Analysieren und Visualisieren mit Python

zweite Übungsaufgabe

Nach langer Suche hat Ihre Freundin Anastasia endlich einen Platz in einer WG mit drei Mathematikern gefunden. Um sich von Anfang an gut mit ihren neuen Mitbewohnern zu stellen, hat sich Anastasia überlegt, ein Plakat aufzuhängen, das das Konvergenzverhalten komplexer Folgen visualisiert. Da Sie Anastasia bei ihrer Seminararbeit bereits so gut geholfen haben, ist Ihre Freundin überzeugt davon, dass Sie eine für die kritischen Augen der Mathematiker ansprechende Schaugrafik berechnen können.

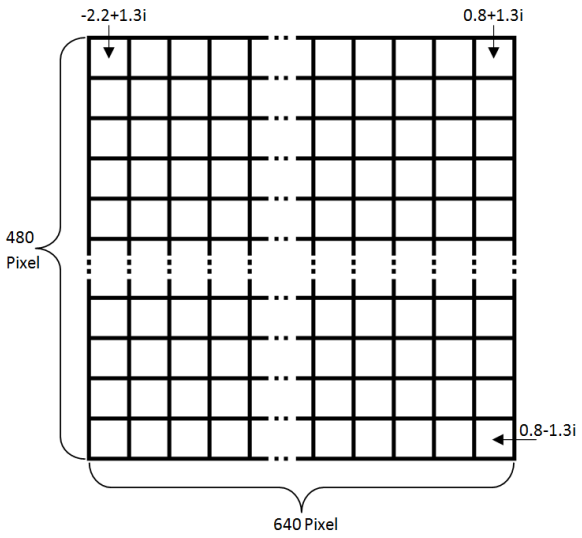
Komplexe Zahlen gehören in Python zu den Built-in-Datentypen (siehe Folien zu Datentypen). Um Anastasia zu zeigen, dass Sie sich bereits mit objektorientierter Programmierung auskennen, sollen Sie zur Lösung der folgenden Aufgaben von der Verwendung dieses Built-in-Typs absehen.

2a) (7 Punkte)
Um dennoch mit komplexen Zahlen rechnen zu können, sollen Sie für diesen Aufgabenteil eine eigene Klasse „*KomplexeZahl*“ schreiben. Neue Zahlen sollen mithilfe der Kontruktormethode durch einen Aufruf der Form
instanzEinerKomplexenZahl = KomplexeZahl(<*Realteil*>, <*Imaginärteil*>)
initialisiert werden können (wobei <*Realteil*> und <*Imaginärteil*> vom Typ *Float* bzw *Integer* sind). Der Realteil soll dann mittels **instanzEinerKomplexenZahl.re**, der Imaginärteil mittels **instanzEinerKomplexenZahl.im** gelesen und geschrieben werden können. Überladen Sie dann die Operatoren __add__ (Addition), __sub__ (Subtraktion), __mul__ (Multiplikation), __div__ (Division) (siehe dazu https://de.wikipedia.org/wiki/Komplexe_Zahl#Rechnen_in_der_algebraischen_Form). Eine komplexe Zahl (auf der linken Seite des Operanden) soll so mit einem *Float*, einem *Integer* oder einer anderen Instanz der Klasse *KomplexeZahl* verrechnet werden können. Überladen Sie weiterhin die Methoden __abs__ (Betrag)
(siehe dazu https://de.wikipedia.org/wiki/Komplexe_Zahl#Betrag_und_Metrik) und __str__.

print instanzEinerKomplexenZahl
soll (durch die Methode __str__) zu einer Ausgabe der Form
<Realteil> ± <Imaginärteil>i
führen.

