Week 14

- Term Project
- Binary Search: questions
- More search
  - Why?
  - Hash functions
  - Let's create our own `set` for O(1) search

Binary Search: Questions from S22

(e) (2 points) `L==[1,4,7,8,10,28]`
At most, how many indices do we need to check with linear search in order to conclude that a number is NOT in L?

◯ 3 ◯ 4 ◯ 5 ◯ 6

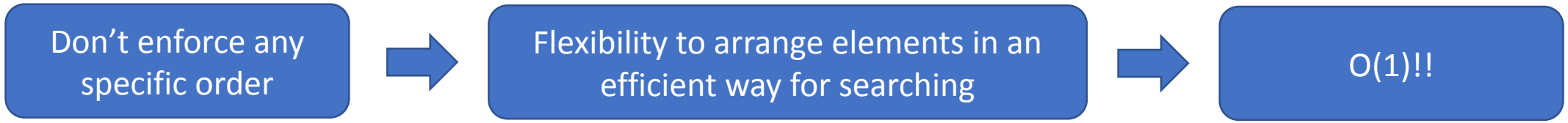
(f) (2 points) `L==[1,4,7,8,10,28]`
At most, how many indices do we need to check with binary search in order to conclude that a number is NOT in L?

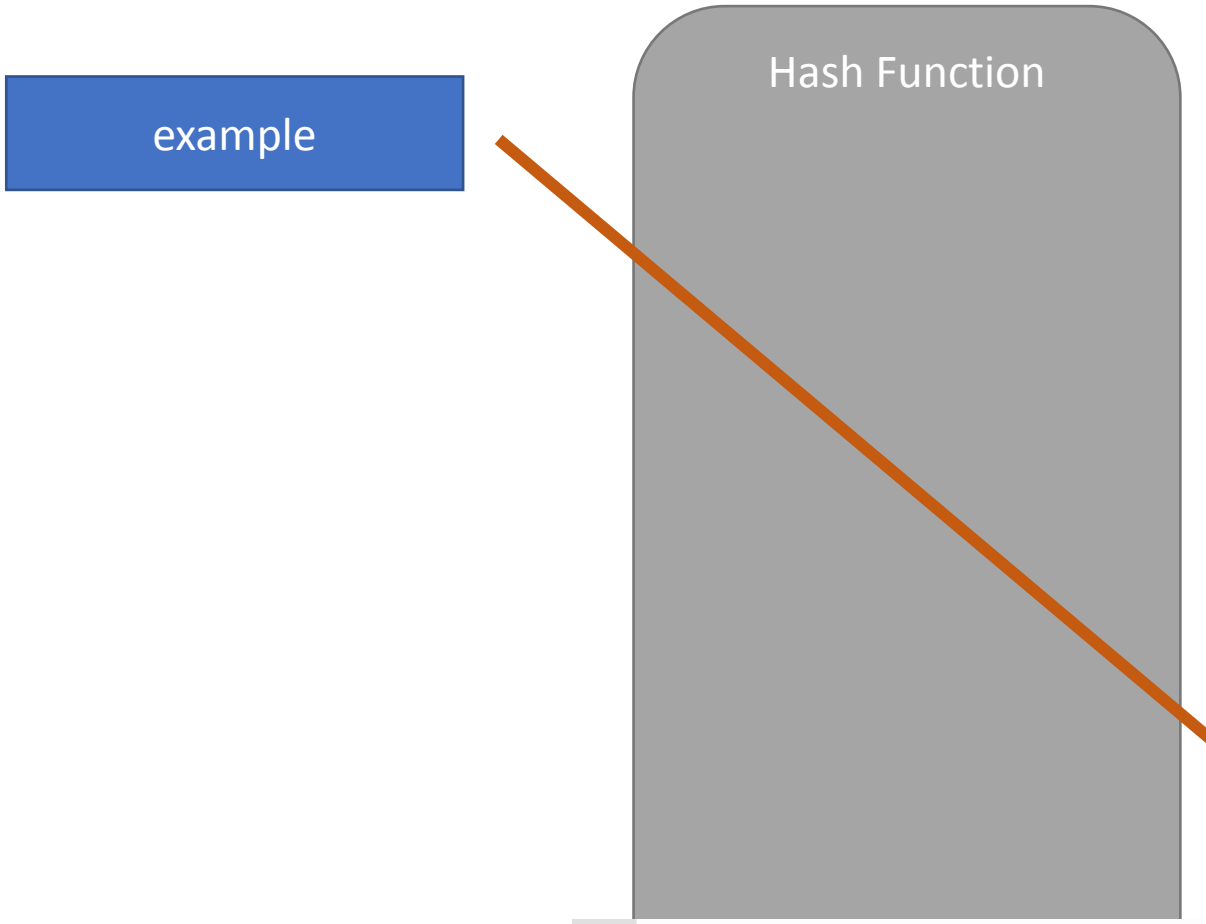
◯ 3 ◯ 4 ◯ 5 ◯ 6

Given `L=[0,4,9,30,38,76,78,80],` which elements will be compared to 38 using Binary Search?

