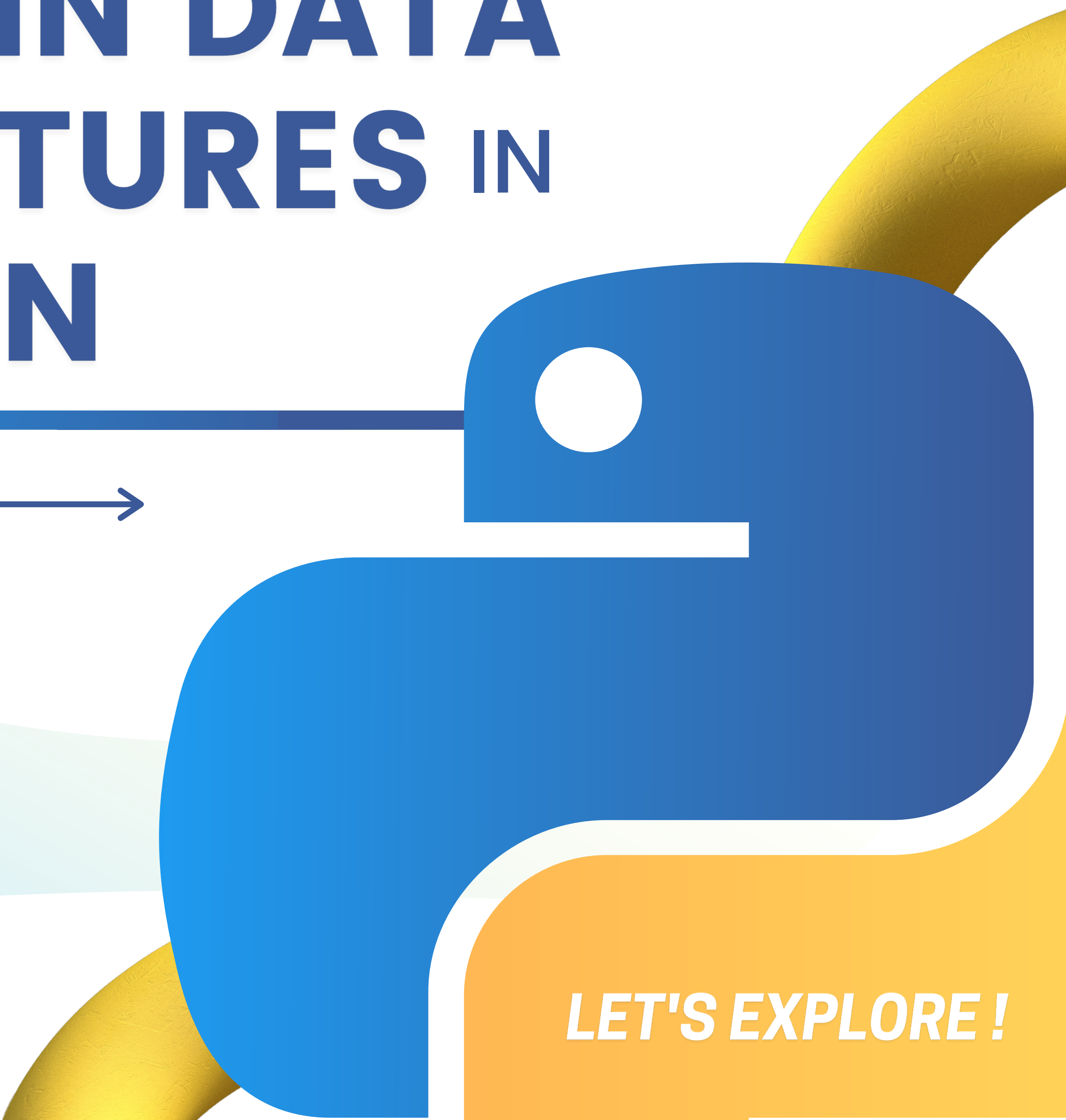
LET THE DATA CONFESS
Understand | Learn | Code | Implement

