Analytics using Python

Learning outcomes

1. You will learn Python , a useful language

2. Use programming for problem solving

Great Lakes Institute of Management

A guide to learn python for analytics

P. V. Subramanian

**A workbook on Analytics using Python**

**P. V. SUBRAMANIAN**

**Visiting faculty – Great Lakes Institute of Management**

*Copyright @ 2019 Great Lakes Institute of Management : All contents in this book, including (but not limited to) its contents are proprietary to Great Lakes Institute of Management, Chennai. Any effort at re-using, modifying and circulating both through Digital or Physical means, without the express written permission of the author, will amount to infringement of copyrights, and will be liable for suitable action.*

**Chapter 2. Basics of Python - continued**

**Table of contents**

Contents

[**1.** **Python Data Structures** 5](#_Toc7032051)

[**1.1** **Tuples** 5](#_Toc7032052)

[1.1.1 How do you create a Python tuple? 5](#_Toc7032053)

[1.1.2 How do you create an empty tuple? 6](#_Toc7032054)

[**1.1.3** How do you create a tuple with a single value? 6](#_Toc7032055)

[1.1.4 Tuple assignment 6](#_Toc7032056)

[1.1.5 Packing and unpacking 7](#_Toc7032057)

[1.1.6 How do you access tuple? 7](#_Toc7032058)

[1.1.7 Slicing a tuple 8](#_Toc7032059)

[1.1.8 How do you delete a tuple? 8](#_Toc7032060)

[1.1.9 Python tuple built-in functions 8](#_Toc7032061)

[1.1.10 Python tuple methods 9](#_Toc7032062)

[1.1.11 Tuple operations 10](#_Toc7032063)

[1.1.12 Iterations of tuples 10](#_Toc7032064)

[**1.2** **Lists** 11](#_Toc7032065)

[1.2.1 How do you create a Python LIST? 11](#_Toc7032066)

[1.2.2 How do you create an empty LIST? 11](#_Toc7032067)

[1.2.3 How do you create a LIST with a single value? 11](#_Toc7032068)

[1.2.4 LIST assignment 11](#_Toc7032069)

[1.2.5 List de-structuring 12](#_Toc7032070)

[1.2.6 Accessing a List 13](#_Toc7032071)

[1.2.7 Slicing a LIST 13](#_Toc7032072)

[1.2.8 list re-assignment 13](#_Toc7032073)

[1.2.9 Deleting a list 14](#_Toc7032074)

[1.2.10 LIST OPERATIONS 14](#_Toc7032075)

[1.2.11 Traversing a list 15](#_Toc7032076)

[1.2.12 List comprehension 15](#_Toc7032077)

[1.2.13 Built-in fuctions 16](#_Toc7032078)

[1.2.14 BUILT-IN METHODS 17](#_Toc7032079)

[**1.3** **Sets** 18](#_Toc7032080)

[1.3.1 How do you create an empty set? 18](#_Toc7032081)

[1.3.2 How do you change a set? 18](#_Toc7032082)

[1.3.3 set methods 19](#_Toc7032083)

[1.3.4 Set Operations 19](#_Toc7032084)

[**1.4** **Dictionaries** 20](#_Toc7032085)

[1.4.1 How do you create a dictionary? 20](#_Toc7032086)

[1.4.2 how do you access values in a dictionary? 21](#_Toc7032087)

[1.4.3 dictionary operations 21](#_Toc7032088)

[1.4.4 dictionary methods 22](#_Toc7032089)

[1.4.5 built-in dictionary functions 24](#_Toc7032090)

[**2.** **File handling continued** 25](#_Toc7032091)

[**1.1** **Access MySQL database using Python** 25](#_Toc7032092)

[1.4.6 Introduction to SQL 25](#_Toc7032093)

[1.4.7 DDL commands: 26](#_Toc7032094)

[1.4.8 DML commands: 28](#_Toc7032095)

[1.4.9 DCL commands: 29](#_Toc7032096)

[1.4.10 TCL commands: 29](#_Toc7032097)

[1.4.11 Python DB-API is the Python standard for database interfaces. 31](#_Toc7032098)

[1.4.12 How do I install MySQLdb? 31](#_Toc7032099)

[1.4.13 Steps to access MySQL through DB-API using MySQLdb 31](#_Toc7032100)

[**1.2** **How to access, create JSON files?** 48](#_Toc7032101)

# **Python Data Structures**

Data structure is a way of organizing and storing data such that we can access the data efficiently. Python provides the following built-in data structures:

1. Tuples
2. Lists
3. Sets
4. Dictionaries

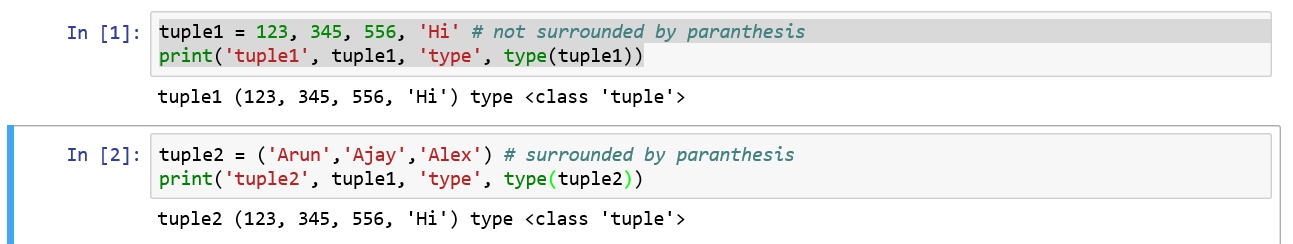
## **Tuples**

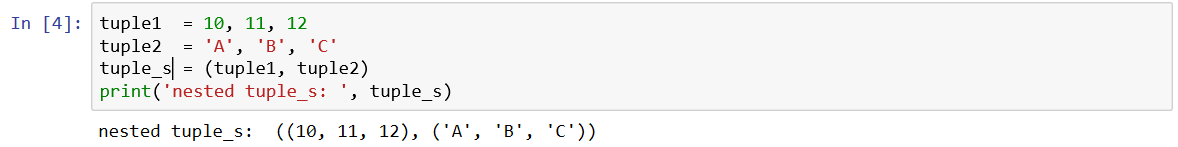
* Tuples hold a sequence of items and it is immutable (cannot be changed).
* A tuple consists of several values separated by commas with or without surrounding parenthesis.
* They hold different type of data(integer, float, list etc.)
* Tuples can be nested.

### How do you create a Python tuple?

You write a sequence of items separated by commas with or without parenthesis and assign it to a variable.

Example:





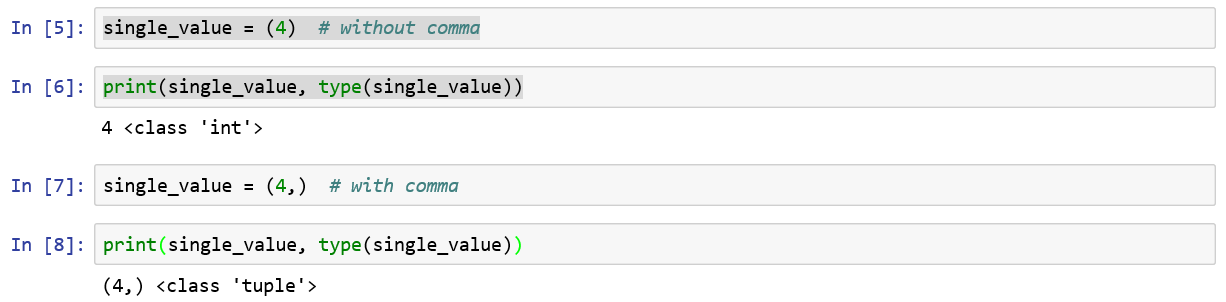
### How do you create an empty tuple?

To create an empty tuple, write as two parentheses containing nothing.



### How do you create a tuple with a single value?

To create a tuple with a single value, you need to include a comma, even though there is only one value. If you don’t include a comma, you will be creating an object of class, integer.



### Tuple assignment

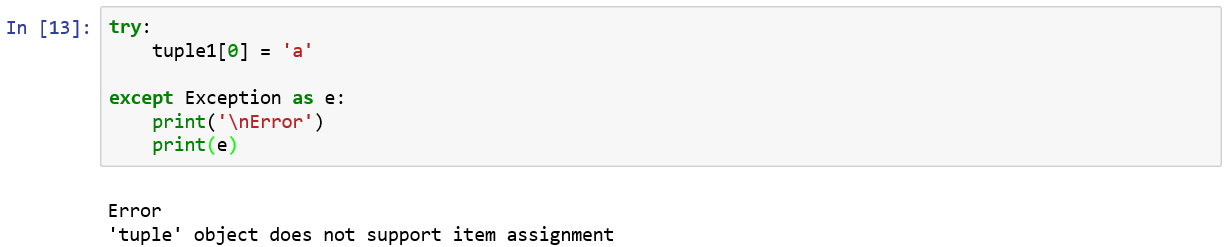
We can have a tuple of variables on the left hand side and a tuple of variables on the right hand side. Each value is assigned to the respective variable using tuple assignment. This eliminates the need of a temporary variable while swapping two variables.



**Tuples can be reassigned**

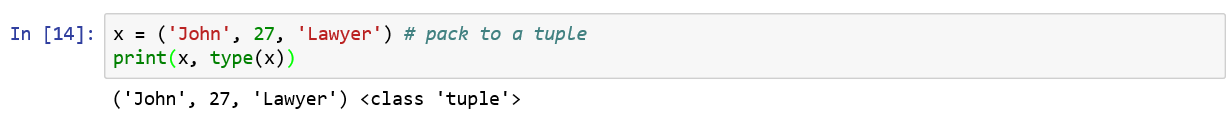
****

**Tuples are immutable meaning you cannot modify an element of a tuple.**

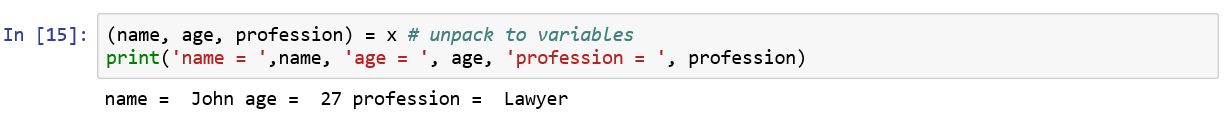
****

### Packing and unpacking

In packing, you place values into a tuple.

****

While unpacking we extract those values from the tuple to the variables.

****

### How do you access tuple?

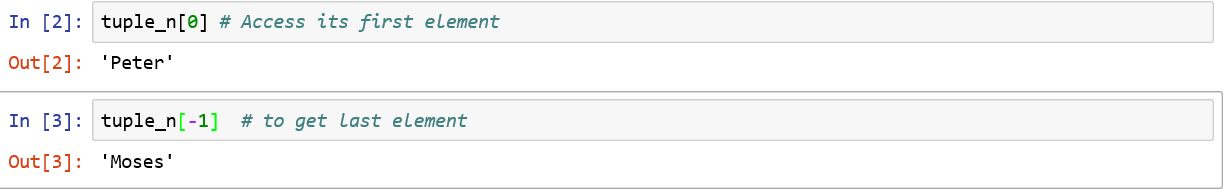
1. To access a tuple in Python, type its name.



