```
In [ ]: # Initialize Otter
        import otter
        grader = otter.Notebook("CT-04-Hurricanes.ipynb")
```

1 1 Hurrikans und Wörterbücher

1.1 1.1 Structure of Arrays vs Array of Structures (Zusatzinformation)

Zwei der Datenhaltungen die wir häufig antreffen sind sog. *Structure of Arrays SoA* und *Array of Structure* (siehe auch AoS and SoA).

1.1.1 1.1.1 Structure of Arrays

Bei *Structure of Arrays* (Struktur von Listen) wird je ein Typ von Datum für alle Objkte in einer Liste gehalten. Alle Objekte zusammengenommen bilden sich dann aus den ganzen Listen, wobei jede Liste soviele Einträge besitzt, wie es Objekte gibt. Zum Beispiel:

```
x = [1.2, 2.3, 4.5, ...]
y = [3.4, 2.1, 3.3, ...]
name = ['rect1', 'rect2', 'rect3', ...]
width = [1.4, 2.1, 3.3, ...]
height = [91.4, 28.1, 6.5, ...]
rectangles = {'x': x, 'y': y, 'name': name, 'width': width, 'height': height}
```

Structure of Arrays SoA werden dann verwendet, wenn man eine bestimmte Operation parallel auf allen Daten eines Typs ausführen möchte. So verwenden wir auf Grafikkarten *Structures of Arrays SoA*. Auch verwenden wir beim wissenschaftlichen Rechnen häufig *Structures of Arrays SoA*.

1.1.2 1.1.2 Array of Structures

Die zweite Form nennt man *Array of Structures AoS* (Liste von Strukturen). Hierbei werden die Daten eines Objekts gebündelt und eine Liste mit all diesen Bündeln erstellt. Zum Beispiel:

```
rect1 = {'name': 'rect1', 'x': 1.2, 'y': 3.4, 'width': 1.4, 'height': 91.4}
rect2 = {'name': 'rect2', 'x': 2.3, 'y': 2.1, 'width': 2.1, 'height': 28.1}
rect3 = {'name': 'rect3', 'x': 4.5, 'y': 3.3, 'width': 3.3, 'height': 6.5}
...
objects = [rect1, rect2, rect3, ...]
```

Ein *Array of Structures* verwenden wir hingegen in der herkömmlichen sog. objektorientierten Programmierung.

Beide Formen haben Vor- und Nachteile. In der folgenden Aufgabe werden wir sowohl mit *Structures of Arrays SoA* als auch mit einem *Array of Structures* arbeiten.

1.2 1.2 Aufgabenbeschreibung

Hurrikans, auch bekannt als Wirbelstürme, Zyklone oder Taifune, sind eine der mächtigsten Naturkräfte der Erde. Aufgrund des vom Menschen verursachten Klimawandels haben Anzahl und Intensität von Hurrikanen zugenommen, was eine bessere Vorbereitung der vielen von ihnen verwüsteten Gemeinschaften erfordert. Als besorgte Umweltschützer*innen möchten Sie sich Daten über die stärksten Wirbelstürme ansehen, die sich ereignet haben.

Im folgenden sind die Daten zu den 34 stärksten atlantischen Hurrikans in einem *Structure of Arrays* `hurricanes` in der Form eines Wörterbuchs `dict` bestehend aus Listen `list` zusammengefasst. Die Daten umfassen:

- `name`: Namen der Hurrikans
- `month`: Monate, in denen die Wirbelstürme aufgetreten sind
- `year`: Jahre, in denen die Wirbelstürme aufgetreten sind
- `speed`: maximal anhaltende Winde (Kilometer pro Stunde) der Hurrikans
- `areas`: Liste der verschiedenen Gebiete, die von jedem der Hurrikans betroffen sind
- `deaths`: Gesamtzahl der durch jeden der Hurrikans verursachten Todesfälle

