Dictionaries in Python

Dictionaries in Python

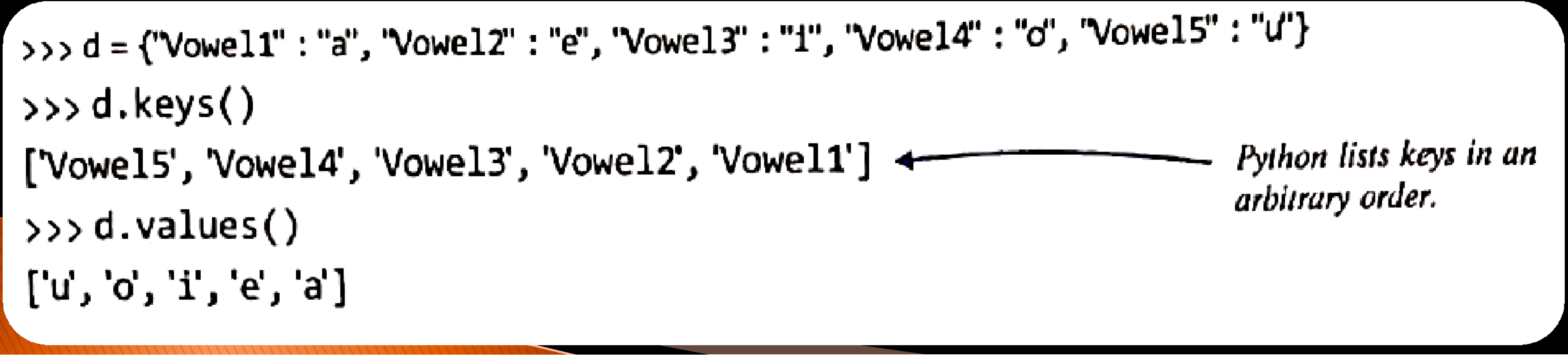
- Python dictionary are a collection of some value pairs.
- Dictionary are mutable, unordered collections with elements in the form of a key : value pairs that associate keys to values.

065 CREATING A DICTIONARY

```
teacher={"dimple":"computer science",
                "Karen":"sociology",
                "harpreet":"mathematics",
                "sabah":"legal studies"}
print(teacher)
```

Accessing Elements of a Dictionary

- To see all keys in our dictionary in one go,
- you may write<dictionary>.value ( ), as shown below



```
>>> d = {"Vowel1" : "a", "Vowel2" : "e", "Vowel3" : "i", "Vowel4" : "o", "Vowel5" : "u"}
>>> d.keys()
['Vowel5', 'Vowel4', 'Vowel3', 'Vowel2', 'Vowel1']
>>> d.values()
['u', 'o', 'i', 'e', 'a']
```

066 ACCESING THE ELEMENTS OF A DICTIONARY

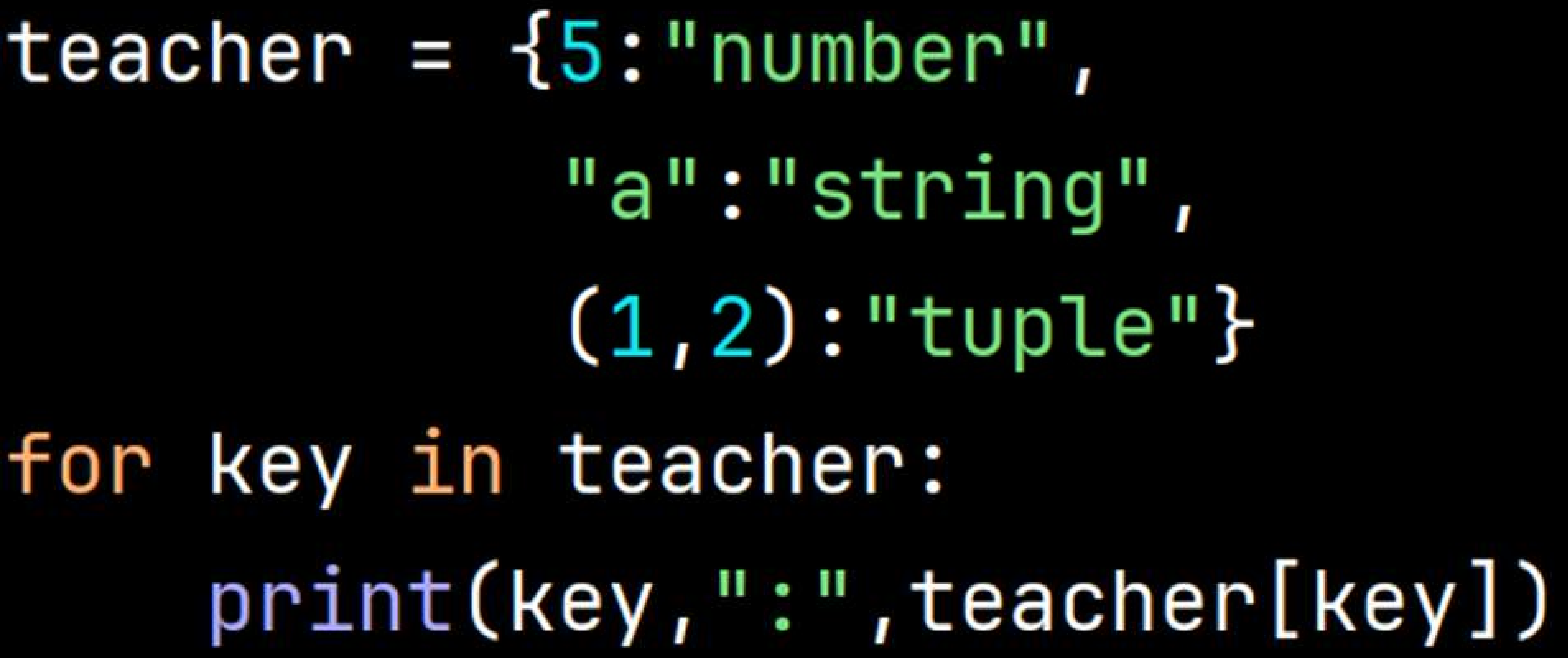
```
teacher={"Abhinav":"best boy",
         "Korku":"worst boy ",
         "Viraj":"average boy",}
print(teacher.keys())
print(teacher.values())
```

Dictionary Operations

- **Traversing a dictionary**
- Reversal of a collection means assigning and processing each element of it.
- The for loop makes it easy to travel for loop over the items in our dictionary, as per following syntax:

```
For <item> in <dictionary>:
        Process each item here
```

067 TRAVERSING A DICTIONARY



```
teacher = {5:"number",
           "a":"string",
           (1,2):"tuple"}
for key in teacher:
    print(key,":",teacher[key])
```

Dictionary Operations

- **Adding Elements to Dictionary**
- You can add new elements (key: value pair) to a dictionary using assignment as per the following syntax.
- But the key being added must not exist in dictionary and must be unique.
- If the key already exist, then this statement will change the value of existing key and no new entry will be added to dictionary

```
        <dictionary>[<key>] = <value>
```

068 ADDING ELEMENTS YO DICTIONARY

```python
employee = {'name':'John','salary':100000,'age':25}
employee['salary'] = 200000
print(employee)
```

Dictionary Operations

- **Updating Existing Elements in a Dictionary**
- Updating an element is similarly to what we did just now.
- That is, you can change value of an existing key using assignment as per following syntax:

```
Dictionary>[<key>]=<value>
```