1. To access a single item from a tuple, use its index in square bracket. Note that index begins at 0 for Python.

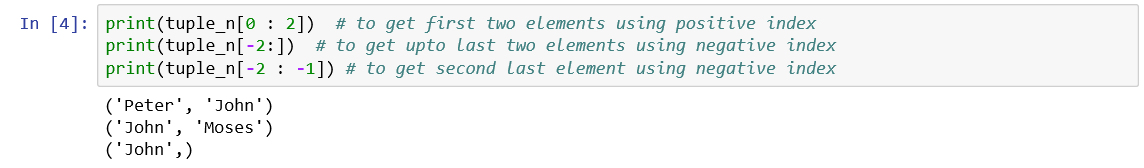
**Note:**

* 1. **Positive indices traverse from the left.**
  2. **Negative indices traverse from the right.**



### Slicing a tuple

Use slicing operator [] for getting a part of a tuple.

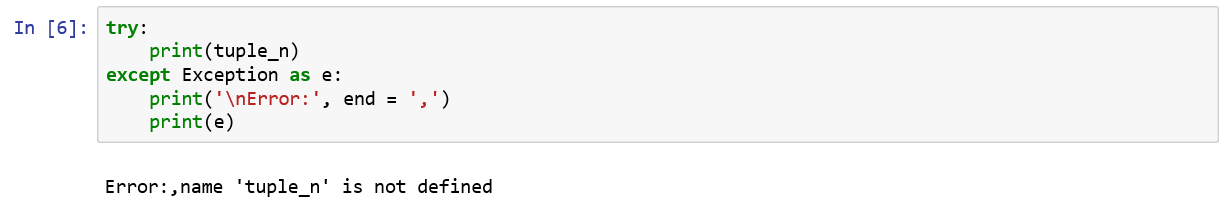
****

### How do you delete a tuple?

By using del() command, you can delete an entire tuple and it is not possible to delete any part of the tuple.

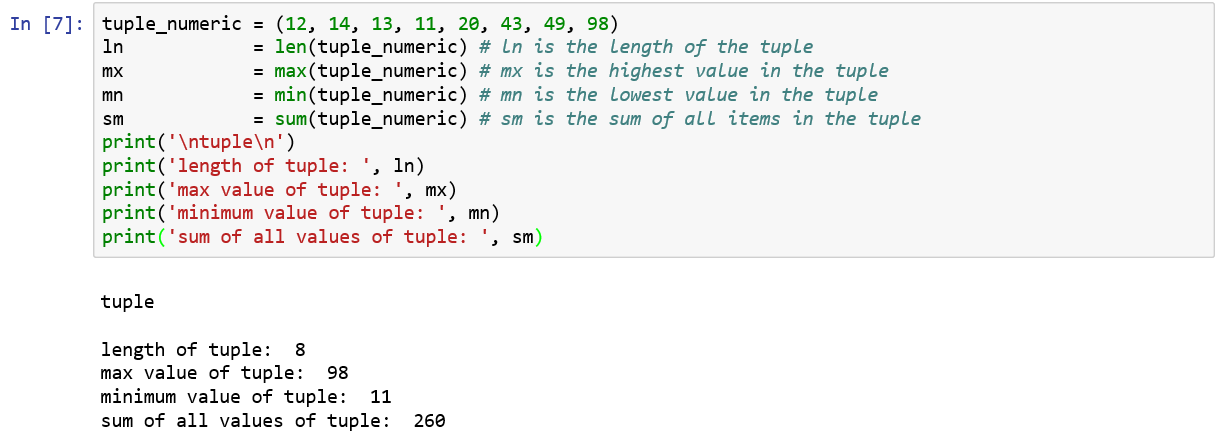


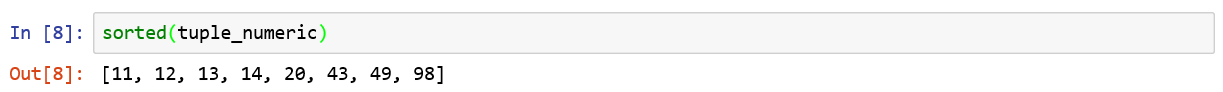
**How do you check, whether the tuple, tuple\_n is deleted or not?**

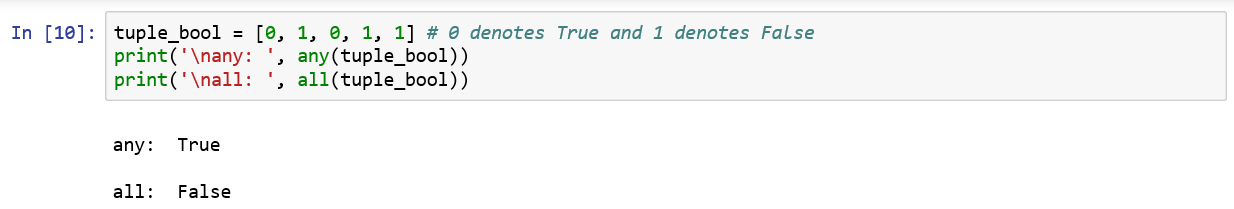


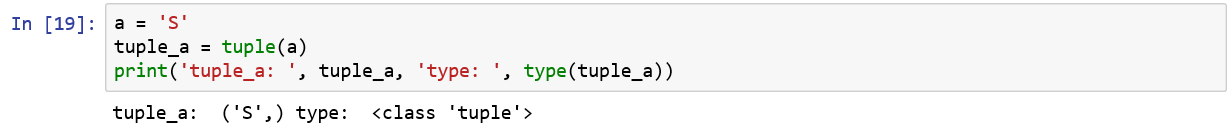
### Python tuple built-in functions

* len() returns the length of the tuple
* max() returns the highest value in the tuple
* min() returns the lowest value in the tuple
* sum() returns the sum of all items in the tuple
* sorted() returns the items in the tuple sorted in ascending order
* any() returns True if any one item in the tuple has a Boolean value of True.
* all() returns True only if all items in the tuple has a Boolean value of True.
* tuple() converts the given construct into a python tuple.



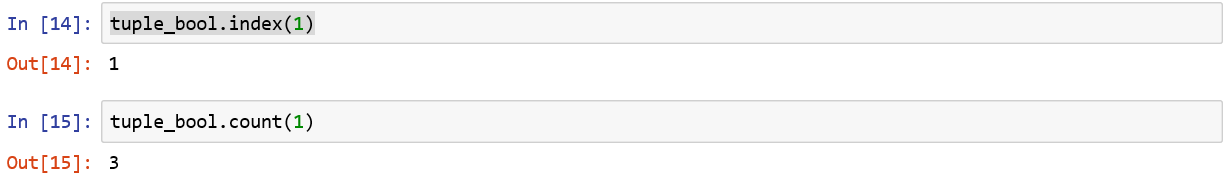






### Python tuple methods

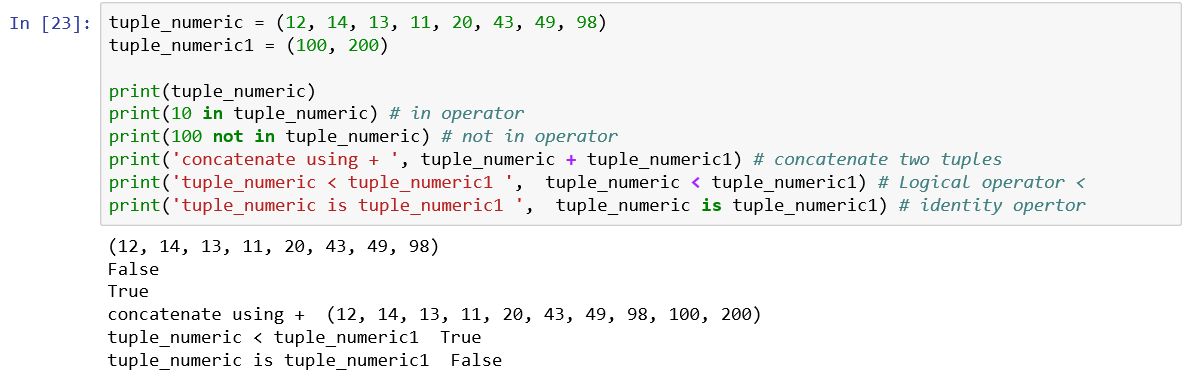
* index() returns the index of the first appearance of an item in a tuple.
* count() returns the number of times a given item appears in the tuple.



### Tuple operations

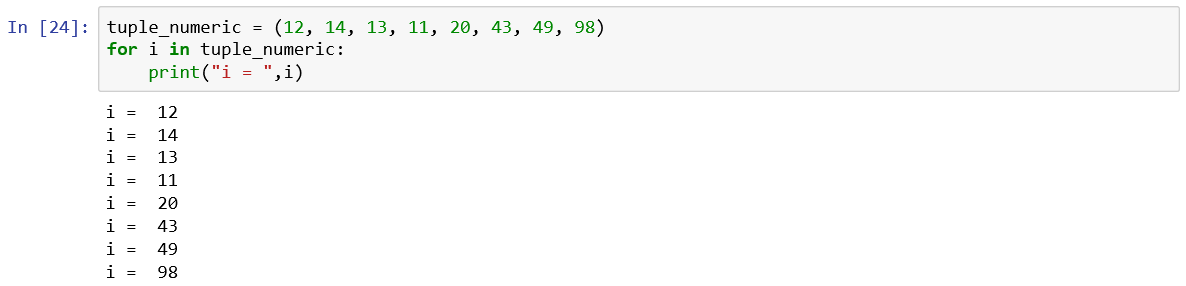
1. Membership informs us whether the given item appears in the tuple.
2. Concatenation joins two tuples using the operator, +.
3. Logical operators such as >, <, >=, ... can be applied on a tuple and it returns True or False.
4. Identity operator ***is*** returns True, if the two operands have the same identity whereas

***is not*** returns True if the two operands have the same identity.



### Iterations of tuples

You can iterate on a tuple as shown below:



**Refences:**

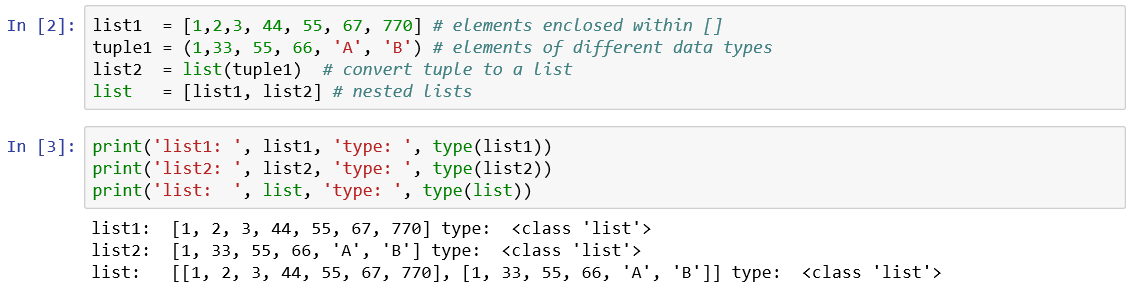
1. <https://data-flair.training/blogs/python-tuple/>

## **Lists**

* A list is an ordered set of values, where each value is identified by an index.
* The values comprising a list are called its elements.
* Lists can hold values of different data types.
* Lists, strings (holding ordered set of characters) - and other objects that behave like ordered sets are called sequences.
* To convert another sequence to a list, enclose within list(). For example. List(tuple1) will convert tuple, tuple1 to a list.

### How do you create a Python LIST?

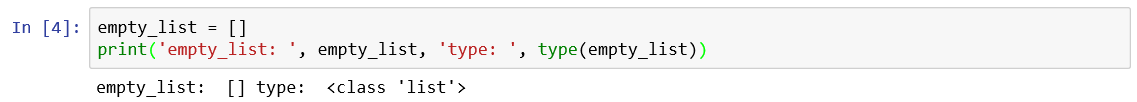
* To create a list, enclose the elements of the list in square brackets.



