

# Хэшируемость: метод `hash()`

# План урока

---

- Хеш-функция
- Метод hash
- Хэш таблица

# ХЕШ-ФУНКЦИЯ

(*hash* – «превращать в фарш», «мешанина»)

это функция, которая сопоставляет объекту  
целое число фиксированной длины



# ХЕШ

Значение возвращаемое Хеш функцией



# **ХЕШ-ФУНКЦИЯ**

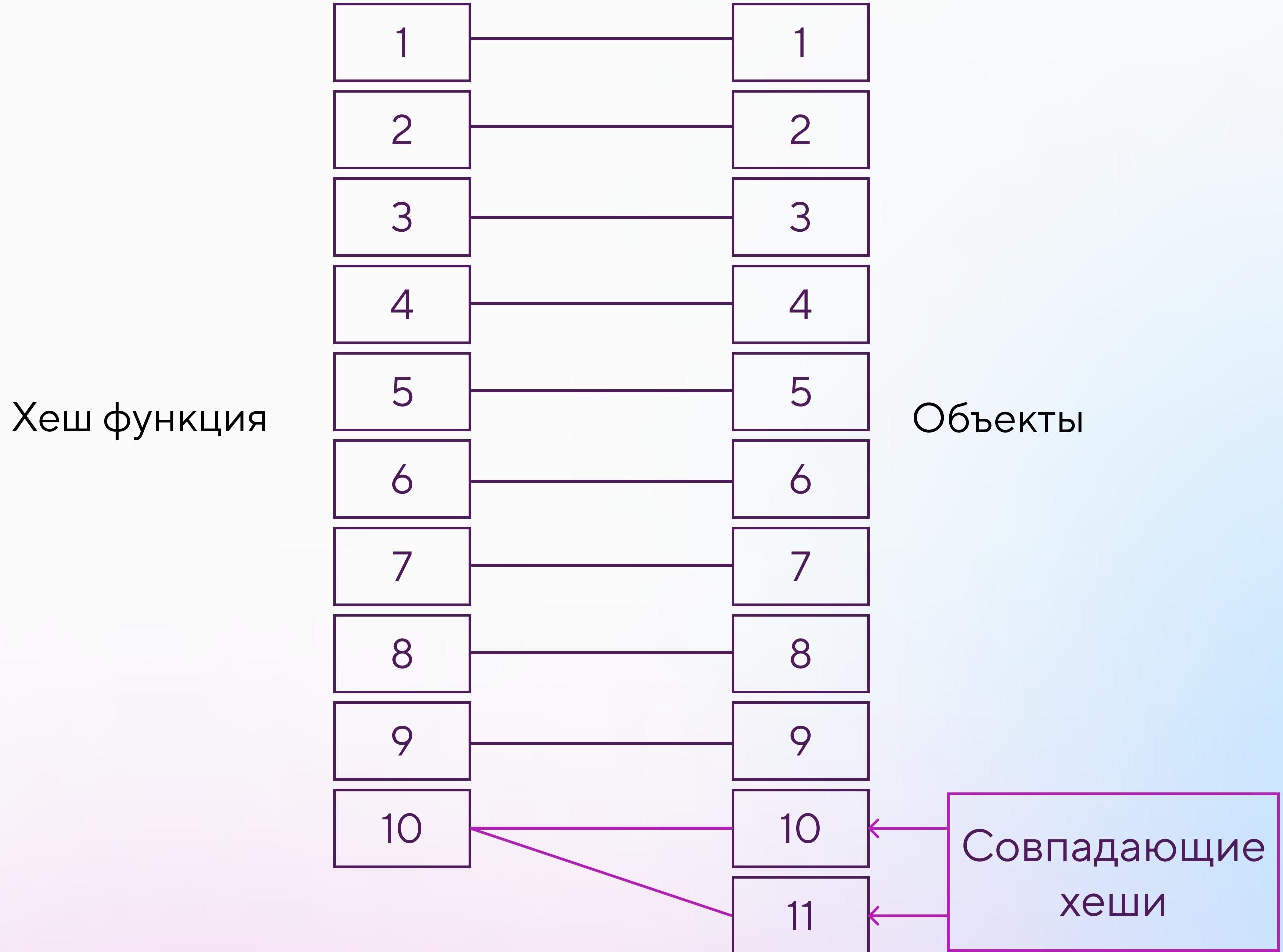
Хеш-функция используется для обеспечения уникальности и целостности данных, быстрого поиска и проверки целостности файлов