069 UPDATING ELEMENTS DICTIONARY

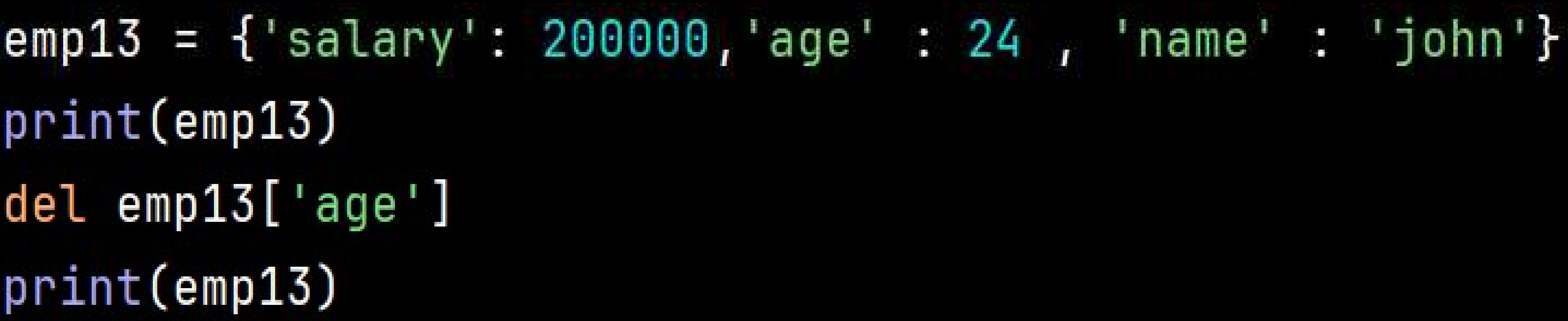
```python
n = int(input("How Many Students ? "))
CompWinners = { }
for a in range(n) :
    key = input("Name of the student : ")
    value = int(input("Number of Competition won :"))
    CompWinners[key] = value
print("The dictionary now is :")
print(CompWinners)
```

Dictionary Operations

- **Deleting Elements from a Dictionary**
- There are two methods for deleting elements from additionally
- (I) To delete a dictionary element for a dictionary entry,
- i.e., a key value pair, you can use delete command. The syntsx for doing so As given below:

del<dictionary>[<key>]

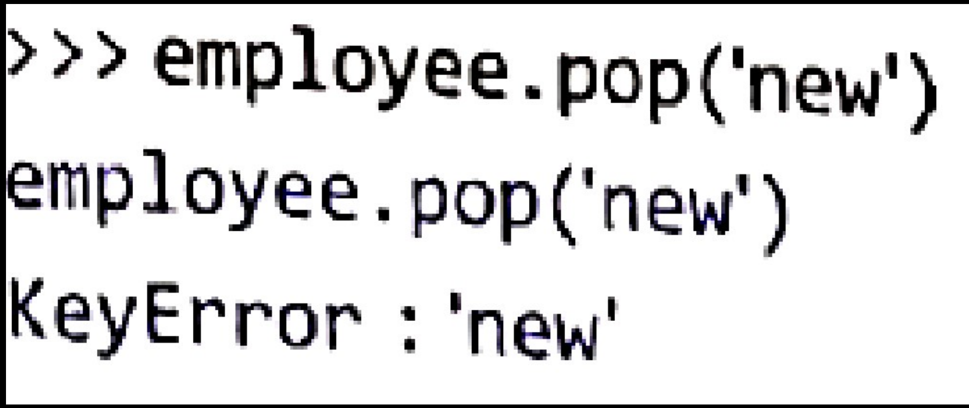
070 DELETING ELEMENTS DICTIONARY



```python
emp13 = {'salary': 200000,'age' : 24 , 'name' : 'john'}
print(emp13)
del emp13['age']
print(emp13)
```

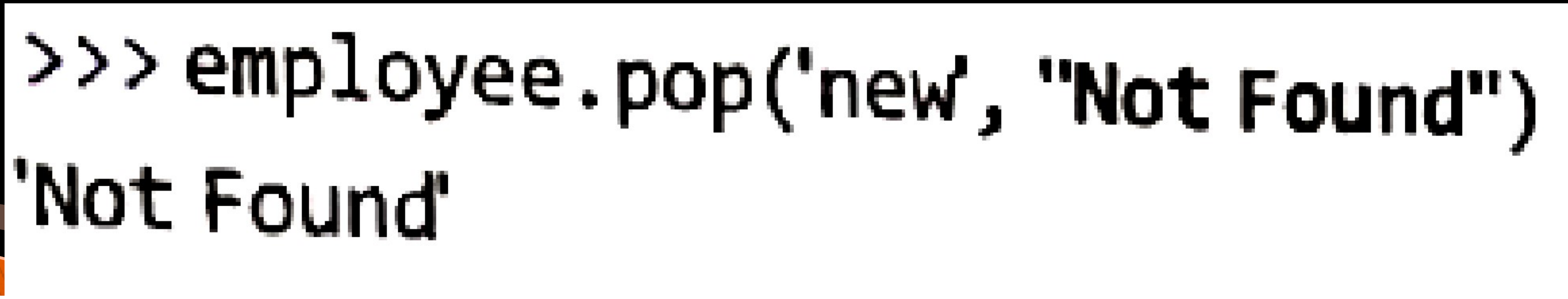
Dictionary Operations

- If you tried to delete a key which does not exist, the python returns error. See below:



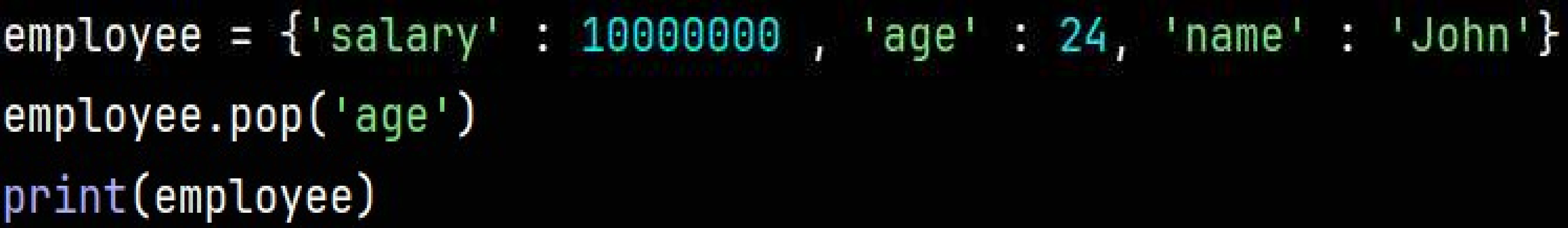
```
>>> employee.pop('new')
employee.pop('new')
KeyError : 'new'
```

- <Dictionary>. Pop (<key in case of error show me>)
- For example:



```
>>> employee.pop('new', "Not Found")
'Not Found'
```

071 POP DICTIONARY

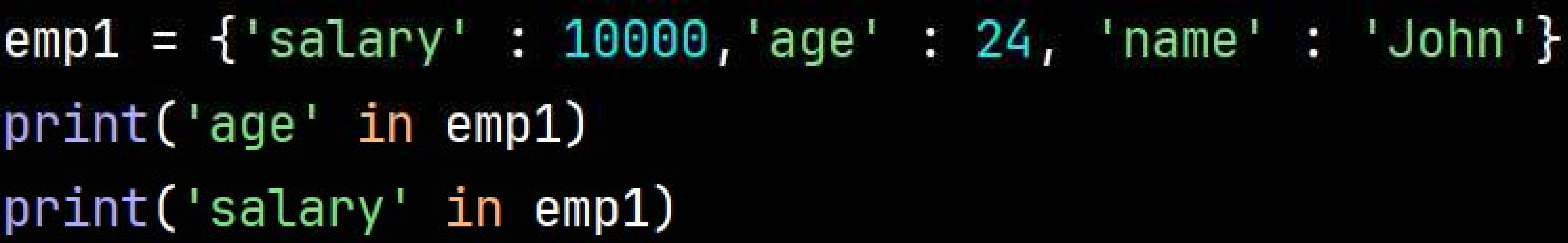


```python
employee = {'salary' : 10000000 , 'age' : 24, 'name' : 'John'}
employee.pop('age')
print(employee)
```

