```

Dieser Code läuft aber nicht nur beim direkten Ausführen des Skripts, sondern auch wenn das Modul importiert oder mit `pytest` getestet wird. Um dies zu verhindern, können Sie auf die spezielle Variable `__name__` zugreifen, die genau dann den Wert `'__main__'` hat, wenn das Skript direkt aufgerufen wurde:

```
def f(x):
    ...
if __name__ == '__main__':
    print(f(2))
    print(f(4))
```

Verwenden Sie diese Technik bitte insbesondere bei Aufgabe 2 und 3, da ansonsten die Tutoren beim Ausführen ihrer Unittests in aufpoppenden Fenstern ertrinken.

Aufgabe 10.1 (Potenzmenge; Datei: `powerset.py`; Punkte: 4+4)

Die Potenzmenge $\mathcal{P}(X)$ einer Menge X ist eine neue Menge, die aus allen Teilmengen von X besteht.

(a) Definieren Sie eine Funktion `powerset(s: list) -> list`, die für eine beliebige Menge s, rekursiv die Potenzmenge berechnet und zurückgibt. Verwenden Sie dabei Python-Listen zur Darstellung von Mengen. Wir nehmen an, dass keine Wiederholungen in den Eingabelisten auftreten. Beispiel:

```
>>> powerset([1, 2, 3])
[[], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]]
```

(b) Schreiben Sie mindestens vier Testfunktionen, welche jeweils die korrekte Berechnung der Potenzmenge zu einer gegebenen Eingabemenge überprüfen. Die Testfälle sollen dabei voneinander verschieden und sinnvoll gewählt sein. Für Testfunktionen sind keine Docstrings erforderlich.

Hinweis: Das Importieren von weiteren Modulen ist bei der Bearbeitung dieser Aufgabe nicht erlaubt. Die Reihenfolge der Elemente innerhalb der Listen spielt keine Rolle (dies erschwert allerdings das Testen).

Lösung:

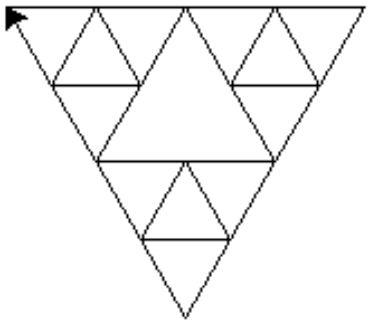
```
def powerset(xs: list) -> list:
    """Computes the power set of `xs`."""
    res = [xs]
    for i, e in enumerate(xs):
        ys = xs[:i] + xs[i + 1:]
        for y in powerset(ys):
            if y not in res:
                res += [y]
    return res

def test_powerset_1():
    ps = powerset([])
    assert len(ps) == 1
    assert [] in ps

def test_powerset_2():
    ps = powerset([1])
    assert len(ps) == 2
    assert [] in ps and [1] in ps

def test_powerset_3():
    ps = powerset([1, 2])
    assert len(ps) == 4
    assert [] in ps and [1] in ps and [1, 2] in ps and [2] in ps

def test_powerset_4():
    for i in range(5):
        assert len(powerset(list(range(i)))) == 2 ** i
```

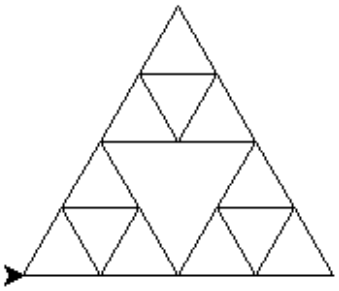
Aufgabe 10.2 (Sierpinski-Dreieck; Datei: `sierpinski.py`; Punkte: 6)
Das *Sierpinski-Dreieck* ist ein Fraktal, welches als $0L$-System wie folgt beschrieben werden kann:

$$\begin{aligned} V &= \{F, G, +, -\} \\ \omega &= F - G - G \\ P &= \{F \mapsto F - G + F + G - F, \\ &\qquad G \mapsto GG\} \end{aligned}$$

Hierbei entsprechen F und G dem Vorwärtszeichnen einer Strecke, $+$ entspricht einer Drehung um $120°$ nach links und $-$ entspricht einer Drehung um $120°$ nach rechts.

Implementieren Sie eine Funktion `sierpinski(size: int, n: int)` zum Zeichnen eines Sierpinski Dreiecks mit Hilfe des `turtle`-Moduls, wobei `size` die jeweilige Streckenlänge und `n` die Rekursionstiefe (bzw. Anzahl Generationen) angibt (analog zur Vorlesung). Implementieren Sie die Funktion direkt über Rekursion, also nicht so wie es in der nächsten Aufgabe gefordert ist. Das Ergebnis für $n = 2$ sollte so, oder so ähnlich aussehen (die Ausrichtung spielt keine Rolle):





Hinweis: Das `turtle`-Modul benötigt das `python3-tk` Paket. Falls dieses nicht installiert ist tritt folgender Fehler auf: `ImportError: No module named '_tkinter'`. In diesem Fall können Sie das Paket auf Ubuntu installieren mit (benötigt Adminrechte): `sudo apt install python3-tk`. Auf den Pool-Rechnern ist dieses Paket bereits installiert.

Lösung:

```
from turtle import forward, right, pendown, penup, Screen, speed, left

def sierpinski_g(size, n):
    """Helper for `sierpinski` implementing G-rule."""
    if n == 0:
        forward(size)
    else:
        sierpinski_g(size, n - 1)  # G
        sierpinski_g(size, n - 1)  # G

def sierpinski_f(size, n):
    """Helper for `sierpinski` implementing F-rule."""
    if n == 0:
        forward(size)
    else:
        sierpinski_f(size, n - 1)  # F
        right(120)                 # -
        sierpinski_g(size, n - 1)  # G
        left(120)                  # +
        sierpinski_f(size, n - 1)  # F
```

```
        left(120)                  # +
        sierpinski_g(size, n - 1)  # G
        right(120)                 # -
        sierpinski_f(size, n - 1)  # F

def sierpinski(size, n):
    """Draw `n` iterations of the Sierpinski Triangle with lines of
    length `size`.
    """
    pendown()
    sierpinski_f(size, n)  # F
    right(120)             # -
    sierpinski_g(size, n)  # G
    right(120)             # -
    sierpinski_g(size, n)  # G
    penup()
```

Aufgabe 10.3 (Sierpinski-Dreieck; Datei: `sierpinski_framework.py`; Punkte: 4)
Implementieren Sie, wie in der vorherigen Aufgabe, eine Funktion `sierpinski(size: int, n: int)` zum Zeichnen eines Sierpinski-Dreiecks, aber verwenden Sie das `lsystem`-Modul von der Vorlesungswebseite[1].

Erstellen Sie zunächst ein `LSystem`-Objekt, das die Regeln des Sierpinski-Dreiecks beschreibt, generieren sie dann mit der `generate`-Methode das Wort der gewünschten Iterationstiefe und iterieren Sie anschließend über die Zeichen des Wortes, um diese mit Hilfe der Funktionen des `turtle`-Moduls entsprechend zu interpretieren.

Eine beispielhafte Verwendung des `lsystem`-Moduls finden Sie in der Datei `mystery2.py`[2], die in der Vorlesung vom 18.12.2019 ab Minute 38 beschrieben wurde. Es bietet sich an, anders wie im Beispiel, den gesamten Code direkt in die `sierpinski`-Funktion zu schreiben.

Lösung:

```
from lsystem import *
from turtle import *
import random

def sierpinski(size : int, n : int):
    '''Draws the Sierpinski Triangle with lines of length `size` at recursion
    depth `n` using the `turtle` module.
    '''
    ls = LSystem(
```

[1] `http://proglang.informatik.uni-freiburg.de/teaching/info1/2019/lecture/lsystem.py`

[2] `http://proglang.informatik.uni-freiburg.de/teaching/info1/2019/lecture/mystery2.py`

```
    "F-G-G",
    {
        'F': "F-G+F+G-F",
        'G': "GG"
    })

w = ls.generate(n)

reset()
speed(0)
hideturtle()
color('black')
pendown()

for c in w:
    if c == 'F' or c == 'G':
        forward(size)
    elif c == '+':
        left(120.0)
    elif c == '-':
        right(120.0)
    else:
        pass
```

Aufgabe 10.4 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet10` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, Bezug zur Vorlesung, Interessantes, benötigter Zeitaufwand, etc.).