Demokratie und Informatik

→ Gerechtigkeit?

****

Abbildung 1: Waage-Gerechtigkeit-Gesetz,   
[no-longer-here](https://pixabay.com/de/users/no-longer-here-19203/), Lizenz [CC0 1.0](https://creativecommons.org/publicdomain/zero/1.0/deed.de), [Pixabay](https://pixabay.com/de/users/no-longer-here-19203/)

**Inhaltsverzeichnis**

[A Überblick 2](#_Toc62566420)

[B Lernaufgabe 3](#_Toc62566421)

[C Bezug zum Rahmenlehrplan 10](#_Toc62566423)

[D Anhang 12](#_Toc62566424)

# A Überblick

|  |  |
| --- | --- |
| Unterrichtsfach | Informatik |
| Jahrgangsstufen | 9 / 10 |
| Niveaustufen | G / H |
| Zeitrahmen | Zwei Doppelstunden |
| Thema | Wahlverfahren – Berechnung der Sitzverteilung bei einer Wahl |

|  |  |
| --- | --- |
| Themenfelder | Problemlösen, Mit Informationen umgehen |

|  |  |
| --- | --- |
| Kontext | Auswertung der Stimmenverteilung bei einer Wahl zu einem Parlament; Verteilung der Sitze an Parteien oder Gruppierungen |
| Schlagwörter | Demokratie, Wahl, Gerechtigkeit, Python, Hare-Niemeyer-Verfahren, D‘Hondt-Verfahren, Sainte-Laguë-Verfahren |

|  |  |
| --- | --- |
| Zusammenfassung | Die Lernaufgabe ‚Demokratie und Informatik → Gerechtigkeit?‘ soll das Verfahren der Sitzverteilung bei Wahlen für Schülerinnen und Schüler nachvollziehbar und damit transparent machen. Mittels der in einem Jupyter-Notebook befindlichen Python-Programmcodes kann interaktiv das Ergebnis des jeweiligen Codeabschnitts dargestellt werden. So experimentieren die Schüler mit den vorgegebenen Programmcodes, verändern Sie und können so das Verfahren verstehen, sowie dessen Probleme (Alabama-Paradox, New-State-Paradox und Population-Paradox) beurteilen. |

# B Lernaufgabe

# Demokratie und Informatik 🡪 Gerechtigkeit ?



Abbildung 2: Waage-Gerechtigkeit-Gesetz, [no-longer-here](https://pixabay.com/de/users/no-longer-here-19203/), Lizenz [CC0 1.0](https://creativecommons.org/publicdomain/zero/1.0/deed.de), [Pixabay](https://pixabay.com/de/users/no-longer-here-19203/)

Text-1:

Eine Grundlage von Demokratie bildet die gerechte Verteilung von Resourcen innerhalb einer Gemeinschaft. Gruppen können Familien, Vereine, Parteien oder andere Organisationen einer Gesellschaft sein. Einfache Verteilungsfragen können durch eine simple Division gerecht gelöst werden. Teste dies innerhalb der folgenden Python-Codezeilen durch Veränderung der Variablenwerte:

Python-Code-1 [in]:

anzahl\_gruppen = 4

resource = 100

verteilte\_menge = resource / anzahl\_gruppen

print (verteilte\_menge)

Python-Code-1 [out]:

25.0

Text-2:

Eine Besonderheit im Zusammenhang politischer Teilhabe bildet die Verteilung von Sitzen in parlamentarischen Gremien wie Gemeinderäten, Länderparlamenten oder dem Bundestag. Dort hat jeder wahlberechtigte Bürger eine Stimme und die Sitze sollen gerecht gemäß der Stimmenanteile an die Gruppierungen / Parteien verteilt werden, die sich zur Wahl gestellt haben. Dabei ergibt sich ein Problem:

Python-Code-2 [in]:

parteien = ("Gruene","Linke","SPD","CDU","FDP","AFD")

stimmen = [5012, 4444, 2336, 5265, 2356, 4390] # Beispielwerte -> diese kannst du durch Ergebnisse aus deinem Wahlkreis ersetzen

sitze = [0,0,0,0,0,0]

anzahl\_sitze = [30] # Beispielwert

gesamtstimmen = 0

for i in range(0,len(stimmen)):

gesamtstimmen += stimmen[i]

for i in range(0,len(stimmen)):

sitze[i] = stimmen[i] \* anzahl\_sitze[0] / gesamtstimmen

print (parteien[i], "=", sitze[i], "Sitze")