Dictionary Operations

- **Checking for existence of a key**
- Usual membership operators in and not in work with dictionary as well.
- But they can check for the existence of key only.
- You may use them as per syntax given below:
- <key> in <dictionary>
- <key> not in <dictionary>

072 CHEKING EXISTENCE DICTIONARY



```python
emp1 = {'salary' : 10000,'age' : 24, 'name' : 'John'}
print('age' in emp1)
print('salary' in emp1)
```

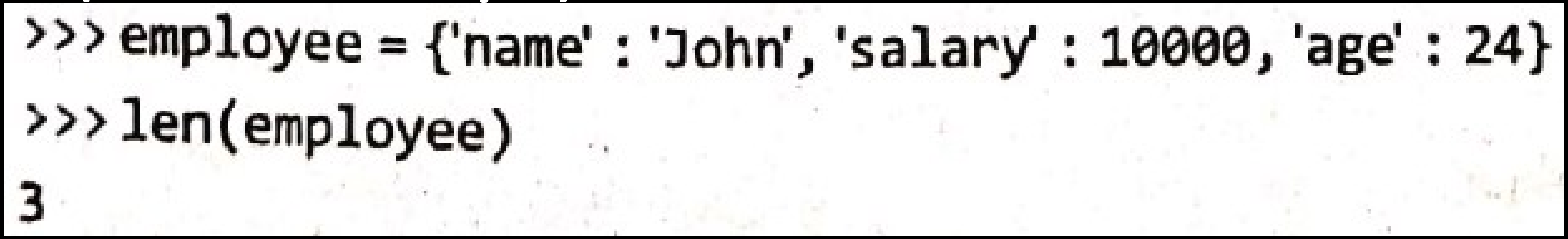
Dictionary Functions And Methods

1. The len( ) Method

- This method returns length of the dictionary,
- i.e., The count of elements (key: value pairs) in the dictionary.
- The syntax to use this method is given below:

  Len(<dictionary>)

- E.g.,



```
>>> employee = {'name' : 'John', 'salary' : 10000, 'age' : 24}
>>> len(employee)
3
```

073 THE LEN METHOD

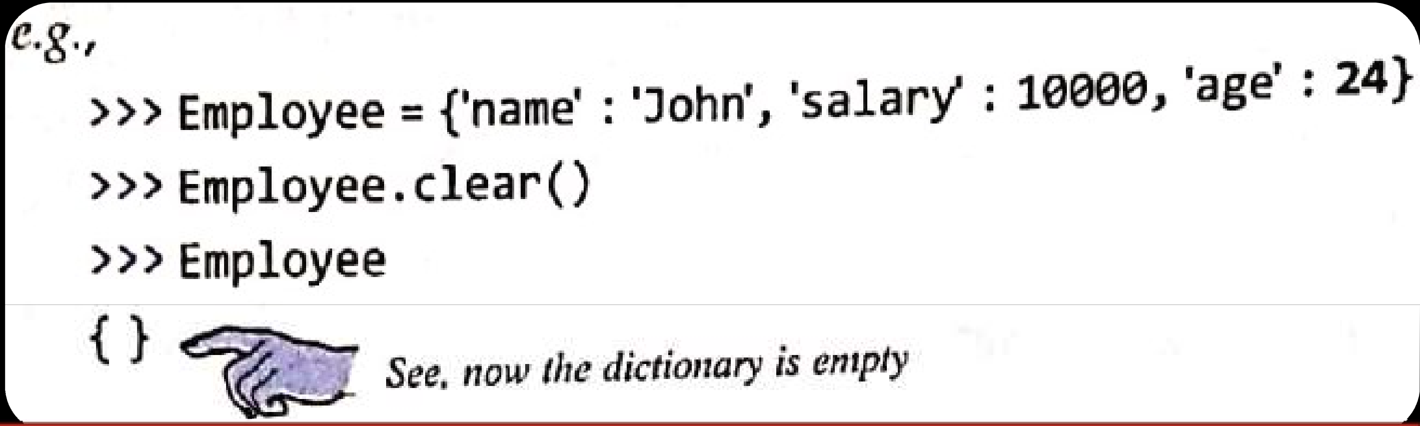
```python
emp1 = {'salary' : 10000,'age' : 24, 'name' : 'John'}
print('age' in emp1)
print('salary' in emp1)
```

Dictionary Functions And Methods

2. The clear ( ) Method

- This method remove all items from the dictionary and the dictionary become entity dictionary post this method.

<Dictionary>.clear()

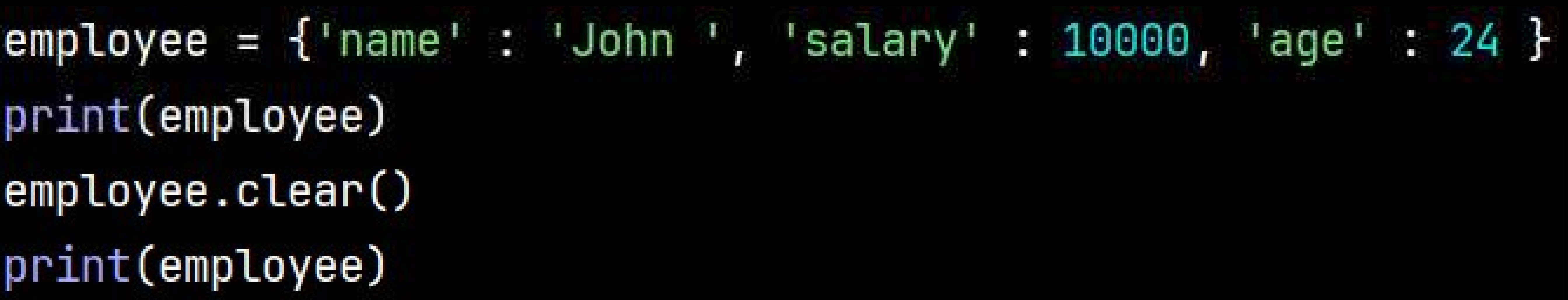


e.g.,

```
>>> Employee = {'name' : 'John', 'salary' : 10000, 'age' : 24}
>>> Employee.clear()
>>> Employee
{ }
```

Note :The clear( ) remove all the elements of a dictionary and mix it mtt dictionary will delete statement removes the complete dictionary as an object. After delete statement with a dictionary name, that dictionary object no more exist, not even empty dictionary.

