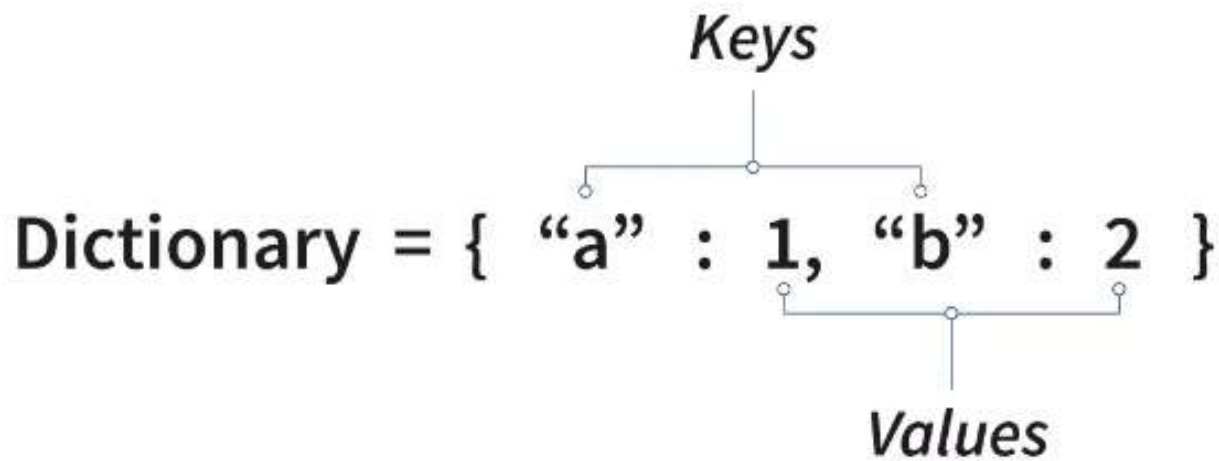
Nested Dictionary

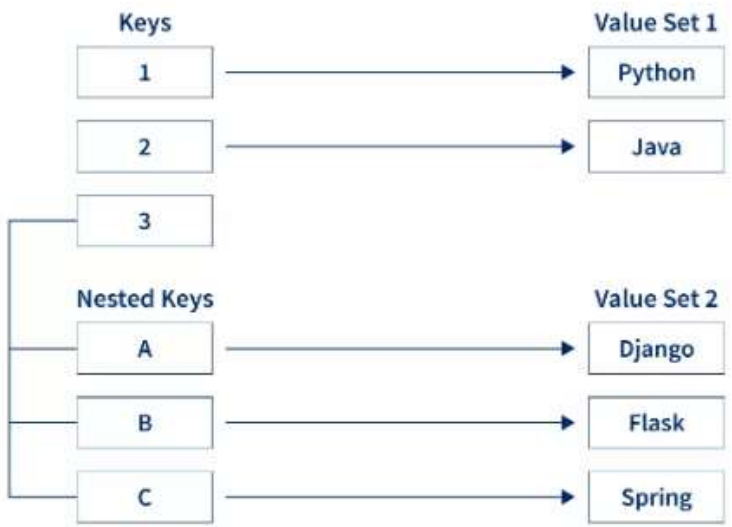
Dictionary in Python is one of the most popular **data structures**. Dictionaries are used to store data in a "key-value" pair format. The keys are always unique within a dictionary. The values of the Python dictionary may or may not be unique. We can define a dictionary by enclosing a comma-separated list of key-value pairs in curly braces ({}). A colon (:) separates each key from its associated value.



So, what are Nested Dictionaries?

In Python, a nested dictionary is a dictionary inside a dictionary. Basically, it is a collection of dictionaries kept inside a single dictionary.

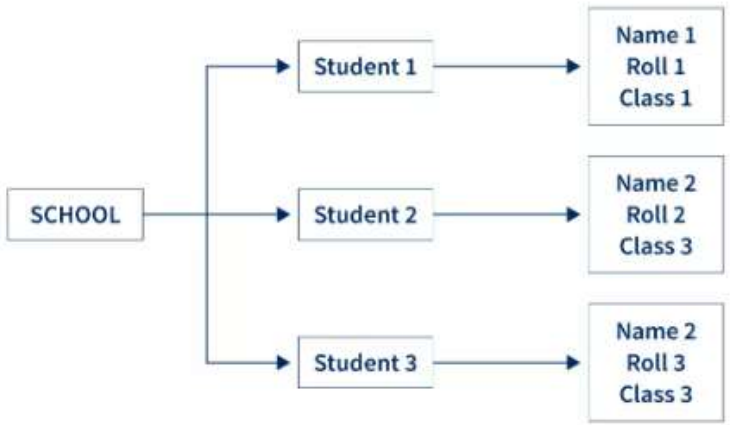
Nested dictionaries are one of the ways to represent and store structured information (similar to some 'records' in other languages).