Die Daten sind so organisiert, dass die Daten für Index i zum i-ten Hurrikan gehören. Anders ausgedrückt diese Daten gehören zu einem modellierten Objekt, den einen Hurrikan.

```
In [1]: # name of the hurricane
        name = ['Cuba I', 'San Felipe II Okeechobee', 'Bahamas', 'Cuba II', 'CubaBrownsville', 'Tampico

        # month of appearence
        month = ['October', 'September', 'September', 'November', 'August', 'September', 'September', 'S

        # year of appearence
        year = [1924, 1928, 1932, 1932, 1933, 1933, 1935, 1938, 1953, 1955, 1961, 1961, 1967, 1969, 197

        # maximal wind speed
        speed = [265, 260, 260, 275, 260, 260, 285, 260, 260, 275, 275, 260, 260, 275, 260, 275, 275, 2

        # regions the hurricane hit
        areas = [['Central America', 'Mexico', 'Cuba', 'Florida', 'The Bahamas'], ['Lesser Antilles', 'T

        # damage in million
        damage = ['Schaden nicht aufgezeichnet', '100M', 'Schaden nicht aufgezeichnet', '40M', '27.9M',
```

```
# caused deaths
deaths = [90,4000,16,3103,179,184,408,682,5,1023,43,319,688,259,37,11,2068,269,318,107,65,19325

hurricanes = dict(
    name=name,
    month=month,
    year=year,
    speed=speed,
    areas=areas,
    damage=damage,
    deaths=deaths)
#hurricanes
```

Aufgabe 1 (Typumwandlung). Die Schäden wurden als Zeichenketten der Form `'100M'` bzw. `'20.9B'` abgelegt wobei `'M'` für Millionen und `'B'` für Billionen steht. Damit lässt sich schlecht rechnen.

Schreiben Sie eine Funktion `damage_to_float(damage)`, welche aus der Schadensliste `damage` bzw. `hurricanes['damage']` eine neue neue Liste aus Fließkommazahlen erzeugt und zurückgibt. Ersetzen Sie dabei `'Schaden nicht aufgezeichnet'` durch `None`.

Führen Sie Ihre Umwandlung duch und legen Sie die Liste in `hurricanes['damage']` ab.

In [2]:
```
# BEGIN SOLUTION
def to_float(text):

    if text == 'Schaden nicht aufgezeichnet':
        return None
    elif text[-1] == 'M':
        return float(text[:-1:]) * 1e6
    else:
        return float(text[:-1:]) * 1e9

def damage_to_float(damage):
    return list(map(lambda text: to_float(text), damage))

hurricanes['damage'] = damage_to_float(hurricanes['damage'])
# END SOLUTION
```

In [ ]:
```
grader.check("q1")
```

Aufgabe 2 (Kostenschätzung). Schreiben Sie eine Funktion `estimate_costs(damage_list)`, die die Gesamtkosten, aller aufgezeichneten Hurrikans schätzt. Fehlt eine Aufzeichnung werten Sie dies als 0 Kosten (Unterschätzung). Führen Sie die Schätzung durch. Nutzten Sie die Ergebnisse aus Aufgabe 1 (d.h. `damage_list` ist eine Liste aus Fließkommazahlen und `None`).

```
In [6]: def estimate_costs(damage_list):
            # BEGIN SOLUTION
            estimate = 0
            for damage in damage_list:
                estimate += damage if damage != None else 0
            return estimate
            # END SOLUTION
```

```
In [ ]: grader.check("q2")
```

Aufgabe 3 (Durchschnittsgeschwindigkeit). Schreiben Sie eine Funktion `avg_speed(speed_list)`, die die durchschnittliche Maximalgeschwindigkeit der Hurrikans berechnet. Dabei ist `speed_list` eine Liste aus Zahalen wie `hurricanes['speed']`.

```
In [10]: def avg_speed(speed_list):
             # BEGIN SOLUTION
             if len(speed_list) == 0:
                 return 0
             return sum(speed_list) / len(speed_list)
             # END SOLUTION
```

```
In [ ]: grader.check("q3")
```

Aufgabe 4 (Array of Structures). Schreiben Sie eine Funktion `extract(hurricanes, i)`, welche den `i`-ten Hurrikan Ihres Wörterbuchs `hurricanes` in ein Wörterbuch der folgenden Form umwandelt:

```
{'name': 'Cuba I',
 'month': 'October',
 'year': 1924,
 'speed': 265,
 'areas': ['Central America', 'Mexico', 'Cuba', 'Florida', 'The Bahamas'],
 'damage': None,
 'deaths': 90}, ...
```

umwandelt. **Hinweis:** Die Schlüssel des Ergebnisses gleichen den Schlüsseln des Wörterbuchs `hurricanes`!

```
In [13]: def extract(hurricanes, i):
             # BEGIN SOLUTION
             hurricane = {}
             for key in hurricanes.keys():
                 hurricane[key] = hurricanes[key][i]
             return hurricane
             # END SOLUTION
```

```
In [ ]: grader.check("q4")
```