074 THE CLEAR METHOD



```python
employee = {'name' : 'John ', 'salary' : 10000, 'age' : 24 }
print(employee)
employee.clear()
print(employee)
```

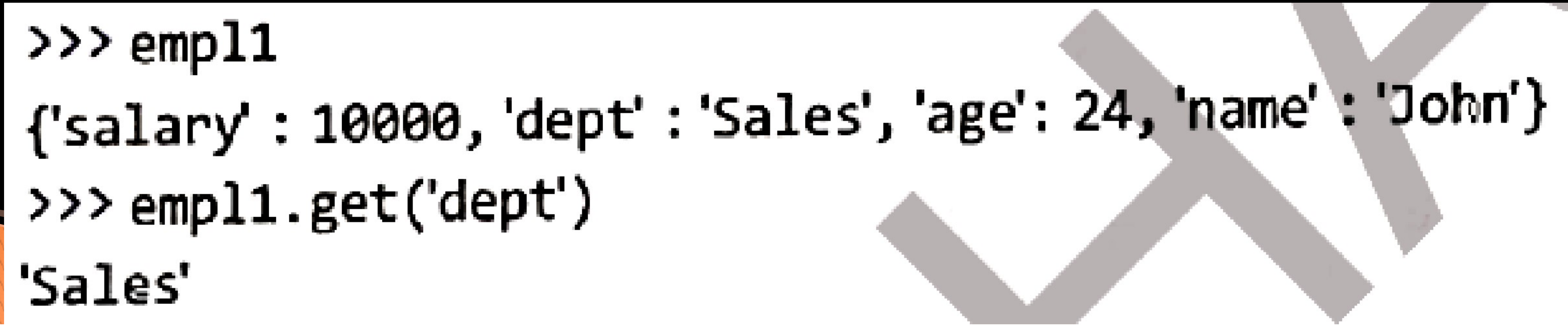
Dictionary Functions And Methods

3. The get ( ) Method

- With this method, you can get the item with the given key, similar to dictionary [key].
- If the key is not present, python will give error

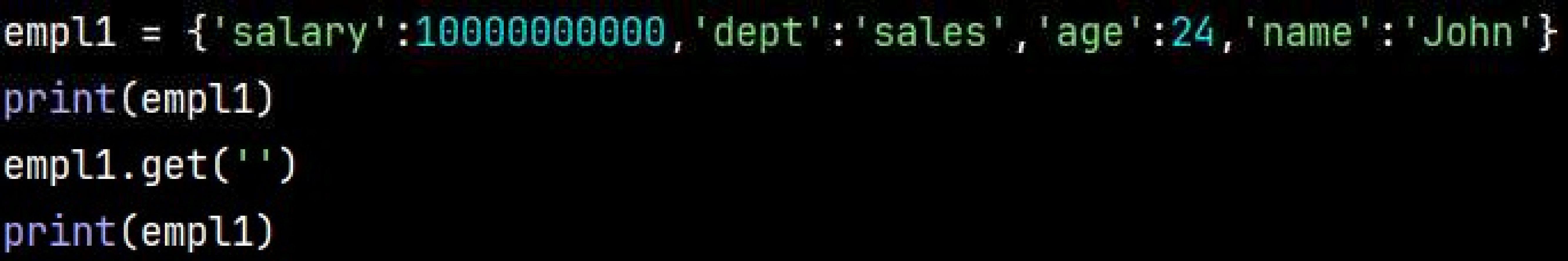
  <Dictionary>.get(key,[default])

- E g.,



```
>>> empl1
{'salary' : 10000, 'dept' : 'Sales', 'age': 24, 'name' : 'John'}
>>> empl1.get('dept')
'Sales'
```

075 THE GET METHOD



```
empl1 = {'salary':10000000000,'dept':'sales','age':24,'name':'John'}
print(empl1)
empl1.get('')
print(empl1)
```

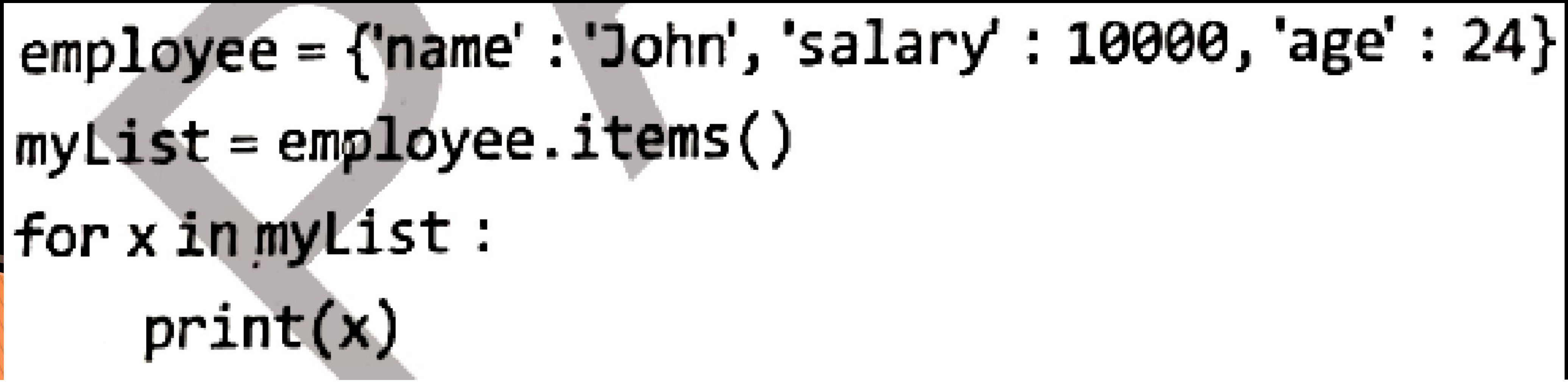
Dictionary Functions And Methods

4. The Items ( ) Method

- This method returns all of the items in the dictionary as a sequence of (key, list) tuples. Note that this are returned in low particular order.

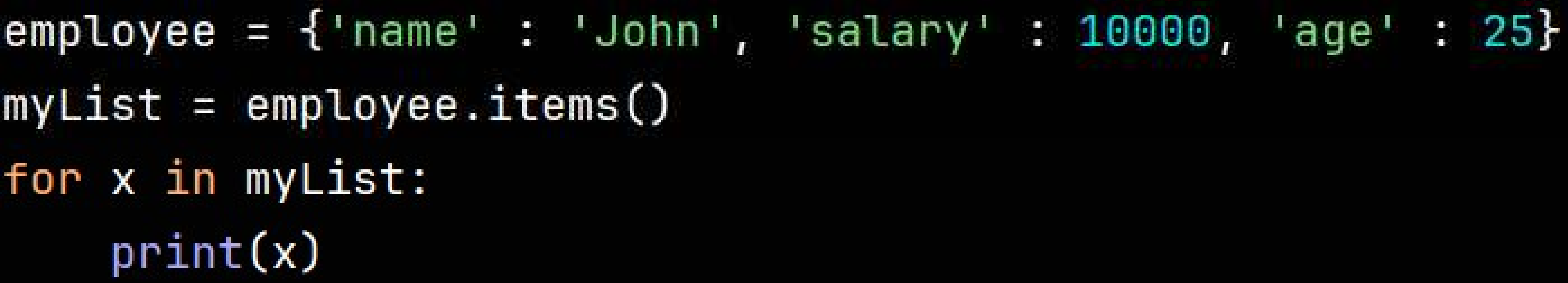
  <Dictionary>.items( )

- E g.,



```
employee = {'name' : 'John', 'salary' : 10000, 'age' : 24}
myList = employee.items()
for x in myList :
    print(x)
```

076 THE ITEMS METHOD



```python
employee = {'name' : 'John', 'salary' : 10000, 'age' : 25}
myList = employee.items()
for x in myList:
    print(x)
```