Python-Code-2 [out]:

Gruene = 6.316850817123892 Sitze

Linke = 5.600974667058774 Sitze

SPD = 2.9441667016762594 Sitze

CDU = 6.635718186783179 Sitze

FDP = 2.9693736083686932 Sitze

AFD = 5.532916018989203 Sitze

Text-3:

Ein Sitz ist unteilbar! Was tun? Erster und wahrscheinlich gerechter Gedanke: aufrunden, bzw. abrunden. Also los...

Python-Code-3 [in]:

for i in range(0,len(stimmen)):

sitze[i] = round(stimmen[i] \* anzahl\_sitze[0] / gesamtstimmen)

print (parteien[i], "=", sitze[i], "Sitze")

Python-Code-3 [out]:

Gruene = 6 Sitze

Linke = 6 Sitze

SPD = 3 Sitze

CDU = 7 Sitze

FDP = 3 Sitze

AFD = 6 Sitze

Text-4:

Aber, stimmt nun die Gesamtanzahl der Sitze? Mal schauen:

Python-Code-4 [in]:

summe = 0

for i in range(0,len(sitze)):

summe += sitze[i]

print (summe)

Python-Code-4 [out]:

31

Text-5:

Nein, das stimmt nicht! Und aufrunden ist auch ungerecht, denn die Partei hat ja gar nicht den Stimmanteil bekommen, um den aufgerundet wird. Also haben der englischen Mathematiker Thomas Hare und der deutsche Mathematikprofessor Horst F. Niemeyer folgendes Verfahren entwickelt.

Schaue dir mal eine Erklärung bei Youtube an: <https://www.youtube.com/watch?v=lEtjrySlQZU>[www.youtube.com/watch?v=IEtjrySIQZU](http://www.youtube.com/watch?v=IEtjrySIQZU)

Hier findest du auch noch eine gute Erklärung:https://www.wahlrecht.de/verfahren/hare-niemeyer.html [www.wahlrecht.de/verfahren/hare-niemeyer.html](http://www.wahlrecht.de/verfahren/hare-niemeyer.html)

Dieses Verfahren kann man in Python so programmieren:

Python-Code-5 [in]:

import math

def hare\_niemeyer(liste\_stimmen, sitze):

# liste\_stimmen -> Stimmen aller Parteien (Datentyp: Liste integer)

# sitze -> Anzahl der zu verteilenden Sitze (Datentyp: integer)

summe\_stimmen = sum(liste\_stimmen);

quoten = [float(stimmen)\*sitze/summe\_stimmen for stimmen in liste\_stimmen]

print ("Quoten: ", quoten)

liste\_nachkommastellen = []

for (i, fp) in enumerate(quoten):

liste\_nachkommastellen.append((math.modf(fp)[0], i)) # Vor dem Sortieren: alte Indexposition merken! = Partei

liste\_nachkommastellen.sort()

liste\_nachkommastellen.reverse()

print ("Nachkommastellen sortiert: ",liste\_nachkommastellen)

ergebnisse = [int(math.modf(quote)[1]) for quote in quoten]

print ("Ergebnisse nach dem Abrunden = Grundverteilung: ", ergebnisse)

restsitze = sitze - sum(ergebnisse)

print ("Zu verteilende Restsitze: ", restsitze)

for (nachkommastelle, i) in liste\_nachkommastellen:

ergebnisse[i] += 1

restsitze -= 1

if restsitze == 0: break

print ( "Ergebnisse nach der Verteilung der Restsitze: ", ergebnisse)

return ergebnisse

# Test:

sitze = []

print ("Stimmenverteilung:", parteien[0],stimmen[0],parteien[1],stimmen[1],parteien[2],stimmen[2],parteien[3],stimmen[3],parteien[4],stimmen[4],parteien[5],stimmen[5])

print ("Anzahl der zu verteilenden Sitze: ", anzahl\_sitze[0])

sitze = hare\_niemeyer(stimmen, anzahl\_sitze[0])