# ИДЕАЛЬНАЯ ХЕШ-ФУНКЦИЯ

это функция, которая обеспечивает уникальный хеш-код для каждого входного значения, не имеет коллизий (совпадений хеш-кодов для разных входных значений) и имеет константное время выполнения



# Хеш-функция



~~Идеальная  
Хеш-функция~~

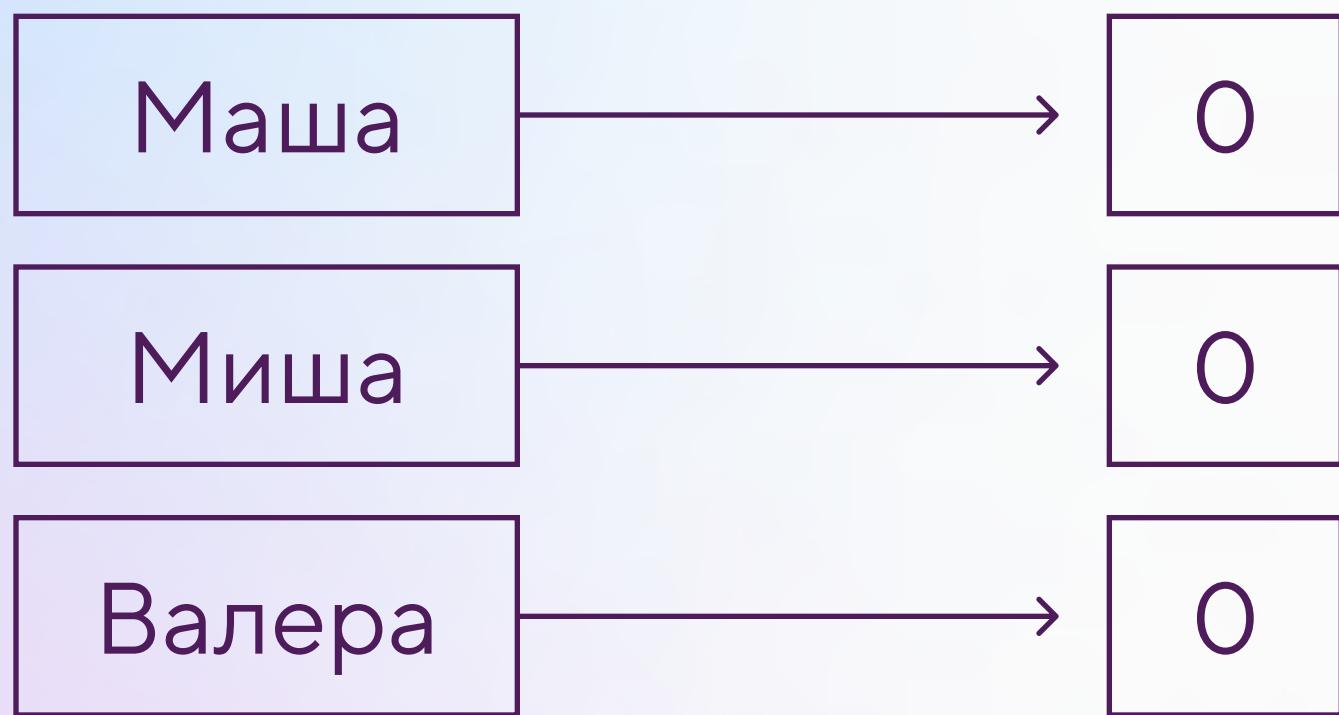


**Хорошая  
хеш-функция**

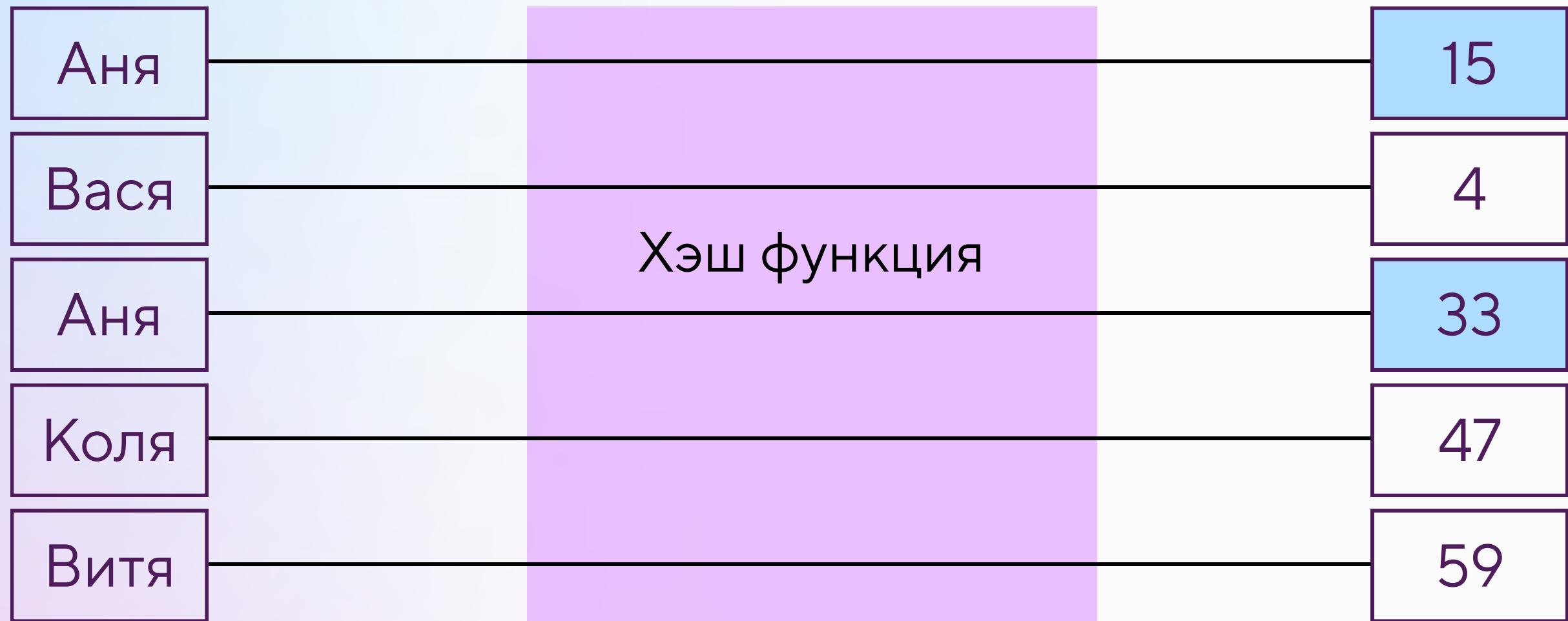
# **ХОРОШАЯ ХЕШ-ФУНКЦИЯ**

это хеш-функция, которая равномерно распределяет хеши объектов по множеству всех допустимых хешей

# плохая хеш-функция



# ХОРОШАЯ ХЕШ-ФУНКЦИЯ





Хеш функция должна  
возвращать одинаковые хеши  
для одинаковых объектов

# Хеш-функция

о о о	код
<pre>n = 1000  # hash values from 0 to 1000 a = 173← p = 1103</pre>	

Простое число до  $n$

# Хеш-функция

о о о

код

```
n = 1000
```

```
# hash values from 0 to 1000
```

```
a = 173
```

```
p = 1103 ←
```