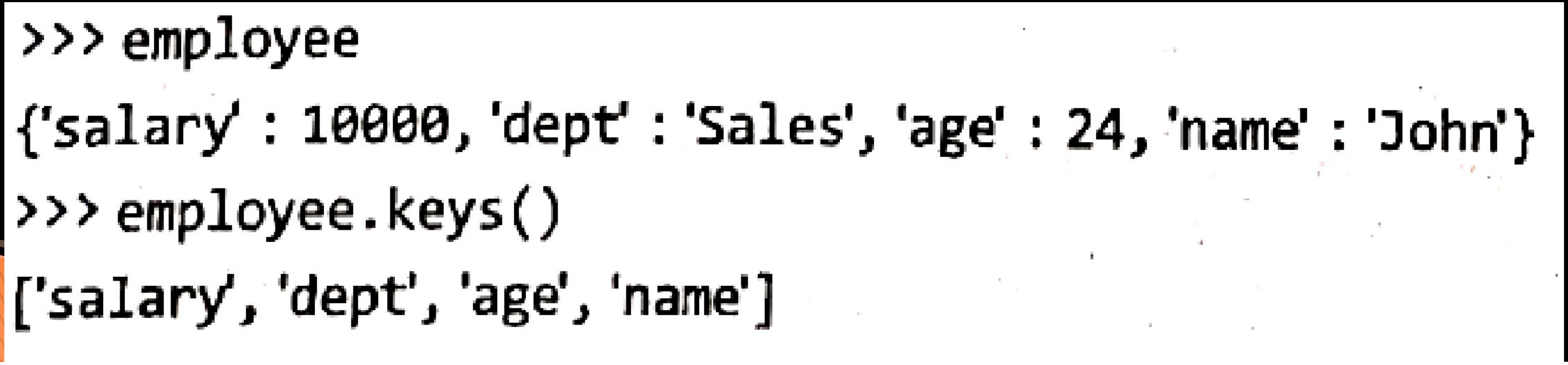
Dictionary Functions And Methods

5. The Keys ( ) Method

- This method returns all of the keys in the dictionary as a sequence of key (inform of a list).
- Note that this are returned in low particular order.

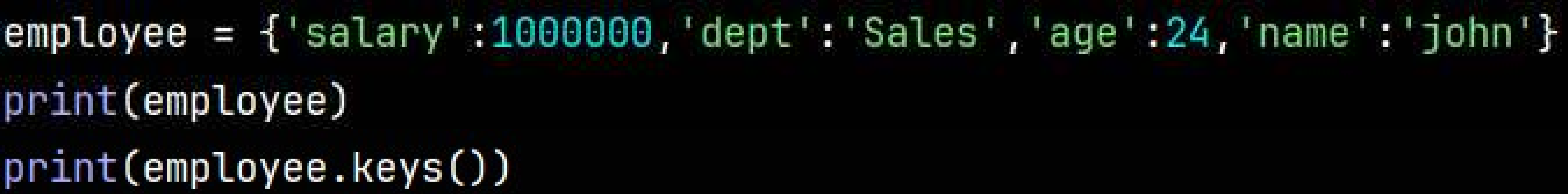
<Dictionary>.keys( )

- E g.,



```
>>> employee
{'salary' : 10000, 'dept' : 'Sales', 'age' : 24, 'name' : 'John'}
>>> employee.keys()
['salary', 'dept', 'age', 'name']
```

077 THE KEYS METHOD



```
employee = {'salary':1000000,'dept':'Sales','age':24,'name':'john'}
print(employee)
print(employee.keys())
```

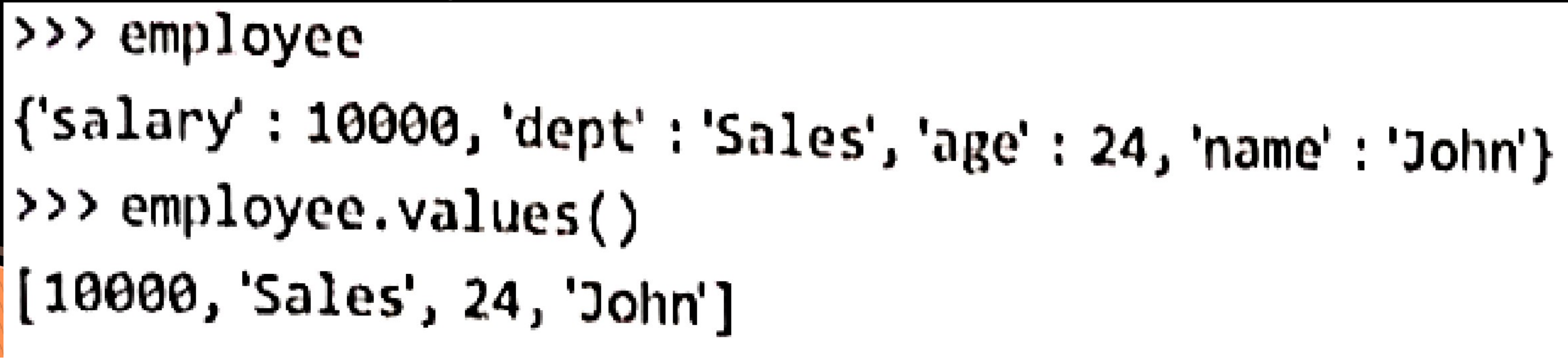
Dictionary Functions And Methods

6. The Values ( ) Method

- This method returns all the values from the dictionary as a sequence (all list).
- Note that these are returned in no particular order.

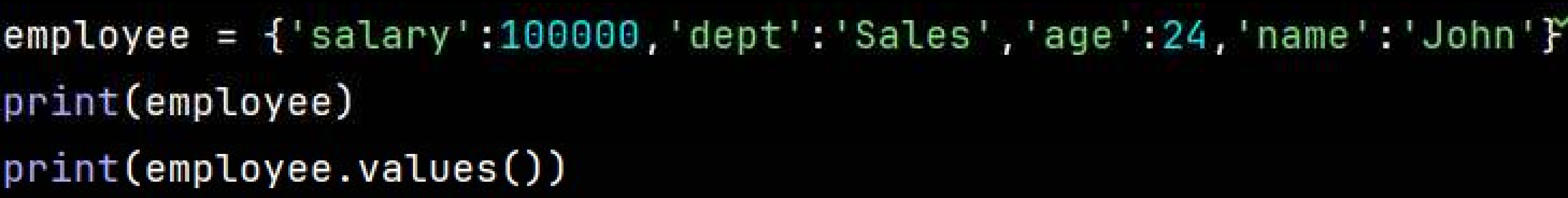
<Dictionary>.values( )

- E.g,



```
>>> employee
{'salary' : 10000, 'dept' : 'Sales', 'age' : 24, 'name' : 'John'}
>>> employee.values()
[10000, 'Sales', 24, 'John']
```

078 THE VALUES METHOD



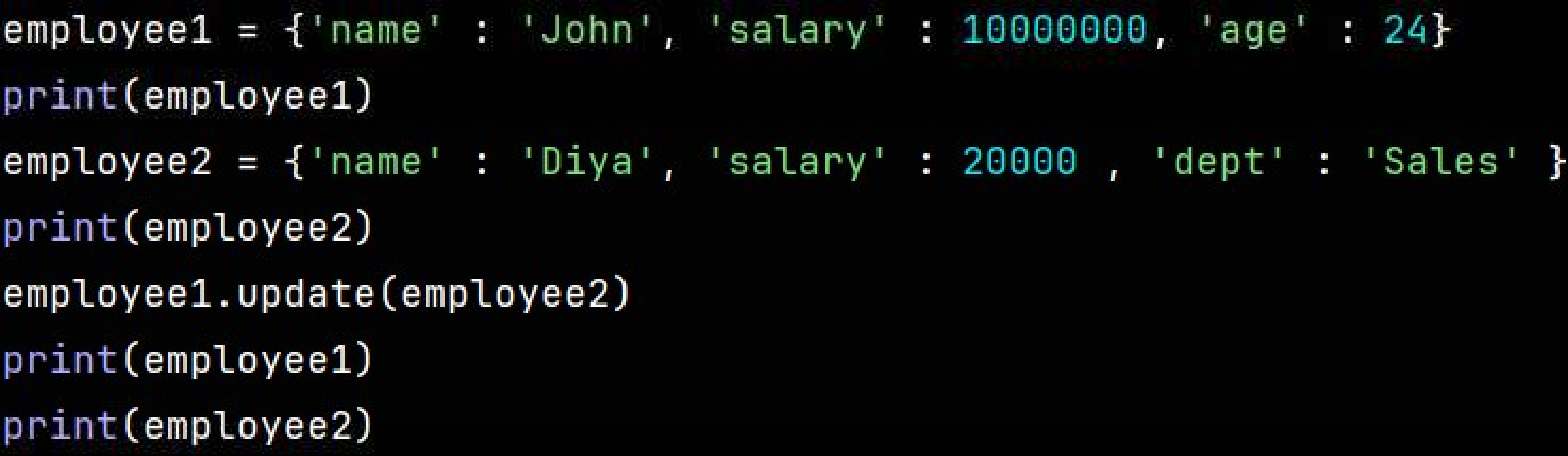
```
employee = {'salary':100000,'dept':'Sales','age':24,'name':'John'}
print(employee)
print(employee.values())
```

Dictionary Functions And Methods

7. The update ( ) Method

- This method merges key value pairs from the new dictionary into the original dictionary, adding or replacing as needed.
- The items in the new dictionary are added to the old one and override any items already there with the same keys.