Python-Code-5 [out]:

Stimmenverteilung: Gruene 2398 Linke 2810 SPD 2323 CDU 2211 FDP 2473 AFD 2960

Anzahl der zu verteilenden Sitze: 43

Quoten: [6.79499176276771, 7.962438220757825, 6.582471169686985, 6.265107084019769, 7.007512355848435, 8.387479406919276]

Nachkommastellen sortiert: [(0.962438220757825, 1), (0.7949917627677099, 0), (0.5824711696869853, 2), (0.38747940691927596, 5), (0.26510708401976935, 3), (0.007512355848435348, 4)]

Ergebnisse nach dem Abrunden = Grundverteilung: [6, 7, 6, 6, 7, 8]

Zu verteilende Restsitze: 3

Ergebnisse nach der Verteilung der Restsitze: [7, 8, 7, 6, 7, 8]

Text-6:

Etwas anschaulicher wird es, wenn wir die Sitzverteilung statt als nackte Zahlen mit einer Tortengrafik darstellen. Hierbei hilft uns eine Softwarebibliothek namens 'matplotlib'. Das ist ein zusammengesetzter Kunstname, der aus den Wörtern 'Mathematik', 'to plot = engl. darstellen' und 'library = engl. Bibliothek' gebildet wurde. Diese Bibliothek beinhaltet Eigenschaften (z.B. 'style') und Methoden (z.B. 'show'). Im Code wird ein Objekt mit dem Objektnamen 'plt' erzeugt und diesem die Bezeichnungen der Parteien (labels) und die Sitzanzahlen (sitze) übergeben.

Python-Code-6 [in]:

import matplotlib.pyplot as plt

plt.style.use('seaborn-bright') #Aussehen der matplotlib-Grafik

farben = ['lightgreen', 'red', 'orange', 'lightblue', 'yellow', 'purple']

def make\_autopct(values): # Eine notwendige Umdefinition der automatischen Prozentanzeige, damit

def my\_autopct(pct): # die ganzzahlige Sitzanzahl im Inneren der Tortenstuecke angezeigt wird.

total = sum(values)

val = int(round(pct\*total/100.0))

return 'Sitze: {v:d}'.format(p=pct,v=val)

return my\_autopct

plt.figure(figsize=plt.figaspect(2))

plt.pie(sitze, # Daten

labels = parteien, # Bezeichnungen

colors = farben, # Farben

autopct= make\_autopct(sitze), # Werte in den Tortenstücken

shadow=True, # Schatten eingeschaltet

startangle=0) # Startwinkel

plt.axis('equal')

plt.show()

Python-Code-6 [out]:

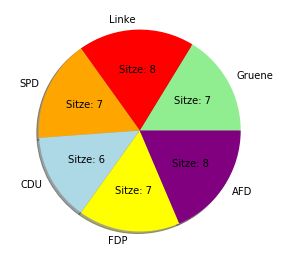


Abbildung 3: Sitzverteilung (Parteien) - Diagramm, M. Abend & A. Mrosch, Lizenz [CC-BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/deed.de), Lernaufgabe „Demokratie und Informatik 🡪 Gerechtigkeit?

Text-7:

Nun wollen wir mit verschiedenen Stimmanteilen und Sitzanzahlen experimentieren. Dazu wird eine weitere Bibliothek namens 'ipwidgets' importiert. Sie stellt uns Einstellregler für die entsprechenden Variablen zur Verfügung. Die eingestellten Werte stehen dann im Arbeitsspeicher unseres Cloudcomputers sofort zur Verfügung und können in einer anderen Python-Zelle des Jupyter-Notebooks verwendet werden, z.B. zum Berechnen und Zeichnen anderer Sitzverteilungen.

