Broadcasting

Gegeben sind die Positionen von 3 Atomen:

```
import numpy as np

atoms = np.array([[0., 0., 0.],
                  [3., 3., 3.],
                  [2., 8., 7.]]
                 )
```

sowie ihre Massen:

```
masses = np.array([1., 1., 5.])
```

- Verschieben Sie die Positionen aller 3 Atome um [1., 2., 3.]
- Berechnen Sie den Schwerpunkt der 3 Atome

Selektives Ändern eines Arrays

Gegeben ist folgender Array:

```
riddle = np.array([[ 533.,  502.,  530.,  591.,  552.,  563.,  508.,  581.,  570.,  557.,  5
       [ 512.,  522.,  540.,  510.,  541.,  544.,  521.,  567.,  324.,  388.,  384.,  495.,
       [ 545.,  584.,  521.,  552.,  589.,  583.,  314.,  301.,  461.,  473.,  436.,  469.,
       [ 576.,  517.,  538.,  588.,  550.,  331.,  529.,  532.,  470.,  469.,  471.,  461.,
       [ 542.,  582.,  520.,  525.,  319.,  575.,  543.,  481.,  413.,  474.,  408.,  446.,
       [ 533.,  588.,  569.,  383.,  581.,  541.,  558.,  448.,  411.,  425.,  403.,  463.,
       [ 576.,  552.,  550.,  365.,  587.,  543.,  575.,  586.,  425.,  430.,  493.,  488.,
       [ 567.,  574.,  374.,  513.,  578.,  593.,  539.,  555.,  512.,  407.,  435.,  483.,
       [ 537.,  515.,  398.,  576.,  525.,  530.,  529.,  584.,  544.,  513.,  450.,  420.,
       [ 539.,  542.,  304.,  522.,  565.,  500.,  559.,  530.,  554.,  520.,  526.,  594.,
       [ 544.,  383.,  567.,  520.,  575.,  372.,  393.,  528.,  546.,  571.,  531.,  538.,
       [ 590.,  326.,  524.,  327.,  357.,  501.,  559.,  348.,  566.,  516.,  522.,  505.,
       [ 571.,  362.,  322.,  544.,  251.,  223.,  523.,  544.,  355.,  542.,  597.,  587.,
       [ 585.,  312.,  225.,  228.,  435.,  484.,  215.,  536.,  336.,  522.,  529.,  557.,
       [ 552.,  232.,  444.,  458.,  454.,  435.,  234.,  564.,  503.,  365.,  589.,  555.,
       [ 279.,  522.,  453.,  406.,  419.,  487.,  444.,  514.,  283.,  398.,  521.,  352.,
       [ 227.,  460.,  430.,  405.,  451.,  108.,  544.,  454.,  409.,  379.,  505.,  595.,
       [ 595.,  218.,  209.,  456.,  108.,  584.,  549.,  576.,  450.,  401.,  491.,  515.,
       [ 524.,  564.,  433.,  400.,  108.,  516.,  575.,  413.,  662.,  648.,  666.,  470.,
       [ 524.,  596.,  549.,  439.,  108.,  108.,  437.,  691.,  693.,  671.,  659.,  628.,
       [ 573.,  577.,  595.,  436.,  468.,  108.,  479.,  679.,  686.,  686.,  678.,  627.,
       [ 501.,  544.,  508.,  539.,  430.,  499.,  108.,  474.,  656.,  669.,  601.,  462.,
       [ 527.,  515.,  584.,  581.,  505.,  411.,  462.,  489.,  465.,  454.,  406.,  108.,
       [ 537.,  562.,  565.,  573.,  559.,  500.,  466.,  464.,  410.,  404.,  108.,  108.,
       [ 586.,  596.,  509.,  579.,  530.,  508.,  579.,  480.,  439.,  629.,  631.,  458.,
```

```
[ 536.,  578.,  502.,  522.,  531.,  525.,  587.,  580.,  501.,  487.,  439.,  432.,
[ 548.,  562.,  550.,  561.,  584.,  598.,  572.,  584.,  581.,  544.,  577.,  591.,
```

Wandeln Sie den Array folgendermaßen um:

- Zahlen zwischen 200 und 300 -> 0
- Zahlen zwischen 300 und 400 -> 40
- Zahlen zwischen 400 und 500 -> 156
- Zahlen zwischen 600 und 700 -> 220

Speichern Sie den Ergebnisarray als Bild

```
Image.fromarray(riddle).save("solution.png")
```

Inverse einer Matrix

- Schreiben Sie eine Funktion, die die Inverse einer 2x2 Matrix berechnet.
- Schreiben Sie eine Funktion, die die Adjunkte einer beliebigen (quadratischen) Matrix bestimmt
- Benutzen Sie die vorherige Funktion, um beliebige quadratische Matrizen zu invertieren

Hinweise:

- np.linalg.det berechnet die Determinante einer Matrix

Apfel-Männchen

Gegeben ist eine rekursive Folge für komplexe Zahlen

```
z_{i+1} = z_i**2 + c
```

mit z_0 = 0

Je nachdem wie man c wählt, bleibt diese Folge beschränkt (d.h. der Betrag der komplexen Zahl (np.abs) bleibt unter einer gewissen Grenze), oder sie divergiert.

