Python Lists Cheat Sheet

Learn computer science with Python:

www.k0nze.dev

Lists

Python lists are a sequential data type that allows you to organize your data in an ordered manner where each list element can be a different data type.

Defining a List

```
>>> l = [True, 42, 0.23, "Hi"]
>>> l
  [True, 42, 0.23, 'Hi']
```

Retrieving List Elements

```
>>> l = [True, 42, 0.23, "Hi"]
>>> l[0]
  True
>>> l[1]
  42
>>> l[-1]
  "Hi"
>>> l[-2]
  0.23
```

Setting List Elements

```
>>> l = [True, 42, 0.23, "Hi"]
>>> l[1] = "Foo"
>>> l
  [True, "Foo", 0.23, "Hi"]
```

.append()

Appends an element to the end of a list.

Syntax

```
list.append(object)
```

Parameters

`object:` element that is append to the end of the list

Return Value

None

```
>>> l = ['a','b','c']
>>> l.append('d')
>>> l
  ['a','b','c','d']
```

.extend() and +

Appends a list (iterable) to the end of a list.

Syntax

```
list.extend(iterable)
list + list
```

Parameters

`iterable:` list/iterable that is append to the end of the list

Return Value

None / result of list additon

```
>>> l = ['a','b','c']
>>> l.extend(['d','e'])
>>> l
  ['a','b','c','d','e']
>>> l + ['f','g']
  ['a','b','c','d','e','f','g']
```

.insert()

Inserts an element at a given index of a list.

Syntax

```
list.insert(index, object)
```

Parameters

`index:` index at which the element is inserted into the list

`object:` element that is inserted into the list

Return Value

None

```
>>> l = ['a','b','c']
>>> l.insert(1,'z')
>>> l
  ['a','z','b','c']
```

.remove()

Removes first occuring element with a given value from a list.

Syntax

```
list.remove(value)
```

Parameters

`value:` value that is removed from the list.

Return Value

None

```
>>> l = ['a','z','z',b','c']
>>> l.remove('z')
>>> l
  ['a','z','b','c']
```

List Slicing

With the []-operator it is not only possible to access single list elements but also to access sub lists of a list.

Syntax

```
l[start:stop:step]
```

Parameters

`start:` start index from which the sub list is returned
`stop:` stop index until the sub list is returned
`step:` which elements are included in the sub list (2 means every 2nd element is returned)

Retrieving Sub Lists

```
>>> l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> l[2:6]
  [2, 3, 4, 5]
>>> l[5:-1]
  [5, 6, 7, 8]
>>> l[1:7:2]
  [1, 3, 5]
>>> l[4:]
  [4, 5, 6, 7, 8, 9]
>>> l[:6]
  [0, 1, 2, 3, 4, 5]
>>> l[::3]
  [0, 3, 6, 9]
>>> l[::-1]
  [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

Setting Sub Lists

```
>>> l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> l[3:7] = ['a','b','c','d']
>>> l
  [0, 1, 2, 'a', 'b', 'c', 'd', 7, 8, 9]
>>> l = [0, 1, 2, 3, 4, 5]
>>> l[::-1] = ['a','b','c','d','e']
>>> l
  ['e', 'd', 'c', 'b', 'a']
```

.clear()

Removes all elements from a list.

Syntax

```
list.clear()
```

Parameters

None

Return Value

None

```
>>> l = ['a',b','c']
>>> l.clear()
>>> l
  []
```

.pop()

Removes and returns an element from a list at the given index.

Syntax

```
list.pop(index=-1)
```

Parameters

`index:` index of the list element that is removed

Return Value

element that was removed from the list

```
>>> l = ['a','b','c','d']
>>> l.pop()
  'd'
>>> l
  ['a','b','c']
>>> l.pop(1)
  'b'
```

.index()

Returns the index of the first occuring element with a given value from a list.

Syntax

```
list.index(value, start=0, stop=inf)
```

Parameters

`value:` value for which the index is returned
`start:` index from which the search for the given value starts
`stop:` index at which the search for the given value stops

Return Value

element that was removed from the list

