**Dictionary (Mapping data)**

**Word & Value -> Key and Value -> Key Value pair**

* a dictionary is a built-in data type that represents a collection of key-value pairs.
* It is also sometimes referred to as an associative array or a hash map in other programming languages. (hash convert into numbers)
* Dictionaries are mutable, unordered, and can store any data type as values.
* Each key in a dictionary must be unique.
* denoted by {}
* primitive datatypes can be a key (except list and dictionary)
* value can be any data type

d**=** {}

type(d)

dict

d **=** {1,2,3}

type(d)

set

#creating a dictionary:

bio **=** { "name":"sheema ", "age": 25, "city" : "karachi"}

**Key characteristics of dictionaries in Python:**

* **Unordered:**

The order of items in a dictionary is not guaranteed.the insertion order of items is maintained, but this behavior is considered an implementation detail and should not be relied upon.

* **Mutable:**

You can add, modify, or remove key-value pairs from a dictionary after its creation.

* **Keys:**

Keys must be immutable (strings, numbers, or tuples), and each key in a dictionary must be unique. (List can not be in dictionary)

keys are created.not edited. you can change the key but cant update the old one

* **Values:**

Values can be of any data type, including other dictionaries.

* **memberships operators** are only applied on keys

input:

{"salaries" : [4000,5000,6000]}

Output:

{'salaries': [4000, 5000, 6000]}

bio

{'name': 'sheema ', 'age': 25, 'city': 'karachi'}

bio["city"]

'karachi'

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']

*#to find keys*

bio**.**keys()

dict\_keys(['name', 'age', 'city'])

*#to find values*

bio**.**values()

dict\_values(['sheema ', 25, 'karachi'])

bio

{'name': 'sheema ', 'age': 25, 'city': 'karachi'}

*#to add keys we have to create new*

bio["school"] **=** "kps"

bio

{'name': 'sheema ', 'age': 25, 'city': 'karachi', 'school': 'kps'}

*#to remove a key-value pair*

bio**.**pop("age")

bio

{'name': 'sheema ', 'city': 'karachi', 'school': 'kps'}

bio["age"] **=** 25

bio

{'name': 'sheema ', 'city': 'karachi', 'school': 'kps', 'age': 25}

**applying for loops in dictionary:**

* by default key will appear in loops

**for** i **in** bio:

print(i)

name

city

school

age

**for** key **in** bio: *#this is not dictionary just appearing to be*

print(key, bio[key])

name sheema

city karachi

school kps

age 25

**for** v **in** bio**.**values():

print(v)

sheema

karachi

kps

25

*#we can use items method to call out each elements in dictionary*

bio**.**items()

dict\_items([('name', 'sheema '), ('city', 'karachi'), ('school', 'kps'), ('age', 25)])

**for** k , v **in** bio**.**items():

print(k , v)

name sheema

city karachi

school kps

age 25

**Sets**

* a set is a collection data type that is unordered, mutable, and does not allow duplicate elements.
* It is defined by enclosing a comma-separated sequence of elements
* within ()
* it is particularly useful for tasks that involve membership testing and eliminating duplicate entries.
* Here are some key characteristics and operations associated with sets in Python

\* unordered-doesnt maintain order for data insertion

\* mutable- can be modified

\* heterogenous - may conatin all types of data

\* uniques - doesnt allow duplicate elemnets

*## creating a set:*

num **=**{1,2,3,4,5}

num

{1, 2, 3, 4, 5}

*#Adding Elements: You can add elements to a set using the add() method.*

num**.**add(6)

num

{1, 2, 3, 4, 5, 6}

*#Removing Elements: You can remove elements from a set using the remove() or discard() method.*

*The remove() method raises a KeyError if the element is not present, while discard() does not.*

num**.**remove(3)

num**.**discard(7)

num

{1, 2, 4, 5, 6}

**Set Operations:**

* Sets support various operations like

\* union (|),

\* intersection (&),

\* difference (-),

\* symmetric difference (^), etc.

Union

union\_set = my\_set | another\_set

Intersection

intersection\_set = my\_set & another\_set

Difference

difference\_set = my\_set - another\_set

Symmetric difference

symmetric\_difference\_set = my\_set ^ another\_set

*#Membership: You can check if an element is in a set using the in keyword.*

5 **in** num

True

*#Length of a Set: You can find the number of elements in a set using the len() function.*

print(len(num))

5

*#Iterating Over Sets: You can iterate over the elements of a set using a for loop.*

**for** i **in** num:

print(i)

1

2

4

5

6

**zip()**

* is a built-in function that is used to combine multiple iterables (lists, tuples, etc.)  
  element-wise.
* It takes iterables as input and returns an iterator of tuples where the i-th tuple contains the i-th element from each of the input iterables.
* It's important to note that if the input iterables are of different lengths, zip() stops as soon as the shortest iterable is exhausted. This means that any extra elements in longer iterables are ignored.

*#****example of how zip() works:***

list1 **=** [1, 2, 3]

list2 **=** ['a', 'b', 'c']

list3 **=** [10, 20, 30]

*#* ***Using zip to combine lists element-wise***

zipped\_lists **=** zip(list1, list2, list3)

zipped\_lists

<zip at 0x1a545d69640>

*#* ***Converting the iterator to a list***

result **=** list(zipped\_lists)

result

[(1, 'a', 10), (2, 'b', 20), (3, 'c', 30)]

**In this example, zip() combines the elements of list1, list2, and list3 into tuples, creating a new list of tuples where each tuple contains elements from the corresponding positions of the input lists.**