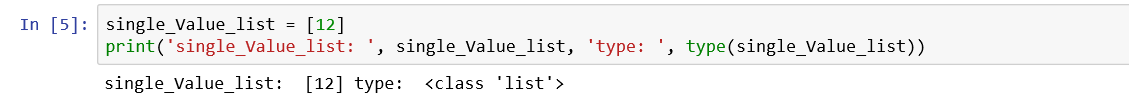
* Using a range() function, you can create a list of consecutive integers.

### How do you create an empty LIST?

To create an empty list, write as two square brackets, [] containing nothing.

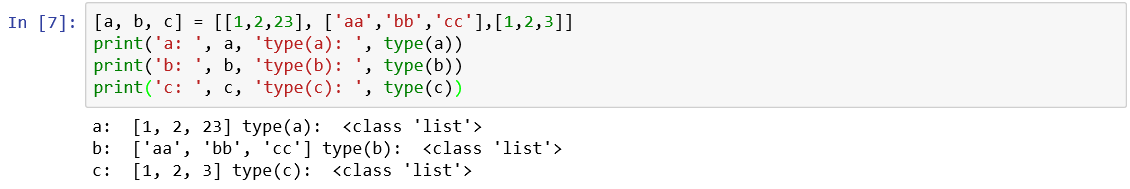


### How do you create a LIST with a single value?



### LIST assignment

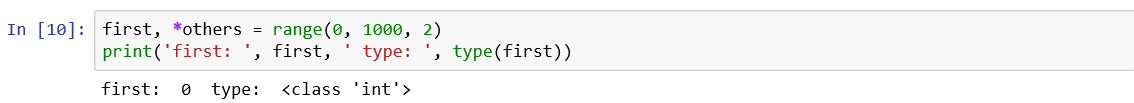
We can have a list of variables on the left hand side and a list of variables on the right hand side. Each value is assigned to the respective variable using list assignment. This eliminates the need of a temporary variable while swapping two variables.

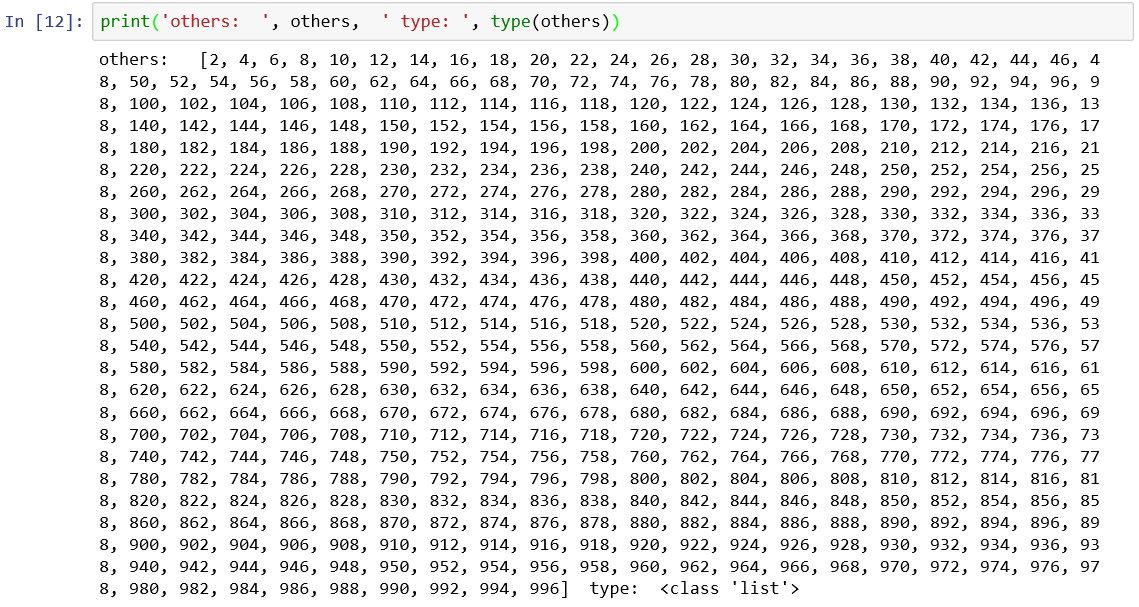
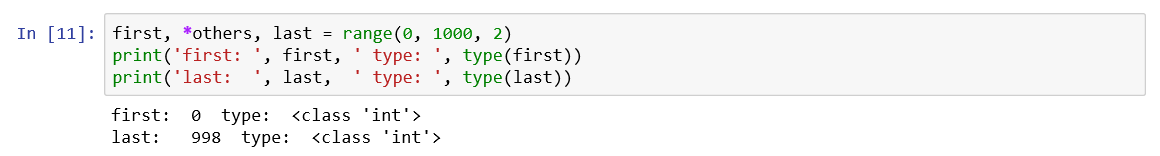


### List de-structuring

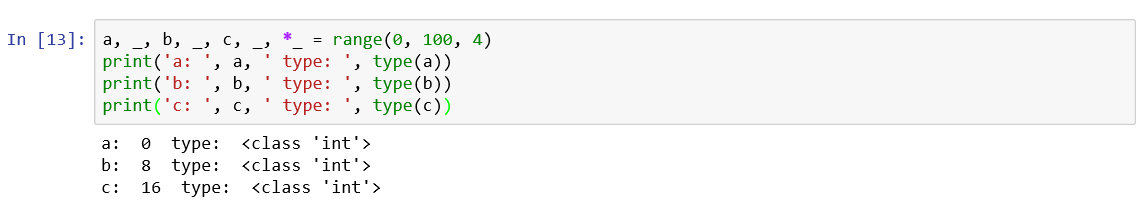
In assignments, you can split an iterable into values using the unpacking syntax.

Unpack a list of unknown length as shown below:





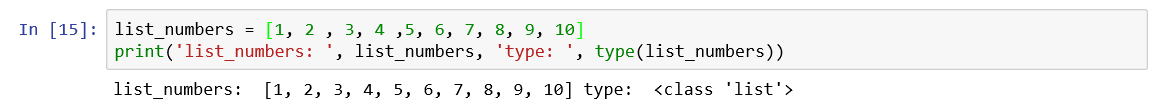
We have extracted the first and last elements and in-between values as list in the above example.

Ignore values, lists while de-structuring a list.

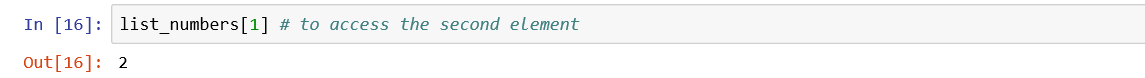
**Here, we have ignored alternative values and the sequence of values from the list formed by the range(). We seldom use \_ and \*\_ variables.**

### Accessing a List

1. To access a list in Python, type its name.



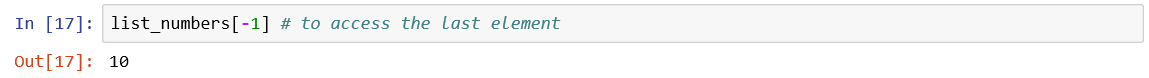
1. To access a single element, type its index.



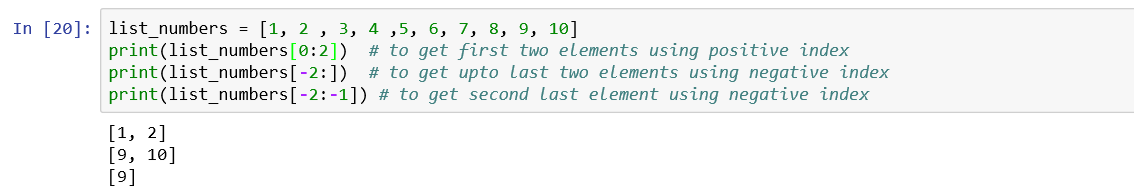
Note:

1. Positive indices traverse from the left.

2. Negative indices traverse from the right.



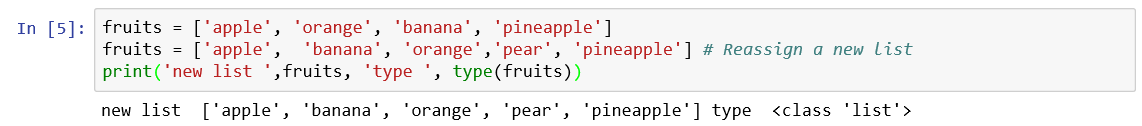
### Slicing a LIST



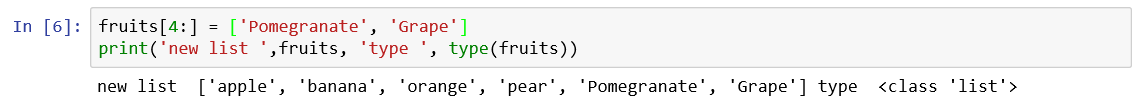
### list re-assignment

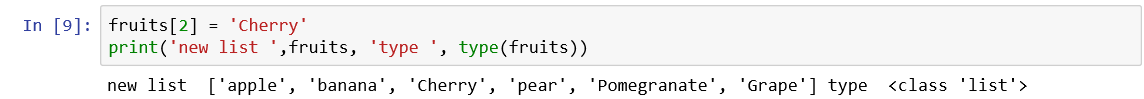
Python lists are mutable meaning that you can reassign it completely or part of it (items).

1. **Reassign a whole new list**



1. **Reassign a part of the list**

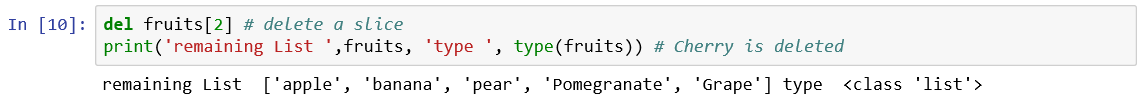




### Deleting a list

By using del() command, you can delete an entire list and it is possible to delete any part of the list.

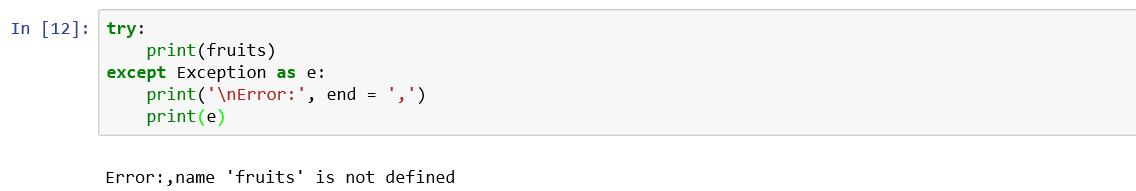
1. Deleting a slice of the list



1. Deleting a list



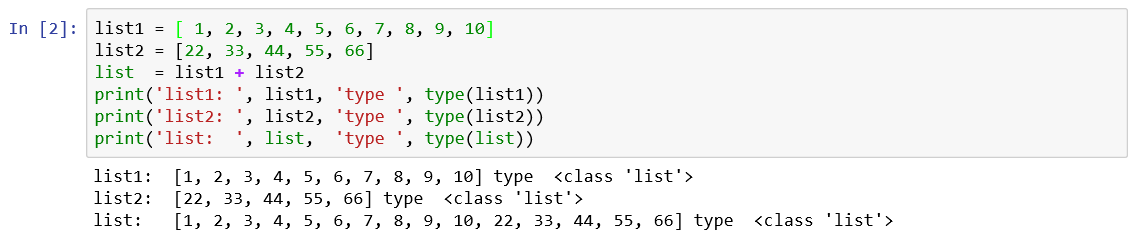
**How do you check, whether the list, fruits is deleted or not?**



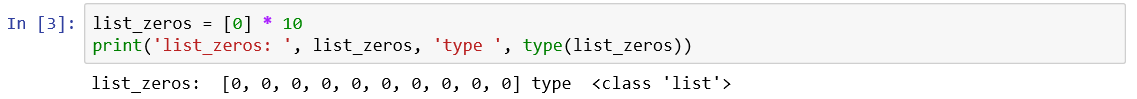
### LIST OPERATIONS

**List Operations**

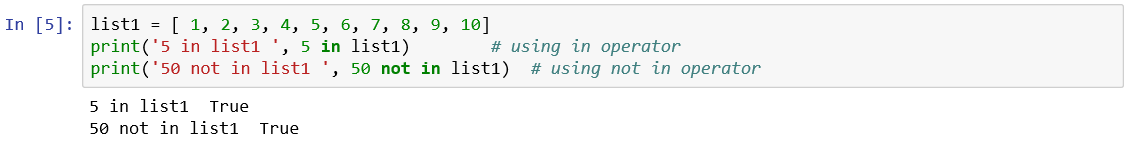
1. **Concatenation:** Using + operator, we concatenate lists.