Searching

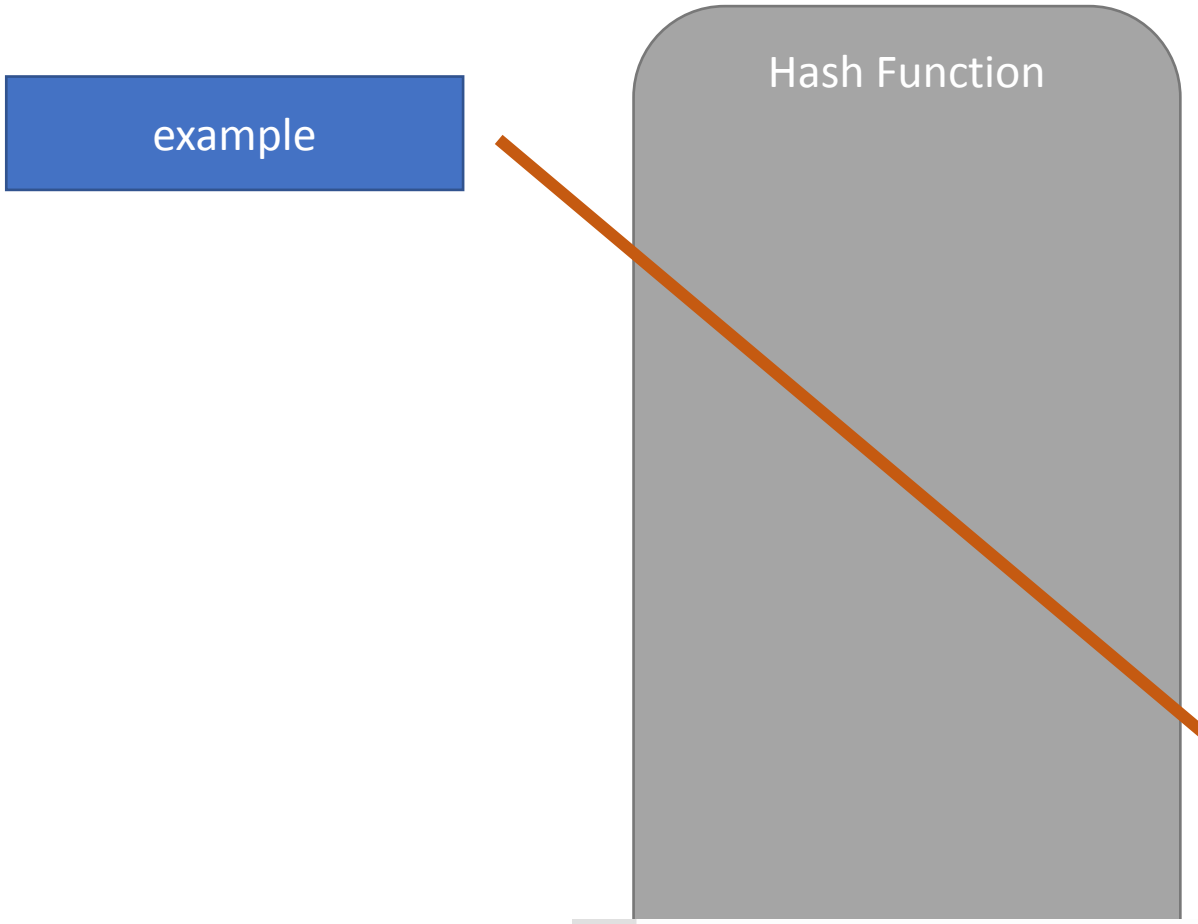
- Binary Search
  - `O(logN)`
  - You should know why.
- What if there's no order among the elements?
- What if we don't care about the order?
- Good News! We can do even better than `O(logN)`





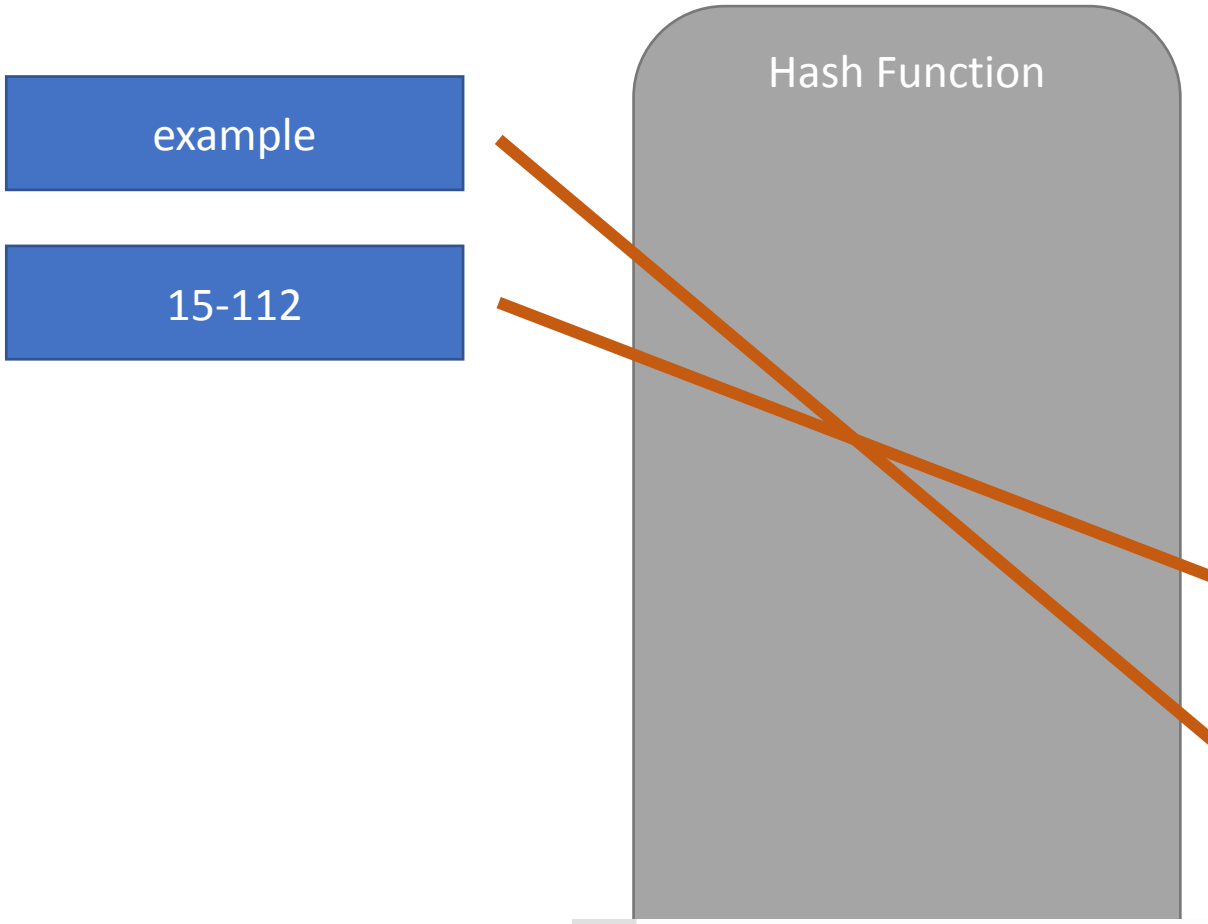
Index	Value
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```