Are you aware of

BUILT-IN DATA STRUCTURES IN PYTHON

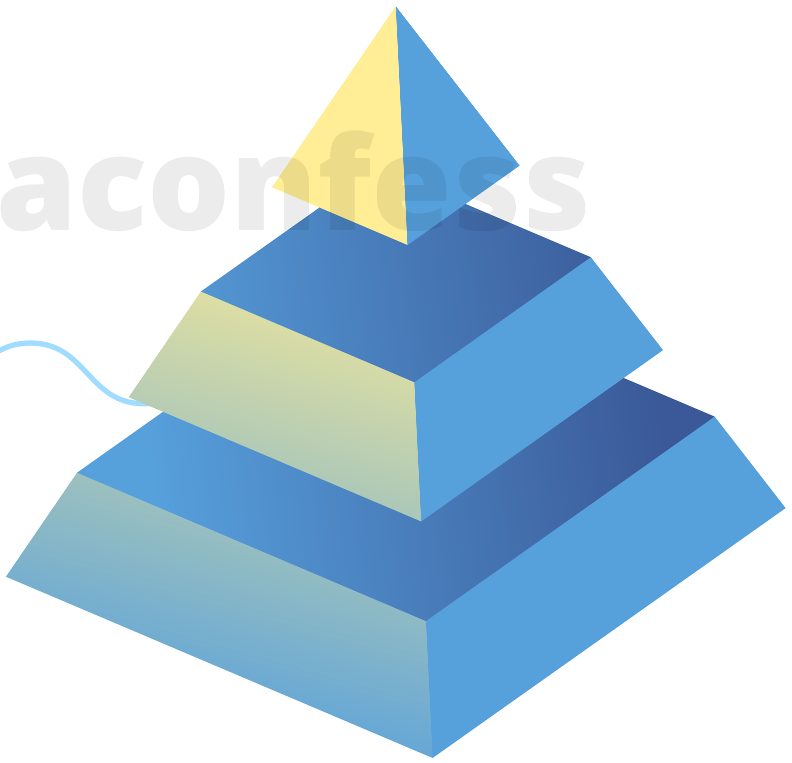
-letthedataconfess



www.letthedataconfess.com

What is meant by Data structure ?

- The data structure is the way of organising the data in such a way that it can be retrieved quickly.
- But one data structure is not sufficient to handle all the use case scenarios.
- That is why we have multiple data structures which can be used for various use cases.



Why do we need **Data structure** ?

- Consider the following scenario, where you want to search for a particular document in file explorer with over 1000 documents stored. One way to do that is by going one by one in a linear way, which is time-consuming.

- Another way is to jump directly go to that place where it is stored or where the related documents are present.

