Dictionary

- Collection of key-value pairs.
- Separated by comma operator
- A Boundary: {...}
- Each element is associated with unique key.
- Syntax : var_name = {key1: value1, key2: value2,...}
- Length: returns the number of keys present in the dictionary.

  len(dict_variable)
- It does not support indexing/slicing.

Different ways of constructing a dictionary:-

- d = {} # empty dictionary
- d = dict()
- Using dictionary constructor
- d = dict(Bangalore=25, goa=35, punjab=30)
- d = dict([("Bangalore", 25), (“goa", 35), (“punjab", 30)])
- d = dict({'Bangalore': 25, “goa": 35, “punjab": 30})

Characteristics of dictionaries:-

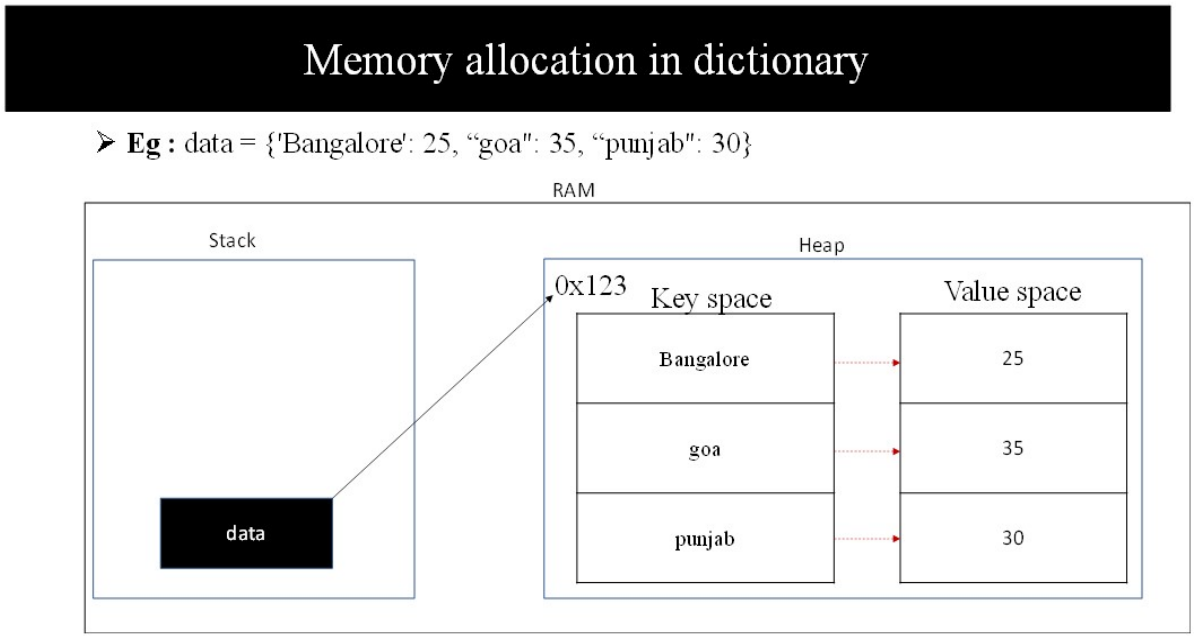
- Key cannot be duplicated.
- Keys will be a single element
- Values can be of any datatype.
- Values can be accessed through keys only.
- Keys must be of immutable data type.

Composite keys:-

- Dictionaries can have composite keys i.e, tuples as keys.
- Eg:

- holidays = {(26, 1): 'Republic Day’,(15, 8): 'Independence Day’,(21, 6): 'Yoga Day’}
- # Nested Dictionary
- prices = {'IBM': {'current': 90.1, 'low': 88.3, 'high': 92.7}, 'HP': {"current": 29.70, "low": 28.30, "high": 31.2} }
- # list inside the dictionary as values.
- location = {'country': 'India', 'states': ['Karnataka', 'Andra', 'Kerala']}
- # points
- points = {'a': 1, 'b': 2, 'c': 3}

Memory allocation in dictionary



Accessing and update values from a dictionary:- using dictionary loopup

- Eg:
- data = {'Bangalore': 25, “goa": 35, “punjab": 30}

- data['Bangalore’] >>> 25
- data.get('Bangalore’) >>> 25
- # Accessing a key that does not exist
- data[“Noida”] >>> KeyError

- # adding (updating)using key and value syntax.
- data[‘delhi'] = 26.5 # Updating the dictionary key with new value
- data
- {'Bangalore': 25, “goa": 35, “punjab": 30,‘delhi';30}

- data["goa"] = 22 #modify the old value by new value if key already existed.
- data
- {'Bangalore': 25, “goa": 22, “punjab": 30,‘delhi';30}

Methods on dictionary:-

>>> dir(dict)
>>>['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ior__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__ror__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']

- *Accessing values from a dictionary:-*

get(): returns the value of the item with the specified key. If keyname and value is not present, then it will give the second parameter[value] as value.
Eg:
data = {'Bangalore': 25, “goa": 35, “punjab": 30}
data['Bangalore’] >>> 25
data.get('Bangalore’) >>> 25