079 THE UPDATE METHOD

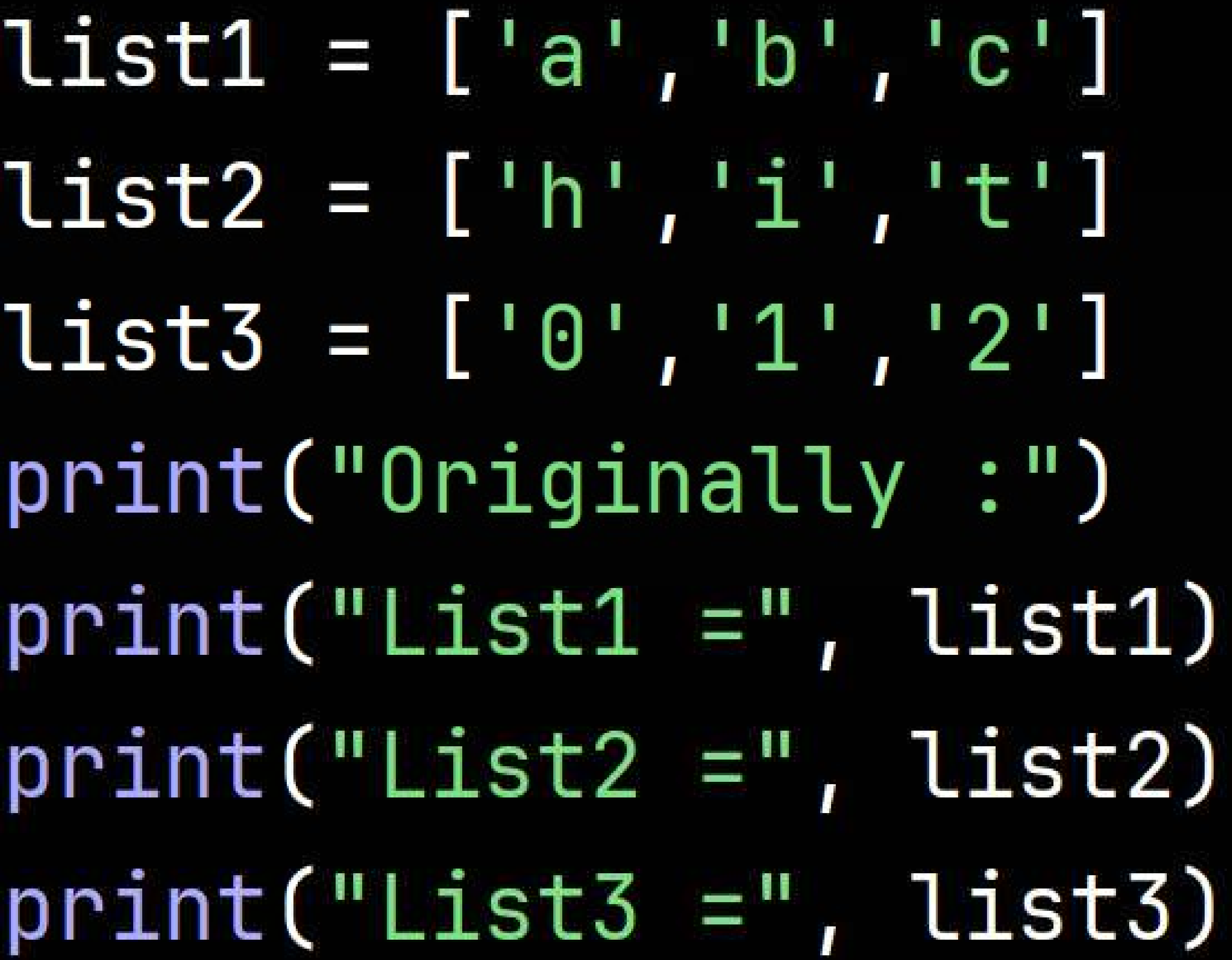


```
employee1 = {'name' : 'John', 'salary' : 10000000, 'age' : 24}
print(employee1)
employee2 = {'name' : 'Diya', 'salary' : 20000 , 'dept' : 'Sales' }
print(employee2)
employee1.update(employee2)
print(employee1)
print(employee2)
```

Dictionary Functions And Methods

- Given three list as list1=['a','b','c'], list^2=[h i t] and list3=[012].
- Write a program that its list two and three two list one as single element each.
- The resultant list should be in the order of list three elements of list one and list two

080 DICTIONARY & LISTS

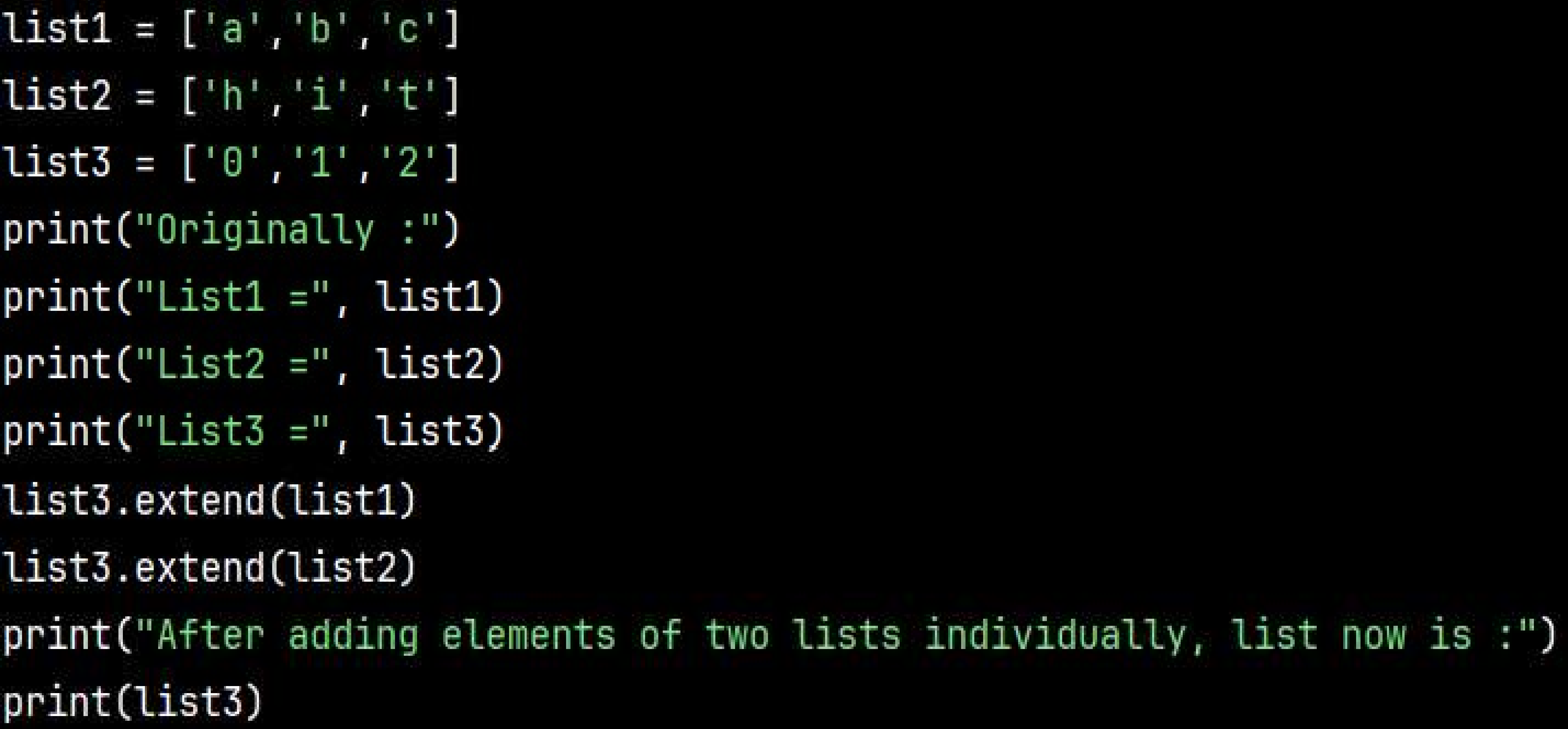


```python
list1 = ['a','b','c']
list2 = ['h','i','t']
list3 = ['0','1','2']
print("Originally :")
print("List1 =", list1)
print("List2 =", list2)
print("List3 =", list3)
```

