Les Itérables

Release 0.8.1

Dec 10, 2021

Contents:

Les Itérables

CHAPTER 1

les_iterables package

Submodules:

1.1 les_iterables.augmenting module

1.1.1 Summary

Functions:

alternate_with	Generate a series from items, alternating with an alternate item.
append	Yield an iterable followed by an item.
append_if	Yield an iterable, conditionally followed by an item.
ensure_contains	Yield items, followed by ensured_item, if ensured_item is not already present.
extend	Extend an iterable by yielding items returned by a factory.
prepend	Yield an item followed by an iterable.
prepend_if	Conditionally yield an item, followed by an iterable.
repeat_first	Repeat the first item from an iterable on the end.
separate_with	Generate a series from items, where the original items are separated by another item.

1.1.2 Reference

les_iterables.augmenting.**repeat_first**(*iterable*)
Repeat the first item from an iterable on the end.

Example

```
>>> ''.join(repeat_first("ABDC"))
"ABCDA"
```

Useful for making a closed cycle out of elements. If iterable is empty, the result will also be empty.

Parameters **iterable series of items.** (*An*) –

Yields All items from iterables, followed by the first item from iterable.

les_iterables.augmenting.**prepend**(*item*, *iterable*)
Yield an item followed by an iterable.

les_iterables.augmenting.**prepend_if**(*item*, *iterable*, *condition*)
Conditionally yield an item, followed by an iterable.

les_iterables.augmenting.**append**(*iterable*, *item*)
Yield an iterable followed by an item.

les_iterables.augmenting.**append_if**(*iterable*, *item*, *condition*)
Yield an iterable, conditionally followed by an item.

les_iterables.augmenting.**alternate_with**(*items*, *alternate_item*)
Generate a series from items, alternating with an alternate item.

items[0], alternate_item, items[1], alternate_item, . . . ,items[n - 1], alternate_item

The last item yielded will be alternate_item

les_iterables.augmenting.**separate_with**(*items*, *separator*)
Generate a series from items, where the original items are separated by another item.

items[0], separator, items[1], separator, items[2] . . . separator, items[n]

The last item yielded will be the last element of items.

les_iterables.augmenting.**ensure_contains**(*items*, *ensured_item*)
Yield items, followed by ensured_item, if ensured_item is not already present.

les_iterables.augmenting.**extend**(*iterable*, *item_factory=<function <lambda>>*)
Extend an iterable by yielding items returned by a factory.

Parameters

- **iterable** – An iterable series of items to be extended.
- **items_factory** – A zero-argument callable that will be invoked once for reach item requested beyond the end of iterator to create additional items as necessary.

1.2 les_iterables.combining module

Functions for combining iterable series.

1.2.1 Summary

Functions:

join_with	Generate a series of items, with separators taken from a second series.

1.2.2 Reference

les_iterables.combining.**join_with**(*items*, *separators*)

Generate a series of items, with separators taken from a second series.

The number of separators consumed will be one fewer than the number of items.

items[0], separators[0], items[1], separators[1], ..., separators[n-2], items[n-1]

Parameters

- **items** – An iterable series of items to return.
- **separators** – A series of items one of which will be returned between each item. The number of available separators must be at least one less than the number of items. Separators will only be consumed as required.

Returns The series of items alternating with items from separators. The first value yielded will be the first item. The last value yielded will be the last item. If items is empty an empty series will be yielded.

Raises ValueError – If there are too few separators to go between the supplied number of items.

1.3 les_iterables.functions module

1.3.1 Summary

Classes:

HeadPartitionIterator	
PartitionedTail	
TailPartitionIterator	

Functions:

elements_at	Select elements from a sequence based on their indexes.
empty_iterable	
extended_unchain	Convert an iterable into an infinite series of lists of containing zero or one items.
false_then_true	A single False value followed by True values.
generate	
group_by_terminator	Group the items of of an iterable series, starting a new group after each terminator.
indexes	The indexes at which item occurs in a sequence.
just	An iterable of just one item.
pairwise_padded	
partition_tail	Lazily partition an iterable series into a head, and tail of no more than specified length.

Continued on next page

Table 4 – continued from previous page

run_length_encode	
split_around	Split an iterable series into groups around specific items.
transform_if	
true_then_false	A single True value followed by False values.
unchain	

1.3.2 Reference

les_iterables.functions.**just**(*item*)
An iterable of just one item.

Parameters **item** – The item to be yielded.

Yields The item.

les_iterables.functions.**generate**(*collection=None*)

les_iterables.functions.**pairwise_padded**(*iterable*, *fillvalue=None*)

les_iterables.functions.**transform_if**(*iterable*, *predicate*, *transform*)

les_iterables.functions.**group_by_terminator**(*iterable*, *predicate*, *group_factory=None*)
Group the items of of an iterable series, starting a new group after each terminator.

