Welcome to Week 6

Tutorial 11

Mid-Course Checkup

How many of you are still alive?

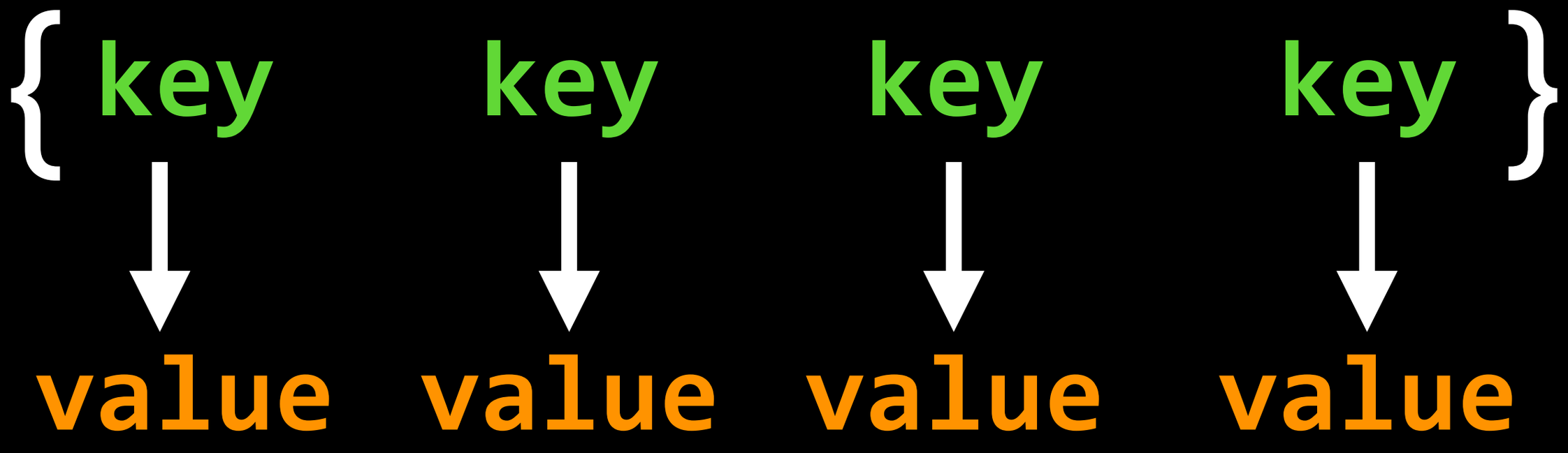
On a scale of 1 to 10:
How difficult do you
find this course?

Which part of this course do you find the most challenging?

What can I do to help you achieve your goals in this course?

Dictionaries

Python Dictionaries



A dictionary is a collection which is unordered, mutable and indexed.
In Python dictionaries are written with curly brackets, and they have keys and values.

Python Dictionaries

key value

```
myDict = { “Name” : “Kara”
           “Age” : 19
           “Occupation” : “TA”
           “Program” : “CS” }
```

Setting Up a Dictionary

You know that to create:

- A new String: `myStr = ""`
- A new List: `myList = []`

We know that dictionaries are denoted with curly braces {} so, intuitively:

```
myDict = {}   OR   myDict = dict()
```

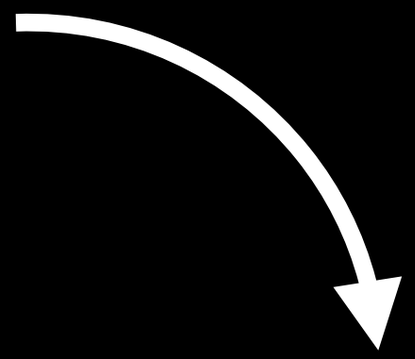
Adding Values to a Dictionary

To add a new value to a dictionary, we must add a key, and give it a value.

```
myDict[key] = value
```

