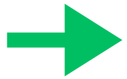
List Comprehension in Python

List Comprehension in Python

Empfohlene Vorkenntnisse für dieses Tutorial

- **Grundkenntnisse über:**
  - Arrays oder Listen
  - For-Schleifen



In jeder beliebigen Programmierungssprache !

Falls diese Vorkenntnisse nicht vorhanden:

- **Mehr konzentrieren und Fragen stellen**
- ~~Pretend you understand everything and keep a poker face.~~

~~#TODO: Remove this Tip later~~

List Comprehension in Python

Überblick

- Was ist List comprehension
- Was ist eine Liste in Python ?
- Was ist ein iterierbaren Objekt?
  Iteration mit for-Schleife
- Beispiel - List Comprehension
- List Comprehension Syntax
- List Comprehension mit Bedingung
- List Comprehension ohne Bedingung
- Wie kann es komplex werden?
- Code Challenge
- Fragerunde

List Comprehension in Python

List Comprehension ist ein einfache, schnellere und elegante Weg, eine (neue) Liste aus einem bestehenden iterierbaren Objekt (z. B. eine andere Liste, Tuples, Sets, Dictionaries, Strings, range-Objekte usw.) in Python zu erstellen.

List Comprehension ist ein einfache, schnellere und elegante Weg, eine (neue) Liste aus einem bestehenden iterierbaren Objekt (z. B. eine andere Liste, Tuples, Sets, Dictionaries, Strings, range-Objekte usw.) in Python zu erstellen.

Zwei Fragen zuerst :

1. Was ist Liste in Python?

2. Was ist ein iterierbaren Objekt?

1.Was ist eine Liste in Python?

Liste ist einer der Datentypen in Python, in dem wir jede Art von Daten speichern können.

Listen werden nach Indexnummern geordnet. Wenn wir eine liste erstellen, python geben diese index nummern automatisch an alle List elemente.

Wir können auf jedes Element in der Liste zugreifen und bearbeiten, mit diese Index Nummern.

Wir können auf jedes Element in der Liste zugreifen und bearbeiten, mit Indexnummern.

```
cities = ["Bremen", "Hamburg", "Berlin", "Duisburg"]
          0         1          2         3
```

Indexnummern

`print(cities[0])` → Bremen

`print(cities[1])` → Hamburg

`print(cities[2])` → Berlin

`print(cities[3]` → Duisburg

Wir können auf jedes Element in der Liste zugreifen und **bearbeiten / ändern**, mit Indexnummern.

```
cities = ["Bremen", "Hamburg", "Berlin", "Duisburg"]
          0         1          2         3          Indexnummern

print(cities)     → ['Bremen', 'Hamburg', 'Berlin', 'Duisburg']

cities[0] = "München"

print(cities[0]) → München
print(cities[1]) → Hamburg

print(cities)     → ['München', 'Hamburg', 'Berlin', 'Duisburg']
```

2.Was ist ein iterierbaren Objekt?

Es handelt sich um ein Objekt in Python/anderen Programmiersprachen

- Dieses Objekt verfügt über eine Kollektion von Items
- Mit diesem Objekt können wir ein Item nach dem anderen Iterieren bzw. durchlaufen (damit wir etwas mit diesen Items machen kann)

via for-Schleifen, while-Schleifen usw.

Beispiel : Liste in Python

Mit iterierebaren Objekten können wir ein Item nach dem anderen Iterieren bzw. durchlaufen (damit wir etwas mit diesen Items machen kann)

```
cities = ["Bremen", "Hamburg", "Berlin", "Duisburg"]

hallo_city = []


for banana in cities:
    hallo_city.append("Hallo "+banana)

print(hallo_city)  →  ['Hallo Bremen', 'Hallo Hamburg',
                       'Hallo Berlin', 'Hallo Duisburg']
```

#in production level code -->‘city’ oder ‘items’ statt banana, als Best Practice

List Comprehension ist ein einfache, schnellere und elegante Weg, eine (neue) Liste aus einem bestehenden iterierbaren Objekt (z. B. eine andere Liste, Tuples, Sets, Dictionaries, Strings, range-Objekte usw.) in Python zu erstellen.

