

LandkartenErstellung

API Documentation

May 17, 2019

Contents

Contents	1
1 Module Berechnung	3
1.1 Class Berechnung	3
1.1.1 Methods	3
1.2 Class StandardBerechnung	3
1.2.1 Methods	3
2 Module InputReader	4
2.1 Functions	4
2.2 Variables	4
2.3 Class InputReader	4
2.3.1 Methods	4
2.3.2 Class Variables	4
3 Module Karte	5
3.1 Variables	5
3.2 Class Karte	5
3.2.1 Methods	5
3.2.2 Class Variables	5
4 Module Land	6
4.1 Variables	6
4.2 Class Land	6
4.2.1 Methods	6
4.2.2 Class Variables	6
5 Module Main	8
5.1 Functions	8
5.2 Variables	8
5.3 Class Main	8
5.3.1 Methods	8
5.3.2 Class Variables	8
6 Module OutputWrite	9
6.1 Variables	9
6.2 Class OutputWriter	9
6.2.1 Methods	9

6.2.2 Class Variables	9
Index	10

1 Module Berechnung

1.1 Class Berechnung

1.1.1 Methods

distance(*self*, *landA*, *landB*)

Berechnet den Kreisabstand zwischen landA und landB anhand der Radien und des direkten Abstandes

berechne(*self*, *karte*)

Abstrakte Methode der Berechnung Muss in der abgeleiteten Klasse implementiert werden

normVektor(*self*, *landA*, *landB*)

Berechnet den normierten Vektor zwischen den Mittelpunkten von landA und landB

1.2 Class StandardBerechnung



1.2.1 Methods

mittelwertRadien(*self*, *karte*)

Berechnet den Mittelwert aller Radien

voriteration(*self*, *karte*, *faktor*=0)

Berechnung der Voriteration durch Skalieren der Mittelpunkte und Verkleinern der Radien

berechne(*self*, *karte*)

Berechnet einen Iterationsschritt fuer alle Laender in karte

Overrides: Berechnung.Berechnung.berechne

Inherited from Berechnung.Berechnung(Section 1.1)

distance(), normVektor()

2 Module InputReader

2.1 Functions

zusammenhaengend(*laenderliste*)

prueft ob alle Laender in der Liste zusammenhaengen; gibt true zurueck wenn alle zusammenhaengen, sonst false

zusammenhaengendRek(*startland, besucht*)

Rekursionsfunktion zum pruefen ob alle laender zusammenhaengen void

addNachbarnToLand(*self, nachbarn, heimatland*)

fuegt einem Land ein anderes als Nachbarn hinzu

2.2 Variables

Name	Description
<code>__package__</code>	Value: None

2.3 Class InputReader

2.3.1 Methods

__init__(*self, filename*)

Konstruktor erhaelt den Dateinamen der Eingabedatei und speichert ihn ab

read(*self*)

liest die Eingabedatei ein und gibt ein Objekt von Karte mit allen eingelesenen Informationen zurueck

2.3.2 Class Variables

Name	Description
<code>mFilename</code>	dateiname fuer eingabedatei Value: ''

3 Module Karte

3.1 Variables

Name	Description
<code>--package--</code>	Value: None

3.2 Class Karte

3.2.1 Methods

<code>--init--(self, l, kennzBz)</code>
Konstruktor der Klasse Karte bekommt eine Liste an Laendern sowie die Kennzahlbezeichnung
<code>scale(self)</code>
Passt den Wertebereich von x und y so an, dass $x_{\max} - x_{\min} = y_{\max} - y_{\min}$
<code>berechneMinimum(self)</code>
Berechnet die minimalen und maximalen x- und y-Werte der Karte

3.2.2 Class Variables

Name	Description
<code>laenderliste</code>	Value: []
<code>kennzahlBz</code>	Value: ''
<code>xmin</code>	minimaler x-Wert der Karte Value: 0.0
<code>xmax</code>	maximaler x-Wert der Karte Value: 0.0
<code>ymin</code>	minimaler y-Wert der Karte Value: 0.0
<code>ymax</code>	maximaler y-Wert der Karte Value: 0.0

4 Module Land

4.1 Variables

Name	Description
<code>--package--</code>	Value: None

4.2 Class Land

4.2.1 Methods

<code>--init--(self, x, y, kennZe, kennZa)</code>
Konstruktor der Klasse Land bekommt Mittelpunkt (x,y) das Laenderkennzeichen kennZe und die Kennzahl kennZa
<code>--str--(self)</code>
Gibt String-Repraesentation fuer Ausgabe zurueck
<code>--repr--(self)</code>
Gibt String-Repraesentation fuer Ausgabe zurueck Wird zu Debugging-Zwecken benoetigt
<code>addNachbar(self, nachbar)</code>
fuegt Nachbarn der Liste hinzu
<code>containsNachbar(self, nachbar)</code>
True wenn nachbar ein Nachbar des self-Objektes