Each group will have as it's last item an item from which the predicate returns True. For all preceding items in the group the predicate will return False. The last group yielded may be incomplete, without a terminator.

Parameters

- **iterable** – An iterable series of items to be grouped.
- **predicate** – A unary callable function used to detect group-terminating items from the iterable series.
- **group_factory** – A callable which creates a group given an sequence of items. By default, a list.

Yields A series of groups.

les_iterables.functions.**split_around**(*iterable*, *predicate*, *group_factory=None*)
Split an iterable series into groups around specific items.

Each item for which the predicate returns True will be in its own group.

Example:

split_around("abc

def ", is_newline) -> ['a', 'b', 'c'], [' '], ['d', 'e', 'f'], [' ']

Args: iterable: An iterable series of items to be grouped.

predicate: A unary callable to detect items which should be placed in their own group.

group_factory: A callable which creates a group given a sequence of items. By default, a list.

Yields: A series of groups.

les_iterables.functions.**elements_at**(*seq*, *indexes*)
Select elements from a sequence based on their indexes.

Parameters

- **seq** – The sequence from which to select elements.
- **indexes** – Indexes into seq indicating the selected elements.

Yields A series of items selected from seq by indexes.

Raises IndexError – If one of the indexes is not valid with seq.

les_iterables.functions.**indexes**(*seq*, *item*)
The indexes at which item occurs in a sequence.

Parameters

- **seq** – A sequence in which to search for occurrences of item.
- **item** – The item for which to determine indexes.

Yields A series of indexes into seq at which item occurs.

les_iterables.functions.**partition_tail**(*items*, *n*)
Lazily partition an iterable series into a head, and tail of no more than specified length.

Parameters

- **items** – An iterable series of items.
- **n** – The maximum number of items to be partitioned into the tail.

Returns A pair of iterators, head and tail. Consuming any items from the tail iterator will cause the entire head iterator to be consumed, so typically the head iterator should be consumed before consuming any items from the tail iterator.

Example

head, tail = partition_tail(range(10), 3) for item in head:

print(item) # Prints all but the last three

for item in tail: print(item) # Prints the last three

class les_iterables.functions.**PartitionedTail**(*items*, *n*)
Bases: object

class les_iterables.functions.**HeadPartitionIterator**(*partition_tail*)
Bases: object

class les_iterables.functions.**TailPartitionIterator**(*partition_tail*)
Bases: object

les_iterables.functions.**unchain**(*iterable*, *box=None*)

les_iterables.functions.**extended_unchain**(*iterable*, *box=<class 'list'>*)
Convert an iterable into an infinite series of lists of containing zero or one items.

les_iterables.functions.**empty_iterable**()

les_iterables.functions.**run_length_encode**(*items*)

les_iterables.functions.**false_then_true**()
A single False value followed by True values.

les_iterables.functions.**true_then_false**()
A single True value followed by False values.

1.4 les_iterables.parsing module

1.4.1 Summary

Functions:

expand_numbered_list	Expands a string containing numbered items into a list of integers.
range_from_text	A range of integers from a textual description.

1.4.2 Reference

les_iterables.parsing.**range_from_text**(*text_range: str*, *separator='-'*) → range
A range of integers from a textual description.

Parameters **text_range** – A string of the form "<first>-<last>" such as "7-10" describing the inclusive ends of a range of integers.

Returns A range object.

Raises ValueError – If the string could not be parsed as an ascending range of at least one item.

les_iterables.parsing.**expand_numbered_list**(*text*, *, *separator=', '*, *range_separator='-'*)
Expands a string containing numbered items into a list of integers.

e.g. "1, 2, 5, 7-10, 15, 20-25" -> [1, 2, 5, 7, 8, 9, 10, 15, 20, 21, 22, 23, 24, 25]

Parameters

- **text** – A string containing separated integers and ascending integer ranges.
- **separator** – The item separator. Defaults to ",".
- **range_separator** – The separator between the beginning and end of a range. Defaults to "-"

Yields An iterable series of integers.

Raises ValueError – If the list could not be parsed.

1.5 les_iterables.searching module

1.5.1 Summary

Functions:

duplicates	Find duplicate items.
first_matching	The first item matching a predicate.
nth_matching	The nth item matching a predicate.

1.5.2 Reference

les_iterables.searching.**nth_matching**(*iterable*, *predicate*, *n*)
The nth item matching a predicate.

Parameters

- **iterable** – An iterable series of items to be searched.
- **predicate** – A callable to which each item will be passed in turn.
- **n** – A zero-based index.

Returns The nth item for which the predicate returns True.

Raises ValueError – If there are no matching items.

les_iterables.searching.**first_matching**(*iterable*, *predicate*)
The first item matching a predicate.