1. Multiplication: Using \* operator, we repeat a given list a specified number of times.

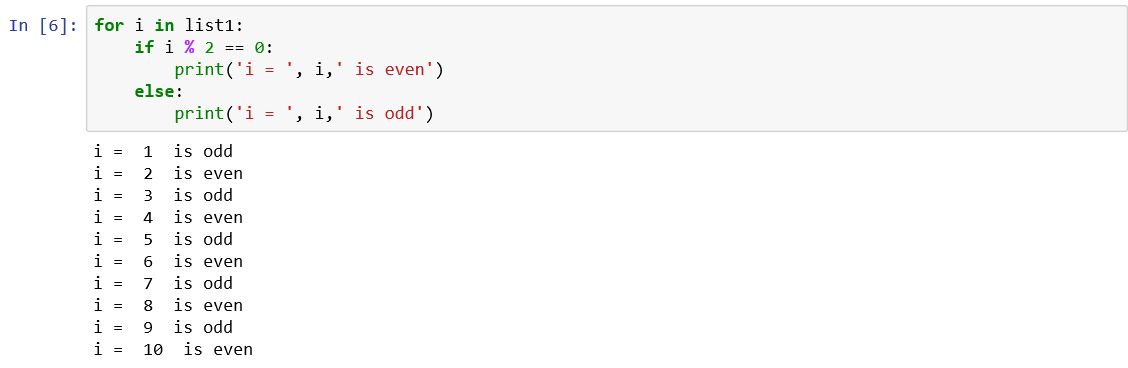


1. Membership operators: Using the operator, “in”, we can check if the specified element exists in the list or not. The operator, “not in”, checks if the specified element does not exist in the given list or not.



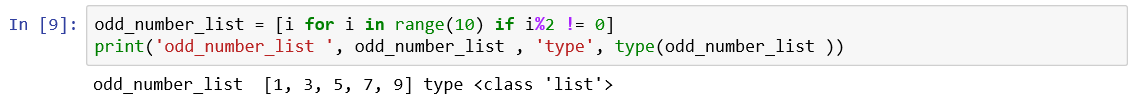
### Traversing a list

**You can iterate on a Python list using for loop.**

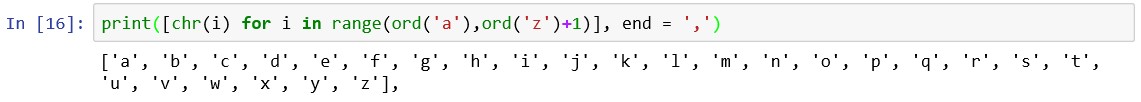


### List comprehension

**This is the most exciting feature of Python which creates lists by iterating over an iterable object.**

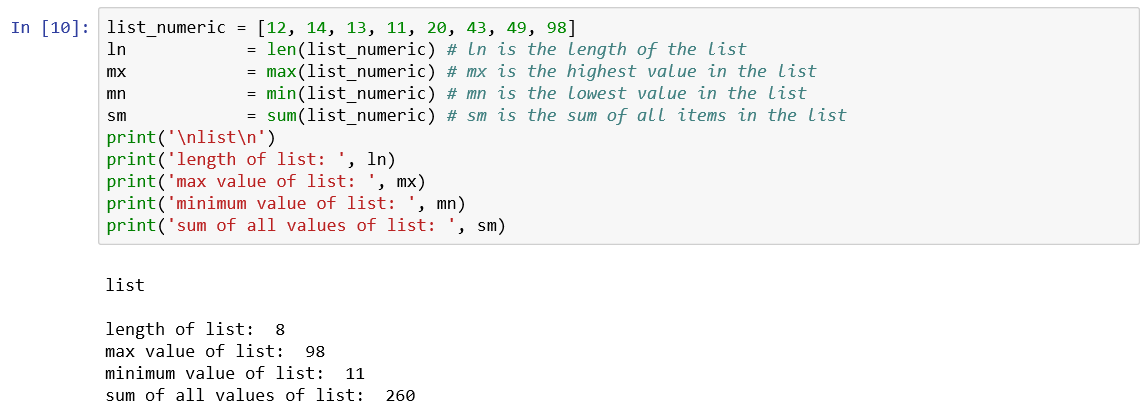


The ord() function which is inbuilt-in python, returns an integer representing the Unicode value of the character when the argument is a Unicode object.



### Built-in fuctions

* len() returns the length of the list
* max() returns the highest value in the list
* min() returns the lowest value in the list
* sum() returns the sum of all items in the list`
* sorted() returns the items in the list sorted in ascending order
* any() returns True if any one item in the list has a Boolean value of True.
* all() returns True only if all items in the list has a Boolean value of True.
* list() converts a different data type to a list

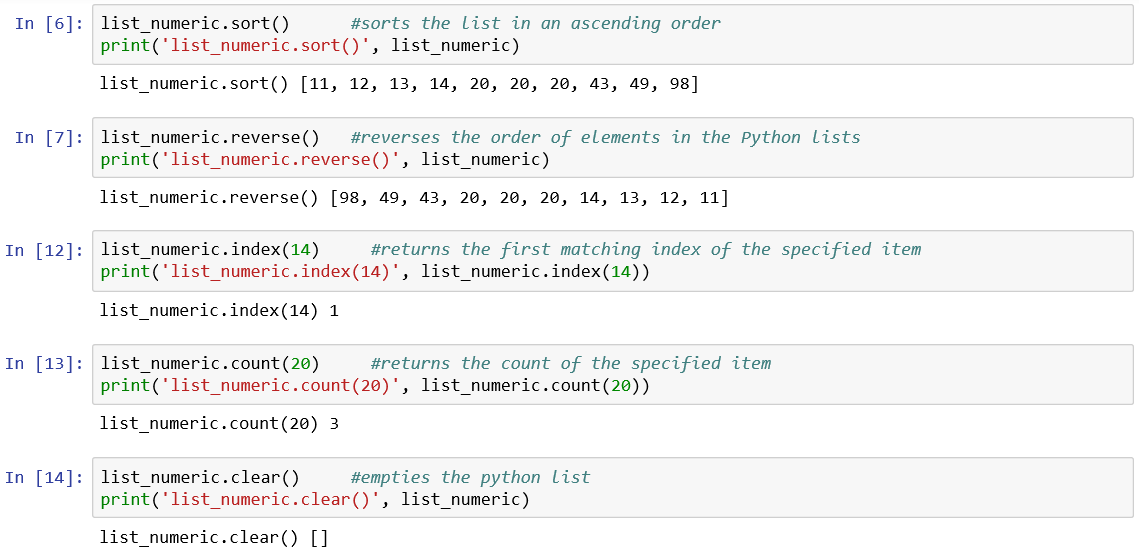
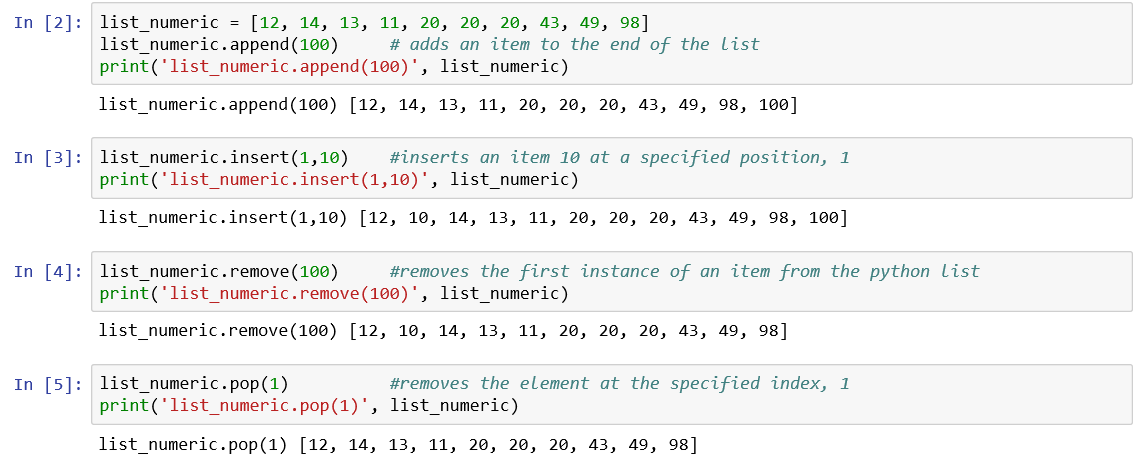






### BUILT-IN METHODS

* append() adds an item to the end of the list
* insert() inserts an item at a specified position
* remove() removes the first instance of an item from the python list
* pop() removes the element at the specified index, and prints it to the screen
* sort() sorts the list in an ascending order
* reverse() reverses the order of elements in the Python lists
* index() returns the first matching index of the specified item
* count() returns the count of the specified item
* clear() empties the python list

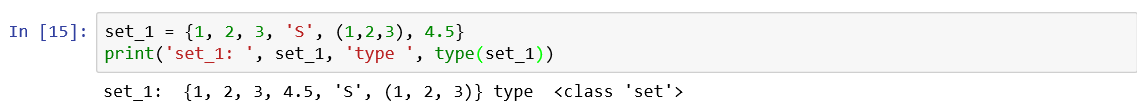


***Note: The reverse() method, has not changed the original list, list\_numeric but only reversed the order in its copy.***

*Refer: https://www.geeksforgeeks.org/nested-list-comprehensions-in-python/*

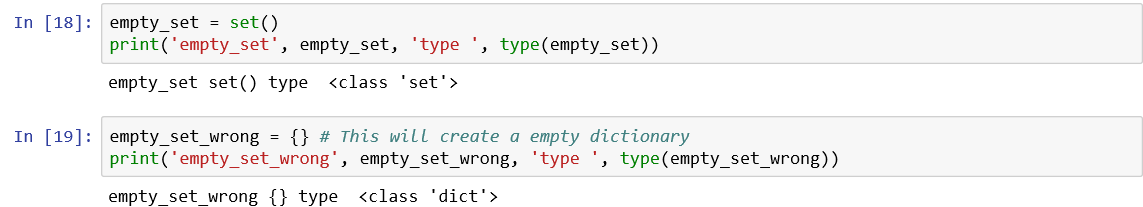
## **Sets**

* A set is an ordered collection of items and it is created by placing it inside {}, curly brackets, separated by commas or by using the built-in function, set().
* Set is mutable but you can add or remove items from it.
* Sets may contain elements of different data types. It can contain a tuple but not list or dictionary which is mutable.