4.2.2 Class Variables

Name	Description
xPos	x-Position des Mittelpunktes Value: 0.0
yPos	y-Position des Mittelpunktes Value: 0.0
kennzeichen	Laenderkennzeichen Value: ''

continued on next page

Name	Description
kennzahl	Kennzahl Value: 0
kraftX	aktuelle Kraft in x-Richtung (nur waehrend einer Iteration benoetigt) Value: 0.0
kraftY	aktuelle Kraft in y-Richtung (nur waehrend einer Iteration benoetigt) Value: 0.0
radius	Radius des Kreises Value: 0.0
nachbarlaender	Nachbarlaender Value: []

5 Module Main

5.1 Functions

main (<i>arguments</i>)

Main-Methode zum Steuern von Input, Berechnung und Output

5.2 Variables

Name	Description
ap	Value: argparse.ArgumentParser("Hello there!")
results	Value: ap.parse_args()

5.3 Class Main

5.3.1 Methods

printUsage (<i>self</i>)

5.3.2 Class Variables

Name	Description
mFilename	string dateiname fuer ausgabe Value: ""
mInput	InputReader fuer eingabe Value: None
mOutput	OutputWriter fuer ausgabe Value: None
mBerechnung	(Standard-)Berechnung fuer die Iterationsberechnung Value: None
karte	eingesene und ggf. schon berechnete Daten Value: None
iterationen	iterationsschritte Value: 100

6 Module OutputWrite

6.1 Variables

Name	Description
<code>--package--</code>	Value: None

6.2 Class OutputWriter

6.2.1 Methods

<code>--init--</code> (<i>self</i> , <i>iFilename</i> , <i>iTemplate</i> , <i>iKarte</i> , <i>iIterationen</i>)
Konstruktor fuer die Ausgabe; <i>iFilename</i> - Dateipfad fuer die Ausgabedatei; <i>iTemplate</i> - Template mit allgemeiner Form; <i>iKarte</i> - berechnete Karte zur Ausgabe; <i>iIterationen</i> - Anzahl der durchgefuehrten Iterationen

<code>write</code> (<i>self</i>)
Schreibt die Datei

6.2.2 Class Variables

Name	Description
<code>mFilename</code>	Dateipfad der Ausgabedatei Value: ''
<code>mTemplate</code>	Dateipfad des Templates Value: ''
<code>mKarte</code>	Karte die ausgegeben werden soll Value: <code>Karte([], "")</code>
<code>mIterationen</code>	Anzahl der berechneten Iterationen Value: 0

Index

- Berechnung (*module*), 3
 - Berechnung.Berechnung (*class*), 3
 - Berechnung.Berechnung.berechne (*method*), 3
 - Berechnung.Berechnung.distance (*method*), 3
 - Berechnung.Berechnung.normVektor (*method*), 3
 - Berechnung.StandardBerechnung (*class*), 3
 - Berechnung.StandardBerechnung.mittelwertRadien (*method*), 3
 - Berechnung.StandardBerechnung.voriteration (*method*), 3
- Main.Main (*class*), 8
 - Main.Main.printUsage (*method*), 8
 - Main.main (*function*), 8
- OutputWrite (*module*), 9
 - OutputWrite.OutputWriter (*class*), 9
 - OutputWrite.OutputWriter.__init__ (*method*), 9
 - OutputWrite.OutputWriter.write (*method*), 9
- InputReader (*module*), 4
 - InputReader.addNachbarnToLand (*function*), 4
 - InputReader.InputReader (*class*), 4
 - InputReader.InputReader.__init__ (*method*), 4
 - InputReader.InputReader.read (*method*), 4
 - InputReader.zusammenhaengend (*function*), 4
 - InputReader.zusammenhaengendRek (*function*), 4
- Karte (*module*), 5
 - Karte.Karte (*class*), 5
 - Karte.Karte.__init__ (*method*), 5
 - Karte.Karte.berechneMinimum (*method*), 5
 - Karte.Karte.scale (*method*), 5
- Land (*module*), 6–7
 - Land.Land (*class*), 6–7
 - Land.Land.__init__ (*method*), 6
 - Land.Land.__repr__ (*method*), 6
 - Land.Land.__str__ (*method*), 6
 - Land.Land.addNachbar (*method*), 6
 - Land.Land.containsNachbar (*method*), 6
- Main (*module*), 8