For example:

```
kara = dict()
kara[“name”] = “Kara”
kara[“age”] = 19
kara[“job”] = “TA”
```



```
{“name”:“Kara”, “age”:19, “job”:“TA”}
```

Reading Values from a Dictionary

To read an existing value to a dictionary, we must reference a key.

```
value = myDict[key]
```

For example:

```
{“name”:“Kara”, “age”:19, “job”:“TA”}
```

```
kara[“name”]    → “Kara”
kara[“age”]     → 19
kara[“job”]     → “TA”
kara[“address”] → ERROR
```

Removing Values from a Dictionary

To remove an existing value to a dictionary, we must pop the key value pair by referencing a key.

```
myDict.pop(key)
```

For example:

```
{“name”:“Kara”, “age”:19, “job”:“TA”}
```

```
kara.pop(“job”)
```

```
{“name”:“Kara”, “age”:19}
```

Updating a Key Value Pair

Sometimes we have to edit values that have been assigned to a key. We can update by referencing a key.

```
myDict.update(key:newValue)
```

For example:

```
{“name”:“Kara”, “age”:19, “job”:“TA”}
```

```
kara.update(“age”:300)
```

```
{“name”:“Kara”, “age”:300}
```

Important Dictionary Methods

`Dictionary.copy()`
Returns a copy of the dictionary

`Dictionary.clear()`
Removes all elements from the dictionary

`Dictionary.keys()`
Returns a list of the dictionary's keys

`Dictionary.values()`
Returns a list of the dictionary's values

Old Friends We Can Rely On

`in` [KEYWORD]

Is a key in our dictionary?

`len()`

How many keys are in our dictionary?

`type()`

Is our variable a dictionary?

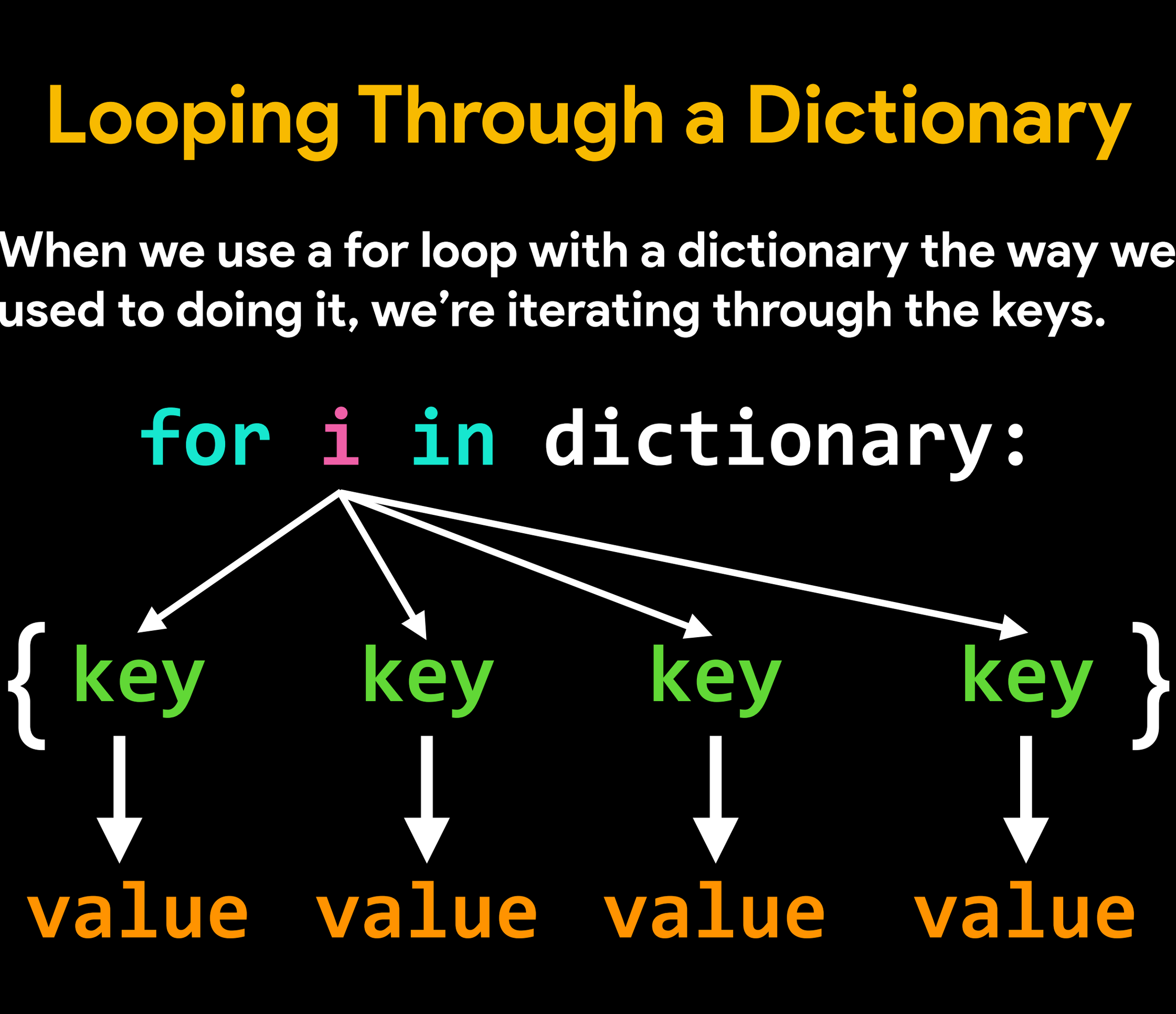
`del` [KEYWORD]

