Dictionary

- **Key-Value Pairs**: Data mapping.
- **Aliases**: Map, associative array.

```
dict = { 'name': 'nitish', 'age': 33, 'gender': 'male' }
```

Traits:

- *Mutable*
- *No indexing*
- *Unique keys*
- *Immutable keys*

In [11]:
```
{"Name": "Ketaki", "Gender":"Female"}
```

Out[11]:
```
{'Name': 'Ketaki', 'Gender': 'Female'}
```

4 Key Rules About Dictionaries

1. **No Indexing**: No index support.
2. **Mutable**: Can modify post-creation.
3. **Keys**: Immutable (str, num, tuple); **Values**: Any type.
4. **Unique Keys**: Keys must be unique; overwrite on reassignment.

In [1]:
```
# Mutable Types   ---> Lists, Sets, Dicts

# Immutable Types ---> Str, Tuples, Int, Float, Bool, Complex
```

1. Create
2. Access
3. Edit

4. Add
5. Delete
6. Operations
7. Functions

1. Create

In [1]:
```
# empty dictionary
D = {}
D
```

Out[1]:
```
{}
```

In [3]:
```
# 1D dictionary
D = {'Name': 'Manish', 'Gender': 'Male'}
D
```

Out[3]:
```
{'Name': 'Manish', 'Gender': 'Male'}
```

In [4]:
```
D1 = {[1,2,3]:"Saurabh"}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[4], line 1
----> 1 D1 = {[1,2,3]:"Saurabh"}

TypeError: unhashable type: 'list'
```

In [5]:
```
# with mixed keys
D1 = {(1, 2, 3):"Saurabh"}
D1
```

Out[5]:
```
{(1, 2, 3): 'Saurabh'}
```

In [6]:
```
# duplicate keys
D2 = {"Name":"Krishna", "Name":"Max"}
D2
```

Out[6]:
```
{'Name': 'Max'}
```

In [7]:
```
D3 = {"Name":"Saurabh", "College":"SGT", "Marks":{"M1":99, "DS":97, "Eng":98}}
D3
```

Out[7]:
```
{'Name': 'Saurabh', 'College': 'SGT', 'Marks': {'M1': 99, 'DS': 97, 'Eng': 98}}
```

In [8]:
```
# 2D dictionary ---> JSON
s = {
    'name':'Aditya',
     'college':'birla',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
s
```

Out[8]:
```
{'name': 'Aditya',
 'college': 'birla',
 'sem': 4,
 'subjects': {'dsa': 50, 'maths': 67, 'english': 34}}
```

In [41]:
```
# immutable items as keys
D4 = {'name':'akanksha', (1, 2, 3):2}
print(D4)
```

```
{'name': 'akanksha', (1, 2, 3): 2}
```

In [13]:
```
# using sequence and dict function
D5 = dict([("Name","Rutik"),("Age",23),("Gender","Male")])
D5
```

Out[13]:
```
{'Name': 'Rutik', 'Age': 23, 'Gender': 'Male'}
```

2. Access

In [14]:
```
D
```

Out[14]:
```
{'Name': 'Manish', 'Gender': 'Male'}
```

In [15]:
```
D[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[15], line 1
----> 1 D[0]

KeyError: 0
```

In [16]:
```
D["Name"]
```

Out[16]:
```
'Manish'
```

In [18]:
```
D.get("Name")
```

Out[18]:
```
'Manish'
```

In [17]:
```
D["Gender"]
```

Out[17]:
```
'Male'
```

In [19]:
```
D3
```

Out[19]:
```
{'Name': 'Saurabh', 'College': 'SGT', 'Marks': {'M1': 99, 'DS': 97, 'Eng': 98}}
```

In [20]:
```
D3.get("M1")
# .get() ---> 1-D dicts only; Not for 2-D/nested dicts.
```

In [21]:
```
D3
```

Out[21]:
```
{'Name': 'Saurabh', 'College': 'SGT', 'Marks': {'M1': 99, 'DS': 97, 'Eng': 98}}
```

