5 Ways to Filter Items in Python 3

Justus Perlwitz

2015-09-11

Contents

Given a list `items`, that contains `int`s and `None`, produce a list that only contains the `int` values with their order of appearance preserved.

0.1 Using `filter`

```
>>> items  = [1, 2, None, 3, None, 4, 5, 6, None, 7]
>>> result = list(filter(bool, items))
>>> print(result)
[1, 2, 3, 4, 5, 6, 7]
```

0.2 Using list comprehensions

```
>>> items  = [1, 2, None, 3, None, 4, 5, 6, None, 7]
>>> result = [i for i in items if i]
>>> print(result)
[1, 2, 3, 4, 5, 6, 7]
```

0.3 Using a for-loop

```
>>> items  = [1, 2, None, 3, None, 4, 5, 6, None, 7]
>>> result = []
>>> for i in items:
...     if i:
...         result.append(i)
...
>>> print(result)
[1, 2, 3, 4, 5, 6, 7]
```

0.4 Using a recursive procedure

```
>>> items  = [1, 2, None, 3, None, 4, 5, 6, None, 7]
>>> filter_stuff = lambda lst: ([lst[0]] if lst[0] else []) + (filter_stuff(lst[1:]) if lst[1:] else [])
>>> result = filter_stuff(items)
>>> print(result)
[1, 2, 3, 4, 5, 6, 7]
```

0.5 In-place

```
>>> items  = [1, 2, None, 3, None, 4, 5, 6, None, 7]
>>> for i, e in enumerate(items):
...     if not e:
...         items.pop(i)
...
>>> print(items)
[1, 2, 3, 4, 5, 6, 7]
```