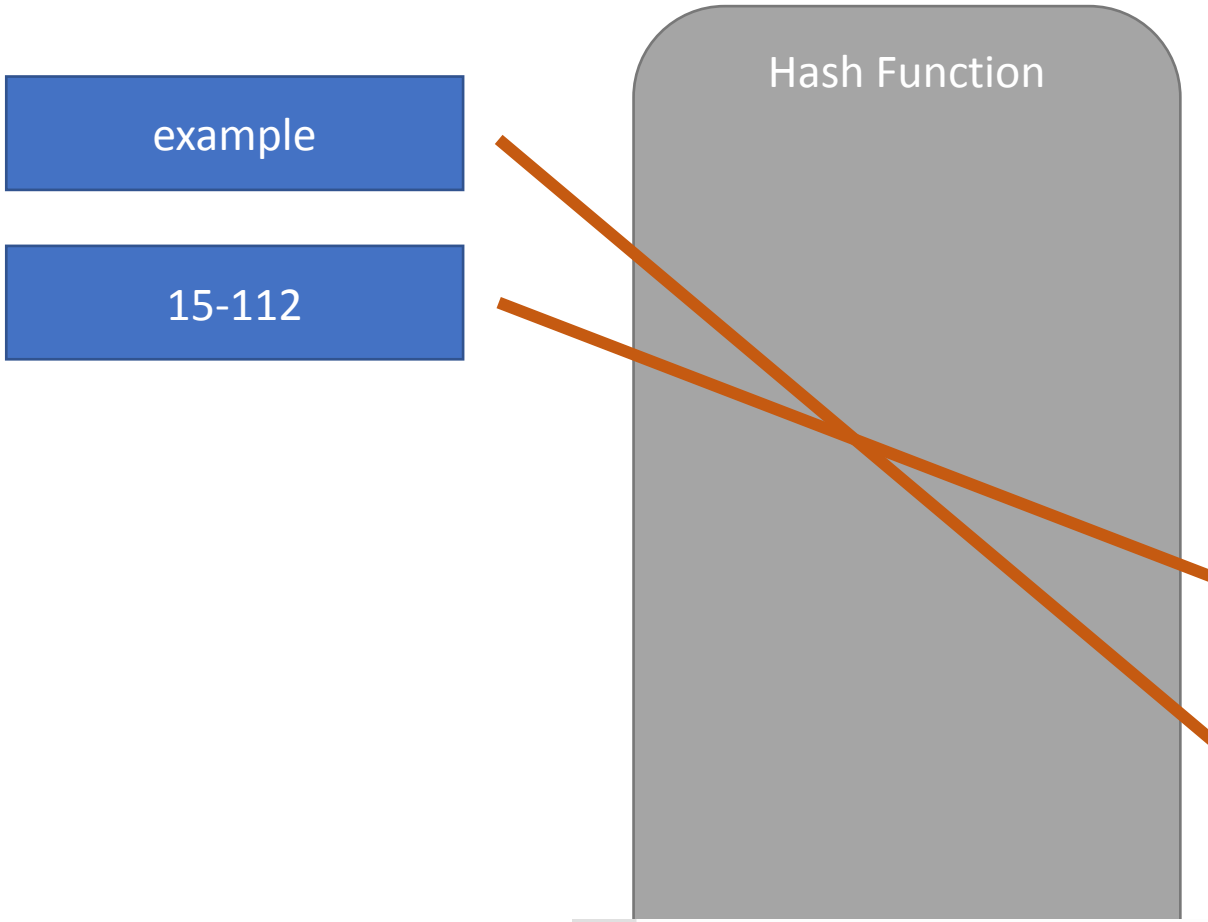
Index	Value
0	
1	
2	
3	
4	
5	
6	
7	
8	“example”
9	

```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```



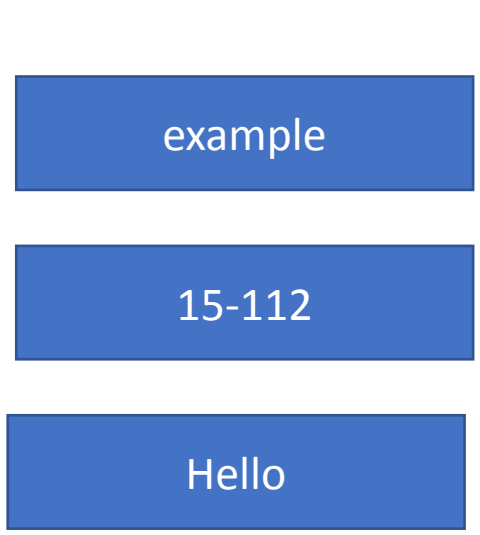
Index	Value
0	
1	
2	
3	
4	
5	
6	
7	
8	“example”
9	

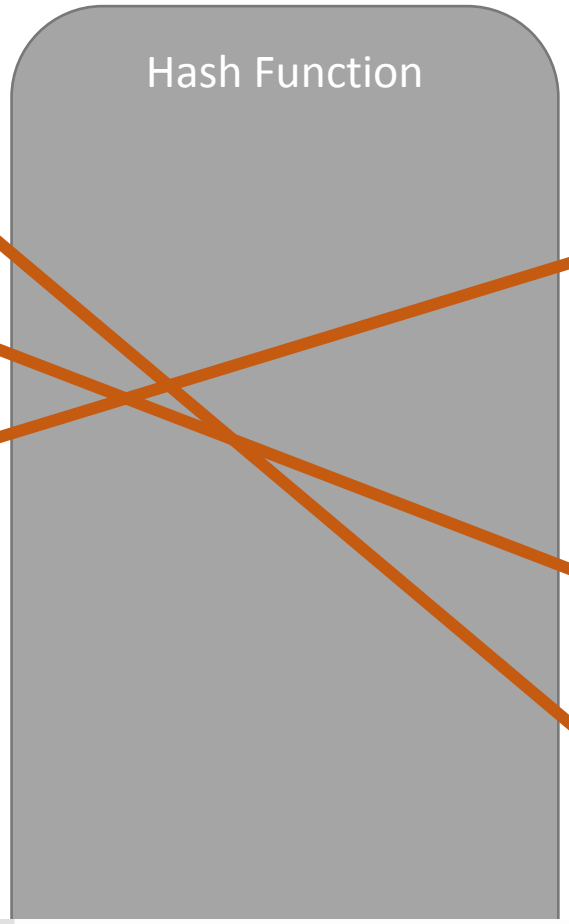
```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```