Parameters

- **iterable** – An iterable series of items to be searched.
- **predicate** – A callable to which each item will be passed in turn.

Returns The first item for which the predicate returns True.

Raises ValueError – If there are no matching items.

les_iterables.searching.**duplicates**(*iterable*, *key=None*)
Find duplicate items.

[1, 3, 6, 3, 6, 2, 9, 3] -> [3, 6, 3]

Parameters

- **iterable** – The items to be searched.
- **key** – An optional function used to generate a key by which items will be compared. If not provided the items themselves will be compared. If the key function returns hashable objects the performance of this function will be O(n); however, the performance will degrade to O(n^2) when the first non-hashable key is encountered.

Yields Items which are considered duplicates according to the key, in the order that they are encountered. Items which are encountered more than twice will be yielded more than once.

1.6 les_iterables.selecting module

1.6.1 Summary

Functions:

element_at	
offset_iterators	Produce two iterators which are offset from each other by a given number of items.
preceding	The item which comes in the series immediately before the specified item.

Continued on next page

Table 7 – continued from previous page

reject_falsy	Reject those items which evaluate to False in a boolean context.
reject_if	Retain those items for which predicate evaluates to True.
reject_truthy	Reject those items which evaluate to True in a boolean context.
relative_to	Return the item relative to the nth occurrence of some existing item.
retain_falsy	Retain those items which evaluate to False in a boolean context.
retain_if	Retain those items for which predicate evaluates to True.
retain_truthy	Retain those items which evaluate to True in a boolean context.
succeeding	The item which comes in the series immediately after the specified item.
take_after_inclusive	Yield items starting with the first match.
take_after_last_match	Yield items in an iterable series after the last matching.
take_before_inclusive	Yield items up to and including the first match.
take_between_inclusive	A list of items from the first matching to the last matching inclusive.
take_between_inclusive_values	A list of items from the first matching to the last matching inclusive.

1.6.2 Reference

les_iterables.selecting.**element_at**(*iterable*, *index*, *, *start=0*)

les_iterables.selecting.**retain_if**(*predicate*, *iterable*)

Retain those items for which predicate evaluates to True.

Example

```
>>> list(retain_if(lambda x: x%2 == 0, range(10)))
[0, 2, 4, 6, 8]
```

Parameters

- **predicate** – A single-argument callable to which each item of iterable will be passed in turn to determine whether it should be retained, by returning True, or rejected, by returning False.
- **iterable** – The iterable series of items to be filtered.

Returns An iterable series of items for which predicate returns True.

les_iterables.selecting.**reject_if**(*predicate*, *iterable*)

Retain those items for which predicate evaluates to True.

Example

```
>>> list(reject_if(lambda x: x%2 == 0, range(10)))
[1, 3, 5, 7, 9]
```

Parameters

- **predicate** – A single-argument callable to which each item of iterable will be passed in turn to determine whether it should be rejected, by returning True, or retained, by returning False.
- **iterable** – The iterable series of items to be filtered.

Returns An iterable series of items for which predicate returns False.

les_iterables.selecting.**retain_truthy**(*iterable*)
Retain those items which evaluate to True in a boolean context.

Parameters **iterable** – The iterable series of items to be filtered.

Returns An iterable series of items for which bool(item) is True.

les_iterables.selecting.**retain_falsy**(*iterable*)
Retain those items which evaluate to False in a boolean context.

Parameters **iterable** – The iterable series of items to be filtered.

Returns An iterable series of items for which bool(item) is True.

les_iterables.selecting.**reject_truthy**(*iterable*)
Reject those items which evaluate to True in a boolean context.

Parameters **iterable** – The iterable series of items to be filtered.

Returns An iterable series of items for which bool(item) is False.

les_iterables.selecting.**reject_falsy**(*iterable*)
Reject those items which evaluate to False in a boolean context.

Parameters **iterable** – The iterable series of items to be filtered.

Returns An iterable series of items for which bool(item) is True.

les_iterables.selecting.**take_after_last_match**(*iterable*, *predicate*)
Yield items in an iterable series after the last matching.

> **Warning:** This function potentially alloctates sufficient space to store the entire series.

Parameters

- **iterable** – An iterable series of items.
- **predicate** – A function of one argument used to select items.

Returns A sequence containing the tail of the iterable series after the last match.

les_iterables.selecting.**take_after_inclusive**(*iterable*, *predicate*)
Yield items starting with the first match.

Parameters

- **iterable** – An iterable series of items.

- **predicate** – A function of one argument used to select the first item.

Yields Items starting with the first match.

les_iterables.selecting.**take_before_inclusive**(*iterable*, *predicate*)
Yield items up to and including the first match.

Parameters

- **iterable** – An iterable series of items.
- **predicate** – A function of one argument used to select the last item.

