③ ハッシュテーブルと辞書

ハッシュテーブルを使って、辞書を実現する方法を学習します。

辞書の実現

- 同じ仕組み辞書も実現することができます
- ここでは、辞書の要素を格納するために、キーと値の対を表現するクラス Node を定義します

```
class Node:
    def __init__(self, key, value):
        self.key = key
        self.value = value
```

keyがキーを、valueが値を表すクラス

```
dict_table = [None] * N
```

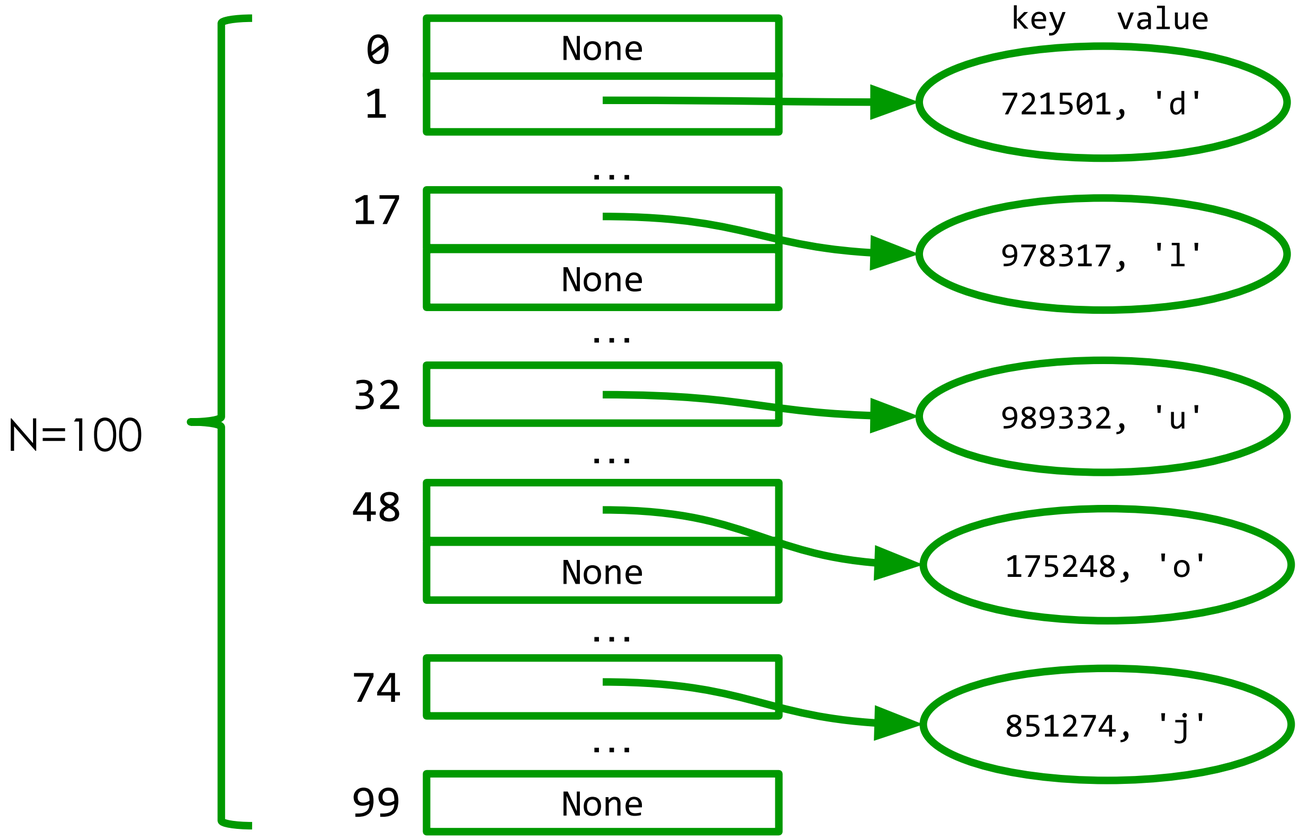
空の配列を用意します

辞書に要素を追加するには？

- 要素を追加する際には、追加するキーのハッシュ値を計算し、配列の該当箇所にノードへのリンクを格納します

```
def put(table, key, value):
    hash_value = h(key)
    if table[hash_value] != None:
        raise Exception('Hash collision!')
    table[hash_value] = Node(key, value)
```

```
put(dict_table, 175248, 'o')
put(dict_table, 721501, 'd')
put(dict_table, 989332, 'u')
put(dict_table, 851274, 'j')
put(dict_table, 978317, 'l')
dict_table
```



辞書の要素を検索するには？

- 辞書から要素を検索する際には、検索するキーのハッシュ値を計算し、配列の該当箇所のノードと比較します

```
def get(table, key):
    hash_value = h(key)
    if table[hash_value] != None and table[hash_value].key == key:
        return table[hash_value].value
    return None
```

```
get(dict_table, 175248)
```

```
'o'
```

48番目の要素のキーと一致するため、対応する値が返る

```
get(dict_table, 100048)
```

48番目の要素のキーと一致しないため、Noneが返る

```
get(dict_table, 101010)
```

10番目の要素がNoneのため、Noneが返る