Accessing a key that does not exist
data[“Noida”] >>> KeyError
print(data.get('Noida’)) >>> None
# get() method does not throw an error, but returns 'None' by default if the second argument is not given.
print(data.get('Noida', 'The Key is not found in the dictionary’))
>>> "The Key is not found in the dictionary"

- *Adding / Updating the dictionary:-*
- data = {'Bangalore': 25, “goa": 35, “punjab": 30}
  # using key and value syntax.
  data[‘delhi'] = 26.5 # Updating the dictionary key with new value
  update(): updates the existing dictionary with the specified items.
  data.update({“delhi": 26, “chennai":28})
  data.update(bdvt=26)
  >>> data = {'Bangalore': 25, “goa": 35, “punjab": 30,“delhi": 26, “chennai":28,'bdvt':26}
  setdefault(keyname, [value]): returns the value of the item with the specified key.
  If the value is not specified then it adds None as the value else the value given will be added

If the key does not exist, it will insert the key, with the specified value.
Eg:
data = {'Bangalore': 25, “goa": 35, “punjab": 30}
data.setdefault("Mysore") >>> None
data >>> {'Bangalore': 25, “goa": 35, “punjab": 30, "Mysore": None}
data.setdefault("Kolkata", 60) >>> 60
data >>> {'Bangalore': 25, “goa": 35, “punjab": 30, "Kolkata": 60}

- *Updating an existing key*
  data = {'Bangalore': 25, “goa": 35, "Mysore": 25}
  # using key and value syntax.
  data['Mysore'] = 26.5 >>> {'Bangalore': 25, “goa": 35, "Mysore": 26.5}
  # update()
  data.update({"Mysore": 26, “Mumbai":28}) >>> {'Bangalore': 25, “goa": 35, "Mysore": 26, “Mumbai":28}
  data.update(Bangalore=26) >>> {'Bangalore': 26, “goa": 35, "Mysore": 25}

fromkeys():-

fromkeys(): returns a dictionary with the specified keys and the specified value.
Syntax : dict.fromkeys(keys, [value])
Keys Required. An iterable specifying the keys of the new dictionary
Value Optional. The value for all keys. Default value is None
Eg:

```
names = ['apple', 'google', 'yahoo', 'gmail', 'google',
'apple']
_count = dict.fromkeys(names, 0)
_count      >>>      {'apple': 0, 'google': 0, 'yahoo': 0,
'gmail': 0}
```

items(), keys(), values():-

- items(): method returns a view object.
- Returns the key-value pairs of the dictionary as tuples in a list.
- The list will reflect any changes done to the dictionary
- Syntax: dictionary.items()
- keys(): returns a list of all the keys present in the dictionary.
- List will reflect any changes done to the dictionary
- Syntax: dictionary.keys()
- values(): returns a list of all the values
- Syntax: dictionary.values()

Deleting the key and value:-

- pop(): removes the specified item from the dictionary.
- Returns the removed value as output.
- Syntax : dictionary.pop(keyname, defaultvalue)
- Defaultvalue value to return if the specified key does not exist.
- popitem(): removes the item that was inserted at last into the dictionary.
- Returns the removed key-value pair as tuple.
- Syntax : dictionary.popitem()
- Eg:
- data = {"name": "Ram", "age":30}

- data.popitem() # Returns and deletes the last key/value pair in the dictionary
- • print(data.pop('age')) # Returns and Deletes the mentioned key from the dictionary
- print(data.pop('phone')) # raises KeyError
- print(data.pop('phone', "Key is not present")) # returns "Key is not present"
- del data['age'] # Deletes the Key 'age' and its value

Merging dictionaries

- d1 = {'fname': 'steve', 'Iname': 'jobs'}
- d2 = {'age': 56, 'company': 'apple'}
- d3 = {**d1, **d2}
- Or
- d3 = d1 | d2

Sequence v/s iterables

A sequence is an object which can be indexed.

- Eg: Strings, lists, tuples

2. All Sequences are Iterables. But all iterables are not sequences.

- Eg: sets and dictionaries

Note: All sequences can be merged using concatenation operator(+)

Properties	String	List	Tuple	Set	Dictionary
Boundary	‘’ / “” / “‘’”	[..]	(...)	{..}	{key: value}
Empty datatype	str() / empty quotes	list() / []	tuple() / ()	set()	dict() / {}
Mutable	No	Yes	No	Yes	Yes
Indexing and slicing	Yes	Yes	Yes	No	No
Duplicates	Allowed	Allowed	Allowed	Not allowed	Key duplication not possible
Datatypes allowed	All	All	All	Immutable/ hashable	Keys - hashable Values - any

Activity

1. points = {'a': 1, 'b': 2, 'c': 3} >>> increment the value of b
2. Difference between pop() and popitem()
3. I = ["a", "b", "c", "d"] → create a dictionary with "No value" as value for all the keys.
4. Merge two dictionaries.
5. points = {'a': 1, 'b': 2, 'c': 3}, add one more key "d" with value 4
6. dict_ = {"flowers": ["Rose", "lily", "lotus"], "animals": ["Cat", "dog"]}
Add "sunflower" to the flowers list and "Lion" to