Below given is a simple example of a nested dictionary.

```
my_dict = { 'dict1': {'key_A': 'value_A'},
        'dict2': {'key_B': 'value_B'}}
```

Let us understand this clearly through an example. Suppose, we have kept the records of 3 students in a dictionary. Say, an individual student has the following properties: **Name, Class**, and **Roll No**.



We can represent this very clearly through a nested dictionary.

```
school = {'student1': {'name': 'Ace', 'roll': '10', 'class': '5'},
     'student2': {'name': 'Bob', 'roll': '15', 'class': '5'},
     'student3': {'name': 'Ron', 'roll': '18', 'class': '6'}}
```

What is Nested Dictionary in Python?

In Python, a nested dictionary is a dictionary inside a dictionary. It's a collection of dictionaries into one single dictionary.

```
nested_dict = { 'dictA': {'key_1': 'value_1'},
                'dictB': {'key_2': 'value_2'}}
```

Here, the `nested_dict` is a nested dictionary with the dictionary `dictA` and `dictB`. They are two dictionary each having own key and value.

Create a Nested Dictionary

We're going to create dictionary of people within a dictionary.

Example 1: How to create a nested dictionary

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}

print(people)
```

Access elements of a Nested Dictionary

To access element of a nested dictionary, we use indexing `[]` syntax in Python.

Example 2: Access the elements using the [] syntax

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}

print(people[1]['name'])
print(people[1]['age'])
print(people[1]['sex'])
```

Add element to a Nested Dictionary

Example 3: How to change or add elements in a nested dictionary?

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}

people[3] = {}

people[3]['name'] = 'Luna'
people[3]['age'] = '24'
people[3]['sex'] = 'Female'
people[3]['married'] = 'No'

print(people[3])
```

Delete elements from a Nested Dictionary

In Python, we use " del " statement to delete elements from nested dictionary.

Example 5: How to delete elements from a nested dictionary?

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'},
          3: {'name': 'Luna', 'age': '24', 'sex': 'Female', 'married': 'No'},
          4: {'name': 'Peter', 'age': '29', 'sex': 'Male', 'married': 'Yes'}}

del people[3]['married']
del people[4]['married']

print(people[3])
print(people[4])
```

Example 6: How to delete dictionary from a nested dictionary?

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'},
          3: {'name': 'Luna', 'age': '24', 'sex': 'Female'},
          4: {'name': 'Peter', 'age': '29', 'sex': 'Male'}}

del people[3], people[4]
print(people)
```

Iterating Through a Nested Dictionary

Using the for loops, we can iterate through each elements in a nested dictionary.

Example 7: How to iterate through a Nested dictionary?

```
people = {1: {'Name': 'John', 'Age': '27', 'Sex': 'Male'},
          2: {'Name': 'Marie', 'Age': '22', 'Sex': 'Female'}}

for p_id, p_info in people.items():
    print("\nPerson ID:", p_id)

    for key in p_info:
        print(key + ':', p_info[key])
```

1. Rename key of a dictionary

Write a program to rename a key `city` to a `location` in the following dictionary.

Given:

```
sample_dict = {
  "name": "Kelly",
  "age":25,
  "salary": 8000,
  "city": "New york"
}
```

Expected output:

```
{'name': 'Kelly', 'age': 25, 'salary': 8000, 'location': 'New york'}
```

2. Get the key of a minimum value from the following dictionary

```
sample_dict = {
  'Physics': 82,
  'Math': 65,
  'history': 75
}
```

3. Change value of a key in a nested dictionary, update salary of emp3 to 8500

Given:

```
sample_dict = {
    'emp1': {'name': 'Jhon', 'salary': 7500},
    'emp2': {'name': 'Emma', 'salary': 8000},
    'emp3': {'name': 'Brad', 'salary': 500}
}
```

Expected output:

```
{
   'emp1': {'name': 'Jhon', 'salary': 7500},
   'emp2': {'name': 'Emma', 'salary': 8000},
   'emp3': {'name': 'Brad', 'salary': 8500}
}
```