Clears the value of a variable

Looping Through a Dictionary

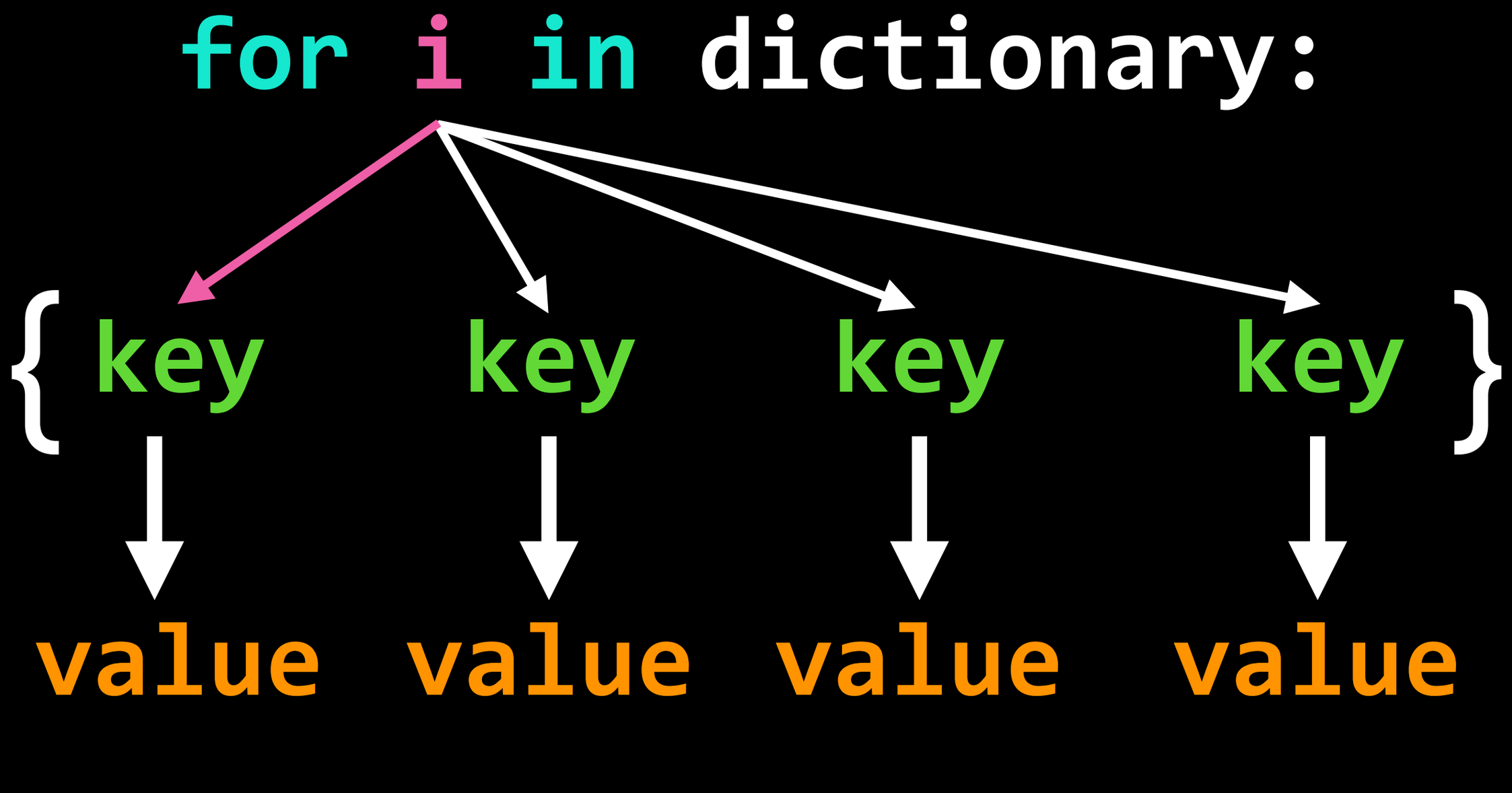
When we use a for loop with a dictionary the way we're used to doing it, we're iterating through the keys.

```
for i in dictionary:
```



