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

update = """def update(m: Mapping[_KT, _VT], /, **kwargs: _VT) -> None
def update(m: Iterable[Tuple[_KT, _VT]], /, **kwargs: _VT) -> None
def update(**kwargs: _VT) -> None
D.update([E, ]**F) -> None.  Update D from dict/iterable E and F.
If E is present and has a .keys() method, then does:  for k in E: D[k]
= E[k]
If E is present and lacks a .keys() method, then does:  for k, v in E:
D[k] = v
In either case, this is followed by: for k in F:  D[k] = F[k]"""

copy = """def copy() -> Dict[_KT, _VT]
D.copy() -> a shallow copy of D"""

keys = """def keys() -> KeysView[_KT]
D.keys() -> a set-like object providing a view on D's keys"""

values = """def values() -> ValuesView[_VT]"""

setdefault = """def setdefault(key: _KT, default: _VT=..., /) -> _VT
Insert key with a value of default if key is not in the dictionary.

Return the value for key if key is in the dictionary, else default."""

popitem = """def popitem() -> Tuple[_KT, _VT]
Remove and return a (key, value) pair as a 2-tuple.

Pairs are returned in LIFO (last-in, first-out) order.
Raises KeyError if the dict is empty."""

items = """def items() -> ItemsView[_KT, _VT]"""

fromkeys = """def fromkeys(iterable: Iterable[_T], /) -> Dict[_T, Any]
def fromkeys(iterable: Iterable[_T], value: _S, /) -> Dict[_T, _S]
Create a new dictionary with keys from iterable and values set to
value."""
```