In [24]:
```
D3["Marks"]["DS"]
```

Out[24]:
```
97
```

3. Edit

In [25]:
```
D
```

Out[25]:
```
{'Name': 'Manish', 'Gender': 'Male'}
```

In [26]:
```
D["Name"] = "Harshal"
D["Gender"] = "Male"
D
```

Out[26]:
```
{'Name': 'Harshal', 'Gender': 'Male'}
```

In [27]:
```
D3["Marks"]["M1"] = 35
D3
```

Out[27]:
```
{'Name': 'Saurabh', 'College': 'SGT', 'Marks': {'M1': 35, 'DS': 97, 'Eng': 98}}
```

4. Add

In [28]:
```
D
```

Out[28]:
```
{'Name': 'Harshal', 'Gender': 'Male'}
```

In [29]:
```
D['Age'] = 22
D
```

Out[29]:
```
{'Name': 'Harshal', 'Gender': 'Male', 'Age': 22}
```

In [30]:
```
D3["Marks"]["Python"] = 95
D3
```

Out[30]:
```
{'Name': 'Saurabh',
 'College': 'SGT',
 'Marks': {'M1': 35, 'DS': 97, 'Eng': 98, 'Python': 95}}
```

5. Delete

In [31]:
```
D5 = {}
D5
```

Out[31]:
```
{}
```

In [32]:
```
del D5
D5
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[32], line 2
      1 del D5
----> 2 D5

NameError: name 'D5' is not defined
```

In [33]:
```
D.pop('Name')
```

Out[33]:
```
'Harshal'
```

In [34]:
```
D
```

Out[34]:
```
{'Gender': 'Male', 'Age': 22}
```

In [35]:
```
D.popitem()
```

Out[35]:
```
('Age', 22)
```

In [36]:
```
D
```

Out[36]:
```
{'Gender': 'Male'}
```

In [37]:
```
del D["Gender"]
D
```

Out[37]:
```
{}
```

In [38]:
```
D.clear()
D
```

Out[38]:
```
{}
```

6. Operations

In [39]:
```
D3
```

Out[39]:
```
{'Name': 'Saurabh',
 'College': 'SGT',
 'Marks': {'M1': 35, 'DS': 97, 'Eng': 98, 'Python': 95}}
```

In [42]:
```
D4
```

Out[42]:
```
{'name': 'akanksha', (1, 2, 3): 2}
```

In [43]:
```
D3 + D4
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[43], line 1
----> 1 D3 + D4

TypeError: unsupported operand type(s) for +: 'dict' and 'dict'
```

In [44]:
```
D3 * 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[44], line 1
----> 1 D3 * 3

TypeError: unsupported operand type(s) for *: 'dict' and 'int'
```

In [45]:
```
D3
```

Out[45]:
```
{'Name': 'Saurabh',
 'College': 'SGT',
 'Marks': {'M1': 35, 'DS': 97, 'Eng': 98, 'Python': 95}}
```

In [49]:
```
# Iteration
for i in D3:
    print(i)
```

```
Name
College
Marks
```

In [52]:
```
for i in D3:
    print(i,":",D3[i])
```

```
Name : Saurabh
College : SGT
Marks : {'M1': 35, 'DS': 97, 'Eng': 98, 'Python': 95}
```

In [54]:
```
# Membership
"Saurabh" in  D3
```

Out[54]:
```
False
```

In [55]:
```
"Name" in  D3
```

Out[55]:
```
True
```

7. Functions

In [56]:
```
len(D3)
```

Out[56]:
```
3
```

In [57]:
```
min(D3)
```

Out[57]:
```
'College'
```

In [58]:
```
max(D3)
```

Out[58]:
```
'Name'
```

In [59]:
```
sorted(D3)
```

Out[59]:
```
['College', 'Marks', 'Name']
```

In [60]:
```
sorted(D3, reverse = True)
```

Out[60]:
```
['Name', 'Marks', 'College']
```

In [61]:
```
D3.items()
```

Out[61]:
```
dict_items([('Name', 'Saurabh'), ('College', 'SGT'), ('Marks', {'M1': 35, 'DS': 97, 'Eng': 98, 'Python': 95})])
```

In [62]:
```
D3.keys()
```

Out[62]:
```
dict_keys(['Name', 'College', 'Marks'])
```

In [63]:
```
D3.values()
```

Out[63]:
```
dict_values(['Saurabh', 'SGT', {'M1': 35, 'DS': 97, 'Eng': 98, 'Python': 95}])
```

In [64]:
```
# update
d1 = {1:2, 3:4, 4:5}
d2 = {4:7, 6:8}
d1.update(d2)
print(d1)
```

```
{1: 2, 3: 4, 4: 7, 6: 8}
```

Dictionary Comprehension

```
{key : value for var in iterable}
```

Creates dicts from iterables.