Index	Value
0	
1	
2	
3	
4	
5	“15-112”
6	
7	
8	“example”
9	

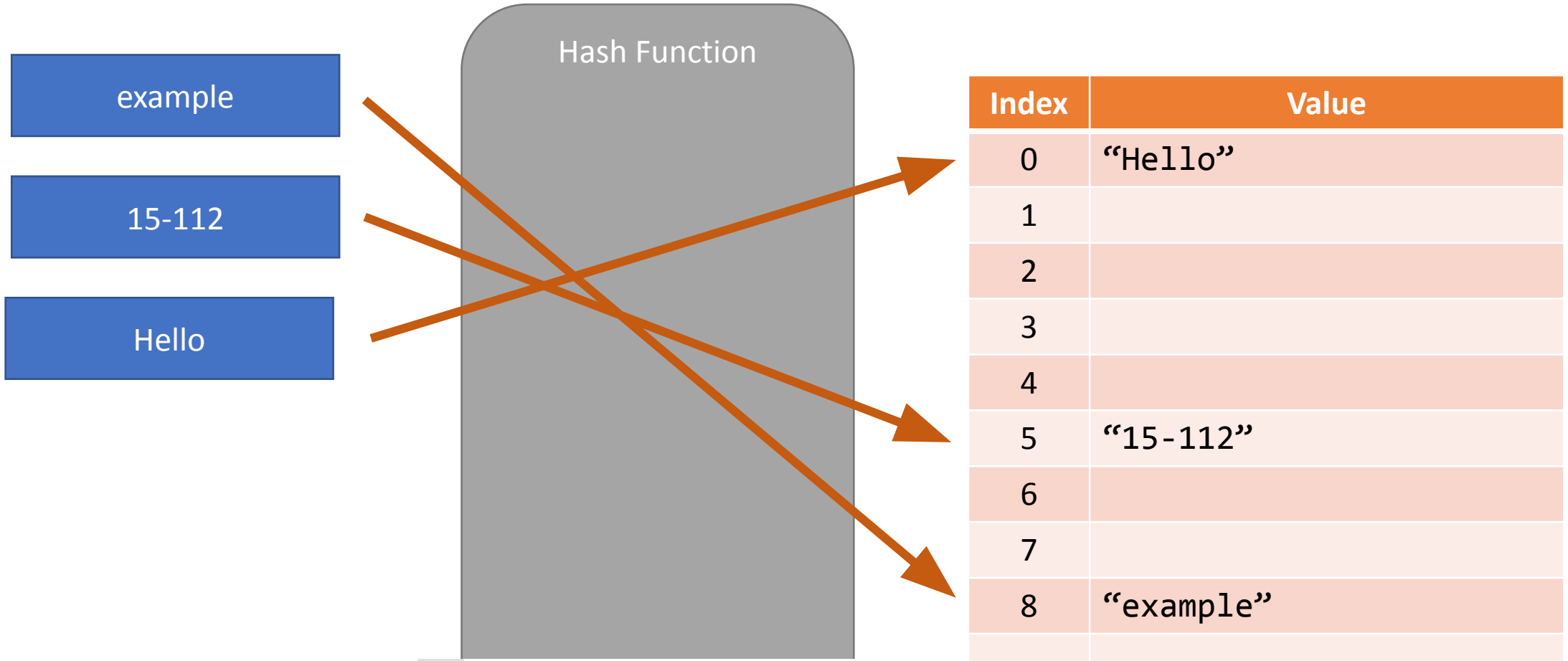
```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```





Index	Value
0	
1	
2	
3	
4	
5	“15-112”
6	
7	
8	“example”
9	

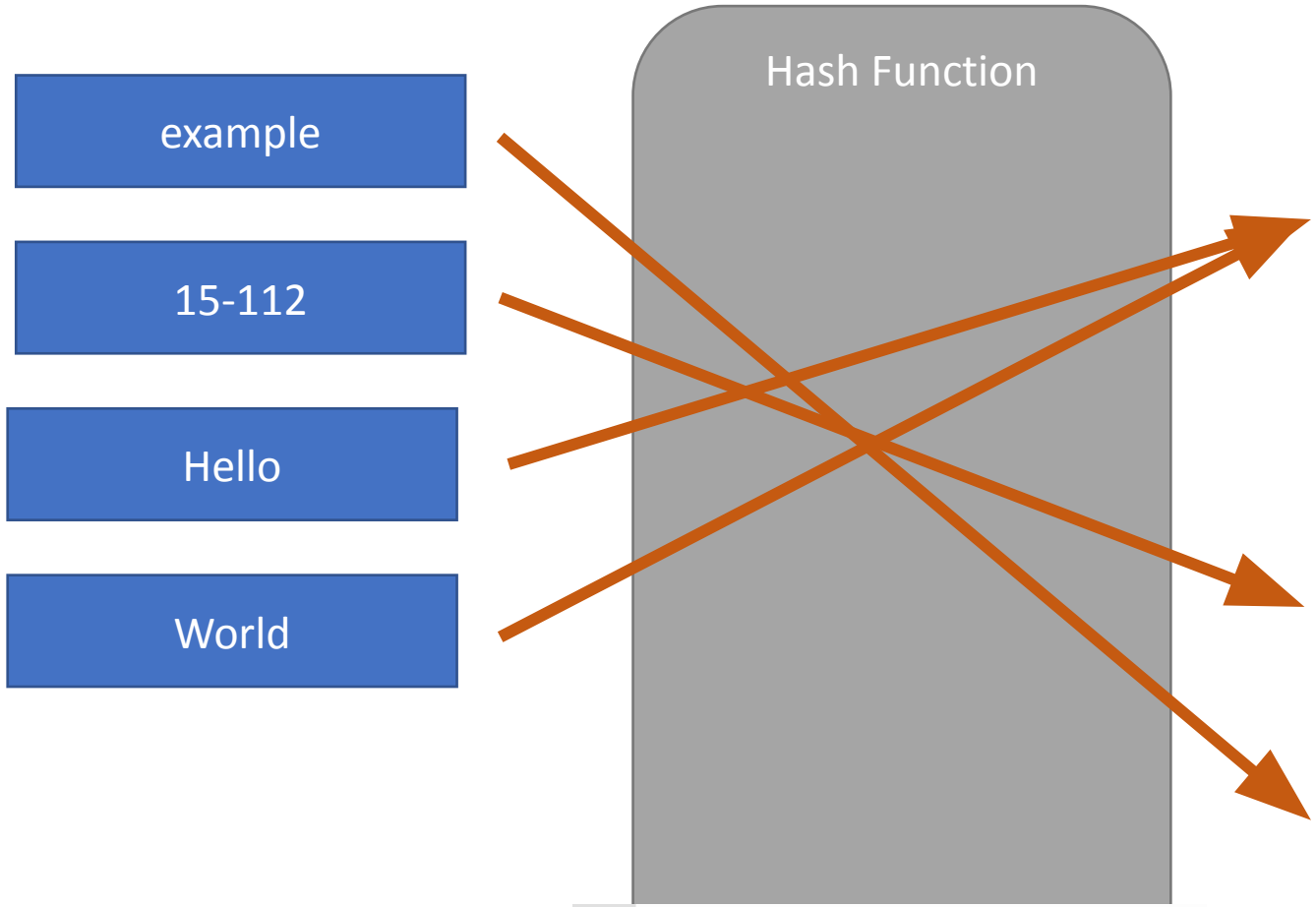
```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```



