```
append = """class object()
The base class of the class hierarchy.

When called, it accepts no arguments and returns a new featureless
instance that has no instance attributes and cannot be given any."""

extend = """def extend(iterable: Iterable[_T], /) -> None
Extend list by appending elements from the iterable."""

remove = """def remove(value: _T, /) -> None
Remove first occurrence of value.

Raises ValueError if the value is not present."""

sort = """def sort(*, key: None=..., reverse: bool=...) -> None
def sort(*, key: Callable[[_T], SupportsLessThan], reverse: bool=...) -
> None
Sort the list in ascending order and return None.

The sort is in-place (i.e. the list itself is modified) and stable
(i.e. the
order of two equal elements is maintained).

If a key function is given, apply it once to each list item and sort
them,
ascending or descending, according to their function values.

The reverse flag can be set to sort in descending order."""

reverse = """def reverse() -> None
Reverse *IN PLACE*."""

clear = """def clear() -> None
Remove all items from list."""

copy = """def copy() -> List[_T]
Return a shallow copy of the list."""

count = """def count(value: _T, /) -> int
Return number of occurrences of value."""

index = """def index(value: _T, start: int=..., stop: int=..., /) ->
int
Return first index of value.

Raises ValueError if the value is not present."""
```

```
insert = """def insert(index: int, object: _T, /) -> None
Insert object before index."""

pop = """def pop(index: int=..., /) -> _T
Remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range."""
```