Example:

```
squares_dict = {x: x**2 for x in range(5)}
```

Maps numbers to squares.

Supports conditions & expressions.

In [70]:
```
D = {"Name":"Shyam", "Gender":"Male", "Age":25}
```

In [71]:
```
D.items()
```

Out[71]:
```
dict_items([('Name', 'Shyam'), ('Gender', 'Male'), ('Age', 25)])
```

In [72]:
```
D1 = {key:value for key, value in D.items() if len(key)>3}
D1
```

Out[72]:
```
{'Name': 'Shyam', 'Gender': 'Male'}
```

In [73]:
```
L = [1, 2, 3, 4, 5, 6, 7]
```

In [74]:
```
D2 = {item : item**2 for item in L}
D2
```

Out[74]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49}
```

In [76]:
```
D2 = {item : item**2 for item in L if item%2 != 0}
D2
```

Out[76]:
```
{1: 1, 3: 9, 5: 25, 7: 49}
```

In [61]:
```
# Print 1st 10 nums & squares
{i:i**2 for i in range(1, 11)}
```

Out[61]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

In [62]:
```
distances = {'delhi':1000, 'mumbai':2000, 'bangalore':3000}
print(distances.items())
```

```
dict_items([('delhi', 1000), ('mumbai', 2000), ('bangalore', 3000)])
```

In [99]:
```
# using existing dict
distances = {'delhi':1000, 'mumbai':2000, 'bangalore':3000}
{key:value*0.62 for (key, value) in distances.items()}
```

Out[99]:
```
{'delhi': 620.0, 'mumbai': 1240.0, 'bangalore': 1860.0}
```

In [59]:
```
# using zip
days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]
temp_C = [30.5, 32.6, 31.8, 33.4, 29.8, 30.2, 29.9]
{i:j for (i, j) in zip(days, temp_C)}
```

Out[59]:
```
{'Sunday': 30.5,
 'Monday': 32.6,
 'Tuesday': 31.8,
 'Wednesday': 33.4,
 'Thursday': 29.8,
 'Friday': 30.2,
 'Saturday': 29.9}
```

In [62]:
```
# using if condition
products = {'phone':10, 'laptop':0, 'charger':32, 'tablet':0}
{key:value for (key, value) in products.items() if value>0}
```

Out[62]:
```
{'phone': 10, 'charger': 32}
```

In [63]:
```
# Nested Comprehension
# Print multiplication tables for 2 to 4
{i:{j:i*j for j in range(1, 11)} for i in range(2, 5)}


# i --> 2 3 4
# for outer dimension j and for inner dimension(i) ---> 1 2 3 4 5 6 7 8 9 10
# J for inner dimension ----> 2 *  1 2 3 4 5 6 7 8 9 10
```

Out[63]:
```
{2: {1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18, 10: 20},
 3: {1: 3, 2: 6, 3: 9, 4: 12, 5: 15, 6: 18, 7: 21, 8: 24, 9: 27, 10: 30},
 4: {1: 4, 2: 8, 3: 12, 4: 16, 5: 20, 6: 24, 7: 28, 8: 32, 9: 36, 10: 40}}
```

In [64]:
```
{
    2:{1:2, 2:4, 3:6, 4:8},
    3:{1:3, 2:6, 3:9, 4:12},
    4:{1:4, 2:8, 3:12, 4:16}
}
```

Out[64]:
```
{2: {1: 2, 2: 4, 3: 6, 4: 8},
 3: {1: 3, 2: 6, 3: 9, 4: 12},
 4: {1: 4, 2: 8, 3: 12, 4: 16}}
```