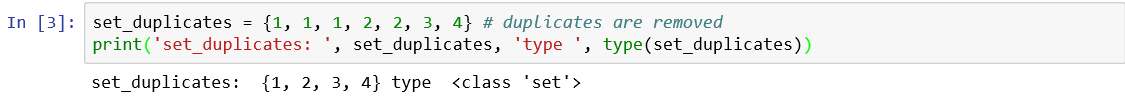
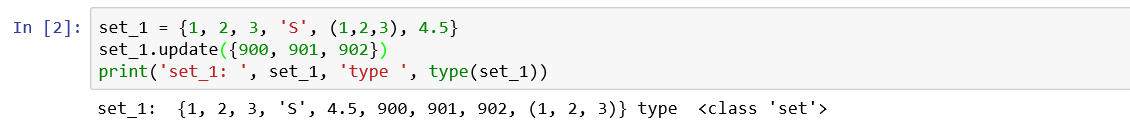
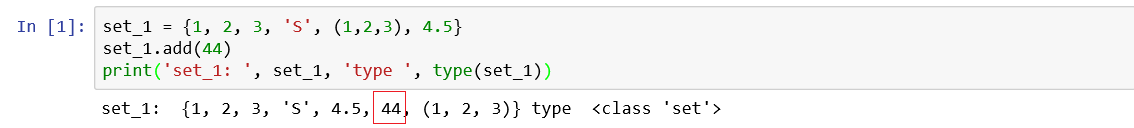
### How do you create an empty set?

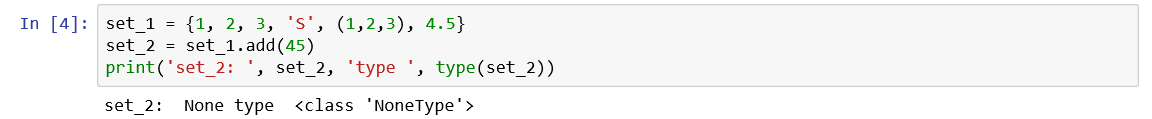
Use the set() function to create an empty set. Using curly braces will create an empty dictionary.



### How do you change a set?

* Sets are mutable and we cannot access or change an element of set using indexing or slicing.
* However, we can add a single element using the add() method and multiple elements using the update() method. If the element is already present, it doesn't add any element.



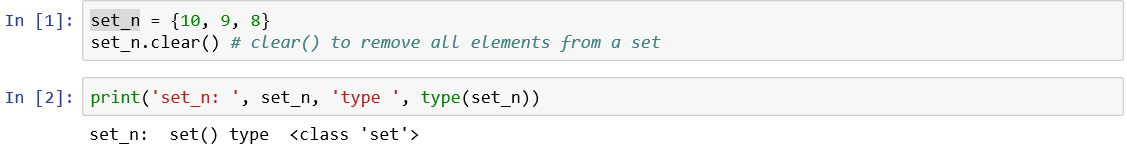


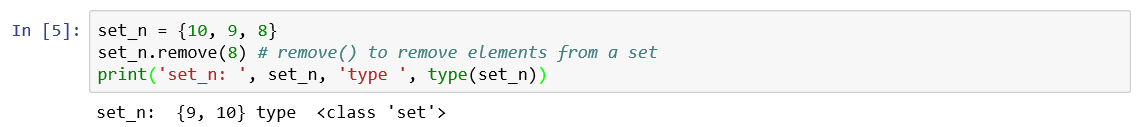
### set methods

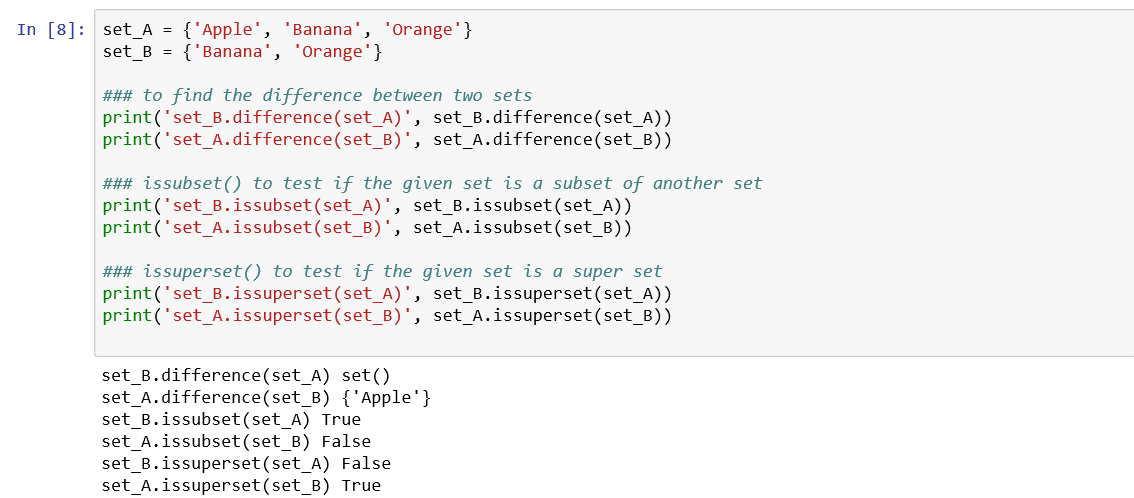
* add() to add elements to a list < as seen earlier>
* clear() to remove all elements from a set
* remove() to remove elements from a set
* issubset() to test if the given set is a subset of another set
* issuperset() to test if the given set is a super set

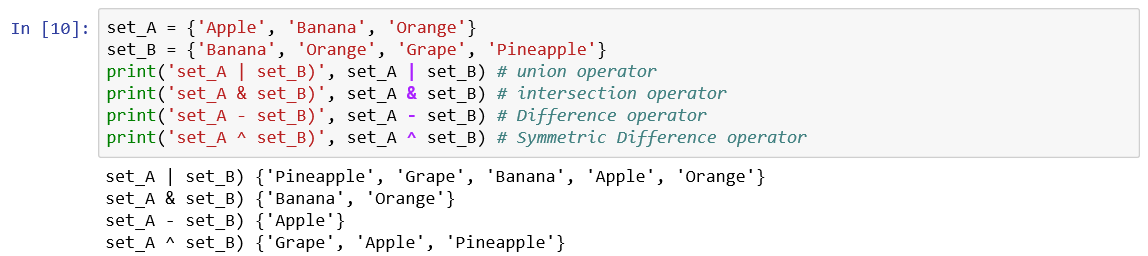
### Set Operations

* Union operation on two sets produces a new set containing all the distinct elements from both the sets by using | operator.
* Intersection operation on two sets produces a new set containing only common elements from both the sets by using & operator.
* Difference operation on two sets produces a new set containing only the elements from the first set and none from the second set by using - operator.
* Symmetric difference of two sets produces a new set of elements which exist in first set and second set but not in both. By using the caret operator (^), we can execute this.









References:

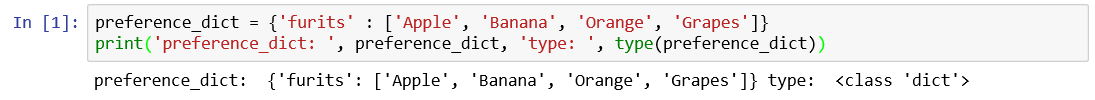
* [*https://www.bing.com/search?q=set+in+python&form=EDNTHT&mkt=en-in&httpsmsn=1&plvar=0&refig=9c5da964954941cfbded433cbd2e46ab&sp=-1&pq=set+in+python&sc=7-13&qs=n&sk=&cvid=9c5da964954941cfbded433cbd2e46ab*](https://www.bing.com/search?q=set+in+python&form=EDNTHT&mkt=en-in&httpsmsn=1&plvar=0&refig=9c5da964954941cfbded433cbd2e46ab&sp=-1&pq=set+in+python&sc=7-13&qs=n&sk=&cvid=9c5da964954941cfbded433cbd2e46ab)
* [*https://www.geeksforgeeks.org/sets-in-python/*](https://www.geeksforgeeks.org/sets-in-python/)

## **Dictionaries**

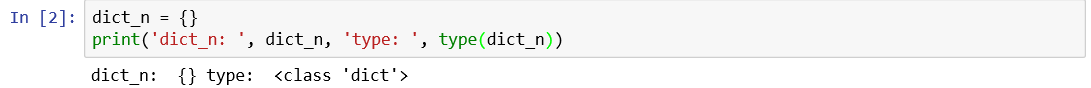
* Dictionary is an unordered collection of data values used for storing key : value pairs and maps from keys to values. The compound types of collection of data such as tuple and lists use integers as indices.
* Each key-value pair in a dictionary is separated by a colon (: ), whereas each key is separated by a comma.

### How do you create a dictionary?

You create a dictionary by enclosing a list of key : value pair within curly braces {}.

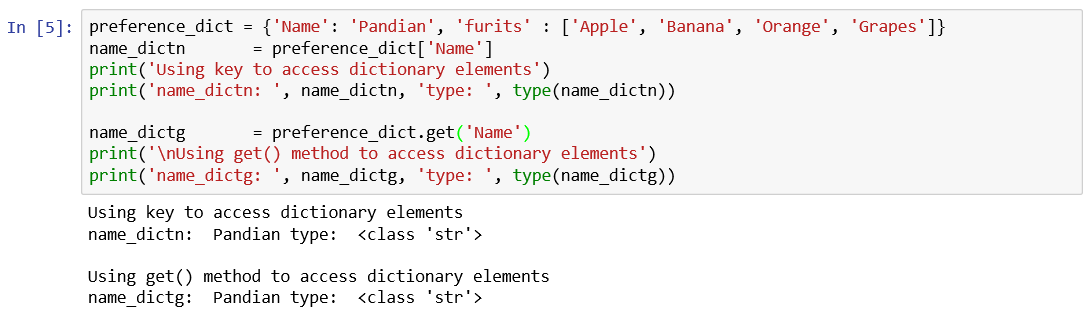


You can create an empty dictionary by typing mere {}



### how do you access values in a dictionary?

* To access dictionary elements, use square bracket along with the key to get its values.
* You can use the get() method to access dictionary elements.

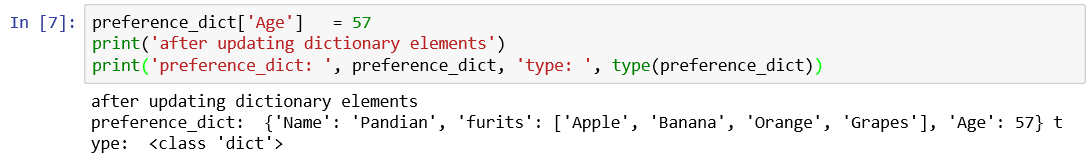


### dictionary operations

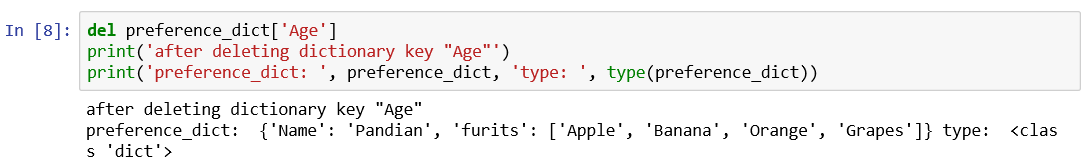
* You **add** elements to the dictionary by adding a new entry or a key : value pair.