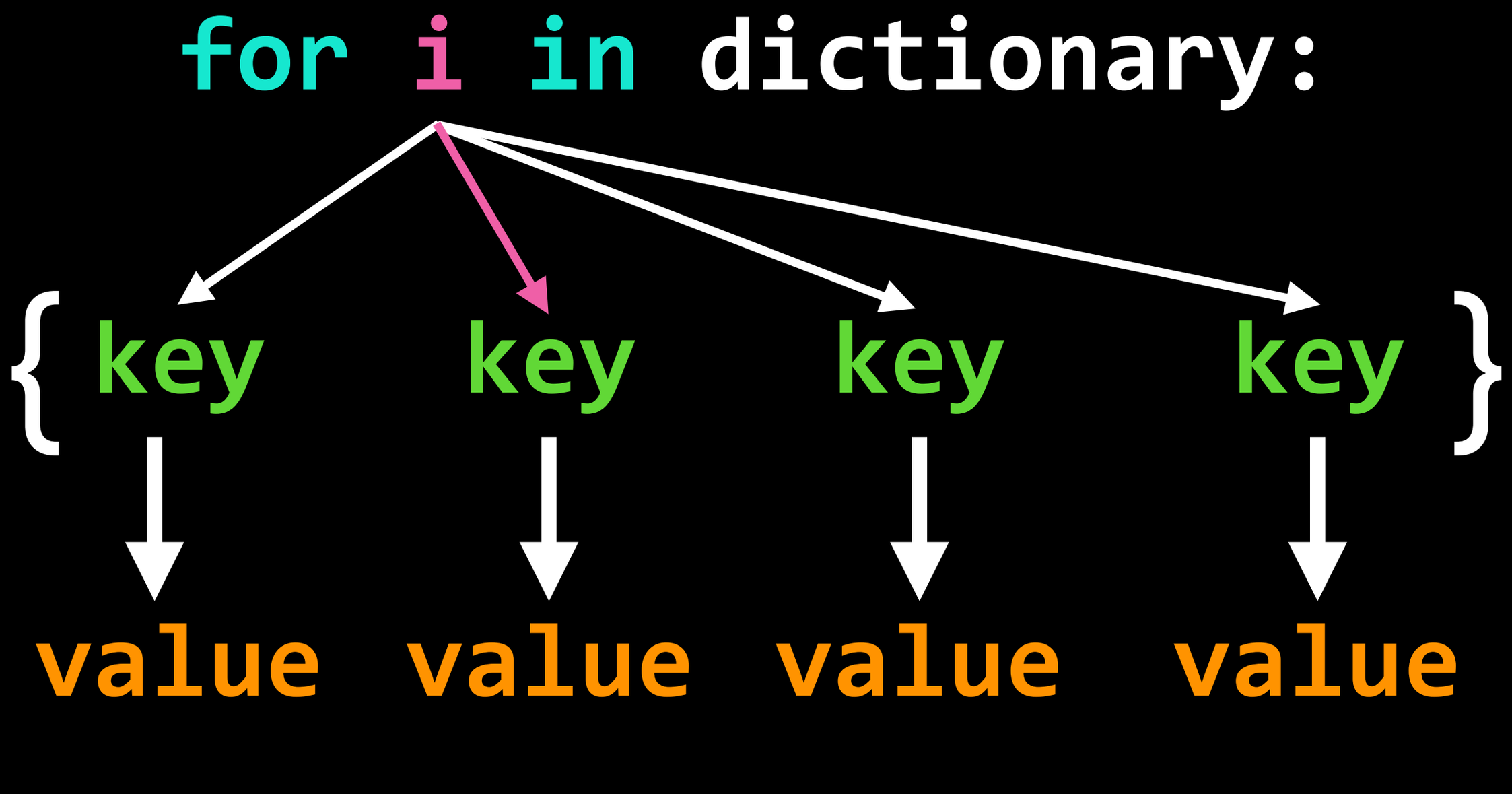
Looping Through a Dictionary

When we use a for loop with a dictionary the way we're used to doing it, we're iterating through the keys.



