Dokumentation zum Arbeitsblatt 5 EWR

Tom Lambert

19. Juni 2017

Inhaltsverzeichnis

1 Einführung in die Problemstellung

In der Vorlesung wurde in Abschnitt 11.3 folgende Aufgabe gestellt:

> Bestimme die Koordinaten der Punkte auf der Einheitskreislinie.

Dort wurden außerdem 3 Algorithmen vorgestellt, mit denen die Aufgabe gelöst werden kann.
Diese und deren Implementierung werden später aufgezeigt.

2 Darstellung des theoretischen Hintergrunds

Wie einmal in der Schule kennen gelernt, kann man zum berechnen der Punkte auf dem Einheitskreis (in Abhängigkeit eines Winkels) mit Hilfe von Sinus und Cosinus verwenden.

$$x(\alpha) = cos(2\pi \cdot \frac{\alpha}{360}) \tag{1}$$

$$y(\alpha) = sin(2\pi \cdot \frac{\alpha}{360}) \tag{2}$$

Zur Berechnung von Sinus und Cosinus sind jedoch relativ aufwendige Rechnungen notwendig, die ggf. auch recht hohe Rundungsfehler aufweisen können.

TODO: effizient beschreiben

TODO: symmetrie beschreiben

3 Aufbau des Programs

Das Programm ist in mehrere Dateien untergliedert:

1. utils.py
   Diese Datei enthält Hilfsfunktionen die in keinem näheren Zusammenhang mit dem eigentlichen Problem stehen.

2. kreis.py
   Diese Datei enthält die Logik für die eigentlichen Berechnungen mit notwendigen, zusätzlichen Algorithmen für eine Auswertung.

3. ab5.py
   Diese Datei enthält das Hauptprogramm und enthält den Code, der die Tests durchführt.

3.1 utils.py

Diese Datei enthält Hilfsfunktionen die in keinem näheren Zusammenhang mit dem eigentlichen Problem stehen. Beispielsweise enthält sie eine Funktion `average` zum Berechnen des Durchschnitts der Werte in einer Liste. Die Datei ist in früheren- bzw. späteren Entwicklungsstadien auch in anderen Projekten zu finden. Der Portabilität wegen wird sie stets kopiert anstatt sie aus einem anderen Ordner zu referenzieren.

3.2 kreis.py

Die Logik wurde mit Hilfe einer Klasse `Kreis` realisiert, welche die notwendigen Methoden enthält.

Zunächst bietet die Klasse 2 Funktionen `_sin` und `_cos`, welche einfach den Sinus bzw. Cosinus mit Hilfe der `Numpy`-Bibliothek berechnen. Das besonder ist jedoch, dass ein Methodenaufruf auch einen internen Zähler um jeweils 1 erhöht. Über die `function_call_counter`-Eigenschaft lässt sich dieser Zähler abfragen und ggf. über die `reset_function_call_counter`-Funktion auf 0 zurück setzen.

Dadurch wird es möglich von außen zu erfahren welcher Algorithmus wie oft Sinus- und Cosinus-Werte berechnet hat.

Das eigentliche Kernstück sind jedoch die 3 Funktionen `naiv`, `effizient` sowie `symmetrie` die die Algorithmen implementieren. Allen gemeinsma ist, dass sie eine Liste mit Punkten auf dem Einheitskreis zurück geben. Und zwar genau 360 stück, einen Punkt je Grad. Weiterhin werden die Listen der Punkte als Wörterbuch zurück gegeben. Dadurch ist es möglich über einen als `Key` gewählten Winkel einen bestimmten Punkt abzufragen.

Zum Berechnen habe ich zunächst eine Hilfsfunktion `_naiv(angles)` erstellt, welche eine Liste mit Punkten zurück gibt, die auf den übergebenen Winkeln liegen. Die `naiv`-Funktion ruft `_naiv` nun einfach mit einer Liste der 360 Winkel auf. Die Funktion `symmetrie` jedoch nur für die Winkel von 1 bis 45 Grad, sodass dann für jeden zurück gegebenen Punkt die 7 passend gespiegelt- und gedrehten Punkte ermittelt werden können.

TODO: symmetrie beschreiben

3.3 ab5.px

Diese Datei enthält als ausführbaren Code lediglich eine `main`-Methode, welche direkt am Ende aufgerufen wird. Diese Implementierung hat den Vorteil, dass bei Erweiterung des Programms um Benutzereingaben einfach eine 2. Funktion implementiert werden kann.

Innerhalb der `main`-Funktion wird zunächst die Präzision des `Decimal`-Typs auf 20 festgelegt. Anschließend wird ein Wörterbuch erstellt, welches einige vorberechnete Punkte des Einheitskreises enthält. Diese wurden mit Wolfram Mathematica auf 20 Nachkommastellen genau berechnet. Diese Werte dienen später zum Vergleich mit den dynamisch berechneten Werten und zum Ermitteln des absoluten und des relativen Fehlers.

Zunächst werden nun die 3 Funktionen `naiv`, `effizient` und `symmetrie` aufgerufen. Nach jedem Aufruf wird dabei `function_call_counter` abgefragt und dem Benutzer ausgegeben.

Im zweiten Schritt werden nun einige ausgewählte berechnete Punkte mit ihren vorberechneten Pendants ausgegeben. Anschließend erscheinen die passenden absoluten- und relativen Fehler in der Benutzerausgabe. Während dieses Schritts werden die Fehler jeweils zu einer Liste hinzugefügt.

Nach dem Durchlauf der Liste der ausgewählten Winkel werden noch die durchschnittlichen Fehler ausgegeben, um dem Benutzer ein besseres Gefühl zu verschaffen wie einzelne Werte in der Menge aller Fehler liegen.

4 Experimente und Beobachtungen

5 Auswertung