Простое число больше n

# Хеш-функция

ооо | код

```
n = 1000
```

```
a = 173
```

```
p = 1103
```

```
x = 12435656
```

```
h = (x * a) % p % n ←
```

```
print(h)
```

```
x = 12435526752
```

```
h = (x * a) % p % n ←
```

```
print(h)
```

Вычисляем хеши

ооо | вывод

```
>>> 78
```

```
>>> 902
```

# Полиномиальный хеш

ооо | код

```
def hash_polynomial(s, x, p):
    h = 0
    for i, ch in enumerate(s):←
        h += ord(ch) * (x ** i % p) % p
        h = h % p

    return h
```

Нумеруем символы строки  
в порядке возрастания  
начиная с нуля

# Полиномиальный хеш

ооо | код

```
def hash_polynomial(s, x, p):
    h = 0
    for i, ch in enumerate(s):
        h += ord(ch) * (x ** i % p) % p ←
        h = h % p

    return h
```

Сумма всех кодов  
символов, умноженных на  $x$   
в степени равной индексу  
элемента в строке

# Полиномиальный хеш

ооо | код

```
def hash_polynomial(s, x, p):
    h = 0
    for i, ch in enumerate(s):
        h += ord(ch) * (x ** i % p) % p ←
        h = h % p

    return h
```

X - какое-то простое число, превышающее размер алфавита. P - простое число, превышающее размер множества допустимых значений хеша

# Полиномиальный хеш

ooo | код

```
def hash_polynomial(s, x, p):
    h = 0
    for i, ch in enumerate(s):
        h += ord(ch) * (x ** i % p) % p
        h = h % p

    return h

x = 263
p = 1000000007

print(hash_polynomial("Hello", x, p))
print(hash_polynomial("world", x, p))
print(hash_polynomial("Hello", x, p))
print(hash_polynomial("Hella", x, p))
```

ooo | вывод

```
>>> 35081703
>>> 407643594
>>> 35081703
>>> 54174318
```

# ХЕШИРУЕМЫЕ ОБЪЕКТЫ

это встроенные неизменяемые типы данных, а также другие типы данных, для которых определена функция `hash` в классе

## Мы уже изучили

- Целые
- Вещественные
- Комплексные числа
- Строки
- Кортежи

# Хеширование целых чисел

о о о

код

```
print(hash(1256))
```

о о о

вывод

```
>>> 1256
```

Хеш для целых чисел это остаток  
от деления числа на число  $2^{61} - 1$

# Хеш-функция

ооо

код

```
print(hash(1256))
print(hash(-1))
print(hash(4258468))
print(hash(35735824681486846854))
print(hash(2305843009213693950))
print(hash(2305843009213693951))
```

ооо

вывод

```
>>> 1256
>>> -2
>>> 4258468
>>> 1148179543281437589
>>> 2305843009213693950
>>> 0
```

# Хеширование вещественных чисел

о о о

код

```
print(hash(23.64))
```

о о о

вывод

```
>>> 1475739525896765463
```

# Хеширование комплексных чисел

о о о

код

```
print(hash(3 + 5j))
```

о о о

вывод

```
>>> 5000018
```

# Хеширование кортежей

ooo

код

```
print(hash((4, -2, "Hello", 2.5)))
print(hash((4, -2, "Hello", 2.5)))
```

ooo

вывод

```
>>> 8242004071891041952
>>> 8242004071891041952
```

Значение хеша кортежа может меняться от запуска программы к запуску, так как при вычислении хешей для коллекций используется случайное число, генерируемое при запуске программы