Index	Value
0	“Hello”
1	
2	
3	
4	
5	“15-112”
6	
7	
8	“example”
…	
N	

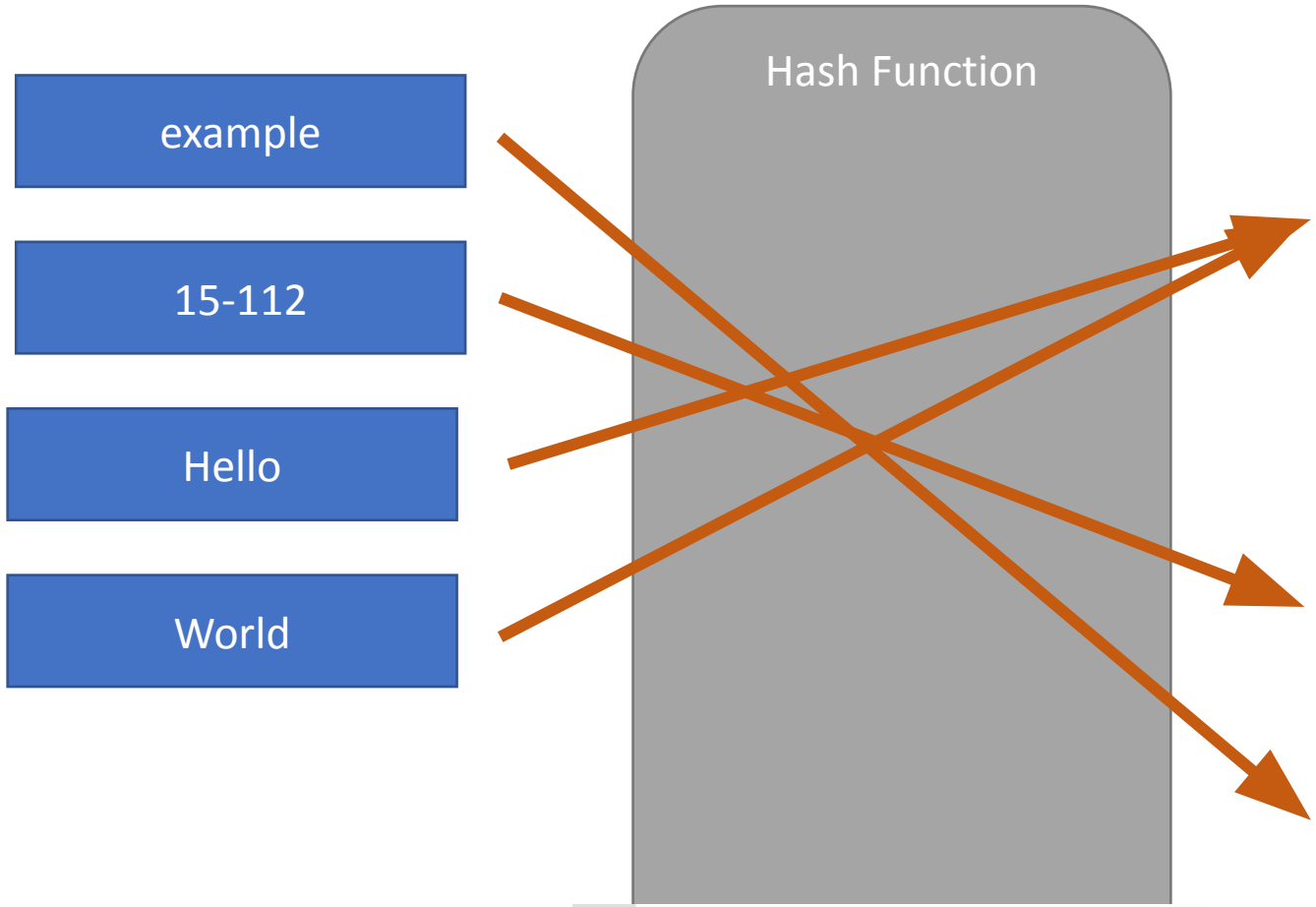
```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```

