Dictionary

- Python dictionary is an **ordered collection of items**.

- As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.
- Each item of a dictionary has a `key-value` pair.
- Dictionaries are **optimized to retrieve values when the key is known.**
- A dictionary is a collection that is **changeable** and **does not allow duplicates**.
- Dictionaries are written **with curly brackets** and **have keys and values.**
- **Syntax**➖

```
dict={
  "key1": value,
  "key2": "value",
}
```

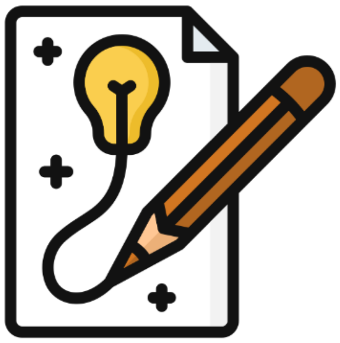
```

```

Creating Python Dictionary

While the values can be of any data type and can repeat, keys must be of **immutable type (string, number, or tuple with immutable elements) and must be unique**.

- **Empty dictionary**

```
mydic={}        # empty dictionary
```



- **Dictionary where keys are integer type**

```
Dic={1: "Hello",2: "World"}        # dictionary where keys are integer type
```

- **Dictionary where keys are of mixed type**

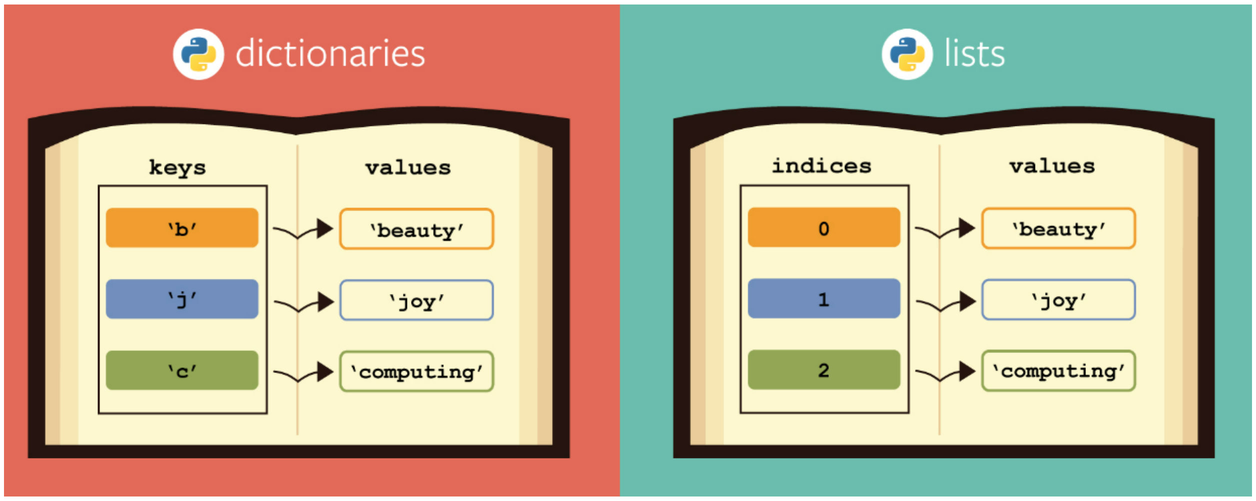
```
myDic={"name":"Sam",2:[5,4,3]}     # dictionary where keys are of mixed type
```

- **Creating Dictionary using in-built function dict()**

```
# dictionary created using inbuilt function

DicFun=dict({1: "Python",2: "Java", 3: "HTML"})
```

Going from List to dictionaries

Here, Instructor needs to compare the List and Dictionaries. What types of things are similar and what things are going to be different



Code 1: Print the dictionary and type of it

```
student = {
"name": "Rahul",
"age": 23,
"nationality: "Indian",
"location": "Nainital",
is_married: false,
highest_degree: "Btech"
}
print(student)
print(type(student))
```

Code 2: Print the "brand" value of the given dictionary

```
# Print the "brand" value of the dictionary:

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"])
```

Duplicate values will overwrite existing values:

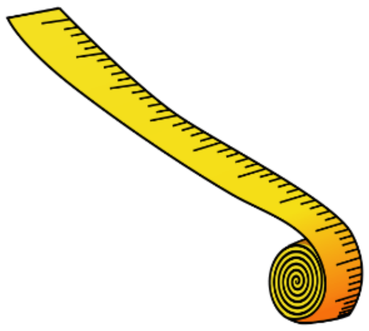
Keys are unique

```
thisdict = {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964,
 "year": 2020,
 "colors": ["red", "white", "blue"]
}
print(thisdict)
```

Dictionary Length

To determine how many items a dictionary has, use the `len()` function.

```
Dict={"name": "ABC", "age": 25, "City": "Delhi"}
print(len(Dict))      # 3
```



Code 3: Getting values of the Index and by using the If-else condition

```
student = {
"name": "Rahul",
"age": 23,
"nationality": "Indian",
"location": "Nainital",
"is_married": False,
"highest_degree": "Btech"
}
if not student['is_married']:
  print("Naam to suna hi hoga")
else:
 print("Sunn ke koi fayda nahin")
```

Accessing Elements from the Dictionary

- To access values, a dictionary uses `keys`
- Keys can be used either inside **square brackets** `[]` or with the `get()` **method**.
- `KeyError` occurs in case a key is not found in the dictionary.

```
Dict={"name": "ABC", "age": 25, "city": "Delhi"}

print(Dict["name"])         # ABC
print(Dict.get("city"))     # Delhi
print(Dict["DOB"])          # KeyError
print(Dict.get("DOB"))      # KeyError
```

Loop Through a Dictionary

- You can loop through a dictionary by using a `for` loop.
- When looping through a dictionary, the return value is the *keys* of the dictionary, but there are methods to return the *values* as well.

```
student = {
"name": "Rahul",
"age": 23,
"nationality": "Indian",
"location": "Nainital",
"is_married": False,
"highest_degree": "Btech",
"pcm_marks": [12,45,78]
}

# Ist way to iterate in a python dictionary
for k in student:
  print(k,student[k])
  print("One key-value ends here")

# IInd way of iterating in a python dictionary
```

```
for k, v in student.items():
   print(k,":",v)
```

Updating Dictionary Elements

- Dictionaries are **mutable**.
- We can **add new items** or **change the value of existing items** using an assignment operator.

```
Dict={"name": "ABC", "age": 25, "city": "Delhi"}

# Updating
Dict["age"]=26
print(Dict)
# {'name': 'ABC', 'age': 26, 'city': 'Delhi'}
```

- If the key is already present, then the existing value gets updated.
- If the key is not present, a new (**key: value**) pair is added to the dictionary.

```
Dict={"name": "ABC", "age": 25, "city": "Delhi"}

# Adding
Dict["country"]="India"
print(Dict)
# {'name':'ABC','age':26,'city':'Delhi','country':'India'}
```

Removing Elements from Dictionary

- We can remove a particular item in a dictionary by using the `pop()` method. This method removes an item with the provided `key` and returns the `value`.

```
Dict={"name": "ABC", "age": 25, "city": "Delhi"}
Dict.pop("name")
print(Dict)
#  {'name': 'ABC', 'city': 'Delhi', 'country': 'India'}
```

- The `popitem()` method can be used to remove and return an arbitrary `(key, value)` item pair from the dictionary.

```
Dict={"name": "ABC", "age": 25, "city": "Delhi", "country": "India"}
Dict.popitem()
print(Dict)          # {'name': 'ABC', 'age': 25, 'city': 'Delhi'}
```

- All the items can be removed at once, using the `clear()` method.

```
Dict={"name": "ABC", "age": 25, "city": "Delhi","country":"India"}
Dict.clear()
print(Dict)                   # {}
```

- We can also use the `del` keyword to remove individual items or the entire dictionary itself.

```
Dict={"name": "ABC", "age": 25, "city": "Delhi", "country": "India"}
del Dict["city"]
print(Dict)     # {'name': 'ABC', 'age': 25, 'country': 'India'}

del Dict
print(Dict)     # Throw Error as the Dict is deleted
```

Happy Coding!