Aufgabe 5 (Array of Structures). Schreiben Sie eine Funktion `to_array_of_structures(hurricanes)`, die aus dem Wörterbuch `hurricanes` eine Liste aus Wörterbüchern erstellt und zurückliefert. (Verwenden Sie Ihre zuvor definierte Funktion `extract(hurricanes, i)`.

```
In [15]: def to_array_of_structures(hurricanes):
             # BEGIN SOLUTION
             hurricane_list = []
             for i in range(len(hurricanes['damage'])):
                 hurricane_list.append(extract(hurricanes, i))
             return hurricane_list
             # END SOLUTION
         hurricane_list = to_array_of_structures(hurricanes)
```

Aufgabe 6 (Gebiete). Schreiben Sie eine Funktion `extract_areas(hurricane_list)` die alle Gebiete in denen ein Hurrikan aufgetreten ist aus Ihrer Liste aus Hurrikanobjekten zusammenfasst und zurückliefert. Es sollen keine doppelten Einträge entstehen. Welche Datenstruktur eignet sich dafür?

```
In [16]: def extract_areas(hurricane_list):
             # BEGIN SOLUTION
             areas = set()

             for hurricane in hurricane_list:
                 areas |= set(hurricane['areas'])
             return areas
             # END SOLUTION
         areas = extract_areas(hurricane_list)
```

Aufgabe 7 (Gebiete). Wir möchten nun anstatt Hurrikans unsere Gebiete besser analyiseren. Dazu haben wir eine Funktion `extract_area_information(hurricane_list)` geschrieben, die ein Wörterbuch aus Wörterbüchern aus der Liste der Hurikans erzeugt.

