DICTONARY& SET

Dictionaries

- ❖ Dictionaries are the **most flexible** built-in data type in Python
- ❖ The chief distinction is that in dictionaries, items are stored and fetched by key, instead of by positional offset

List $\longrightarrow$ Ordered collection of Objects

Dictionaries $\longrightarrow$ Un-ordered collection of Objects

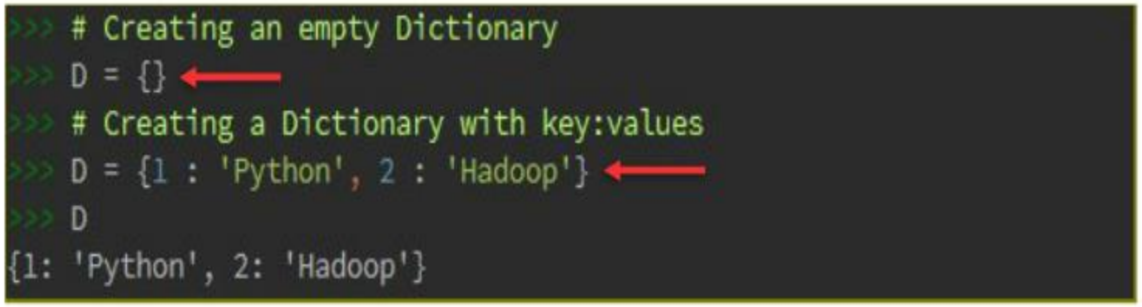
Dictionaries

- ✓ Accessed by key, not offset (index)
- ✓ Unordered collections of arbitrary object
- ✓ Variable-length, heterogeneous, and arbitrarily nestable.
- ✓ Mutable mapping
- ✓ Stored as Tables of object references (hash table)

Dictionaries - Creation

- ✓ Dictionary is written as a series of **key: value** pairs, separated by commas, enclosed in curly braces {} or using **dict()** function
- ✓ An empty dictionary is an empty set of braces, and dictionaries can be nested by writing one as a value inside another dictionary, or within a list or tuple

Example

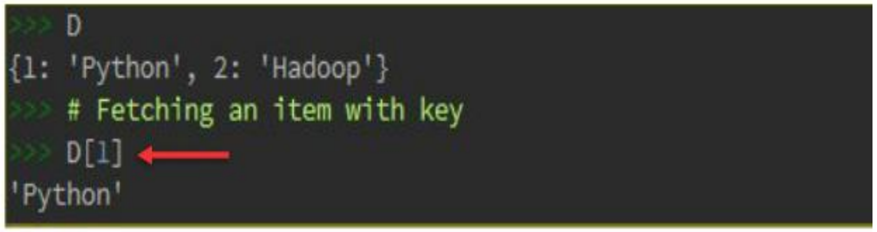


```
>>> # Creating an empty Dictionary
>>> D = {}
>>> # Creating a Dictionary with key:values
>>> D = {1 : 'Python', 2 : 'Hadoop'}
>>> D
{1: 'Python', 2: 'Hadoop'}
```

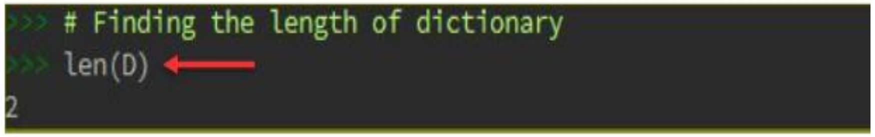
Dictionaries – Access and Methods

- ✓ Dictionary items can be fetched by passing KEY => D['key']
- ✓ Functions like len() and **in** works on dict also.
- ✓ dict.keys() lists all the keys in dictionary
- ✓ Operations like adding, deleting, changing works on dict

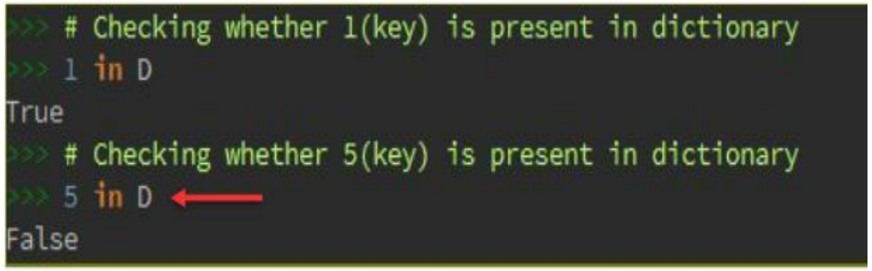
❖ Note : Slicing and concatenation on dict doesn't works because we can fetch values only by key, **not by position**.