What can we do??



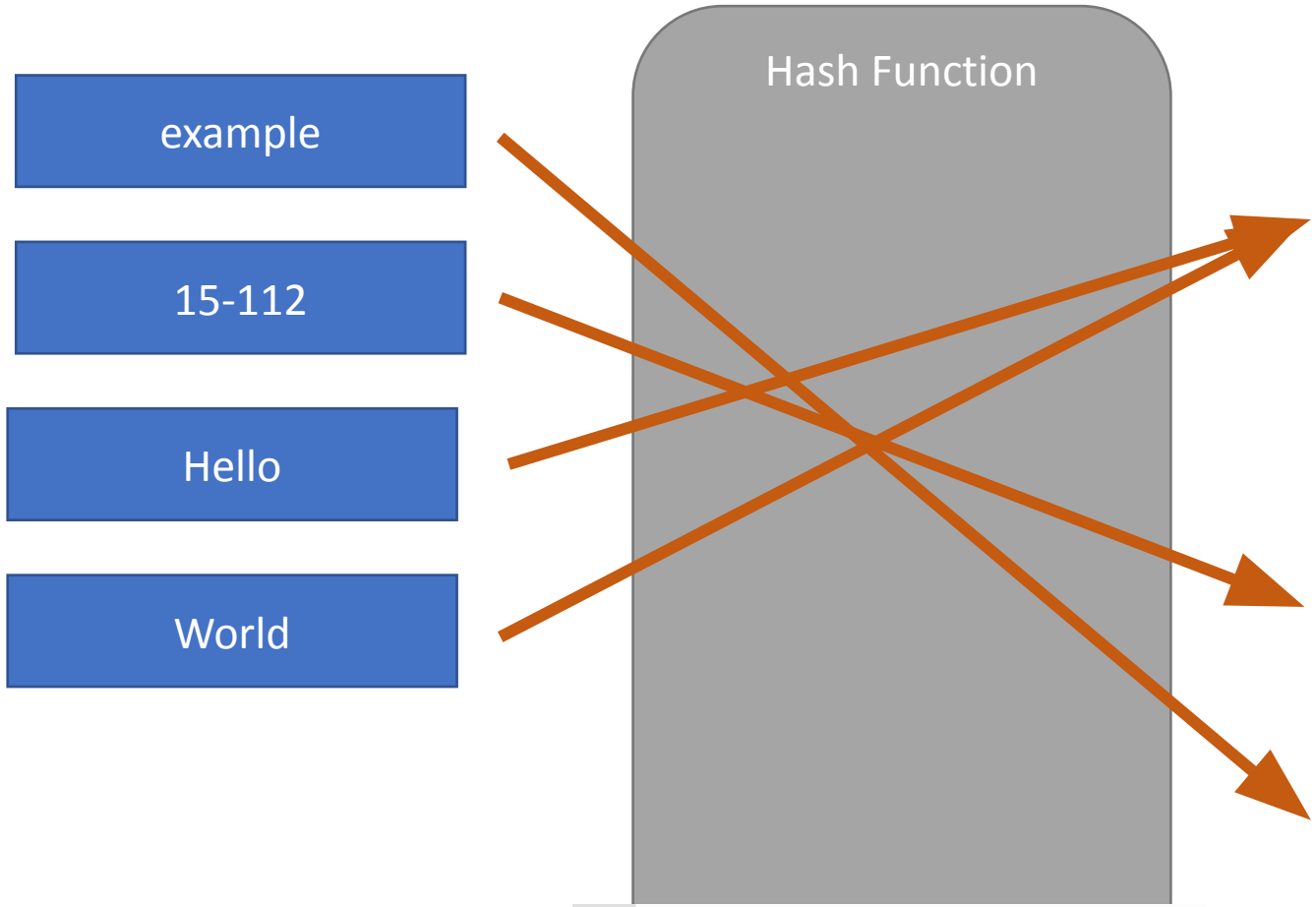
Index	Value
0	“Hello”
1	
2	
3	
4	
5	“15-112”
6	
7	
8	“example”
9	