Returns A sequence of items finishing with the first match.

les_iterables.selecting.**take_between_inclusive**(*iterable*, *first_predicate*, *last_predicate*)
A list of items from the first matching to the last matching inclusive.

Parameters

- **iterable** – An iterable series of items.
- **first_predicate** – A function of one argument used to select the first item in the result.
- **last_predicate** – A function of one argument used to select the last item in the result.
- **Returns** – Either a sequence of at least two elements, or an empty sequence if elements matching the first predicate and the last predicate were not both found.

les_iterables.selecting.**take_between_inclusive_values**(*iterable*, *first*, *last*)
A list of items from the first matching to the last matching inclusive.

Parameters

- **iterable** – An iterable series of items.
- **first** – A value marking the start of the result sequence.
- **last** – A value marking the end of the result sequence.
- **Returns** – Either a sequence of at least two elements, or an empty sequence if elements matching the first predicate and the last predicate were not both found.

les_iterables.selecting.**preceding**(*iterable*, *item*)
The item which comes in the series immediately before the specified item.

Parameters

- **iterable** – The iterable series in which to search for item.
- **item** – The item to search for in iterable.

Returns The previous item.

Raises ValueError – If item is not present in iterable beyond the first item.

les_iterables.selecting.**succeeding**(*iterable*, *item*)
The item which comes in the series immediately after the specified item.

Parameters

- **iterable** – The iterable series in which to search for item.
- **item** – The item to search for in iterable.

Returns The next item.

Raises ValueError – If the item is not present before the penultimate item.

les_iterables.selecting.**relative_to**(*iterable*, *item*, *, *offset*, *n=0*, *default=<object object>*)
Return the item relative to the nth occurrence of some existing item.

Parameters

- **iterable** – The iterable series from which to search for item.
- **item** – The value to relative to which the returned item should be.
- **offset** – The positive or negative offset relative to the found item.
- **n** – Where it is the nth occurrence of item which will be searched for.
- **default** – The default value to return if not found.

Returns The item at a given offset from a specific value, or the default value if given.

Raises ValueError – If there is not item matching the criteria an no default value has been specified.

les_iterables.selecting.**offset_iterators**(*iterable*, *offset: int*)
Produce two iterators which are offset from each other by a given number of items.

Once offset_iterators() has been called, and if the original itererable is an iterator (as opposed to a iterablecollection), the iterator should not be used anywhere else; otherwise, the iterator could get advanced without the offset_iterators objects being informed.

The produced iterators will start with the requested offset but are independent and can be advanced independently. To keep them synchronized, consider iterating over them in lockstep using zip().

Note that:

```
p, q = offset_iterators(iterable, 0)
```

is equivalent to:

```
p, q = itertools.tee(iterable)
```

Parameters

- **iterable** – An iterable from which to return two iterators separated by offset.
- **offset** – An integer offset by which the two iterators should be separated. This offset can be positive or negative.

Returns Two iterators, the second of which will be offset from the first by the specified number (positive, negative or zero) places.

Raises ValueError – If the iterable is not long enough to accommodate two iterators separated by the specified offset.

1.7 les_iterables.sentinels module

1.7.1 Summary

1.7.2 Reference

1.8 les_iterables.version module

1.9 Summary

__all__ Functions:

alternate_with	Generate a series from items, alternating with an alternate item.
append	Yield an iterable followed by an item.
append_if	Yield an iterable, conditionally followed by an item.
duplicates	Find duplicate items.
ensure_contains	Yield items, followed by ensured_item, if ensured_item is not already present.
expand_numbered_list	Expands a string containing numbered items into a list of integers.
extend	Extend an iterable by yielding items returned by a factory.
first_matching	The first item matching a predicate.
join_with	Generate a series of items, with separators taken from a second series.
nth_matching	The nth item matching a predicate.
prepend	Yield an item followed by an iterable.
prepend_if	Conditionally yield an item, followed by an iterable.
range_from_text	A range of integers from a textual description.
reject_falsy	Reject those items which evaluate to False in a boolean context.
reject_if	Retain those items for which predicate evaluates to True.
reject_truthy	Reject those items which evaluate to True in a boolean context.
repeat_first	Repeat the first item from an iterable on the end.
retain_falsy	Retain those items which evaluate to False in a boolean context.
retain_if	Retain those items for which predicate evaluates to True.
retain_truthy	Retain those items which evaluate to True in a boolean context.
separate_with	Generate a series from items, where the original items are separated by another item.
take_after_inclusive	Yield items starting with the first match.
take_after_last_match	Yield items in an iterable series after the last matching.
take_before_inclusive	Yield items up to and including the first match.
take_between_inclusive	A list of items from the first matching to the last matching inclusive.
take_between_inclusive_value	A list of items from the first matching to the last matching inclusive.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index

S

T

U