# Хеширование строк

ооо

код

```
print(hash("Hello"))
```

ооо

вывод

```
>>> -7331187157525728317
```

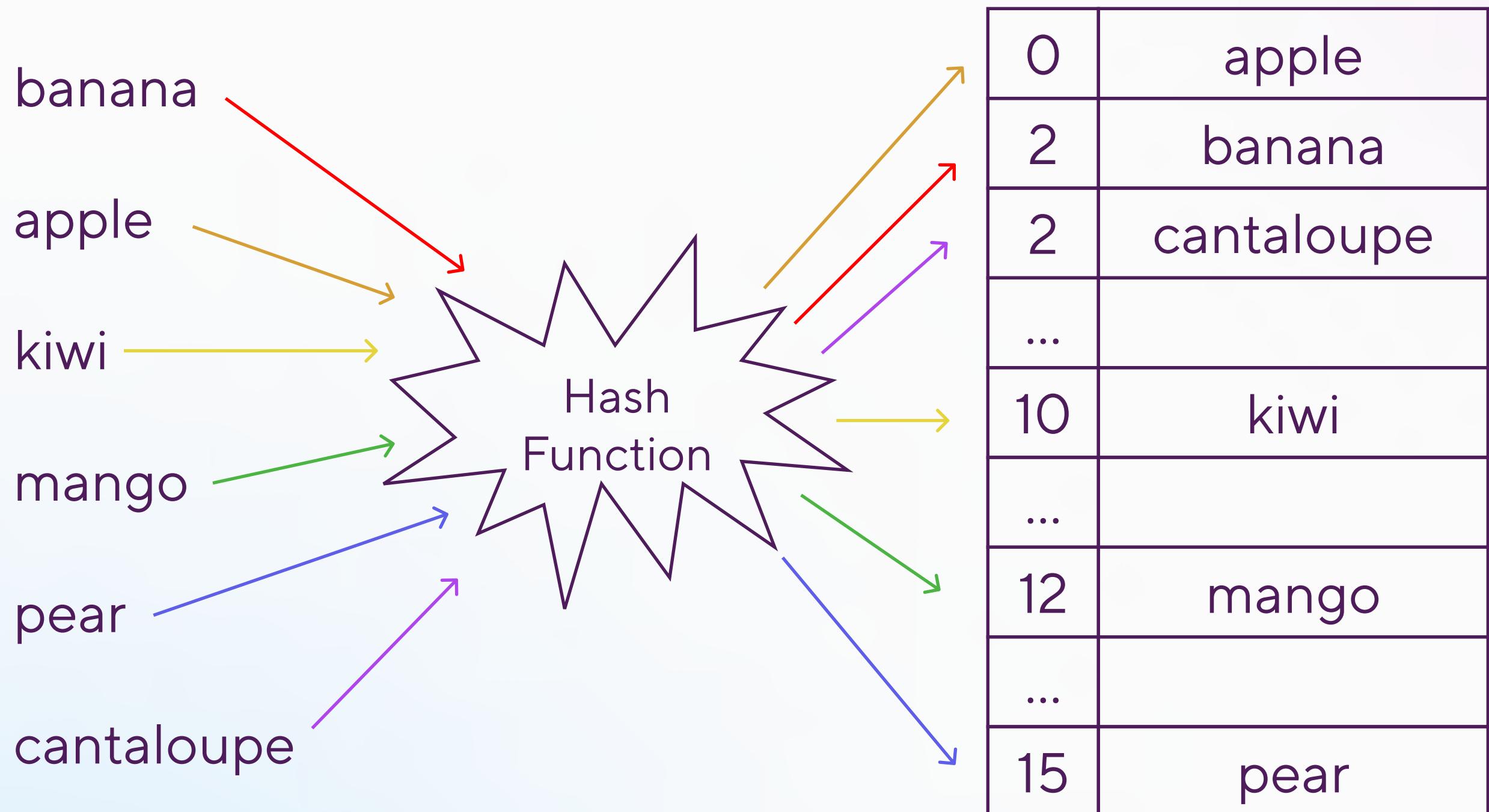
# ХЕШ-ТАБЛИЦА

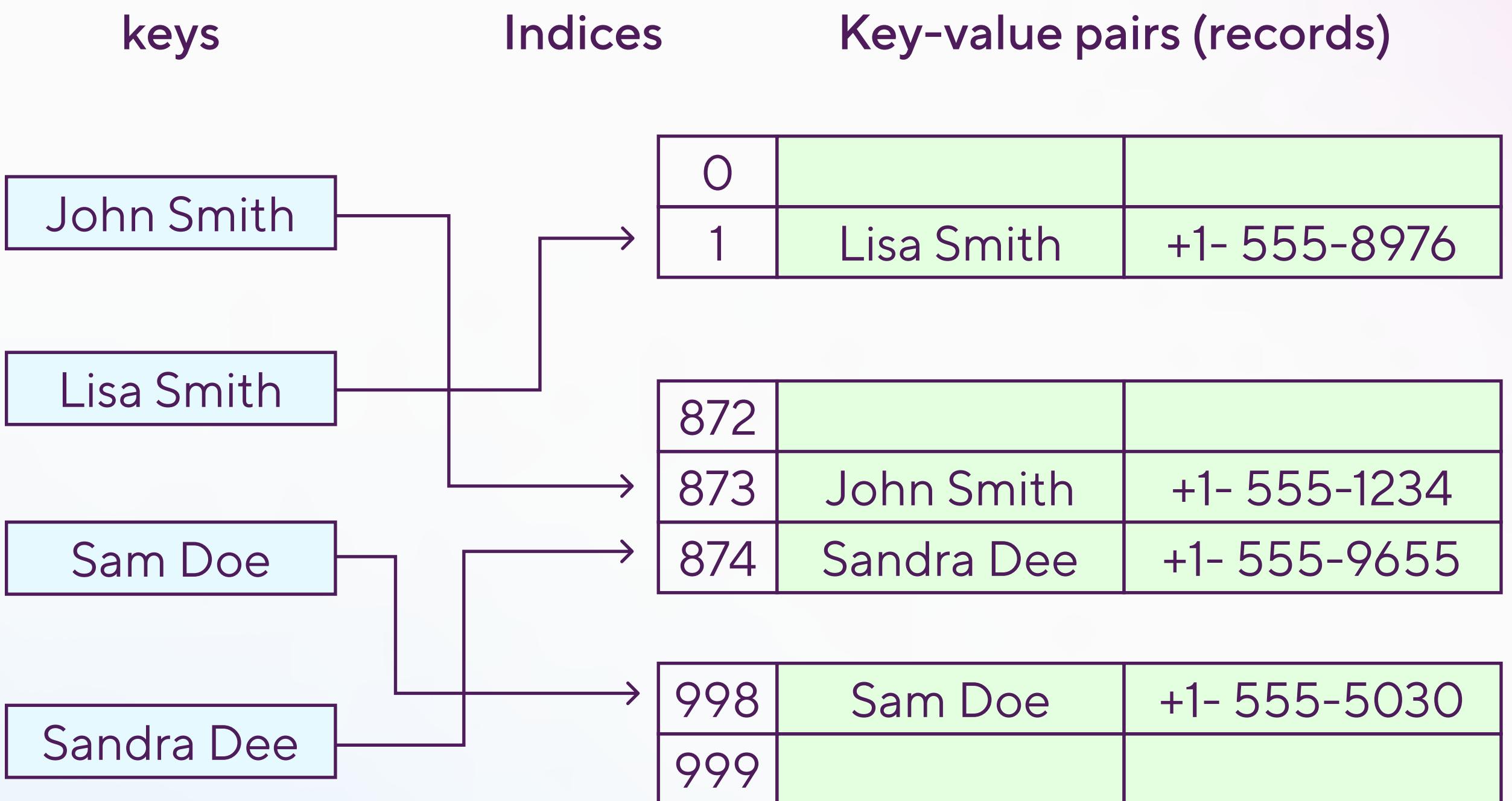
это неупорядоченная коллекция, которая представляет из себя множество ключей, связанных с какими-то данными

## КЛЮЧИ

это хешируемые объекты. Ключи – это уникальные объекты для хеш-таблицы

# Элементы в таблице хранятся в неупорядоченном виде





# КОЛЛИЗИЯ

это ситуация, когда хеши двух разных объектов совпадают



# Разрешение коллизий: Бакеты