```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```



Index	Value
0	["Hello", "World"]
1	
2	
3	
4	
5	["15-112"]
6	
7	
8	["example"]
9	

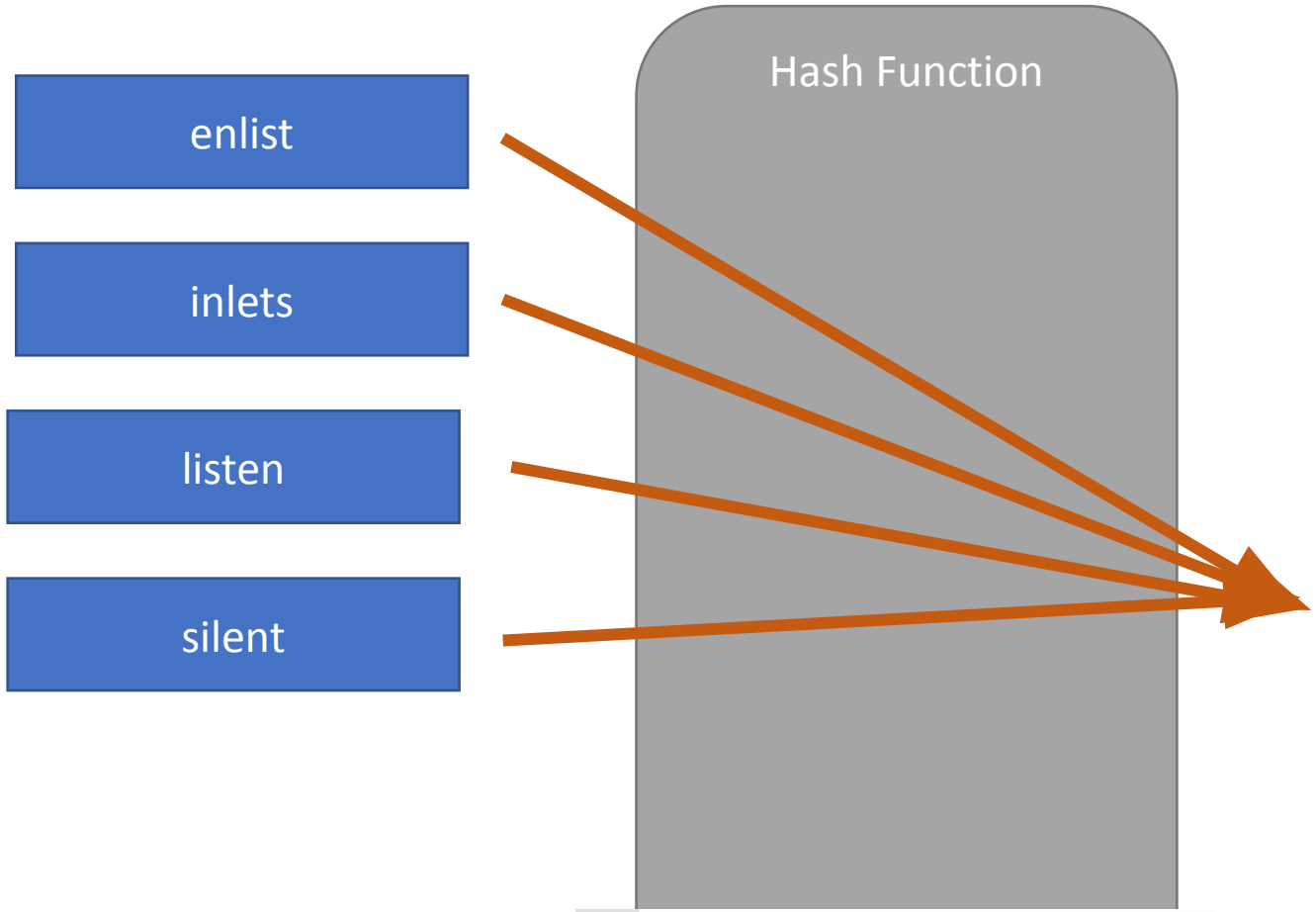
```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```



Index	Value
0	["Hello", "World"]
1	
2	
3	
4	
5	["15-112"]
6	
7	
8	["example"]
9	

