Iteratoren / RUnit Tests / Objektgleichheit

VERWENDEN SIE FÜR DIE LÖSUNG DIE ENTSPRECHENDEN VERZEICHNISSE SCRIPTE UND KLASSEN IN DEM MITGELIEFERTEN PROJEKT!!!!!!!

A Summen berechnen mit inject

Schreiben Sie ein Programm, das die nachfolgenden Formeln berechnet.

1. $\sum_{i=0}^{n} \frac{1}{2^i} \approx 2$
2. $\sum_{i=1}^{n} \frac{1}{(2*i-1)(2*i+1)} \approx \frac{1}{2}$
3. $\sum_{i=1}^{n} (-1)^{(i+1)} \frac{(x-1)^i}{i} \approx, für\ 0 < x \leq 2$

- Schreiben Sie für jede Formel eine eigene Funktion/Methode.
- Prüfen Sie, wenn immer nötig die Wertebereiche der Parameter und geben Sie bei Verletzung des Wertebereichs eine -99 zurück.
- Beachten Sie die Effekte der ganzzahligen Division!
- Lösen Sie die Teilaufgabe nur unter Verwendung der Methode ***inject*** des Moduls ***Enumerable***. Verwenden Sie Ranges zur Abbildung der Intervalle für den Index i.

B RUnit Tests für Aufgabenteil 5-A

Schreiben Sie für die 3'te Funktion aus Aufgabenteil A RUnit-Tests. Testen Sie die positiven und negativen Fälle.

Prüfen Sie die Berechnungsergebnisse für Gleitkommazahlen abzüglich eines kleinen Epsilon mit der Methode ***assert_in_epsilon(expected,actual,epsilon)***. Ihre Tests sollen auch Ergebnisse mit epsilon = 1*e-10 enthalten

Beispiel: assert_in_epsilon(Math::PI, 22.2/7, 0.002)

expected = Math::PI ≈ 3.14159265358979323846264338327 95

actual = 22.2/7 ≈ 3,142857142857142857142857142857 1

epsilon = 0.002

Math::PI - 22.2/7 < 0.002

C Iterieren über Arrays

1. Schreiben Sie eine Methode/Funktion, die auf einem Array die Funktion $f(x) = \sin x$ berechnet. Dabei sind die x die Werte des Arrays. Sie können davon ausgehen, dass die Werte korrekt übergeben werden. Das Ergebnis ist wieder eine Array. Verwenden Sie eine geeignete Methode von ***Enumerable***.
   Beispiel:
   ```
   p sin_auf_array([Math::PI, Math::PI/2, 2, 1.5, 7.8])
   ```

   liefert das Ergebnis

   ```
   [1.2246063538223773e-16, 1.0, 0.9092974268256817, 0.9974949866040544,
   0.998543345374605]
   ```

2. Schreiben Sie eine Methode/Funktion, der zwei Arrays übergeben werden und die das Kreuzprodukt zweier Arrays berechnet. Das Kreuzprodukt kombiniert jedes Element des ersten Arrays mit jedem Element des zweiten Arrays. Das Ergebnis soll ein Array mit 2-elementigen Arrays sein.
   Beispiele:

   ```
   p kreuzprodukt([1,2,3],[:a,:b])
   p kreuzprodukt([],[:a,:b])
   p kreuzprodukt([1,2,3],[])
   ```

   liefert die nachfolgenden Ergebnisse

   ```
   [[1, :a], [1, :b], [2, :a], [2, :b], [3, :a], [3, :b]]
   []
   []
   ```
3. Schreiben Sie eine Methode/Funktion, die für ein Array prüft, ob alle Elemente durch 3 teilbar und größer als 7 sind. Das Ergebnis der Methode ist ein boolescher Wert. Verwenden Sie eine geeignete Methode von ***Enumerable***.
4. Schreiben Sie eine Methode/Funktion, die aus einem Array all die Elemente einsammelt, die gerade Zahlen und > 0 sind. Das Ergebnis der Methode ist ein Array. Verwenden Sie eine geeignete Methode von ***Enumerable***.

D Iterieren über Hashes

1. Schreiben Sie eine Methode/Funktion, die für einen Hash das Maximum der Schlüssel berechnet und diesen Schlüssel zurückgibt.
2. Schreiben Sie eine Methode/Funktion, die für einen Hash das Maximum der Werte berechnet und diesen Wert zurückgibt.
3. Schreiben Sie eine Methode/Funktion, die für einen Hash die maximalen Werte zählt.
   Beispiel:
   ```
   puts count_max_values({:a=> 7, :b=> 11, :c=> 6, :d=> 11, :e=> 11, :f => 9})
   ```
   liefert
   ```
   3
   ```
4. Schreiben Sie eine Methode/Funktion, die aus einem Hash die Paare einsammelt, für die die Summe aus key und value eine gerade Zahl ergibt. Das Ergebnis soll wieder ein Hash sein.
   Beispiel:
   ```
   puts key_value_sum_even({3=> 7, 4=> 11, 2=> 6, 5=> 11, 6=> 11, 8 => 9})
   ```
   liefert
   ```
   {3=>7, 2=>6, 5=>11}
   ```

E Re-Implementieren eigener Methoden von Enumerable

1. Schreiben Sie die Methode ***my_to_a*** für das Modul ***Enumerable***. Die Funktionsweise ist wie bei ***to_a*** von ***Enumerable***.
2. Schreiben Sie die Methode ***my_select*** für das Modul ***Enumerable***. Die Funktionsweise ist wie bei ***select*** von ***Enumerable***.
3. *** Schreiben Sie die Methode ***my_min*** für das Modul ***Enumerable***. Die Funktionsweise ist wie bei ***min*** von ***Enumerable***. ***my_min*** soll mit und ohne Block aufgerufen werden können. Mit ***block_given?()*** können Sie prüfen, ob der Methode ***my_min*** beim Aufruf ein Block übergeben wurde.

Es ist **nicht** erlaubt die Methoden ***to_a, select*** und ***min*** für die Lösung zu verwenden!

Die mit *** gekennzeichnete Aufgabe ist anspruchsvoll und muss nicht von allen gelöst werden.

F Objektgleichheit Arrays, Sets und Hashes

Präparieren Sie die im Projekt zur Aufgabe mitgelieferte Klassen derart, dass

1. Gleiche Objekte in Arrays wiedergefunden werden.
2. Keine zwei gleichen Objekte in eine Menge eingefügt werden können.
3. Keine zwei gleichen Schlüssel in einem Hash existieren können.