Code Implementations of Hashmap Collision Handling

→ Separate Chaining

```
class ChainHashMap(HashMapBase):
  def _bucket_getitem(self, j, k):
    bucket = self._table[j]
    if bucket is None:
      raise KeyError
    return bucket

  def _bucket_setitem(self, j, k, v):
    if self._table is None:
      self._table[j] = UnsortedTableMap()          // bucket new to table
    oldsize = len(self._table[j])
    self._table[j][k] = v
    if len(self._table[j]) > oldsize:              // key new to table
      self._n += 1                                 // increase mapsize

  def _bucket_delitem(self, j, k):
    bucket = self._table[j]
    if bucket is None:
      raise KeyError
    del bucket[k]

  def __iter__(self):
    for bucket in self._table:
      if bucket is not None:
        for key in bucket
          yield key
```

Linear Probing

```
class ProbeHashMap(  ):
  _AVAIL = object()
  def is_available(self, j):
    return self._table[j] is None or
        self._table[j] is _AVAIL

  def find_slot(self, j, k):
    firstAvail = None
    while True:
      if self._is_available(j):
        if firstAvail is None:
          firstAvail = j
        if self._table[j] is None:
          return (False, firstAvail)
      elif k == self._table[j]._key:
          return (True, j)
      j = (j+1) % len(self._table)
```