Python-Code-7 [in]:

# Um die sogenannten 'widgets' (dt. = Komponente einer Benutzeroberfläche - GUI) nutzen zu können,

# müssen die folgenden iPython-Klassenbibliotheken eingebunden werden.

from \_\_future\_\_ import print\_function

from ipywidgets import interact, interactive, fixed, interact\_manual

import ipywidgets as widgets

Python-Code-8 [in]:

# Die folgenden Zeilen dienen der interaktiven Eingabe der Stimmanteile.

a = widgets.IntSlider(description='Gruene', orientation='vertical', min=0, max=5000, value=100)

b = widgets.IntSlider(description='Linke', orientation='vertical', min=0, max=5000, value=100)

c = widgets.IntSlider(description='SPD', orientation='vertical', min=0, max=5000, value=100)

d = widgets.IntSlider(description='CDU', orientation='vertical', min=0, max=5000, value=100)

e = widgets.IntSlider(description='FDP', orientation='vertical', min=0, max=5000, value=100)

g = widgets.IntSlider(description='AFD', orientation='vertical', min=0, max=5000, value=100)

h = widgets.IntSlider(description='Sitze', orientation='vertical', min=0, max=100, value=30)

ui = widgets.HBox([a, b, c, d, e, g, h])

def f(a, b, c, d, e, g, h):

stimmen[0] = a

stimmen[1] = b

stimmen[2] = c

stimmen[3] = d

stimmen[4] = e

stimmen[5] = g

anzahl\_sitze[0] = h

print((a, b, c, d, e, g, h))

out = widgets.interactive\_output(f, {'a': a, 'b': b, 'c': c, 'd': d, 'e': e, 'g': g, 'h': h})

display(ui, out)

Python-Code-8 [out]:

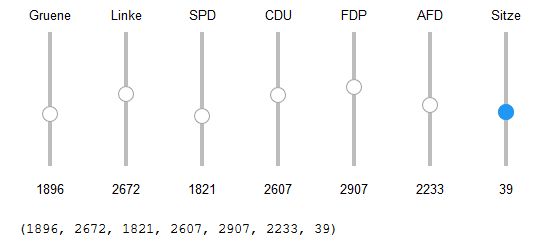


Abbildung 4:Sitzverteilung (Parteien), M. Abend & A. Mrosch, Lizenz [CC-BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/deed.de), Lernaufgabe „Demokratie und Informatik 🡪 Gerechtigkeit?

Text-8:

Aufgabe 1:

Stelle nun unterschiedliche Stimmenverteilungen mit den Reglern ein und beobachte die Sitzverteilungen. Führe hierzu die Codeabschnitte zum Hare-Niemeyer-Verfahren und anschließend die Grafikausgabe jeweils erneut durch Drücken auf 'Run' aus.

Text-9:

Aufgabe 2:

Recherchiere zu den Stichworten:

'Alabama-Paradox', 'New-State-Paradox' und 'Population-Paradox'.

Versuche diese Paradoxien mit simulierten Werten in den obigen Codeabschnitten nachzuvollziehen.

Text-10:

Aufgabe 3:

Diskutiere mit deinen Mitschülern über die Vorteile und Nachteile des Hare-Niemeyer-Sitzzuteilungsverfahrens.

Text-11:

Information:

Nach diesem Verfahren wurde zwischen 1987 und 2005 die Sitzzuteilung im deutschen Bundestag durchgeführt.

# C Bezug zum Rahmenlehrplan

**Evtl. didaktischer Kommentar**

|  |  |
| --- | --- |
| Lernervoraus-  setzungen | Kenntnisse in der Programmiersprache Python |

|  |  |
| --- | --- |
| Kompetenzen | Standards (Die Schülerinnen und Schüler können....) |
| Mit Fachwissen umgehen | Algorithmen entwerfen, implementieren und beurteilen  (RLP Berlin-Brandenburg - Informatik S. 18) |
| Erkenntnisse gewinnen | Die algorithmischen Grundstrukturen in Kombination zielgerichtet anwenden (RLP Berlin-Brandenburg - Informatik S. 18) |
| Kommunizieren | Digitale Visualisierung von Daten  (RLP Berlin-Brandenburg - Informatik S. 28) |
| Bewerten | Informationen in Bezug auf Glaubwürdigkeit, Zuverlässigkeit etc. beurteilen (RLP Berlin-Brandenburg - Informatik S. 16) |