- Erstellen Sie einen komplexen 2D-Array c, der die komplexe Ebene darstellen soll. Der Realteil soll dabei von -2 bis 2 verlaufen, der Imaginärteil von -i bis i. Dies erreichen Sie durch folgende Schritte:
- Erstellen Sie einen Float-Array für den Realteil und einen für den Imaginärteil:

  ‘python real = np.linspace(-2, 2, 100) imag = np.linspace(-1, 1, 50) * 1j
- Mit folgendem Befehl machen Sie aus dem imag Array einen Spaltenvektor:

  ```
  imag = imag[:, np.newaxis]
  ```

Addieren Sie nun die beiden Vektoren, und Sie sollten einen zweidimensionalen Array erhalten, der Punkte der komplexen Ebene zwischen -2-1j und 2+2j enthält.

- Erstellen Sie einen komplexen Array z, sowie einen Integer-Array counter, die beide die selbe shape wie c haben
- Nun beginnt die Hauptschleife: berechnen Sie in 100 Schritten die Werte z_1 bis z_100, indem Sie ihren Array z immer wieder neu berechnen. Überprüfen Sie nach jeder Rechnung, welche Elemente des z-Arrays noch einen Betrag (np.abs) kleiner 10 haben (-> den resultierenden Boolschen Array können Sie zu counter hinzuaddieren. True wird dann als Wert 1, und False als Wert 0 aufgefasst).
- Zum Schluss können Sie sich ein Bild des Counters ausgeben lassen. Dazu müssen Sie zu Beginn des Skripts folgendes Paket importieren:

  ```
  from PIL import Image
  ```

  anschließend können Sie mit folgenden 3 Zeilen ein Bild speichern:

  ```
  counter = np.array(counter/100. * 255, dtype=np.uint8)
  img = Image.fromarray(counter)
  img.save("mandelbrot.png")
  ```

- Experimentieren Sie auch mit anderen Grenzen des Arrays c. Sie können so weiter in das Apfelmännchen "zoomen".

Monte Carlo

Quelle: https://en.wikipedia.org/wiki/Monte_Carlo_method

Bestimmen Sie die Kreiszahl Pi, indem Sie zufällig Punkte in einem Koordinatensystem der Länge 1 und Höhe 1 verteilen. Wenn Sie einen Kreis ziehen, dessen Zentrum im Ursprung des Koordinatensystems liegt, ist ein Viertel davon in dem Koordinatensystemausschnitt zu sehen. Einige der zufällig gewählten Punkte werden nun innerhalb des Viertelkreises liegen, andere außerhalb. Bestimmen Sie über das Verhältnis <Anzahl der Punkte im Kreis>/<Gesamtanzahl der Punkte> Pi.

Hinweis: Mittels

```
np.random.random(shape)
```

können Sie einen Array mit Zufallszahlen im Intervall [0, 1) erzeugen.

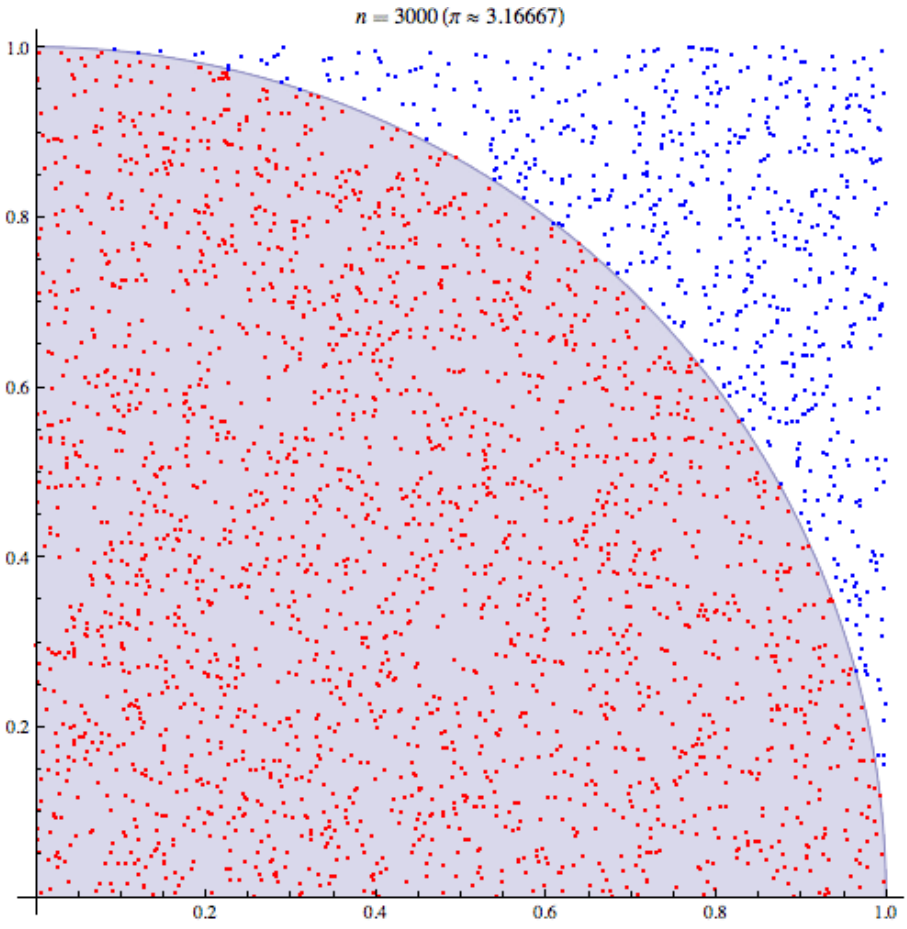

Figure 1: PI_Berechnung