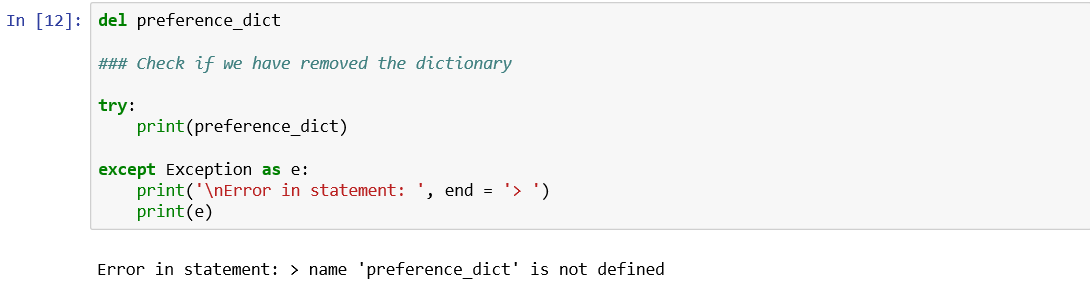
* You **modify** elements of the dictionary by assigning a new value to a key.



* You **delete** elements of the dictionary by using the del() for the key.

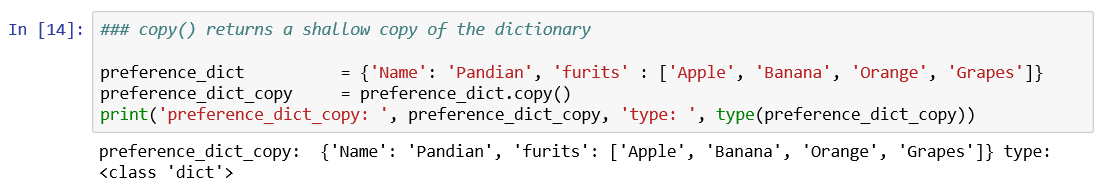
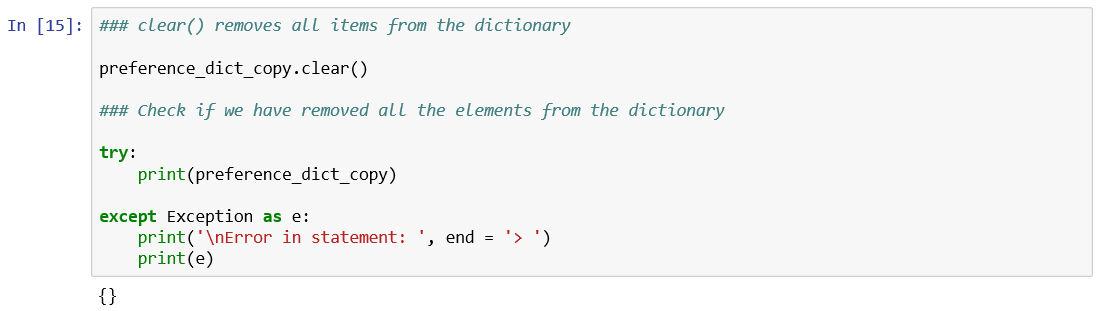


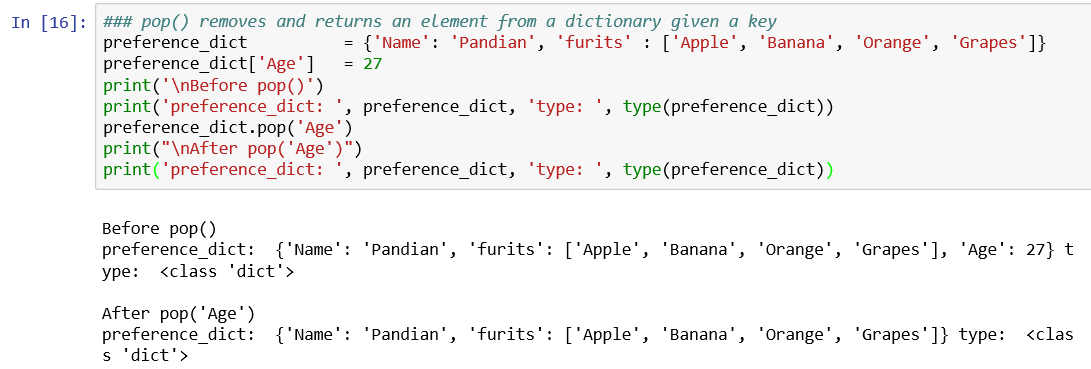
* You can **delete the entire dictionary** by using del() method.

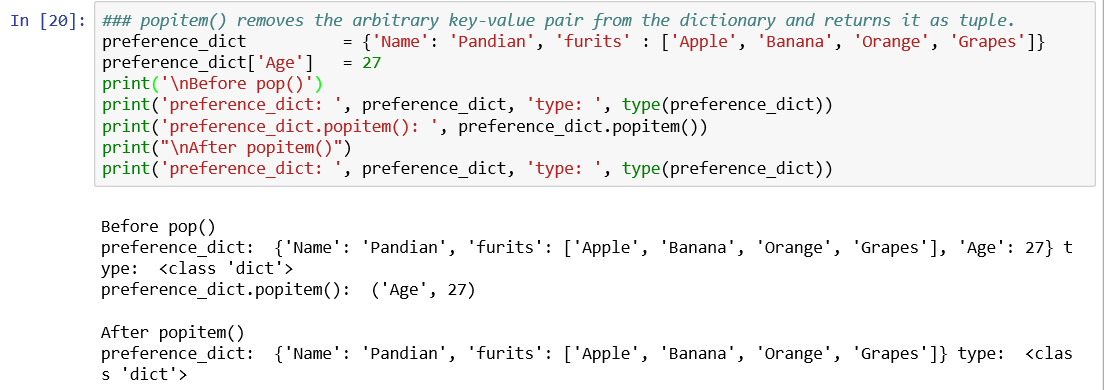


### dictionary methods

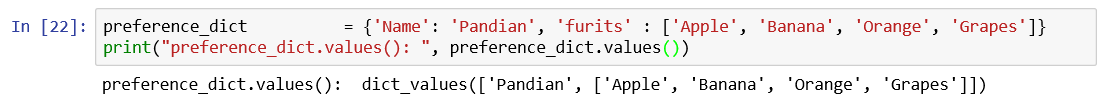
* copy() returns a shallow copy of the dictionary
* clear() removes all items from the dictionary
* pop() removes and returns an element from a dictionary given a key
* popitem() removes the arbitrary key-value pair from the dictionary and returns it as tuple.
* get is used to access a value of a given key
* <dictionary>.values() returns a list of all the values available in a given dictionary
* keys() returns a list of dictionary keys
* items() returns a list of key : value tuple pairs
* in <dictionary>.keys() returns true if key is found in the dictionary; false otherwise
* fromkeys() creates a new dictionary with keys from seq and values set to value. fromkeys() will help us to generate a dictionary from the given keys.

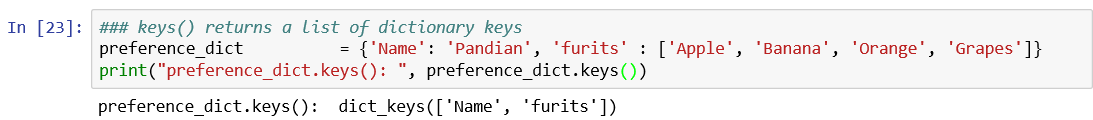
 

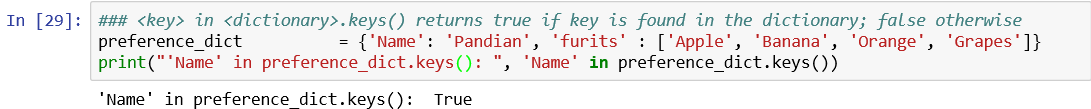
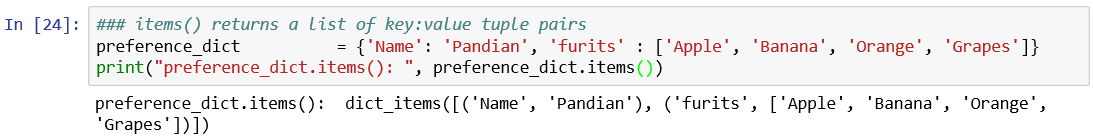


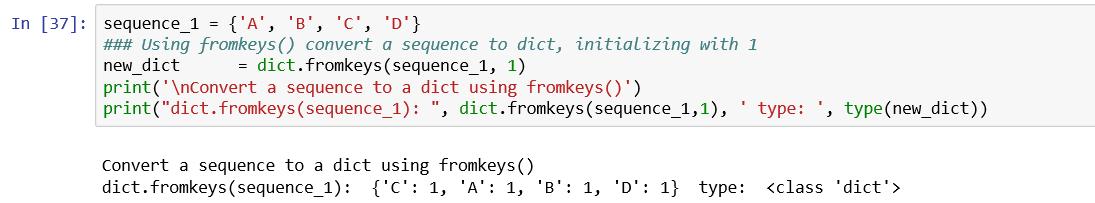






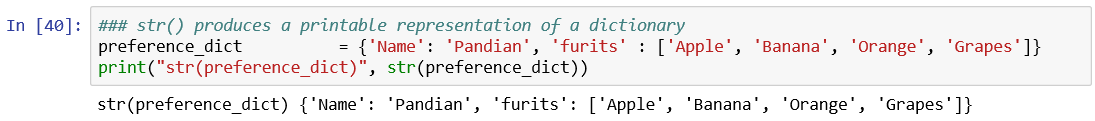




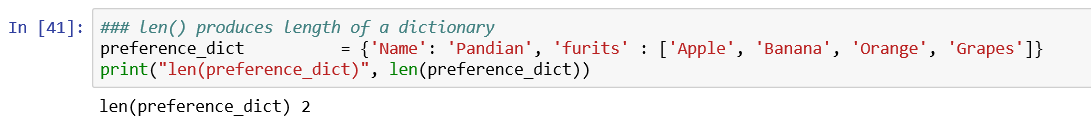


### built-in dictionary functions

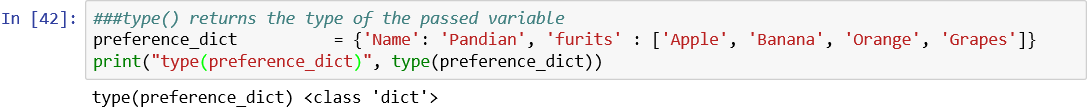
* str() produces a printable representation of a dictionary



* len() produces a printable representation of a dictionary



* type() returns the type of the passed variable



Refer:

1. https://www.geeksforgeeks.org/python-dictionary/
2. https://www.tutorialspoint.com/python/python\_dictionary.htm
3. https://docs.python.org/2/tutorial/datastructures.html
4. https://data-flair.training/blogs/python-data-structures-tutorial/

# **File handling continued**

In Chapter 1, we have seen how to handle text files using Python. There are occasions, when you need to access databases like MySQL or Oracle etc. using Python. We also note that JSON (JavaScript Object Notation) is often used for data interchanging and used in web services because it is lightweight.

We shall learn how to

1. access MySQL database from Python
2. access, create JSON files

*https://stackoverflow.com/questions/383692/what-is-json-and-why-would-i-use-it*

*http://www.json.org/*

*http://www.mysqltutorial.org/python-mysql/*

## **Access MySQL database using Python**

All most all companies store their data in large databases to retrieve & process to cater to their information needs of their clients. Databases are essentially a collection of tables and connected with each other through columns.