Bezüge zum Basiscurriculum Sprachbildung[[1]](#footnote-1)

|  |  |
| --- | --- |
| **Standards des BC Sprachbildung** | Die Schülerinnen und Schüler können… |
| Produktion / Sprechen | * Sachverhalte und Abläufe veranschaulichen, erklären und interpretieren (G) * zu einem Sachverhalt oder zu Texten Stellung nehmen (G)   (RLP- Berlin - Brandenburg Teil B S. 8) |
| Interaktion | * eigene Gesprächsbeiträge unter Beachtung der Gesprächssituation, des Themas und des Gegenübers formulieren (RLP- Berlin - Brandenburg Teil B S. 10) |

Bezüge zum Basiscurriculum Medienbildung[[2]](#footnote-2)

|  |  |
| --- | --- |
| **Standards des**  **BC Medienbildung** | Die Schülerinnen und Schüler können … |
| Informieren | * bei der Bearbeitung von Lern- und Arbeitsaufgaben mediale Quellen gezielt zur Informationsgewinnung und zum Wissenserwerb nutzen (RLP- Berlin - Brandenburg Teil B S. 15) |

Bezüge zu übergreifenden Themen[[3]](#footnote-3)

|  |  |
| --- | --- |
| Demokratiebildung | RLP Berlin-Brandenburg - Fachübergreifende Kompetenzentwicklung S.26: ‚Mit dem Wissen um das Wesen demokratischen Handelns in einem demokratisch verfassten Staat ...‘ |

Bezüge zu anderen Fächern

|  |
| --- |
| * **Mathematik:** Beim Hare-Niemeyer-Verfahren handelt es sich um ein mathematisches Verfahren mit dem Ziel, die Sitzverteilung in einem Parlament möglichst gerecht anhand der Stimmenverteilung zu berechnen. |

**Inklusive Aspekte der Lernaufgabe:**

|  |  |
| --- | --- |
|  | Standards der iMINT-Akademie |
| Aufgabenstellungen | Lesen, Experiment, Recherche, Diskussion |
| Methode | Visualisierung der numerischen Experimente zur Sitzverteilung |
| Experimente | Interaktives Arbeiten in den Codeabschnitten durch Anpassung der Python-Codeabschnitte und durch die Variation von Parametern |
| IT | Cloudbasierte Bearbeitungsmöglichkeit innerhalb eines Jupyter-Notebooks |

# D Anhang

**Didaktische Hinweise**

Der Aufbau des Material erfolgt nach dem Muster zunächst zu lesender Textinformation und jeweils anschließender Codeanalyse und Codeausführung. Hierbei sollen die Schülerinnen und Schüler immer im eigenen Lerntempo vorgehen. Der Schwierigkeitsgrad der einzelnen Abschnitte steigt von einfach zu komplex über die Abschnitte eins bis acht.

Von den Aufgabenstellungen ist die Aufgabe 1 durch alle Schülerinnen und Schüler zu lösen. Die Aufgabe 2 stellt eine Differenzierungsmöglichkeit im Umfang und in der Rechercheleistung dar. Die Quellen [2] bis [4] können auch durch die Lehrkraft vorgegeben werden.

Die Aufgabe 3 (Diskussion der Vor- und Nachteile des Hare-Niemeyer-Verfahrens) kann von allen Schülerinnen und Schüler durchgeführt werden. Die Lehrkraft entscheidet auf Grundlage ihrer Lerngruppe über die Organisationsform (Plenum, Kleingruppen), sowie über die Präsentation des Diskussionsergebnisses (z.B. Tafelbild mit Pro- und Contraargumenten oder verbale Zusammenfassung)

**Material für den Einsatz dieser Lernaufgabe**

Das Material steht als Datei ‚Gerechtigkeit.ipynb‘ für ein Jupyter-Notebook zur Verfügung. Ein solches kann entweder bei einem Anbieter für Cloud-Computing, wie Microsoft Azure Notebooks oder Google Colaboratory eingerichtet werden oder durch Installation eines eigenen Jupyter-Servers realisiert werden.

|  |  |
| --- | --- |
| Anzahl | Name des Materials |
| 1 | **Jupyter-Notebook:** **Gerechtigkeit.ipynb** |