Dictionary Functions And Methods

- Given three list as list1=['a','b','c'], list²=[h i t] and list3=[012].
- Write a program that individual element list two and three two list one as single element each.
- The resultant list should be in the order of list three elements of list one and list two

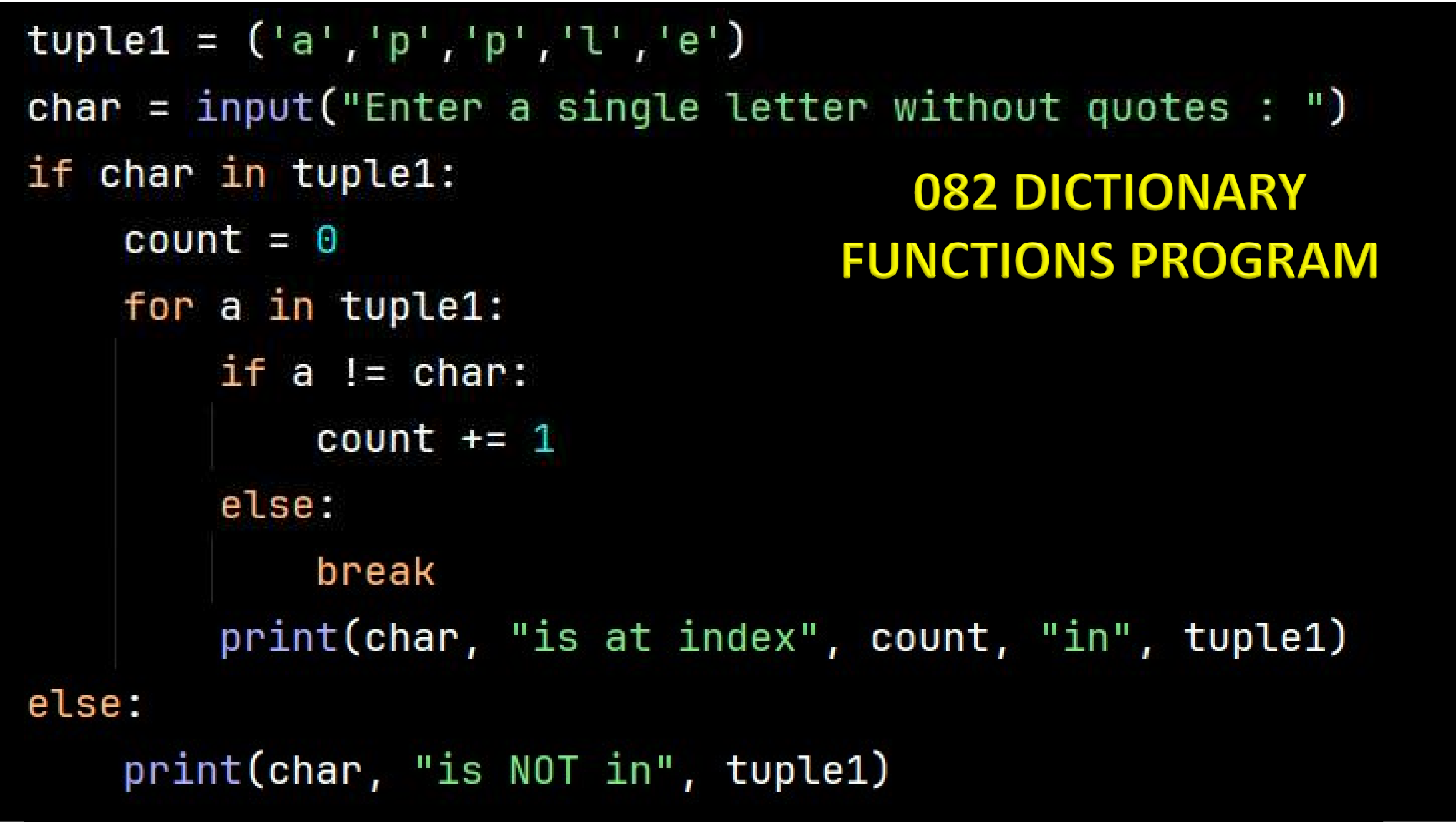
081 DICTIONARY & LISTS



```python
list1 = ['a','b','c']
list2 = ['h','i','t']
list3 = ['0','1','2']
print("Originally :")
print("List1 =", list1)
print("List2 =", list2)
print("List3 =", list3)
list3.extend(list1)
list3.extend(list2)
print("After adding elements of two lists individually, list now is :")
print(list3)
```

Dictionary Functions And Methods

- Write the program that finds an elements index/position in a couple without using index ()