These database systems support Structured Query Language (SQL), which is used to create, access and use the relationships between the stored data. Python language has very powerful features for handling database.

Get friendly with a MySQL Data Base Administrator or a MySQL programmer and install MySQL database and phpMyAdmin, which is an open source free software, designed to handle the administration and management of MySQL databases through a graphic user interface in your PC.

### Introduction to SQL

Structured Query Language (SQL) is the database language used for creating, accessing, manipulating databases.

There are four categories of SQL:

1. Data Definition Language or **DDL**
2. Data Manipulation Language or **DML**
3. Data Control Language or **DCL**
4. Transaction Control Language or **TCL**

### DDL commands:

1. **CREATE** is used to create the database or a table or index etc.
2. **DROP** is used to delete objects such as database, table, index etc. from the database.
3. **ALTER** is used to alter the structure of the database objects such as table.
4. **TRUNCATE** is used to remove all records from a table, including all spaces allocated for the records are removed.
5. **COMMENT** is used to add comments to the data dictionary.
6. **RENAME** is used to rename an existing object in the database.

We shall use only CREATE command in this chapter. To know more about DDL, DML, DCL and TCL commands, please refer to:

<https://www.geeksforgeeks.org/sql-ddl-dml-dcl-tcl-commands/>

<https://www.edureka.co/blog/mysql-tutorial/>

**How do you create a database, test?**

The CREATE SCHEMA statement is used to create a database.

Syntax:

CREATE SCHEMA database\_Name.

Example: CREATE SCHEMA test;

**How do you create a table, STUDENTS in the data base, test?**

The CREATE TABLE statement is used to create a table.

Syntax:

CREATE TABLE table\_name (

column1 datatype (size),

column2 datatype (size),

column3 datatype (size),

...

column datatype

)

Example:

Use test;

CREATE TABLE STUDENTS

(

ID int(4),

NAME varchar(255),

ADDRESS varchar(255)

);

*The column, ID is of type integer and can store an integer number of size 4. The next two columns, NAME and ADDRESs are of type varchar and can store characters and size 255 specifies that these two fields can fold a maximum of 255 characters.*

### DML commands:

1. **USE**  is used to specify the database to be used to perform all the operations.
2. **INSERT** is used to insert new records in an existing table.
3. **UPDATE** is used to modify the existing records in a table.
4. **SELECT** is used to select data from a table of a specified database and the data returned is stored in a result table, called the result set.
5. **DELETE** is used to delete existing records in a table.

**How do you specify which data base you are going to use for creating a new table or extracting information from database tables, etc.?**

Answer is by using the **USE** command.

**USE**

Syntax:

USE database\_name;

Example:

Use test;

**How do you insert records into a data base table of a specified database?**

Answer is by using the **INSERT** command.

**INSERT**

Syntax:

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...)

Example:

INSERT INTO STUDENTS (ID, NAME, ADDRESS) VALUES (1235, 'John William', '12 Sarangapani Road, T. Nagar, Chennai 600 017');

**UPDATE**

**How do you carry out corrections to an existing table in the database?**

Answer: By using the UPDATE statement, we can modify the existing records of a table.

Syntax:

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

Example:

UPDATE STUDENTS

SET ADDRESS = '121 Dr Radhakrishnan Salai, Mylapore Chennai 600004'

WHERE ID = 1235;

**DELETE**

**How do you delete record with ID = 1234 from STUDENTS, an existing table in the database?**

Answer: By using the DELETE statement, we can delete records from an existing table.

Syntax:

DELETE FROM table\_name

WHERE condition;

Example:

DELETE FROM STUDENTS

WHERE ID = 1234;

### DCL commands:

Data base administrators grant and revoke data base access privileges for each user based on the role played by the user.

GRANT command is used to provide user's access privileges for the database objects based on user role.

REVOKE command is used to withdraw user's access privileges for the database objects based on user role.

We are not covering these portions in this chapter.

### TCL commands:

1. **COMMIT**  is used to save all the transactions to the database since the last COMMIT or ROLLBACK command.
2. **ROLLBACK** is used to UNDO all the transactions to the database since the last COMMIT or ROLLBACK command.
3. **SAVE POINT** is used to create points within the groups of transactions in which to rollback.
4. **SET TRANSACTION** is used to name a transaction and to take actions at a group level. Certain transactions are committed or rolled back at a transaction level for example, credit and debit entries. For a voucher entry, we need both debit and credit entries. So either both credit and debit entries are committed or rolled back.

*We will focus on both COMMIT and ROLLBACK.*

**COMMIT**

Syntax:

COMMIT;

Example;

DELETE FROM STUDENTS

WHERE ID = 1234;

COMMIT;

**ROLLBACK**

This is very useful, when you realize there is a mistake in the query of the SQL statement recently executed but not committed.

Syntax:

ROLLBACK;

Example;

DELETE FROM STUDENTS

WHERE ID = 1235;

ROLLBACK;

*There are other functions for doing manipulation such as a) LOGICAL OPERATORS such as AND, OR.*

*The AND operator is used to filter records that rely on more than one condition. The conditions separated by AND operator displays the records which satisfy all the conditions and give the output TRUE.*

*The OR operator is used to filter records that rely on more than one condition. The conditions separated by OR operator displays the records which satisfy any of the conditions and give the output TRUE.*

*b) AGGREGATE FUNCTIONS such as MIN(), MAX(), SUM(), AVG() and COUNT()*

*We shall use COUNT() function in this chapter which returns the number of rows that matched the*

*specified criteria.*

### Python DB-API is the Python standard for database interfaces.

Python Database API supports a wide range of database servers such as −

* MySQL
* PostgreSQL
* Microsoft SQL Server 2000
* Oracle

**Can I use just one common Database API to access all the above databases?**

**Answer** is a big NO, at this point in time. You need to download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

DB-API is one module that provides a data base application programming interface. To use DB-API for writing Python scripts, use MySQLdb, a driver for MySQL.

http://www.kitebird.com/articles/pydbapi.htmldevelopment.

### How do I install MySQLdb?

Type the following command at Anaconda command prompt with administrator privilege.

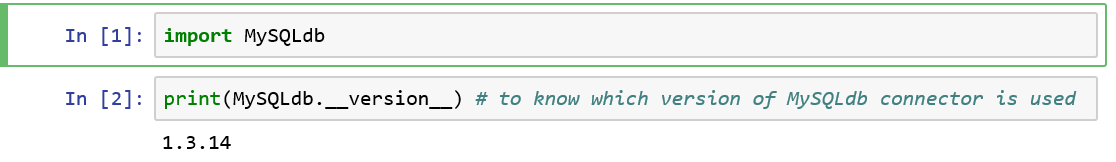
pip install MySQL-python

### Steps to access MySQL through DB-API using MySQLdb

You need to perform the following steps to access MySQL through DB-API using MySQLdb.

1. Import the MySQLdb module
2. Open a connection to the MySQL server
3. Issue SQL statements and retrieve the results
4. Close the server connection

**Step 1: Import the MySQLdb module**



The import statement tells Python that to load the stated module to meet our needs to access the functions in the module (such as MySQLdb).

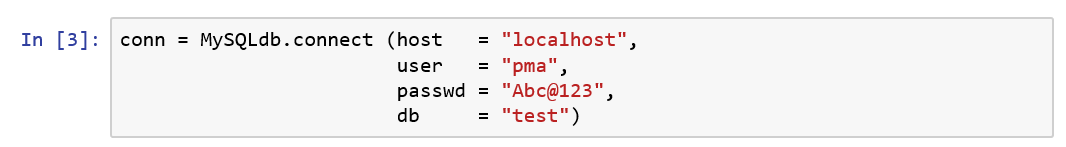
This must be the first statement for accessing MySQL database and precede any attempt to connect to the MySQL server. We observe that we are using version 1.3.14 of MySQLdb module.

**Step 2: Open a connection to the MySQL server**

Having the made available the module MySQLdb for our use in python environment, we need to establish the connection by invoking the connect() method of the MySQLdb driver with appropriate connection parameters such as the hostname where the server is running, the username and password for your MySQL account, and the name of the database you intend to use.

Please note the connect() function list syntax varies among MySQLdb drivers. For MySQLdb, the arguments are allowed to be given in name = value pair format.

The following code makes a connection to the MySQL server on the **local host** to access the **test** database with username **pma** and password, Abc@123.



Usually the database administrator creates a database, users and passwords for the database users and specify the location of the database which you pass through the host = command.

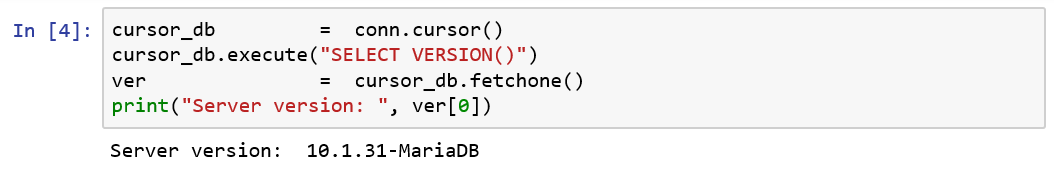
Since the connect() call has succeeded, a connection object, conn is returned which is used for processing out SQL statements.

**Step 3: Issue SQL statements and retrieve the results**

**How do I get the MySQL server version using Python?**

We create a cursor object for processing statements by using the command cursor() method.

We make a cursor, cursor\_db to issue a SELECT VERSION() statement which returns a string containing server version information as shown below:

****

* The cursor object's execute() method sends the statement to the server.
* The method fetchone() retrieves a row as a tuple which contains a single value.
* We print this value which holds the MySQL server version number.

MySQL server version is 10.1.31- MariaDB

**Note:**

MariaDB is a community-developed, commercially supported fork of the MySQL relational database management system, intended to remain free and open-source software under the GNU GPL**. Latest stable release is 10.3.14 (April 2, 2019).**

For more details, please refer to https://en.wikipedia.org/wiki/MariaDB

**Let us create following tables in MySQL database, test.**

1. EMPLOYEE containing the following fields and the data types:

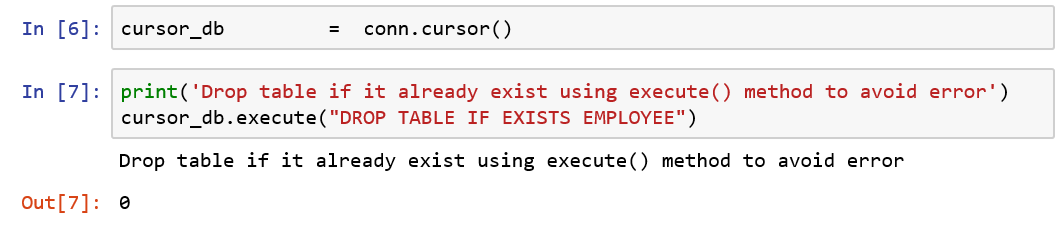
|  |  |  |  |
| --- | --- | --- | --- |
| # | Column name | Data type | Description |
| 1 | EMPID | CHAR(5) NOT NULL | A string column of length 5 which cannot be null or empty |
| 2 | FIRST\_NAME | CHAR(30) NOT NULL | A string column of length 30 which cannot be null or empty |
| 3 | LAST\_NAME | CHAR(30) | A string column of length 30 which can be null or empty |
| 4 | AGE | INT(2) | two digit integer |
| 5 | SALARY | FLOAT(8,2) | A floating point number containing 8 digits as total length out of which two digits are reserved after decimal point |

-.

