6. Erweiterte Pythonsyntax und Colormaps

1 String Operationen

- **str:** Built-In Klasse: verwendet zum Konvertieren von Objekten, hat auch einige nützliche Methoden zum Modifizieren von Strings.
- **string:** Modul mit einigen nützlichen Klassen und Methoden für Operationen mit Strings
- Operationen der Klasse str werden von dem Objekt selbst aufgerufen, also entweder:

```
myString.strip()
```

oder

```
myObject.str.strip()
```

falls Objekt noch in String konvertiert werden muss. Die Befehle können gechained werden, also ohne die Neudefinition der Variable in einem Statement (≡ einer Zeile):

```
myObject.str.strip().str.replace('a','o').\
str.replace(' ','_')
```

- **myString.strip([chars]):** Entfernt chars vom Beginn und vom Ende des Strings, gibt eine Kopie des Strings zurück. Wenn Argument leer bleibt, werden Whitespaces (Leerräume) entfernt, also Leerzeichen, return-Zeichen, newline-Zeichen, tabulatoren, etc.
- **myString.split(separator, maxsplit):** Gibt eine Liste an Strings zurück, die den ursprünglichen String an einem bestimmten char (seperator) splitted.
- **myString.isnumeric():** Eine Boolean Abfrage, die überprüft, ob alle chars des Strings zu einer numerischen Klasse gehören.
- **myString.lower() oder upper():** Gibt einen String in nur Kleinbuchstaben oder nur Großbuchstaben zurück.
- **myString.find(sub, start end):** In einem String wird ein Substring gesucht, der erste Index des ersten Auftretens wird zurückgegeben. Es kann ein Suchbereich (Start:Ende) angegeben werden.

2 List Comprehensions

Methode um Schleifen über Loops bei der Erzeugung von Listen zu verkürzen.

```
newList = [expression for item in Oldlist if conditional]
```

Das erstezt folgende Listenoperation:

```
newList = []
for item in Oldlist:
    if conditional:
        newList.append(expression)
```

3 Lambda Expressions

- wird verwendet, um **anonyme Funktionen** zu erzeugen.
- können nur aus einem Statement bestehen
- geben automatisch Resultat des Statements zurück
- können ohne weitere Variablendefinition weitergegeben werden
- haben keinen Namen und keinen Docstring, also keine Dokumentation

```
lambda arguments : expression
```

Die Nutzung der Lambdafunktion:

```
Dataframe.apply(lambda x: x**2)
```

erstezt also

```
def square(x):
    """
    Function to return the square of an input value
    :param: x: numeric
    :return: x**2
    """
    return x**2

for idx, item in Dataframe.items():
    Dataframe[idx] = square(item)
```

4 Colormaps

- Colormaps werden benutzt um eine gleichmäßige Verteilung von Farben für eine bestimmte Menge an Kurven oder Werte (wenn es ein Bild ist) darzustellen.
- Sie können selbst erstellt werden, aber es lohnt sich die vorgefertigten Maps von matplotlib zu verwenden: `https://matplotlib.org/users/colormaps.html#vischeck`
- durch die Verwendung von Colormaps mit nicht linearer Helligkeitsverteilung können dem Auge Features suggeriert werden, die nicht aus den Daten stammen, man sollte also die Map bewusst wählen: `https://arxiv.org/ftp/arxiv/papers/1712/1712.01662.pdf`
- Zuordnung von Colormaps zu bestimmten Daten:
  - Sequentiell: Daten hängen miteinander zusammen und es gibt einen Verlauf ohne Herausstechende Features.

  - Diverging: es gibt einen zentralen Punkt (z.B. die Null bei +- Werten), der herausgehoben werden sollte

  - Miscellaneous: Daten haben keine direkte Beziehung zueinander (z.B. Bundesländer)