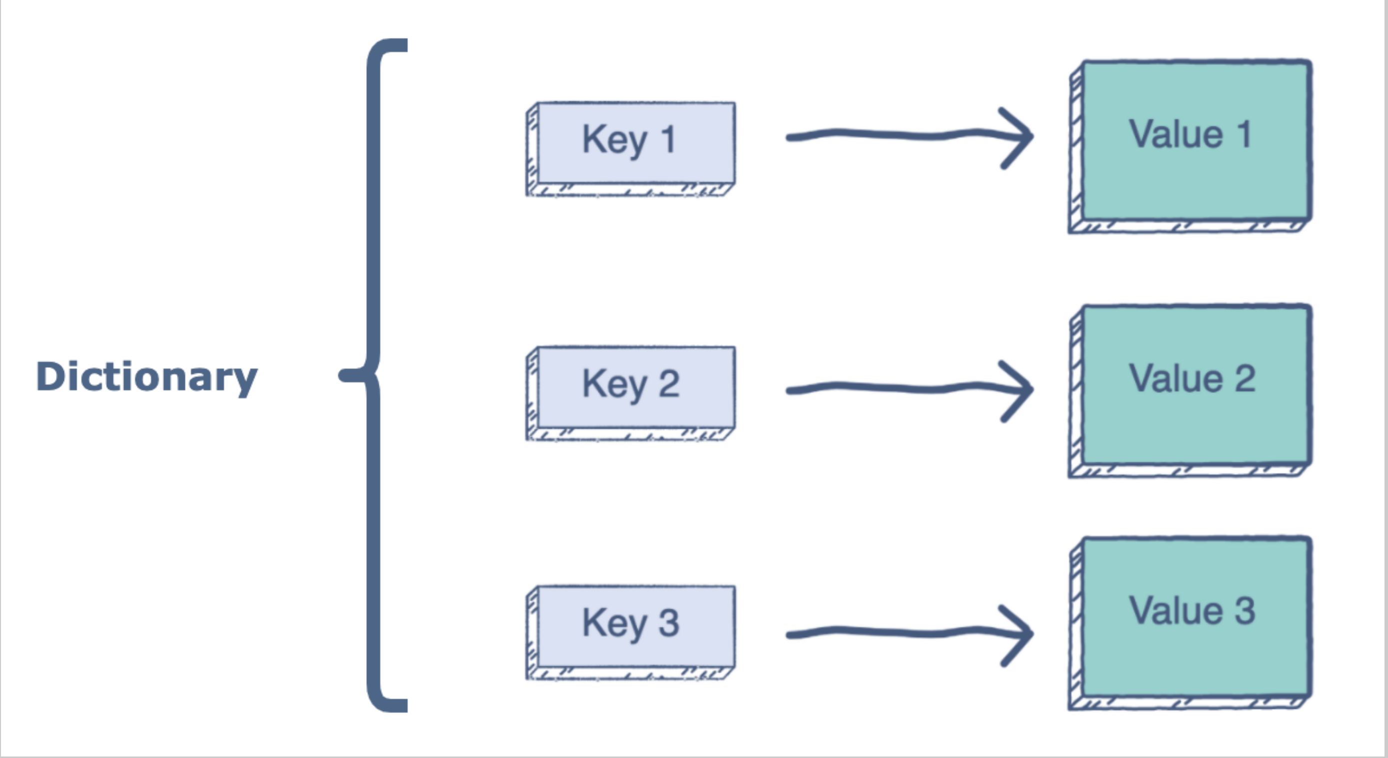
How does the dictionary work in Python?

python3 | dictionary | basics | communitycreator

A **dictionary** contains a collection of `indices` and `values` (indices are also called **keys**). Each key is associated with a single value. The association of a key and a value is called a **key-value pair** or an **item**.

Each key is separated from its value by a colon `:`, key-value pairs are separated by commas `,`, and the whole thing is enclosed in curly braces `{}`. An empty dictionary without any items is written with just two curly braces, like this: `{}`.



Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type (i.e., strings, numbers, or tuples).

Syntax

dictionary = { `key1` : `Value1`, `key2` : `Value2`, ...}

```
# example
dict = {'Name': 'Sam', 'Age': 6, 'Class': 'First'}
# All keys are unique:
# 'Name'
# 'Age'
# 'Class'
```

Accessing the dictionary

To access dictionary elements, you can use the familiar square brackets `[]`, along with the key, to obtain its value.

```
dict = {'Name': 'Sam', 'Age': 6, 'Class': 'First'}

print(dict['Name'])
```

> If we attempt to access a key-value with a key that is not part of the dictionary, we get an error.

Insertion & updation

We can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry

```
dict = {'Name': 'Sam', 'Age': 6, 'Class': 'First'}
print(dict)

# Adding new key-value pair
dict['School'] = "GITAM"
print(dict)

# updating existing key-value pair
dict['Age'] = 8
print (dict)
```

Deletion

We can either remove individual dictionary elements or clear the entire contents of a dictionary. We can also delete the entire dictionary in a single operation.

```
dict = {'Name': 'Sam', 'Age': 6, 'Class': 'First'}
print(dict)

# remove entry with key 'Name'
del dict['Name']
print(dict)

# remove all entries in dict
dict.clear()
print(dict)

# delete entire dictionary
del dict
print(dict)
```

Try here

Properties of Dictionary Keys

1. Duplicate keys are not allowed. When duplicate keys are encountered during the assignment, the value will be the last assigned one.
2. Keys must be immutable. This means you can use strings, numbers, or tuples as dictionary keys, but something like ['key'] is not allowed.

Dictionary functions:

1. **`cmp(dict1, dict2)`** : Compares elements of both dictionarys.
2. **`len(dict)`** : Gives the total length of the dictionary. This would be equal to the number of key-value pairs in the dictionary.
3. **`str(dict)`** : Produces a printable string representation of a dictionary.
4. **`type(variable)`** : Returns the type of the passed variable. If the passed variable is a dictionary, it will return a dictionary type.

Dictionary built-in methods

1. **`dict.clear()`** : Removes all elements of dictionary dict.
2. **`dict.copy()`** : Returns a shallow copy of dictionary dict.
3. **`dict.fromkeys()`** : Create a new dictionary with keys from seq and values set to value.
4. **`dict.get(key, default = None)`** : For key, returns value or default if the key is not in the dictionary.
5. **`dict.has_key(key)`** : Returns true if a key exists in the dictionary and false if otherwise.
6. **`dict.items()`** : Returns a list of dict's (key, value) tuple pairs.
7. **`dict.keys()`** : Returns a list of dictionary keys.
8. **`dict.setdefault(key, default = None)`** : Similar to `get()`, but it will set dict[key]=default if the key does not already exist in the dictionary.
9. **`dict.update(dict2)`** : Merges dictionary `dict2's` key-values pairs with `dict`.
10. **`dict.values()`** : Returns list of dictionary values.

Contributor: Swasthika Jain T J