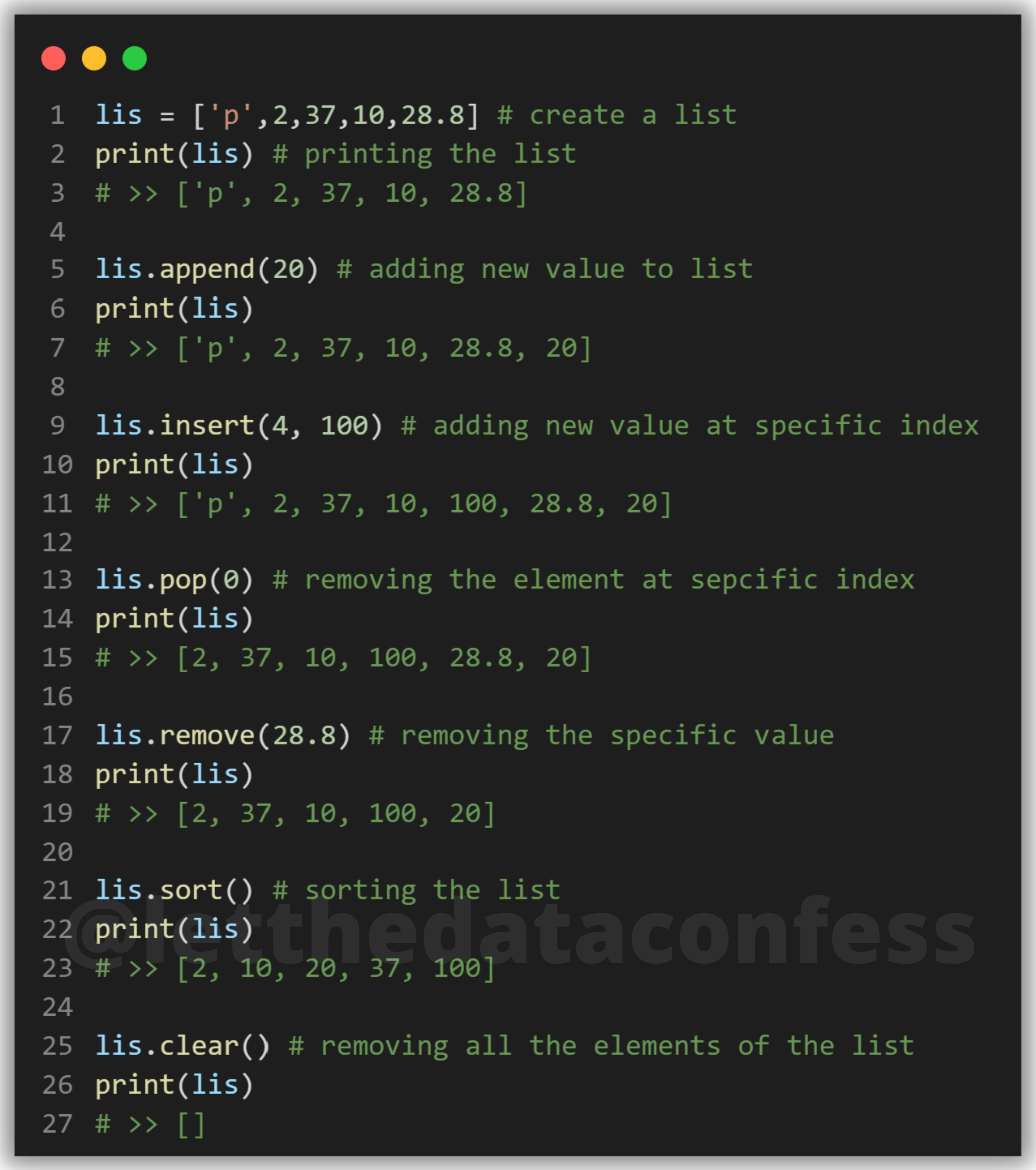
- Yes your OS does this, using indexing and hashtables which is a type of data structure. This reduces the time required to search even if there lots of file present. This is why data structures are important.

Python data structures

01. List

- The list is a linear data structure where the data is present sequentially.
- It's a heterogeneous collection of data which means it can store items of different data types.
- The list comes with multiple methods to perform operations on them.