Beschreiben Sie in Ihren Worten **was** `extract_areas(hurricane_list)` erzeugt. Führen Sie die Funktion aus. Was können Sie aus den Einträgen des Wörterbuchs herauslesen?

```
In [17]: def extract_area_information(hurricane_list):
             areas = {}
             for hurricane in hurricane_list:
                 for areaname in hurricane['areas']:
                     if areaname not in areas:
                         areas[areaname] = {}

                     if 'years' not in areas[areaname]:
                         areas[areaname]['years'] = []
```

```
                if 'avg_speed' not in areas[areaname]:
                    areas[areaname]['avg_speed'] = 0
                if 'avg_damage' not in areas[areaname]:
                    areas[areaname]['avg_damage'] = 0
                if 'avg_deaths' not in areas[areaname]:
                    areas[areaname]['avg_deaths'] = 0

                nareas = len(hurricane['areas'])
                areas[areaname]['years'].append(hurricane['year'])
                areas[areaname]['avg_damage'] += hurricane['damage'] / nareas if hurricane['damage
                areas[areaname]['avg_deaths'] += hurricane['deaths'] / nareas
                areas[areaname]['avg_speed'] += hurricane['speed']

        for areaname, area in areas.items():
            areas[areaname]['avg_speed'] /= len(areas[areaname]['years'])
        return areas
    extract_area_information(hurricane_list)
```

```
Out[17]: {'Central America': {'years': [1924,
            1955,
            1961,
            1971,
            1988,
            1998,
            2005,
            2007,
            2018],
           'avg_speed': 271.6666666666667,
           'avg_damage': 26816216666.666668,
           'avg_deaths': 7451.516666666667},
          'Mexico': {'years': [1924, 1967, 1971, 1977, 1980, 1988, 2005],
           'avg_speed': 270.7142857142857,
           'avg_damage': 2058183333.3333333,
           'avg_deaths': 402.6833333333334},
          'Cuba': {'years': [1924, 1932, 1933, 1969, 2005, 2017],
           'avg_speed': 272.5,
           'avg_damage': 17522246666.666668,
           'avg_deaths': 785.9666666666666},
          'Florida': {'years': [1924, 1933, 1935, 1992, 2005, 2017],
           'avg_speed': 275.0,
           'avg_damage': 29438913333.333336,
           'avg_deaths': 209.06666666666663},
          'The Bahamas': {'years': [1924, 1928, 1932, 1932, 1933, 1935, 1992],
           'avg_speed': 268.57142857142856,
           'avg_damage': 8870580000.0,
           'avg_deaths': 1682.2333333333331},
          'Lesser Antilles': {'years': [1928, 1932, 1955, 2017],
           'avg_speed': 271.25,
           'avg_damage': 18384566666.666668,
           'avg_deaths': 2640.0666666666666},
          'United States East Coast': {'years': [1928, 1979, 1989, 2016],
           'avg_speed': 265.0,
           'avg_damage': 8815000000.0,
```

```
 'avg_deaths': 2208.1},
'Atlantic Canada': {'years': [1928, 1953, 2016],
 'avg_speed': 261.6666666666667,
 'avg_damage': 3045666666.6666665,
 'avg_deaths': 1122.2666666666667},
'Northeastern United States': {'years': [1932, 1938],
 'avg_speed': 260.0,
 'avg_damage': 102000000.0,
 'avg_deaths': 235.33333333333334},
'Jamaica': {'years': [1932, 1933, 1988, 2005],
 'avg_speed': 270.0,
 'avg_damage': 1681666666.6666667,
 'avg_deaths': 677.0166666666667},
'Cayman Islands': {'years': [1932],
 'avg_speed': 275.0,
 'avg_damage': 6666666.666666667,
 'avg_deaths': 517.1666666666666},
'Bermuda': {'years': [1932, 1953],
 'avg_speed': 267.5,
 'avg_damage': 7333333.333333334,
 'avg_deaths': 518.8333333333333},
'Texas': {'years': [1933, 1961, 1967, 2005],
 'avg_speed': 263.75,
 'avg_damage': 436080000.0,
 'avg_deaths': 283.7166666666667},
'Tamaulipas': {'years': [1933],
 'avg_speed': 260.0,
 'avg_damage': 5580000.0,
 'avg_deaths': 35.8},
'Yucatn Peninsula': {'years': [1933, 1980, 1998],
 'avg_speed': 276.6666666666667,
 'avg_damage': 2379166666.666667,
 'avg_deaths': 6600.916666666667},
'Georgia': {'years': [1935],
 'avg_speed': 285.0,
 'avg_damage': 0,
 'avg_deaths': 81.6},
'The Carolinas': {'years': [1935],
 'avg_speed': 285.0,
 'avg_damage': 0,
 'avg_deaths': 81.6},
'Virginia': {'years': [1935],
 'avg_speed': 285.0,
 'avg_damage': 0,