```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```

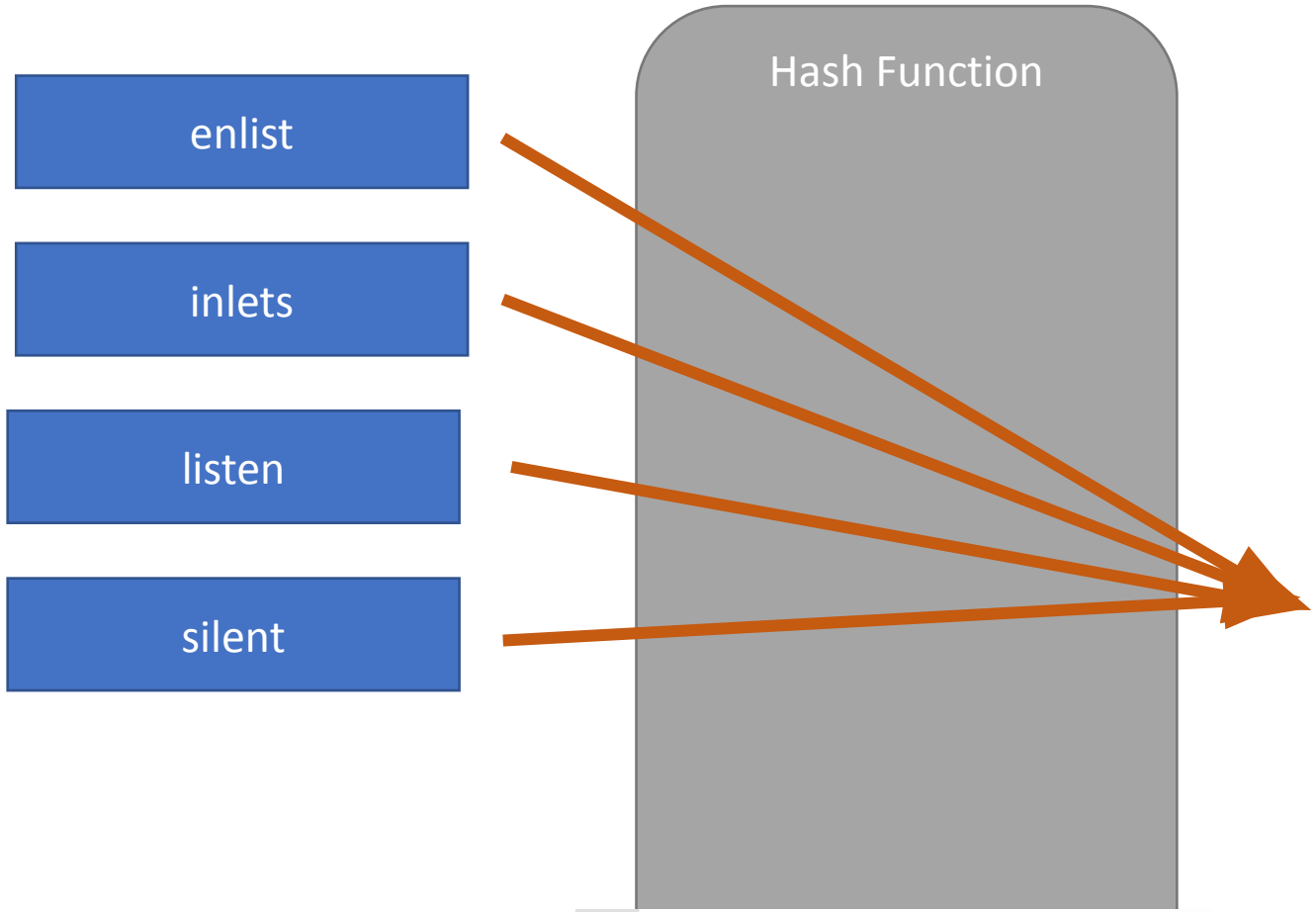
Problem?



Index	Value
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```

Problem?



Index	Value
0	
1	
2	
3	
4	
5	[“enlist”, “inlets”, “listen”, “silent”]
6	
7	
8	
9	

```
1  def myhash(s):
2      total = 0
3      for c in s:
4          total += ord(c)
5      return total%10
```

What makes a good hash function?

- Fast
- Minimize duplicate values

You can use the built-in `hash(s)` function