LET THE DATA CONFESS
Understand | Learn | Code | Implement

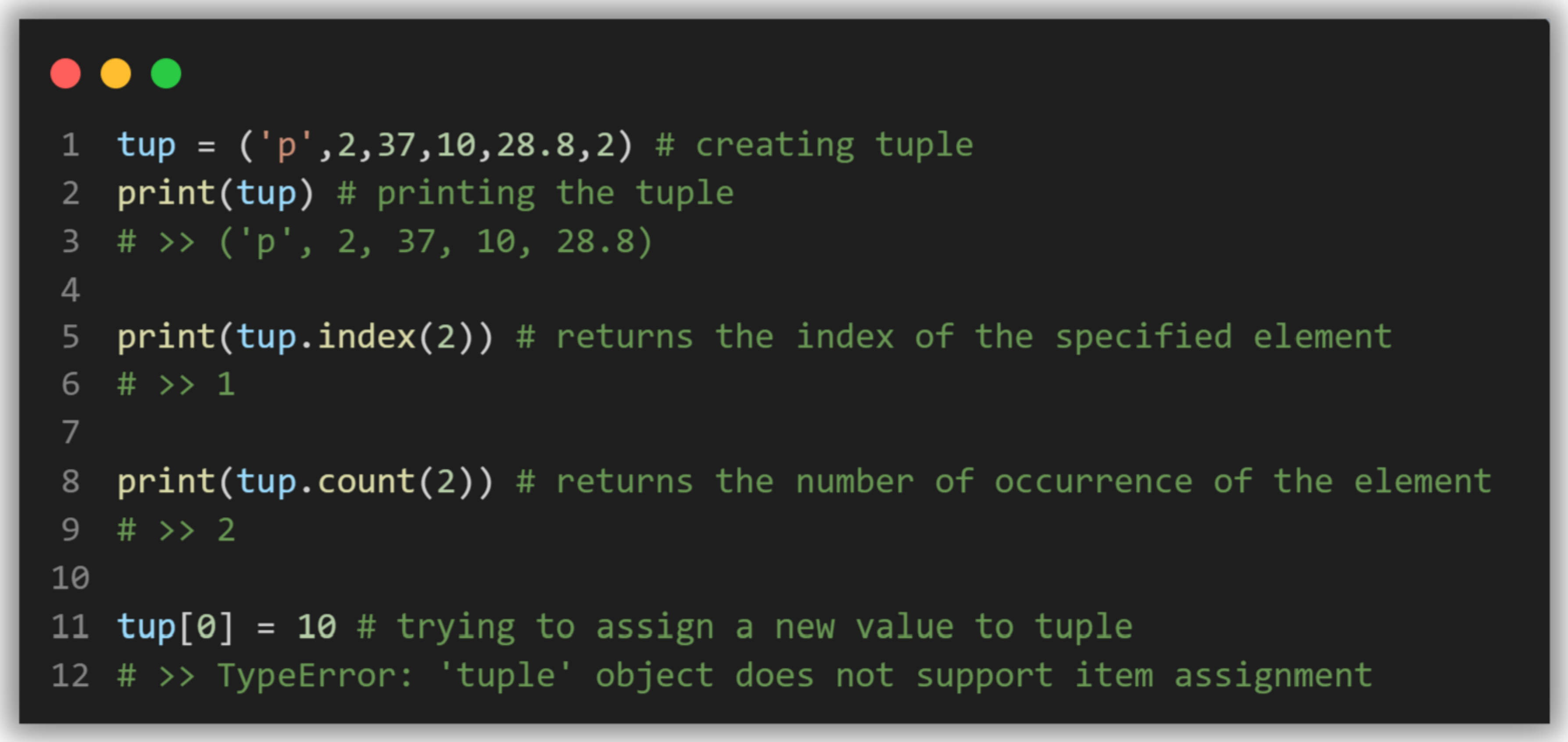


```python
lis = ['p',2,37,10,28.8] # create a list
print(lis) # printing the list
# >> ['p', 2, 37, 10, 28.8]

lis.append(20) # adding new value to list
print(lis)
# >> ['p', 2, 37, 10, 28.8, 20]

lis.insert(4, 100) # adding new value at specific index
print(lis)
# >> ['p', 2, 37, 10, 100, 28.8, 20]

lis.pop(0) # removing the element at sepcific index
print(lis)
# >> [2, 37, 10, 100, 28.8, 20]

lis.remove(28.8) # removing the specific value
print(lis)
# >> [2, 37, 10, 100, 20]

lis.sort() # sorting the list
print(lis)
# >> [2, 10, 20, 37, 100]

lis.clear() # removing all the elements of the list
print(lis)
# >> []
```

02. Tuples

- A tuple is another data structure that is almost similar to a list except that they are immutable, which means once created then, tuple elements cannot be manipulated means you cannot add or remove elements to it.