**Musterlösung der Lernaufgabe**

Es gibt keine separate Musterlösung der Lernaufgabe, da das Jupyter-Notebook den Weg von der Problemstellung bis zur Lösung beinhaltet. Die Schülerinnen und Schüler müssen jedoch interaktiv tätig werden, um die einzelnen Codeabschnitte auszuführen. Der experimentelle Teil, der in drei Arbeitsaufträgen mündet, kann anhand der Quellen [2] bis [4] erschlossen werden. Je nach Könnensstand der Schülerinnen und Schüler ist es der Lehrkraft jedoch möglich, einzelne Codeabschnitte auszuwählen, diese aus dem Jupyter-Notebook entfernen und durch die Schülerinnen und Schüler eigenhändig Python-Codeabschnitte erstellen zu lassen.

Die dritte Aufgabenstellung besteht in einer fachlich fundierten Diskussion über die Sitzzuteilungsverfahren. Sind diese gerecht? Welches ist besser? Hierfür müssen die Schülerinnen und Schüler ihr in der Lernaufgabe erworbenes Wissen anwenden und argumentativ mit den Mitschülern austauschen. Dieser demokratische Diskurs stellt das fachübergreifende Lernziel dieser Lernaufgabe dar.

**Quellen**

Hinweis: Aus lizenzrechtlichen Gründen dürfen die verlinkten Inhalte nicht gespeichert oder verändert werden, sofern sie nicht unter einer entsprechenden Lizenz stehen.

[1] [www.wahlrecht.de/verfahren/hare-niemeyer.html](http://www.wahlrecht.de/verfahren/hare-niemeyer.html) [Letzter Abruf: 12.5.2019]https://www.wahlrecht.de/verfahren/hare-niemeyer.html

[2] [www.wahlrecht.de/verfahren/paradoxien/alabama.html](http://www.wahlrecht.de/verfahren/paradoxien/alabama.html) [Letzter Abruf: 12.5.2019]

[3] [www.wahlrecht.de/verfahren/paradoxien/parteizuwachs.html](http://www.wahlrecht.de/verfahren/paradoxien/parteizuwachs.html)   
[Letzter Abruf: 12.5.2019]

[4] [www.wahlrecht.de/verfahren/paradoxien/population.html](http://www.wahlrecht.de/verfahren/paradoxien/population.html) [Letzter Abruf: 12.5.2019][http](http://www.wahlrecht.de/verfahren/paradoxien/population.html)

[5] pixabay.com/de/illustrations/waage-gerechtigkeit-gesetz-316888/   
[Letzter Abruf: 12.5.2019]

**Bildnachweis**

|  |  |  |
| --- | --- | --- |
| Bildtitel | Seite | Bildquelle |
| Waage-Gerechtigkeit-Gesetz | 1, 3 | Waage-Gerechtigkeit-Gesetz, [no-longer-here](https://pixabay.com/de/users/no-longer-here-19203/), Lizenz [CC0 1.0](https://creativecommons.org/publicdomain/zero/1.0/deed.de), [Pixabay](https://pixabay.com/de/users/no-longer-here-19203/) |
| Sitzverteilung | 8 | Sitzverteilung (Parteien) - Diagramm, M. Abend & A. Mrosch, Lizenz [CC-BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/deed.de), Lernaufgabe „Demokratie und Informatik 🡪 Gerechtigkeit? |
| Sitzverteilung 2 | 6 | Sitzverteilung (Parteien), M. Abend & A. Mrosch, Lizenz [CC-BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/deed.de), Lernaufgabe „Demokratie und Informatik 🡪 Gerechtigkeit? |

1. vgl. Rahmenlehrplan Jahrgangsstufen 1-10, Teil B, S. 6-10, Berlin, Potsdam 2015 [↑](#footnote-ref-1)
2. vgl. Rahmenlehrplan Jahrgangsstufen 1-10, Teil B, S. 15-22, Berlin, Potsdam 2015 [↑](#footnote-ref-2)
3. vgl. Rahmenlehrplan Jahrgangsstufen 1-10, Teil B, S. 24ff, Berlin, Potsdam 2015 [↑](#footnote-ref-3)