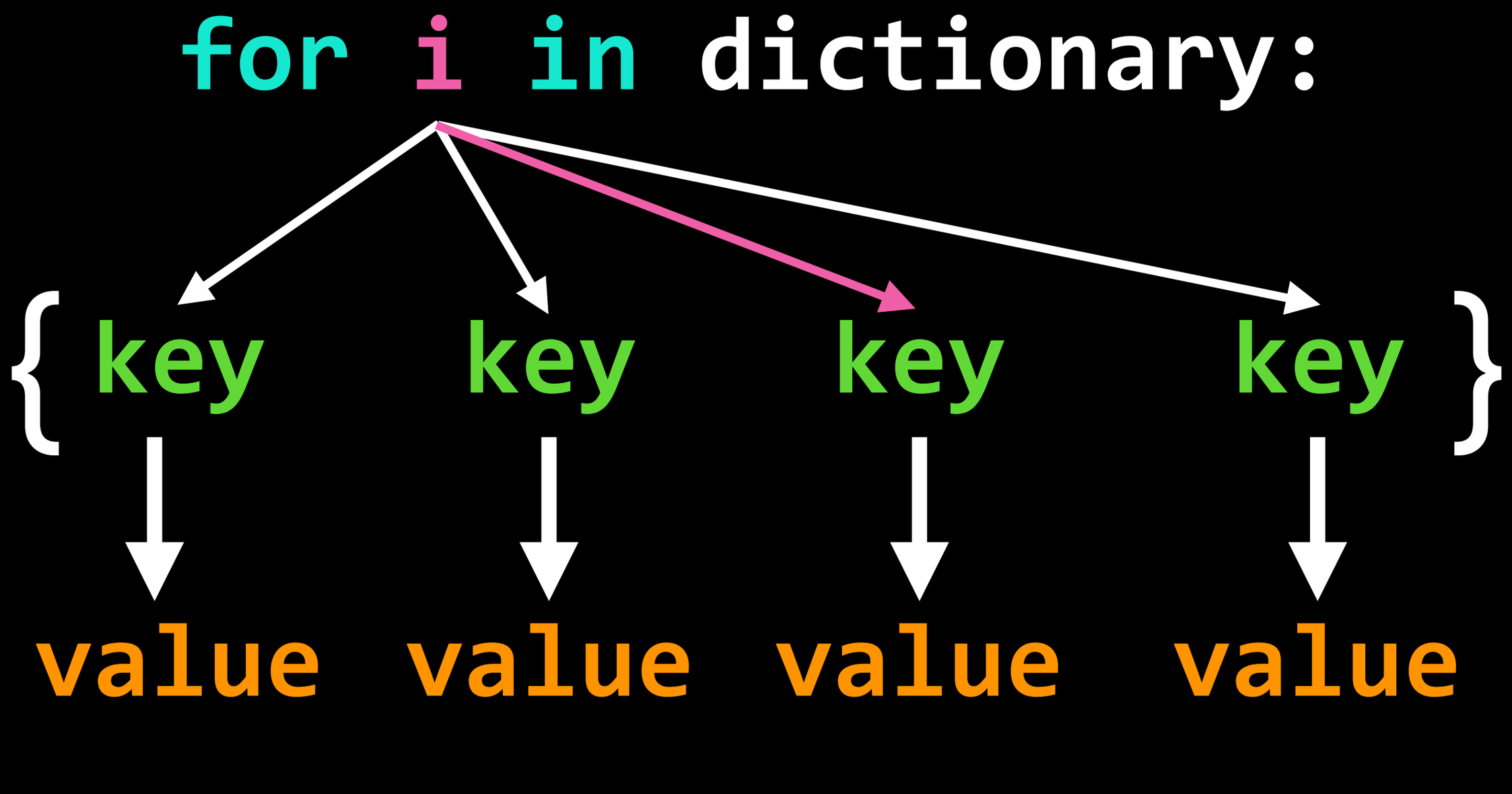
Looping Through a Dictionary

When we use a for loop with a dictionary the way we're used to doing it, we're iterating through the keys.