 'avg_deaths': 81.6},
'Southeastern United States': {'years': [1938],
 'avg_speed': 260.0,
 'avg_damage': 102000000.0,
 'avg_deaths': 227.33333333333334},
'Southwestern Quebec': {'years': [1938],
 'avg_speed': 260.0,
 'avg_damage': 102000000.0,
 'avg_deaths': 227.33333333333334},
'New England': {'years': [1953],
```

```
 'avg_speed': 260.0,
 'avg_damage': 666666.6666666666,
 'avg_deaths': 1.6666666666666667},
'Louisiana': {'years': [1961],
 'avg_speed': 275.0,
 'avg_damage': 108666666.66666667,
 'avg_deaths': 14.333333333333334},
'Midwestern United States': {'years': [1961],
 'avg_speed': 275.0,
 'avg_damage': 108666666.66666667,
 'avg_deaths': 14.333333333333334},
'The Caribbean': {'years': [1967, 1971, 1979, 1980, 1989, 2004, 2007, 2017],
 'avg_speed': 270.625,
 'avg_damage': 25602350000.0,
 'avg_deaths': 1480.1666666666665},
'United States Gulf Coast': {'years': [1969, 1971, 1992, 2004, 2005, 2005],
 'avg_speed': 271.6666666666667,
 'avg_damage': 85816350000.0,
 'avg_deaths': 1182.25},
'South Texas': {'years': [1980],
 'avg_speed': 290.0,
 'avg_damage': 310000000.0,
 'avg_deaths': 67.25},
'Venezuela': {'years': [1988, 2004, 2016],
 'avg_speed': 271.6666666666667,
 'avg_damage': 12206666666.666668,
 'avg_deaths': 225.53333333333333},
'Hispaniola': {'years': [1988],
 'avg_speed': 285.0,
 'avg_damage': 1420000000.0,
 'avg_deaths': 63.6},
'South Florida': {'years': [1998],
 'avg_speed': 280.0,
 'avg_damage': 2066666666.6666667,
 'avg_deaths': 6441.666666666667},
'Greater Antilles': {'years': [2003, 2005],
 'avg_speed': 275.0,
 'avg_damage': 11142500000.0,
 'avg_deaths': 41.75},
'Bahamas': {'years': [2003, 2005],
 'avg_speed': 270.0,
 'avg_damage': 63842500000.0,
 'avg_deaths': 930.75},
'Eastern United States': {'years': [2003],
 'avg_speed': 265.0,
 'avg_damage': 1342500000.0,
 'avg_deaths': 12.75},
'Ontario': {'years': [2003],
 'avg_speed': 265.0,
 'avg_damage': 1342500000.0,
 'avg_deaths': 12.75},
'Windward Islands': {'years': [2005],
 'avg_speed': 260.0,
 'avg_damage': 252500000.0,
```

```
 'avg_deaths': 4.25},
'Nicaragua': {'years': [2007],
 'avg_speed': 275.0,
 'avg_damage': 360000000.0,
 'avg_deaths': 66.5},
'Honduras': {'years': [2007],
 'avg_speed': 275.0,
 'avg_damage': 360000000.0,
 'avg_deaths': 66.5},
'Antilles': {'years': [2016],
 'avg_speed': 265.0,
 'avg_damage': 3020000000.0,
 'avg_deaths': 120.6},
'Colombia': {'years': [2016],
 'avg_speed': 265.0,
 'avg_damage': 3020000000.0,
 'avg_deaths': 120.6},
'Cape Verde': {'years': [2017],
 'avg_speed': 280.0,
 'avg_damage': 10800000000.0,
 'avg_deaths': 23.0},
'British Virgin Islands': {'years': [2017],
 'avg_speed': 280.0,
 'avg_damage': 10800000000.0,
 'avg_deaths': 23.0},
'U.S. Virgin Islands': {'years': [2017],
 'avg_speed': 280.0,
 'avg_damage': 10800000000.0,
 'avg_deaths': 23.0},
'Virgin Islands': {'years': [2017],
 'avg_speed': 275.0,
 'avg_damage': 18320000000.0,
 'avg_deaths': 611.4},
'Puerto Rico': {'years': [2017],
 'avg_speed': 275.0,
 'avg_damage': 18320000000.0,
 'avg_deaths': 611.4},
'Dominican Republic': {'years': [2017],
 'avg_speed': 275.0,
 'avg_damage': 18320000000.0,
 'avg_deaths': 611.4},
'Turks and Caicos Islands': {'years': [2017],
 'avg_speed': 275.0,
 'avg_damage': 18320000000.0,
 'avg_deaths': 611.4},
'United States Gulf Coast (especially Florida Panhandle)': {'years': [2018],
 'avg_speed': 260.0,
 'avg_damage': 12550000000.0,
 'avg_deaths': 37.0}}
```


Type your answer here, replacing this text.

Die Schlüssel des Wörterbuchs welches uns die Funktion `extract_area_information(hurricane_list)` erstellt sind die Namen der Gebiete. Jeder Wert ist ein Wörterbuch welches die Informationen über das Gebiet bzgl. Auswirkungen der Hurrikans enthält. Da aus den Daten nicht hervorgeht wie viel Schaden und Tote und für ein Gebiet insgesamt entstanden sind, wird dieser auf die Gebiete in gleichen Anteilen aufgeteilt (was natürlich eine grobe Schätzung ist). Zudem wird die durschnittliche Maximalgeschwindigkeit der Hurrikans, welche das Gebiet getroffen haben berechnet.

To double-check your work, the cell below will rerun all of the autograder tests.

```
In [ ]: grader.check_all()
```

1.3 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit.

```
In [ ]: grader.export(pdf=False, force_save=True)
```