```
>>> l = ['a','b','z','c','d',z','f']
>>> l.index('b')
  1
>>> l.index('z', 3)
  5
>>> l.index('z', 1, 3)
  2
```

Lists in if,elif,else

The `in` operator can be used to check if an element is contained in a list.

Syntax

```
>>> l = ['a','b','c','d']
>>> if 'z' in l:
...     print("found z")
... elif 'a' in l:
...     print("found a")
... else:
...     print("did not find a or z")
  found a
>>> x = 1 if 'a' in l else 0
>>> x
  1
```

.count()

Returns how often a given value is contained in a list.

Syntax

```
list.count(value)
```

Parameters

`value:` value that is counted

Return Value

amount of how often value is in the list

```
>>> l = ['a','b','b','c']
>>> l.count('b')
  2
>>> l.count('z')
  0
```

max() & min()

Returns the maximum/minimum element of a list (iterable).

Syntax

```
max(iterable, key=None)
min(iterable, key=None)
```

Parameters

`iterable:` list (iterable) that is sorted
`key:` sorting key

Return Value

maximum/minimum element

```
>>> l = ['a', 'aa', 'b', 'c']
>>> max(l)
  'c'
>>> max(l, key=len)
  'aa'
>>> min(l)
  'a'
```

.copy()

Returns a copy of a list.

Syntax

```
list.copy()
```

Parameters

None

Return Value

copy of the list

Copy vs. Reference

A reference to an object is created with the = operator. When the `.copy()` functions is callend on an object the object is duplicated and returned.

Multiple variables reference a single object:

```
>>> l = ['a','b','c','d']
>>> m = l
>>> m
  ['a','b','c','d']
>>> l[1] = 'x'
>>> l
  ['a','x','c','d']
>>> m
  ['a','x','c','d']
```

An object copy is created and assigned to a new variable:

```
>>> l = ['a','b','c','d']
>>> n = l.copy()
>>> n
  ['a','b','c','d']
>>> l[1] = 'x'
>>> l
  ['a','x','c','d']
>>> n
  ['a','b','c','d']
```

Reference

Copy

Lists in Loops

The `in` operator can be used to loop over each element in a list or to check if an element is included in a list.

for each loop

```
>>> l = ['a','b','c',d']
>>> for x in l:
...     print(x, end=',')
  a,b,c,d,
```

for loop over range

```
>>> l = ['a','b','c',d']
>>> for i in range(len(l)):
...     print(l[i], end=',')
  a,b,c,d,
```

for loop enumerate

```
>>> l = ['a','b','c',d']
>>> for i, x in enumerate(l):
...     print(f"l[{i}]={x}", end=',')
  l[0]=1,l[1]=2,l[2]=3,l[3]=4,
```

while loop emptying queue

```
>>> while 'a' in l:
...     x = l.pop()
...     print(x, end=',')
  a,b,c,d,
>>> l
  []
```

.sort()

Sorts a list by the alphanumeric order when no key is set.

Syntax

```
list.sort(key=None, reverse=False)
```

Parameters

`key:` sorting key
`reverse:` when set to True the list is sorted in decending order

Return Value

None

```
>>> l=['f','c','aaa','ab','aa','bbbb','d']
>>> l.sort()
>>> l
  ['aa','aaa','ab','bbbb','c','d','e','f']
>>> l.sort(reverse=True)
>>> l
  ['f','e','d','c','bbbb','ab','aaa', 'aa']
>>> l.sort(key=len)
>>> l
  ['f','e','d','c','ab','aa','aaa','bbbb']
>>> l.sort(key=len)
>>> l
  ['bbbb','aaa','ab','aa','f','e','d','c']
```

.reverse()

Reverses the order of a list.

Syntax

```
list.reverse()
```

Parameters

None

Return Value

None

```
>>> l = ['a','b','c','d']
>>> l.reverse()
>>> l
  ['d','c','b','a']
```

len()

Returns the length of a list or any object that implements `__len__(self)`.

Syntax

```
len(object)
```

Parameters

`object:` object that implements `__len__(self)`

Return Value

length of the object

```
>>> l = ['a','b','c',d']
>>> len(l)
  4
```


k0nze builds
k0nze_
discord.k0nze.dev
patreon.com/k0nze

www.k0nze.dev

© 2021 - k0nze aka. Konstantin Lübeck