**Hashing**

Normal Searching: Feed in a Key value: could be a number, char, a string and you scan your way through the data structure, finding an item that contains the Key value.

Hashing: Making the above as simple + efficient as possible. Take a key value and use it as an index into an array.

* Take the key and go straight into a spot in the array and find the item that has the key
* Effectively, we are doing **indexing with arbitrary values**
* Generally dealing with strings
* Cost would be **O(1)** (perfect search performance) as we are:
  + Taking Key value, using it as the index
  + Going straight to the right location with no searching required

Magic function **h( )** which takes in a string key and gives us an index value

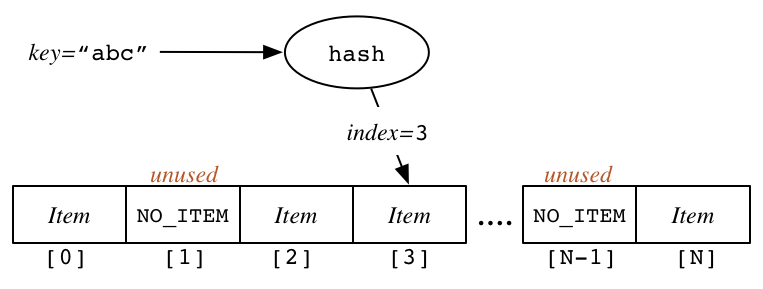
To use arbitrary values as keys, we need three things:

1. **Set of Key values** where each key is unique to identify one item.
2. An **Array (size N)** to store (ptrs to) **Items**
3. A **Hash Function** h() of type **Key**->[0..N-1]

Requirements of Hash Function

* If x == y (same key), then h(x) == h(y)
* h(x) always returns same value for given x

Use a hash function for both storing and retrieving



**The size of the Hash Table must be known when created and cannot be changed, as it will change the hash function, therefore anything inserted before the change won’t be retrievable anymore**

Suitable NoItem values (needed to represent no value in an array slot)

* If keys are ints 🡪 - 1
* If keys are strings 🡪 Use an empty string
* If items[] is an array of (Item \*) 🡪 NULL value