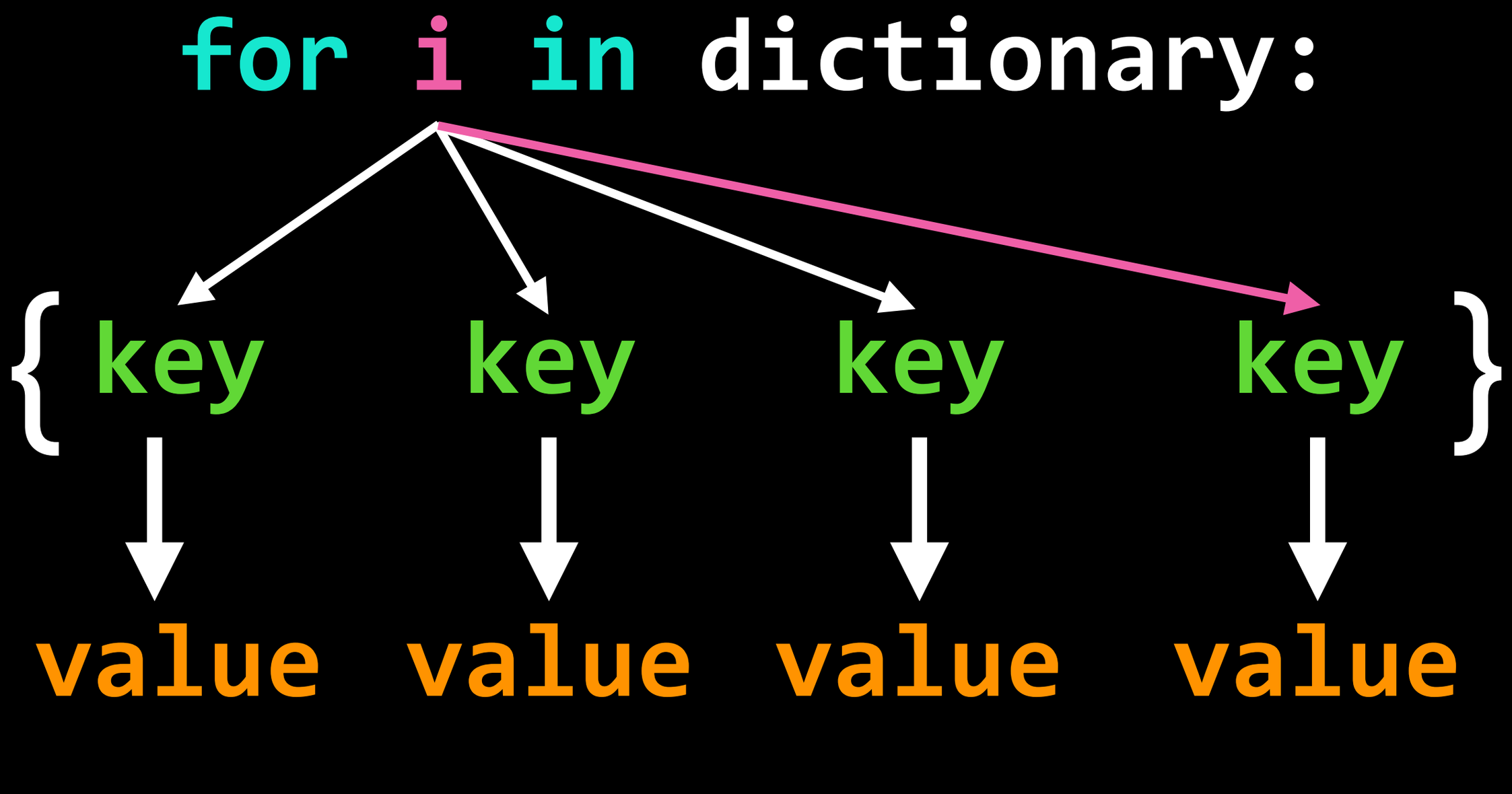
Looping Through a Dictionary

When we use a for loop with a dictionary the way we're used to doing it, we're iterating through the keys.



Looping Through a Dictionary

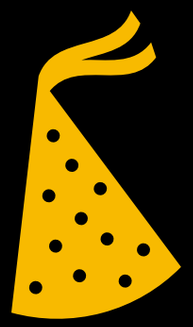
When we use a for loop with a dictionary the way we're used to doing it, we're iterating through the keys.



Fibonacci Numbers are defined as:

- The 0th fibonacci number is 0
- The 1st fibonacci number is 1
- Every fibonacci number is the sum of the previous two fibonacci numbers.

```
1   def fib(n):
2       a = 0
3       b = 1
4       if n == 0:
5           return a
6       elif n == 1:
7           return b
8       else:
9           for i in range(2, (n + 1)):
10              c = (a + b)
11              a = b
12              b = c
13          return b
```



EXTRA SPECIAL

ONE LINE CHALLENGE (big prize)

Fibonacci Numbers are defined as:

- The 0th fibonacci number is 0
- The 1st fibonacci number is 1
- Every fibonacci number is the sum of the previous two fibonacci numbers.

Using only 1 line of code and no loops:

- Write a function fib(n) which returns the nth fibonacci number; where n can be any positive integer.

Using only 1 line of code and no loops:

- Write a function fib(n) which returns the nth fibonacci number; where n can be any positive integer.

SOLUTION:

```
1  def fib(n):
2      return(n if n < 2 else fib(n - 2) + fib(n - 1))
```