2b) (8 Punkte)
In diesem Aufgabenteil soll ein Plakat berechnet werden, in dem den Bildpunkten einer PNG-Grafik verschiedene Farbwerte zugeordnet werden. Damit die Berechnung des Plakates nicht zu lange dauert, soll es eine Auflösung von 640 (Breite) x 480 (Höhe) Bildpunkten erhalten. Welche Farbe ein einzelner Bildpunkt abhängig von seiner Position bekommt, wird nun im Einzelnen beschrieben:
Zunächst wird jedem Pixel eine komplexe Zahl im Bereich von **-2.2 - 1.3i** bis **0.8 + 1.3i** zugeordnet. Wie in der Gaußschen Zahlenebene soll sich der Realteil von links nach rechts und der Imaginärteil von unten nach oben erstrecken. Dem Pixel mit der Position **0, 0** (oben links)

soll also die Zahl **-2.2 + 1.3i** zugeordnet werden, dem Pixel mit der Position **639, 479** (unten rechts) soll die Zahl **0.8 – 1.3i** zugeordnet werden. Der Realteil (von -2.2 bis 0.8) muss also in 640 Teile und der Imaginärteil (von -1.3 bis 1.3) in 480 Teile unterteilt werden.



Jedem einzelnen Pixel wird durch das obig beschriebene Verfahren also ein Wert (k_0) zugeordnet. Für jeden Wert k_0, soll im Weiteren das Konvergenzverhalten der rekursiv definierten Folge

$$k_n = k_{n-1} * k_{n-1} + k_0$$

untersucht werden. Je nach Ausgang dieser Untersuchung soll dem Pixel dann ein unterschiedlicher Farbwert zugeordnet werden. Dabei gilt es, die folgenden Fälle zu unterscheiden:

- Wenn nach 20 Iterationen (Sie müssen also k_1 bis k_{20} ermitteln) kein einziges k berechnet wurde, dessen Betrag den Wert 2.0 überschreitet, soll davon ausgegangen werden, dass die Folge konvergiert. Alle Pixel, deren zugeordnete Werte zu solch einer Folge führen, sollen im ausgegebenen Bild denselben Farbwert erhalten (die Wahl dieser Farbe bleibt dabei Ihnen überlassen).
- Wenn innerhalb der ersten 20 Iterationsschritte ein k berechnet wird, dessen Betrag den Wert 2.0 überschreitet, soll davon ausgegangen werden, dass die Folge divergiert (bzw. gegen ±∞ geht). Je nach Anzahl der Iterationsschritte, die berechnet werden mussten, bis der Betrag einen Wert größer 2.0 erreichte, sollen die Pixel wiederum mit unterschiedlichen Farben versehen werden (wieder dürfen Sie die verwendeten Farben frei wählen). Bei null Iterationsschritten könnte also beispielsweise ein helles Rot verwendet werden, bei einem Iterationsschritt ein dunkles Ocker, bei zwei ein grünliches Grau etc.

Speichern Sie das so erzeugte Bild unter dem Dateinamen „plakat.png“ ab.
Wie Sie ein Bild erzeugen und den einzelnen Pixeln Farbwerte zuweisen, wird in der Mitgelieferten Datei *abgabe2.py* demonstriert.
Gegebenenfalls müssen Sie zusätzlich die Erweiterung „*Pillow*“ installieren:
https://python-pillow.github.io/

Ergänzen Sie zur Lösung der folgenden Aufgaben die Datei *abgabe2.py* und reichen Sie sie über GRIPS bis zum 18.11.2015 um 24:00 Uhr ein. Bei Problemen mit der Abgabe können Sie Ihre Lösung auch an manuel-tonio.mueller@ur.de senden.
Bitte geben Sie ausschließlich die (unverpackte) Datei abgabe2.py ab. Die erzeugte Grafik *plakat.png* wird nicht benötigt.

Da es sich um eine Gruppenabgabe handelt, bekommen alle Mitglieder in Ihrer Gruppe die erreichten Punkte gutgeschrieben. Geben Sie bitte dennoch die Bearbeiter der Aufgabe namentlich an, da jeder Teilnehmer im Laufe des Kurses bei mindestens zwei Aufgaben und am abschließenden Projekt mitgewirkt haben muss, um zu bestehen.

Viel Erfolg!