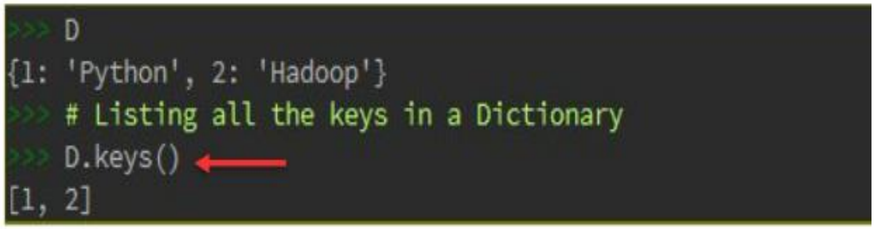
```
>>> D
{1: 'Python', 2: 'Hadoop'}
>>> # Fetching an item with key
>>> D[1]
'Python'
```



```
>>> # Finding the length of dictionary
>>> len(D)
2
```



```
>>> # Checking whether 1(key) is present in dictionary
>>> 1 in D
True
>>> # Checking whether 5(key) is present in dictionary
>>> 5 in D
False
```



```
>>> D
{1: 'Python', 2: 'Hadoop'}
>>> # Listing all the keys in a Dictionary
>>> D.keys()
[1, 2]
```

Dictionaries – Methods

✓ fromkeys()	Create a new dictionary with keys from seq and values set to value.
✓ get(key)	For key key, returns value or default if key not in dictionary
✓ pop(key)	Remove specified key and return the corresponding value. If key is not found, d is returned if given, otherwise KeyError is raised
✓ popitem()	Returns and removes an arbitrary element (key, value) pair from the dictionary.
✓ clear()	Removes all elements of dictionary
✓ update(dict2)	Adds dictionary dict2's key-values pairs to dictionary

Dictionary contains below iterator functions which handles iteration on dictionary items:

✓ items()	Returns a list of dict's (key, value) tuple pairs
✓ keys()	Returns list of dictionary dict's keys
✓ values()	Returns list of dictionary dict's values

Dictionaries – Comprehension

Dictionary Comprehension:

```
squares = {i: i*i for i in range(1,11)}
print(squares)
```

Sets

- A set is an unordered collection of items. Every element is unique (no duplicates) and must be immutable (which cannot be changed).
- However, the set itself is mutable. We can add or remove items from it.
- Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.

Sets - Creating

Curly braces or the set() function can be used to create sets.

Sets creation using set()

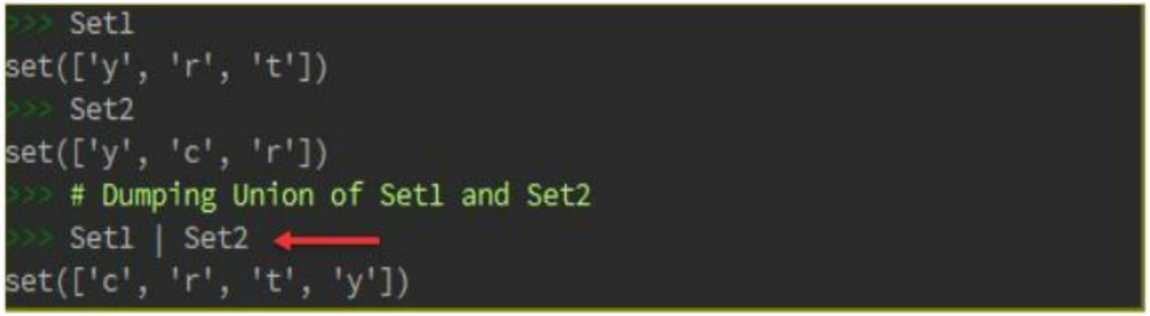
```
>>> set1=set(("Python","Java","Hadoop“))
>>> set1
    {'Hadoop', 'Python', 'Java'}
```

