# Data Structure - Dictionary

## 1. Dictionary

Dictionary is a common feature of modern languages (often known as maps, associative arrays, or hashmaps) which let you associate pairs of values together.

In Python, dictionaries are defined in **dict** data type.

- It stores keys and their corresponding values.
- Keys must be **unique** and **immutable**.
- It is **mutable**, i.e. you can add and remove items from a dictionary.
- It is **unordered**, i.e. items in a dictionary are not ordered.

### How to create a dictionary?

Dictionary is created with listed of items surrounded by curly brackets **"{ }"**, and seperated by comma **","**.

- To create an empty dictionary, simple use **"{}"**
- Key and value are separated by colon **":"**
- Key needs to be immutable type, e.g. data type like scalar, string or tuple

**Exercise:**

In [5]:

```
# empty dictionary
d0 = {}

# dictionary with mixed data type
d1 = {'a': 'alpha', 'b': 123}
print(d1)
```

```
{'a': 'alpha', 'b': 123}
```

**Question:**

Create a disctionary `fruits` which has following keys and values.

| key | value |
|---|---|
| a | Apple |
| b | Banana |
| c | Cherries |

In [6]:
```
fruits = {'a':'Apple', 'b':'Banana', 'c':'Cherries'}
fruits
```

Out[6]:
```
{'a': 'Apple', 'b': 'Banana', 'c': 'Cherries'}
```

## 2. Access Item(s)

Items in dictionary can be accessed by their respective keys.

- Key can be used either inside square brackets or with the **get()** method.
- When using square brackets, it will throw a `KeyError` Exception if the key is not found.

**Exercise:**

Print corresponding value in `fruits` for key `'b'`.

In [11]:
```
print(fruits['b'])
```

```
Banana
```

**Question:** What happens when you try to use a non-existence key, e.g. 'z'?

In [10]:
```
print(fruits['z'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-10-47aefa25407c> in <module>
----> 1 print(fruits['z'])

KeyError: 'z'
```

### Use `get()`

By using `get()` function, you can specify a default value to be returned if key is not found.

Check out the documentation of `dict.get()` function.

In [25]:
```
dict.get?
```

**Exercise:**

Use `dict.get()` function to get item with key = `'z'`.

In [ ]:
```
fruits.get('z', 'Not Fruit Today')
```

### Length

To find the length of the list or the number of elements in a list, **len( )** is used.

**Question:**

- Find the lengh of `fruits` dictionary.

In [ ]:
```
1 len(fruits)
```

### keys(), values(), items()

- **keys()** return a new view of the dictionary's keys.
- **values()** return a new view of the dictionary's values.
- **items()** return a new view of the dictionary's items (key, value).

**Exercise:** Print keys(), values(), and items() of fruits.

In [13]:
```
1 print(fruits.keys())
2 print(fruits.values())
3 print(fruits.items())
```

```
dict_keys(['a', 'b', 'c'])
dict_values(['Apple', 'Banana', 'Cherries'])
dict_items([('a', 'Apple'), ('b', 'Banana'), ('c', 'Cherries')])
```

## 3. Working with Dictionary

Dictionary is mutable. We can add new items or change the value of existing items using assignment operator.

- If the key exists in the dictionary, existing value will be updated.
- If the key doesn't exists in the dictioinary, new key:value pair is added to dictionary.

### Copy a Dictionary

Since a dictionary is immutable, we cannot use assignment statement `=` to copy an dictionary. Use one of following ways instead.

- Use its `copy()` function.
- Use `dict()` constructor function.

In [15]:
```
1 f1 = fruits.copy()
2 print(f1 is fruits)
3 print(f1)
```

```
False
{'a': 'Apple', 'b': 'Banana', 'c': 'Cherries'}
```

## Update an Item

- Create a dictionary `mixed` by copying `fruits` object.
- Update its key `a` value to `['Apple', 'Avocado']`

In [27]:
```
mixed = dict(fruits)
mixed['a'] = ['Apple', 'Avocado']
print(mixed)
```

```
{'a': ['Apple', 'Avocado'], 'b': 'Banana', 'c': 'Cherries', 'f': 'Fig'}
```

## Add an Item

- Add another key-value pair `{'f':'Fig'}` to `fruits` object

In [21]:
```
fruits['f']='Fig'
print(fruits)
```

```
{'a': 'Apple', 'b': 'Banana', 'c': 'Cherries', 'f': 'Fig'}
```

## Merge Dictionaries

**update()** method is used to merge items from another dictionary.

- Adds element(s) to the dictionary if the key is not in the dictionary.
- If the key is in the dictionary, it updates the key with the new value.

**Question:**

- Create a dictionary `f1` by copying `fruits` object
- Create another dictionary `f3` with items `{'d':'Dates', 'e':'Eldercherry', 'f':'Fig', 'g':'Grape'}`
- Add/update items from `f3` to `f2`

In [25]:
```
f2 = fruits.copy()
print(f2)

f3 = {'a': 'avacado', 'd':'Durian', 'f':'Fig'}
f2.update(f3)
print(f2)
```

```
{'a': 'Apple', 'b': 'Banana', 'c': 'Cherries', 'f': 'Fig'}
{'a': 'avacado', 'b': 'Banana', 'c': 'Cherries', 'f': 'Fig', 'd': 'Durian'}
```

## Remove an Item

**pop()**

- It is used to remove an item by key and returns the value.
- It throws exception if key is not found.

In [40]:
```
mixed = dict(fruits)
val = mixed.pop('c')
print(val)
print(mixed)
```

```
Cherries
{'a': 'Apple', 'b': 'Banana', 'd': 'Durian'}
```

## 4. Membership Test

We can use `in` statement to check membership of a **key** in a dictionary.

**Question:**

- Check whether key `a` and `z` are in the `fruits` dictionary.

In [26]:
```
print('a' in fruits, 'z' in fruits)
```

```
True False
```

**Question:**

- How to test if a value `Apple` is in a dictionary?
- How to test if a key-value pair `{'a':'Apple'}` is in the dictionary?

In [31]:
```
print(fruits)
print('Apple' in fruits.values())
print(('a', 'Apple') in fruits.items())
```

```
{'a': 'Apple', 'b': 'Banana', 'c': 'Cherries', 'd': 'Durian'}
True
True
False
```

**Question:**

In a dictionary, how to find key by matching its value?

In [38]:
```
for k,v in fruits.items():
    if v == 'Banana':
        print(k)
        break
```

```
b
```