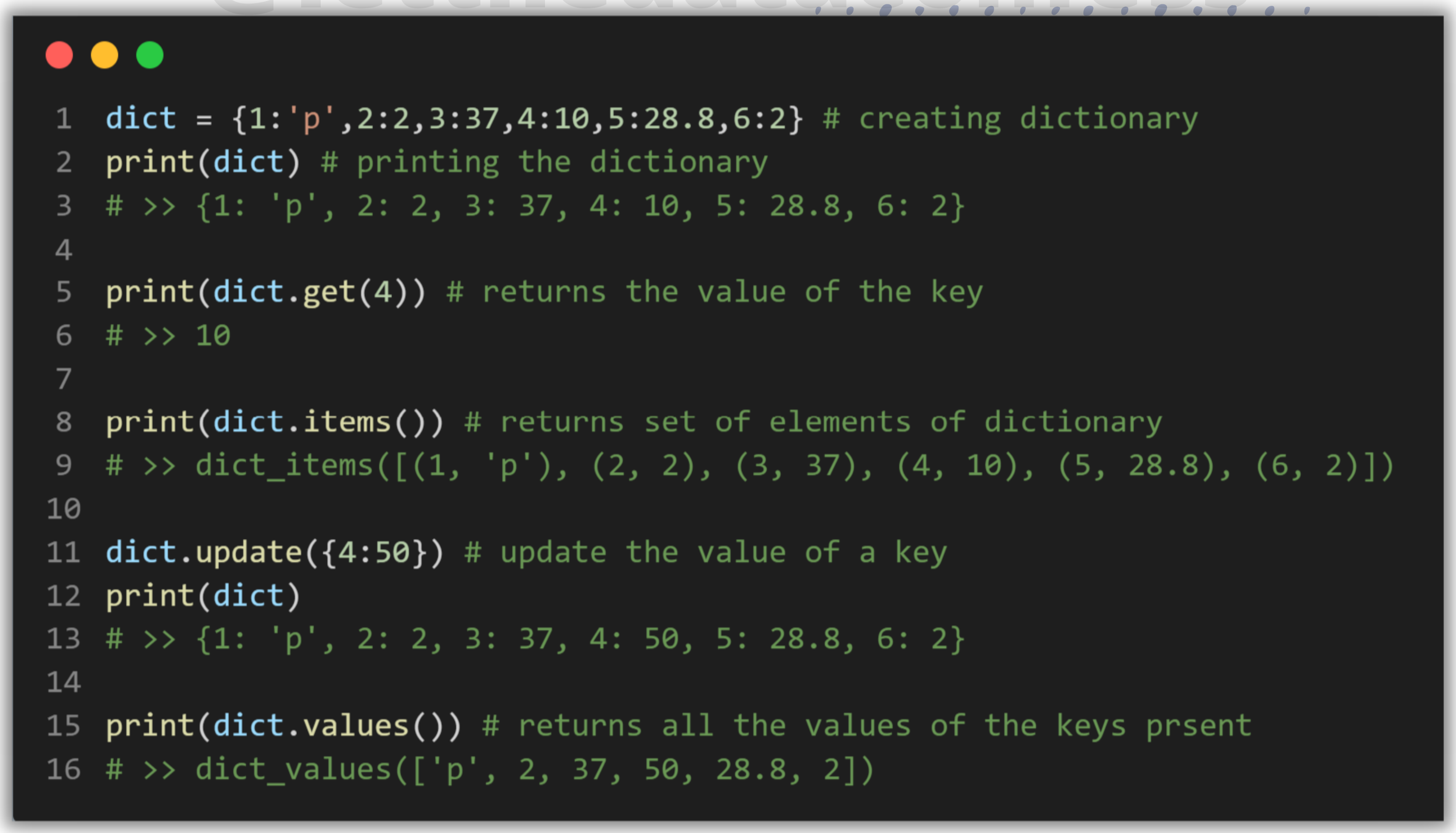
- But you can get the counts and indexes of the elements present.



```python
tup = ('p',2,37,10,28.8,2) # creating tuple
print(tup) # printing the tuple
# >> ('p', 2, 37, 10, 28.8)

print(tup.index(2)) # returns the index of the specified element
# >> 1

print(tup.count(2)) # returns the number of occurrence of the element
# >> 2

tup[0] = 10 # trying to assign a new value to tuple
# >> TypeError: 'tuple' object does not support item assignment
```

03. Dictionary

- Dictionary is a very important data structures that store the data in key-value pairs similar to the hash table.
- Dictionary is an unordered collection of data that are mutable means, you can add or update the existing values for a given key.

