
SoSe 2018
Prof. Dr. Margarita Esponda
Objektorientierte Programmierung
2. Übungsblatt

Ziel: Auseinandersetzung mit Rekursion vs. Iteration und erste Analysen der Klassenkomplexität von Algorithmen.

1. Aufgabe (5 Punkte)

Definieren Sie eine Python-Funktion **apply_if**, die als Eingabe eine einstellige Funktion **f**, ein Prädikat **p** (Funktion) und eine Liste **xs** bekommt und die Funktion **f** nur auf die Elemente der Liste, die das Prädikat **p** erfüllen, anwendet.

Anwendungsbeispiel:

```
>>> apply_if ( factorial, odd [2, 5, 7, 4, 9, 6] )  
[2, 120, 5040, 4, 362880, 6]
```

Haskell-Implementierung:

```
apply_if f p [] = []  
apply_if f p (x:xs) | p x = (f x): apply_if f p xs  
                  | otherwise = x : apply_if f p xs
```

Zum Testen verwenden Sie die **factorial** und **odd** Funktionen aus der Vorlesungsfolien.

2. Aufgabe (8 Punkte)

Betrachten Sie folgende Haskell-Funktion höherer Ordnung:

```
zipWith f (x:xs) (y:ys) = (f x y) : (zipWith f xs ys)  
zipWith f _ _ = []
```

- a) Programmieren Sie eine rekursive Python-Funktion, die genau der gleichen Semantik entspricht. D.h. das Ergebnis soll in eine neue Liste zurückgegeben werden.
- b) Programmieren Sie eine effiziente Version der Funktion, die aber trotzdem korrekte Ergebnisse liefert.
- c) Verwenden Sie das Time-Modul von Python und messen Sie damit die Ausführungszeit beider Lösungen. Beispiele zur Verwendung des Time-Moduls werden in der Vorlesung gezeigt.

3. Aufgabe (5 Punkte)

Schreiben Sie eine Python-Funktion, die die Funktion **randint** des **random**-Moduls von Python benutzt und berechnet, nach wie vielen Zufallszahlen die erste Wiederholung vorkommt.

- a) Verwenden Sie dafür eine Dictionary-Datenstruktur.
- b) Sie müssen die zwei Argumente für die **randint**-Funktion in Ihre Funktion miteingeben.

4. Aufgabe (12 Punkte) Geburtstagsparadox

Die Wahrscheinlichkeit, dass 2 Personen am gleichen Tag Geburtstag haben, kann mit Hilfe eines Zufallsgenerators empirisch approximiert werden.

- a) Schreiben Sie eine Funktion **double_birthday**, die zufällige Geburtstage produziert und zählt, nach wie vielen Versuchen eine Wiederholung vorkommt. Verwenden Sie für die Zwischenspeicherung der Daten ein Wörterbuch (Dictionary) als Datenstruktur.
- b) Programmieren Sie eine zweite Funktion **repeat_double_birthday**, die das Experiment lange genug wiederholt und mit Hilfe eines Arrays aufzählt, nach wie vielen Versuchen (2, 3, 4,...,bis max. 365) eine Wiederholung vorkommt.
- c) Mit der Ergebnisliste aus b) können Sie dann eine Funktion **birthday_paradox** programmieren, die die Wahrscheinlichkeit, dass auf einer Party mit **n** Gästen zwei Personen am gleichen Tag Geburtstag haben, empirisch berechnet.

Wichtige Hinweise:

- 1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Funktionalität der Funktionen darstellen.
- 2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.
- 3) Kommentieren Sie Ihre Programme und schreiben Sie Hilfsdokumentation.
- 4) Verwenden Sie geeignete Hilfsvariablen und Hilfsfunktionen in Ihren Programmen.
- 5) Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.
- 6) Schreiben Sie getrennte Test-Funktionen für alle 4 Aufgaben für Ihren Tutor.