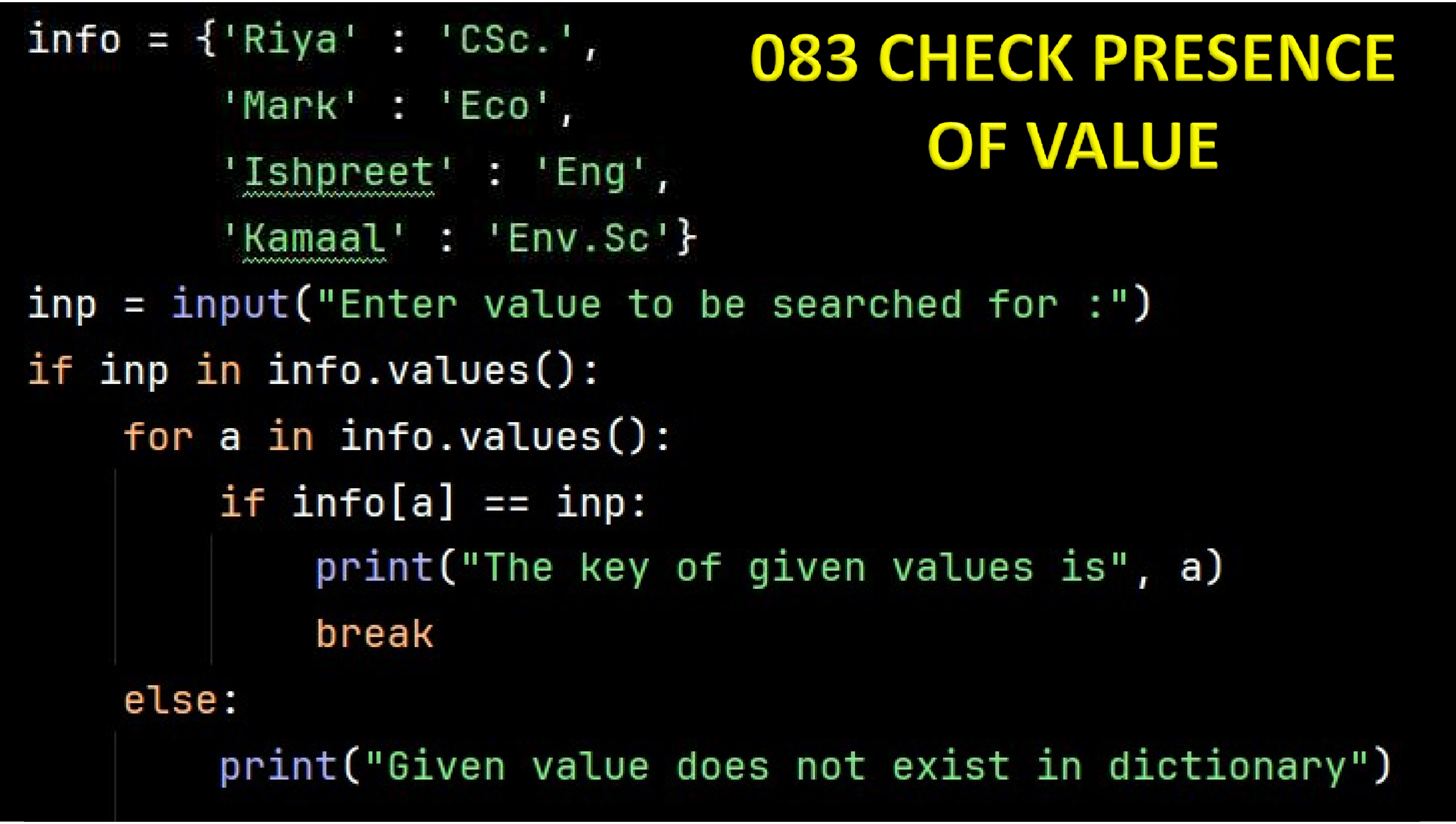


082 DICTIONARY FUNCTIONS PROGRAM

```python
tuple1 = ('a','p','p','l','e')
char = input("Enter a single letter without quotes : ")
if char in tuple1:
    count = 0
    for a in tuple1:
        if a != char:
            count += 1
        else:
            break
        print(char, "is at index", count, "in", tuple1)
else:
    print(char, "is NOT in", tuple1)
```

Dictionary Functions And Methods

- Write a program that checks for presence of a value inside a dictionary and print its key.

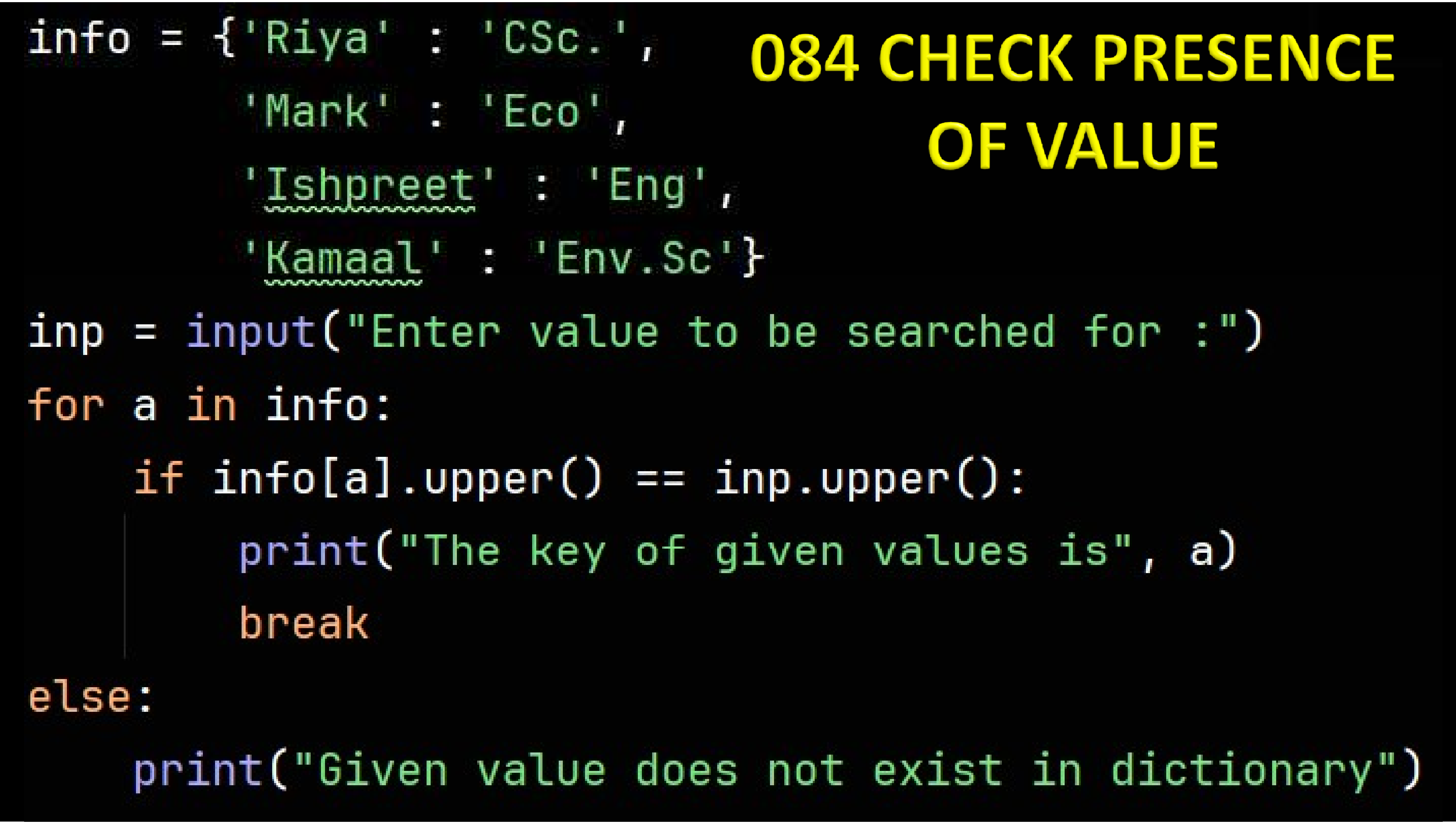


083 CHECK PRESENCE OF VALUE

```python
info = {'Riya' : 'CSc.',
        'Mark' : 'Eco',
        'Ishpreet' : 'Eng',
        'Kamaal' : 'Env.Sc'}
inp = input("Enter value to be searched for :")
if inp in info.values():
    for a in info.values():
        if info[a] == inp:
            print("The key of given values is", a)
            break
else:
    print("Given value does not exist in dictionary")
```

Dictionary Functions And Methods

- The code of previous will not work if the case of the given value and value inside dictionary are different.



084 CHECK PRESENCE OF VALUE

```
info = {'Riya' : 'CSc.',
        'Mark' : 'Eco',
        'Ishpreet' : 'Eng',
        'Kamaal' : 'Env.Sc'}
inp = input("Enter value to be searched for :")
for a in info:
    if info[a].upper() == inp.upper():
        print("The key of given values is", a)
        break
else:
    print("Given value does not exist in dictionary")
```