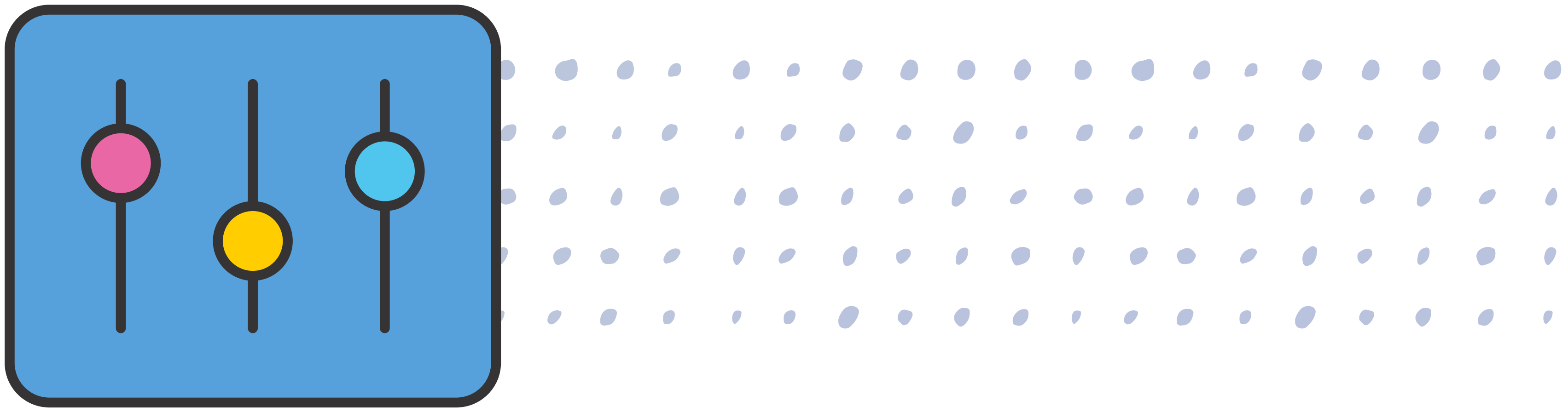


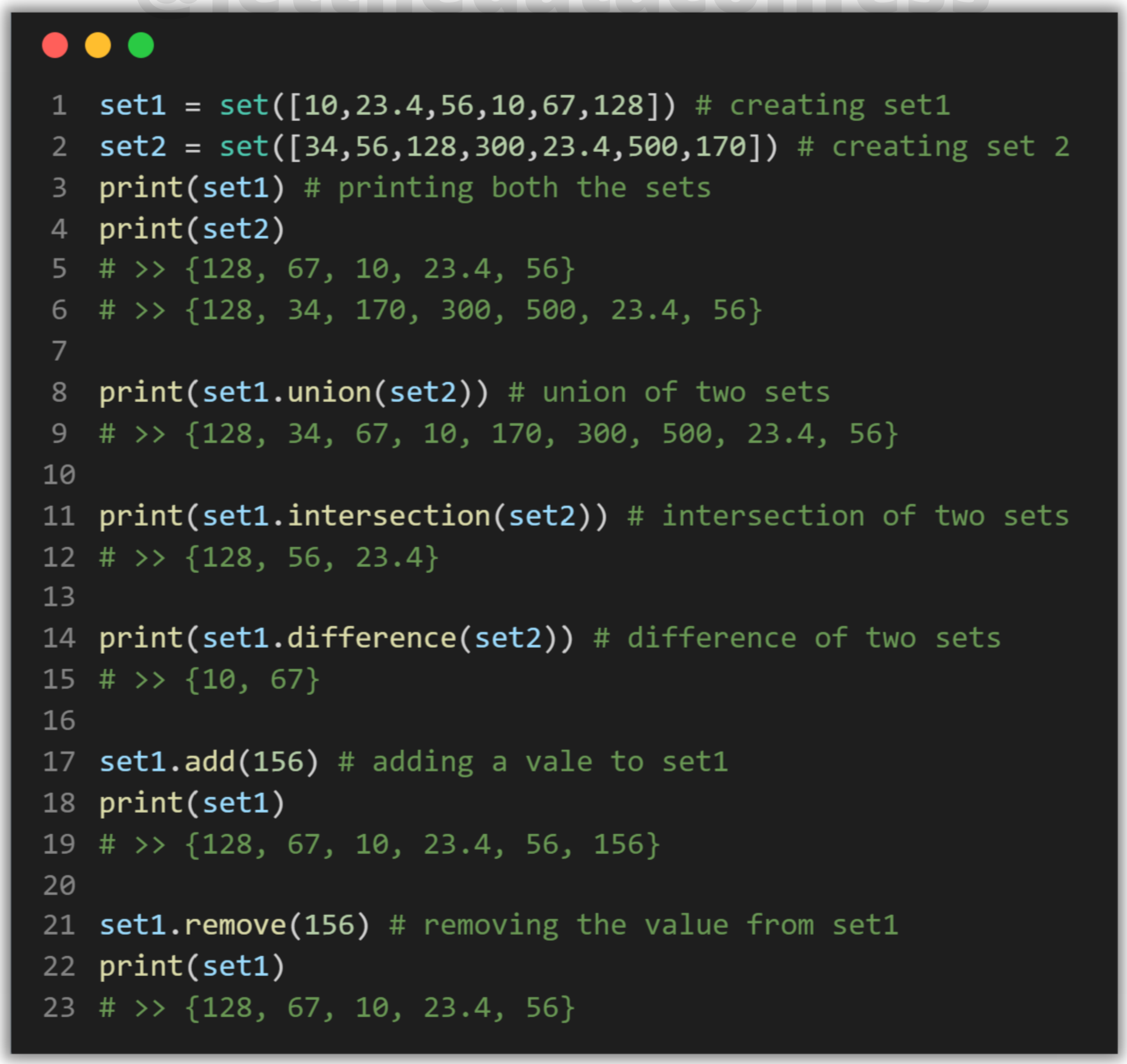
```python
dict = {1:'p',2:2,3:37,4:10,5:28.8,6:2} # creating dictionary
print(dict) # printing the dictionary
# >> {1: 'p', 2: 2, 3: 37, 4: 10, 5: 28.8, 6: 2}

print(dict.get(4)) # returns the value of the key
# >> 10

print(dict.items()) # returns set of elements of dictionary
# >> dict_items([(1, 'p'), (2, 2), (3, 37), (4, 10), (5, 28.8), (6, 2)])

dict.update({4:50}) # update the value of a key
print(dict)
# >> {1: 'p', 2: 2, 3: 37, 4: 50, 5: 28.8, 6: 2}

print(dict.values()) # returns all the values of the keys prsent
# >> dict_values(['p', 2, 37, 50, 28.8, 2])
```

04. Sets

- This is another type of data structure on which we can perform set operation like union, intersection, difference, etc.

- Set doesn't allow duplicate entries. Each value in a set is unique. It's also an unordered collection.





```python
set1 = set([10,23.4,56,10,67,128]) # creating set1
set2 = set([34,56,128,300,23.4,500,170]) # creating set 2
print(set1) # printing both the sets
print(set2)
# >> {128, 67, 10, 23.4, 56}
# >> {128, 34, 170, 300, 500, 23.4, 56}

print(set1.union(set2)) # union of two sets
# >> {128, 34, 67, 10, 170, 300, 500, 23.4, 56}

print(set1.intersection(set2)) # intersection of two sets
# >> {128, 56, 23.4}

print(set1.difference(set2)) # difference of two sets
# >> {10, 67}

set1.add(156) # adding a vale to set1
print(set1)
# >> {128, 67, 10, 23.4, 56, 156}

set1.remove(156) # removing the value from set1
print(set1)
# >> {128, 67, 10, 23.4, 56}
```

LET THE DATA CONFESS
Understand | Learn | Code | Implement

www.letthedataconfess.com

LET THE DATA CONFESS
Understand | Learn | Code | Implement

Follow | Support