Sets creation using { }

```
>>> set2={"Python","Java","Hadoop"}
>>> set2
{'Hadoop', 'Python', 'Java'}
```

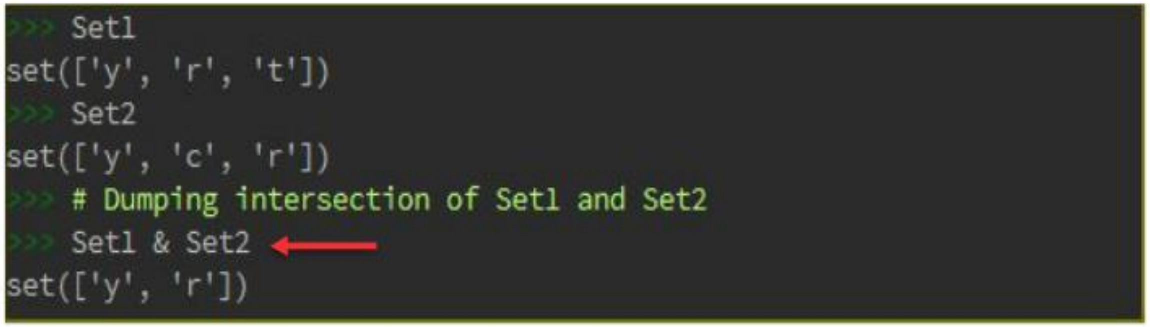
Sets Operations

→Union



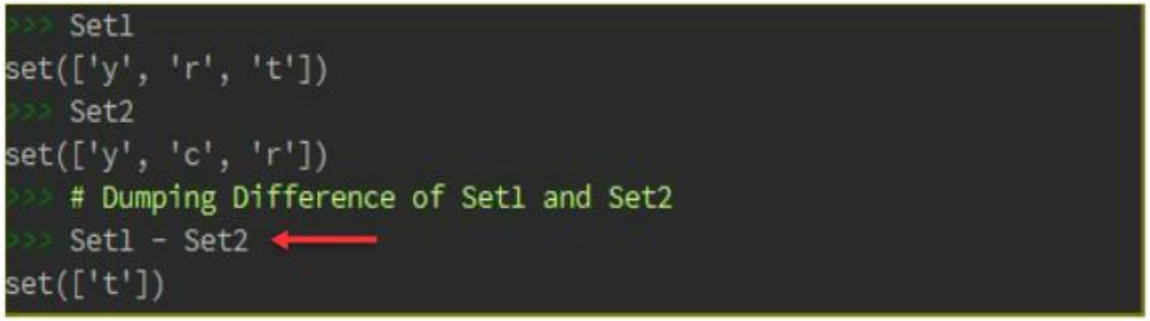
```
>>> Set1
set(['y', 'r', 't'])
>>> Set2
set(['y', 'c', 'r'])
>>> # Dumping Union of Set1 and Set2
>>> Set1 | Set2
set(['c', 'r', 't', 'y'])
```

→Intersection



```
>>> Set1
set(['y', 'r', 't'])
>>> Set2
set(['y', 'c', 'r'])
>>> # Dumping intersection of Set1 and Set2
>>> Set1 & Set2
set(['y', 'r'])
```

→Difference



```
>>> Set1
set(['y', 'r', 't'])
>>> Set2
set(['y', 'c', 'r'])
>>> # Dumping Difference of Set1 and Set2
>>> Set1 - Set2
set(['t'])
```

Methods

✓ add()	Add an element to a set
✓ clear()	Remove all elements form a set
✓ difference()	Return the difference of two or more sets as a new set
✓ symmetric_difference()	Return the symmetric difference of two sets as a new set
✓ discard()	Remove an element from set if it is a member.
✓ intersection()	Return the intersection of two sets as a new set
✓ isdisjoint()	Return True if two sets have a null intersection
✓ issubset()	Return True if another set contains this set
✓ issuperset()	Return True if this set contains another set
✓ pop()	Remove and return an arbitary set element. Raise KeyError if the set is empty
✓ remove()	Remove an element from a set. If the element is not a member, raise a KeyError
✓ union()	Return the union of sets in a new set
✓ update()	Update a set with the union of itself and others

THANK YOU!!