

# DICTIONARIES & ?

Dictionaries are mappings, this basically contains key-value pairs. The Mapping won't retain order since they have objects defined by a key.

## Creating a dictionary:

In: `myd = {'k1': 'v1', 'k2': 'v2'}`

In: `myd['k2']` # calling my value

O/P: `'v2'`

## I can hold any data type

In: `myd = {'k1': 123, 'k2': [12, 23, 33], 'k3': {'k1': 'v1', 'k2': 'v2'}}`

In: `myd['k3']`

O/P: `{'k1': 'v1', 'k2': 'v2'}`

## Index on that value

In: `myd['k2'][0]`

O/P: `12`

## Method on the value

In: `myd = {'k1': 'v1', 'k2': 'v2'}`

In: `myd['k1'].upper()`

o/p: ~~myd~~ V1

In: `myd = {'k1': 123}`

## We can also affect the values

In: `myd['k1'] = 123`

In: `myd['k1']`

o/p: 0

## WE CAN ALSO CREATE KEYS BY

## ASSIGNMENT

In: `d = {}`

In: `d['animal'] = 'Dog'`



In: `d['ans'] = 42`

In: `d`

O/p: `{'animal': 'Dog', 'ans': 42}`

## NESTING WITH DICTIONARIES

In: `d = {'key1': {'nestkey': {'subnestkey': 'value'}}}`

In: `d['key1']['nestkey']['subnestkey']`

O/p: `'value'`

## METHODS (Object.method())

In: `d = {'k1': 1, 'k2': 2, 'k3': 3}`

In: `d.keys()` # Prints all keys

O/p: `dict_keys(['k1', 'k2', 'k3'])`

O/p: ~~`dict_keys(['k1', 'k2', 'k3'])`~~

In: `d.values()` # Prints all values

O/p: ~~`d.items()`~~

O/p: `dict_values([1, 2, 3])`

In: `d.items()` # Prints all items

O/p: `dict_items([('k1', 1), ('k2', 2), ('k3', 3)])`

# SET AND BOOLEANS

Sets →

They are an unordered collection of unique elements. We can construct them by using `set()` function.

Q. Creating a set

gn: `x = set()`

gn: `x.add(1)`

Q. ADDING ELEMENTS (ADD METHOD)

gn: `x.add(1)`

gn: `x`

o/p: `{1}`

gn: `x.add(2)`  
`x`

o/p: `{1, 2}`

gn: `x.add(1)`  
`x`

o/p: `{1, 2}` // Only Unique elements



We can cast a list or tuple to a set

gn: `l1 = [1, 1, 2, 2, 3, 3, 4, 4, 5, 6, 1, 1]`

gn: `set(l1)`

o/p: `{1, 2, 3, 4, 5, 6}`

## BOOLEANS

Python comes with Booleans (with predefined `True` & `false` displays that are basically just the integers `1` & `0`). It also has a placeholder called `None`.

gn: `a = True`

a

o/p: `True`

gn: `1 > 2`

o/p: `False`

gn: `b = None`

o/p: `print(b)`

o/p: `None`