Python und NumPy Cheat Sheet

Onur Yilmaz

Grundlagen

Hier werden die grundlegenden Konzepte und Funktionen von Python, IPython, Jupyter-Notebook, NumPy und Hypothesentests behandelt.

Python, IPython und Jupyter-Notebook

Python ist eine beliebte Programmiersprache, die für ihre Einfachheit und Lesbarkeit bekannt ist. IPython ist eine interaktive Python-Shell, die zusätzliche Funktionen bietet, wie z.B. Autovervollständigung und Tab-Vervollständigung. Jupyter Notebook ist eine webbasierte Umgebung, die es ermöglicht, Code, Text und Visualisierungen in einer einzigen Datei zu kombinieren.

Beispiel: Ausgabe von 'Hallo Welt' in Python

```
print("Hallo Welt")
```

In Python integrierte Datenstrukturen, Funktionen und Dateien

In Python sind verschiedene Datenstrukturen wie Tupel, Listen, Dictionaries und Sets integriert. Es gibt auch eine Vielzahl von integrierten Funktionen für mathematische Operationen, Zeichenkettenmanipulationen, Dateioperationen usw.

Beispiel: Erstellen und Manipulieren einer Liste in Python

```
my_list = [1, 2, 3, 4, 5]
print(my_list)

my_list.append(6)
print(my_list)

```

```
my_list.remove(3)
print(my_list)
```

Tupel, Listen, Dictionarys und Sets

Tupel, Listen, Dictionaries und Sets sind grundlegende Datenstrukturen in Python mit unterschiedlichen Eigenschaften und Verwendungszwecken.

Beispiel: Verwendung von Tupeln in Python

```
my_tuple = (1, 2, 3)
print(my_tuple)

print(my_tuple[0])

# my_tuple[0] = 4

my_tuple = ("Hallo", 3.14, True)
print(my_tuple)
```

Funktionen, Gültigkeitsbereiche und mehrere Rückgabewerte

Funktionen sind benannte Codeblöcke, die eine bestimmte Aufgabe ausführen. Sie können Parameter akzeptieren und Werte zurückgeben. Gültigkeitsbereiche beziehen sich auf den Bereich, in dem Variablen sichtbar und zugänglich sind. Python ermöglicht auch die Rückgabe von mehreren Werten aus einer Funktion.

Beispiel: Funktion mit mehreren Rückgabewerten in Python

```
def get_name_and_age():
name = "John"
age = 30
return name, age

result_name, result_age = get_name_and_age()
print("Name:", result_name)
print("Alter:", result_age)
```

Grundlagen von NumPy

Beispiel: Erzeugung und Operationen mit NumPy-Arrays

```
import numpy as np

my_array = np.array([1, 2, 3, 4, 5])
print(my_array)

my_array_squared = my_array ** 2
print(my_array_squared)

array_sum = np.sum(my_array)
print(array_sum)
```

Beispiel: Transponieren eines NumPy-Arrays

```
import numpy as np

my_array = np.array([[1, 2, 3], [4, 5, 6]])
print("Original-Array:")
print(my_array)

transposed_array = np.transpose(my_array)
print("Transponiertes Array:")
print(transposed_array)
```

Universelle Funktionen, mathematische und statische Methoden und Sortieren von Arrays

NumPy bietet universelle Funktionen für die elementweise Berechnung auf Arrays, mathematische Funktionen und statische Methoden zur statistischen Analyse von Daten. Darüber hinaus gibt es Funktionen zum Sortieren von Arrays.

Beispiel: Anwendung einer universellen Funktion auf ein NumPy-Array

```
import numpy as np

my_array = np.array([1, 2, 3, 4, 5])
print("Original-Array:")
print(my_array)
```

```
6
7       squared_array = np.square(my_array)
8       print("Quadrat des Arrays:")
9       print(squared_array)
10
11      array_mean = np.mean(my_array)
12      print("Mittelwert des Arrays:")
13      print(array_mean)
```

Beispiel: Sortieren eines NumPy-Arrays

```
1       import numpy as np
2
3       my_array = np.array([5, 2, 8, 1, 6])
4       print("Original-Array:")
5       print(my_array)
6
7       sorted_array = np.sort(my_array)
8       print("Sortiertes Array:")
9       print(sorted_array)
```

Hypothesentests

Hypothesentests sind statistische Verfahren, die verwendet werden, um Schlussfolgerungen über Populationen oder Stichproben zu ziehen. NumPy bietet Funktionen zur Durchführung von Hypothesentests.

Beispiel: Durchführung eines t-Tests mit NumPy

```
1       import numpy as np
2       from scipy import stats
3
4       sample1 = np.array([1, 2, 3, 4, 5])
5       sample2 = np.array([2, 4, 6, 8, 10])
6
7       t_statistic, p_value = stats.ttest_ind(sample1, sample2)
8       print("t-Statistik:", t_statistic)
9       print("p-Wert:", p_value)
```