Was ist List comprehension

Was ist eine Liste in Python ?

Was ist ein iterierbaren Objekt?
Iteration mit for-Schleife

Beispiel - List Comprehension

```
cities = ["Bremen", "Hamburg", Berlin","Duisburg"]

hallo_city = ["Hallo "+city  for city in cities]

print(hallo_city)
```

→ ['Hallo Bremen', 'Hallo Hamburg', 'Hallo Berlin', 'Hallo Duisburg']

List Comprehension ist ein einfache, schnellere und elegante Weg, eine (neue) Liste aus einem bestehenden iterierbaren Objekt (z. B. eine andere Liste, Tuples, Sets, Dictionaries, Strings, range-Objekte usw.) in Python zu erstellen.

Die Syntax der List Comprehension

Der List Comprehension besteht aus drei Schritten. Der zweite Schritt ist optional

Ergebnis = [**Ausgabe(Schritt3)** **For-Schleife(Schritt1)** **Bedingung(Schritt2)**]

#neue Liste #bei Bedarf modifizieren #jedes Element aus dem iterablen Objekt #optional

```
cities = ["Bremen", "Hamburg", "Berlin", "Duisburg"]
```

```
hallo_city = [ "Hallo "+city  for city in cities ]
```

Ergebnis

Schritt3: Ausgabe: Modifizierung der einzelnen Listenelemente

Schritt 1: Durchlaufen (Iteration) der einzelnen Listenelemente mit einer for-Schleife

Schritt2 (optional): Keine Bedingungen angegeben. Alle Elemente sind zulässig.

```
[ 'Hallo Bremen', 'Hallo Hamburg', 'Hallo Berlin', 'Hallo Duisburg' ]
```

Typ 1/2 : List Comprehension **ohne den optionalen** zweiten Schritt

Ergebnis = [**Ausgabe(Schritt3)** **For-Schleife(Schritt1)** **Bedingung(Schritt2)**]

#neue Liste #bei Bedarf modifizieren #jedes Element aus dem iterablen Objekt #optional

```
cities = ["Bremen", "Hamburg", "Berlin", "Duisburg"]

hallo_city = [ "Hallo "+city  for city in cities  if len(city) <= 6 ]
```

Ergebnis

Schritt3:
Ausgabe:
Modifizierung der einzelnen Listenelemente

Schritt 1:
Durchlaufen (Iteration) der einzelnen Listenelemente mit einer for-Schleife

Schritt2 (optional):
Kontrolle der Zulässigkeit der Elemente basierend auf den Bedingung/en

```
[ 'Hallo Bremen',  'Hallo Berlin']
```

Typ 2/2 : List comprehension **mit** dem optionalen zweiten Schritt

Ergebnis = [**Ausgabe(Schritt3)** **For-Schleife(Schritt1)** **Bedingung(Schritt2)**]

#neue Liste #bei Bedarf modifizieren #jedes Element aus dem iterablen Objekt #optional

List Comprehension Challenge

```
original_list  =  [ "Dog", 10, 11.5, 12.5, 15, 16, 19, 20, "Cat" ]

bow_meaw_list = [ str(x)+" bow bow" if x % 2 == 0 else str(x)+" meaw"
                  for x in original_list if (isinstance(x, int) and x <= 19) ]
```

Bedingungen in Schritt2 und Ausgabe modifizierungen in schritt 3 kann etwas Komplex werden.

List Comprehension ist ein einfache, schnellere und elegante Weg, eine (neue) Liste aus einem bestehenden iterierbaren Objekt (z. B. eine andere Liste, Tuples, Sets, Dictionaries, Strings, range-Objekte usw.) in Python zu erstellen.

- Was ist List comprehension
- Was ist eine Liste in Python ?
- Was ist ein iterierbaren Objekt?
  - Iteration mit for-Schleife
- Beispiel - List Comprehension
- List Comprehension Syntax
- List Comprehension mit Bedingung
- List Comprehension ohne Bedingung
- Wie kann es komplex werden
- Code Challenge

Zusammenfassung

Or Schnell click ‘Approve this Dozent’ Button and skip to Lunch..!