1. LEAVES containing the following fields and the data types:

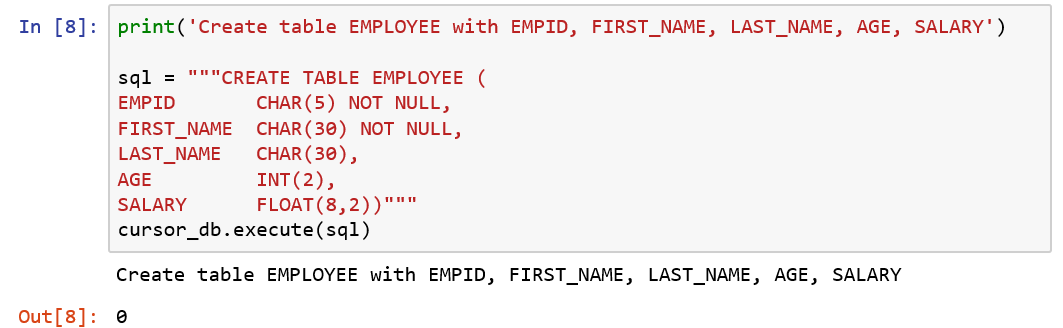
|  |  |  |  |
| --- | --- | --- | --- |
| # | Column name | Data type | Description |
| 1 | EMPID | CHAR(5) NOT NULL | A string column of length 5 which cannot be null or empty |
| 2 | LEAVE\_DAYS | INT(5, 2) | A floating point number containing 5 digits as total length out of which two digits are reserved after decimal point |

* Before creating the table, check if the table already exists or not. If the table already exists, drop the table. You can do so by using execute() method of cursor object.

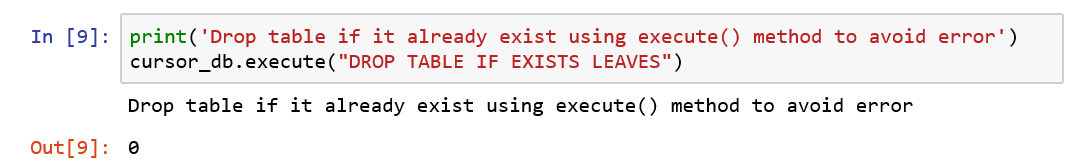


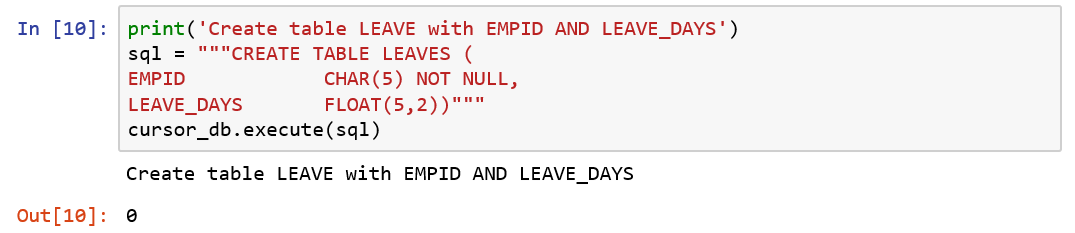
The output 0 indicates success of the SQL statement execution.

* Using the execute() method of cursor object, you can create table EMPLOYEE as shown below:

****

Repeat the above two statements for LEAVES table creation.

****

****

**INSERT**

**Having created two tables, EMPLOYEE and LEAVES, let us insert data into them.**

1. **Insert a single record into the table, EMPLOYEE**

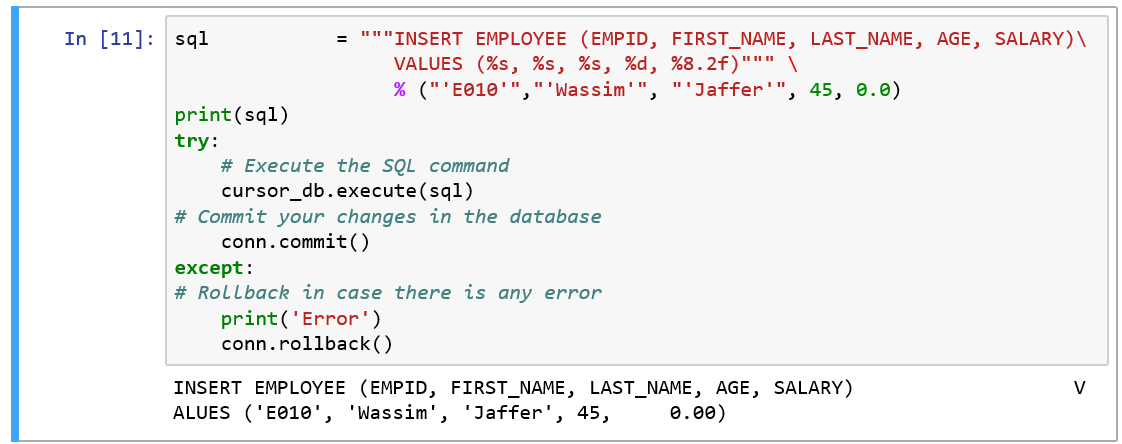
We have already established a database connection. Let us insert a single record into the table, EMPLOYEE.

**EMPLOYEE table values**

|  |  |  |
| --- | --- | --- |
| **#** | **Column** | **Values** |
| 1 | EMPID | [‘E010’] |
| 2 | FIRST\_NAME | [‘Wassim’] |
| 3 | LAST\_NAME | [‘Jaffer’] |
| 4 | AGE | [45] |
| 5 | SALARY | [0.00] |

INSERT EMPLOYEE (EMPID, FIRST\_NAME, LAST\_NAME, AGE, SALARY)

VALUES (‘E010’, ‘Wassim’, ‘Jaffer’, 45,0.0)

****

1. **Insert a bulk of records into the table, EMPLOYEE**

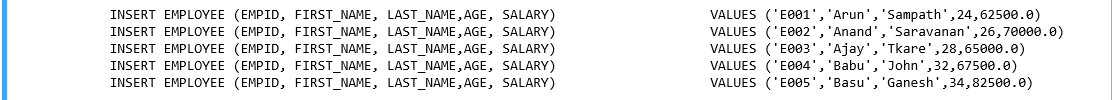
We had seen how to insert a single record into a database table. Life is not that simple!

In real life situations, we have to insert a bulk of records and not just one! Let us introduce a bulk of records by building a dictionary to hold the relevant attributes of an EMPLOYEE and get the length of the dictionary.

**EMPLOYEE table values**

|  |  |  |
| --- | --- | --- |
| **#** | **Column** | **Values** |
| 1 | EMPID | [‘E001’, ‘E002’, ‘E003’, ‘E004’, ‘E005’, ‘E006’, ‘E007’, ‘E008’, ‘E009’, ‘E011’] |
| 2 | FIRST\_NAME | [‘Arun’,’Anand’,’Ajay’,’Babu’,’Basu’,’Chinna’,’Christy’,’Deepa’,’Elizabeth’,’Fawlin’] |
| 3 | LAST\_NAME | [‘Sampath’,’Saravanan’,’Tkare’,John’,’Ganesh’,’Tanmbi’,’John’,’Gopalan’,’Rose’,’Pandian’] |
| 4 | AGE | [24, 26, 28, 32, 34, 37, 40, 41, 42, 50] |
| 5 | SALARY | [62500.00, 70000.00, 65000.00, 67500.00, 82500.00, 92500.00, 99500.00, 90500.00, 99900.00, 94500.00] |





For each sequence of dictionary elements, retrieve empid, fn, ln, age and sal from the emp\_dict, dictionary.

You may notice that the first line of the sql variable is assigned exactly

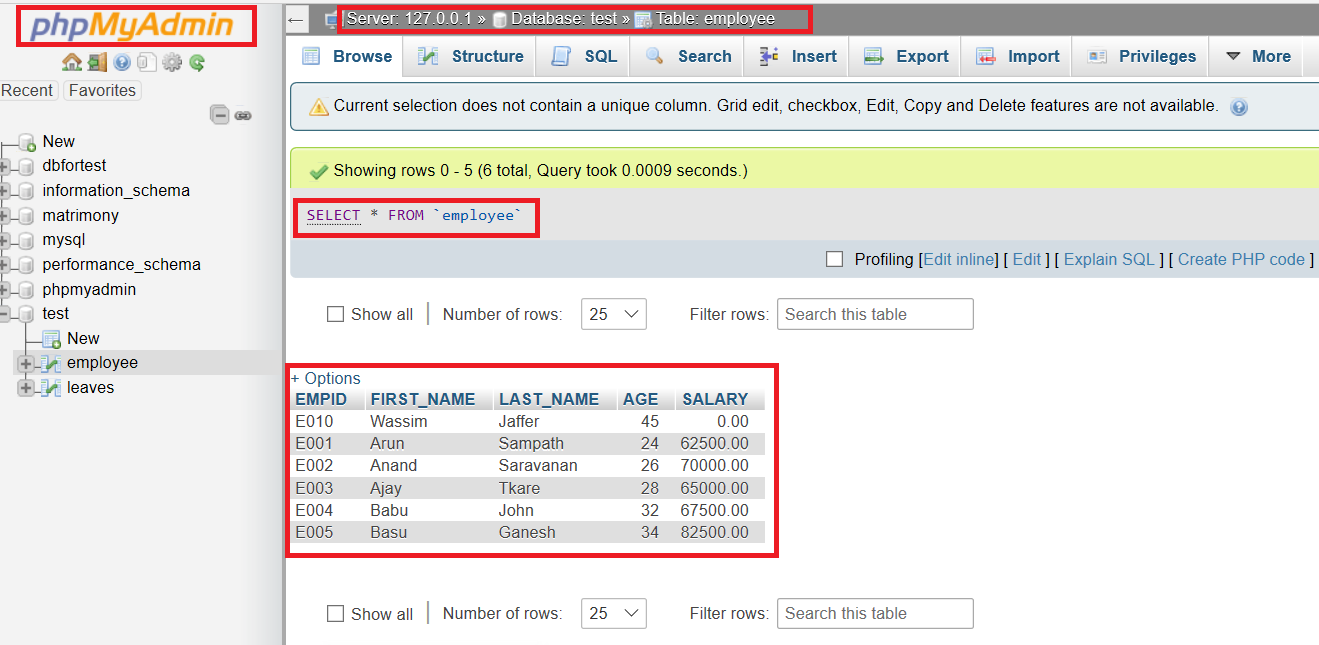
INSERT EMPLOYEE (EMPID, FIRST\_NAME, LAST\_NAME, AGE, SALARY)

In the value\_string, we first create the string ‘(‘ and then for each of the string columns such as empid, fn, ln, we prefix “\’” and suffix “\’,”. For numeric values such as age and sal, we transform them to string values by applying str() to them.

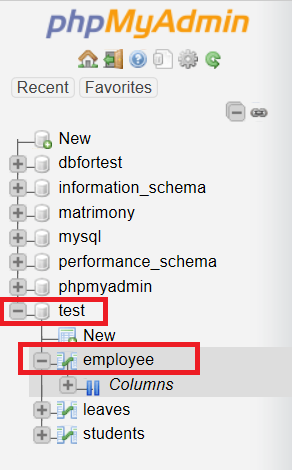
Note: The symbol, \ is the escape character to remove the special effect of the character followed by \. We don’t want to treat the variable name as a string instead of showing the value of that variable sandwiched between single quotes such as ‘Arun’.

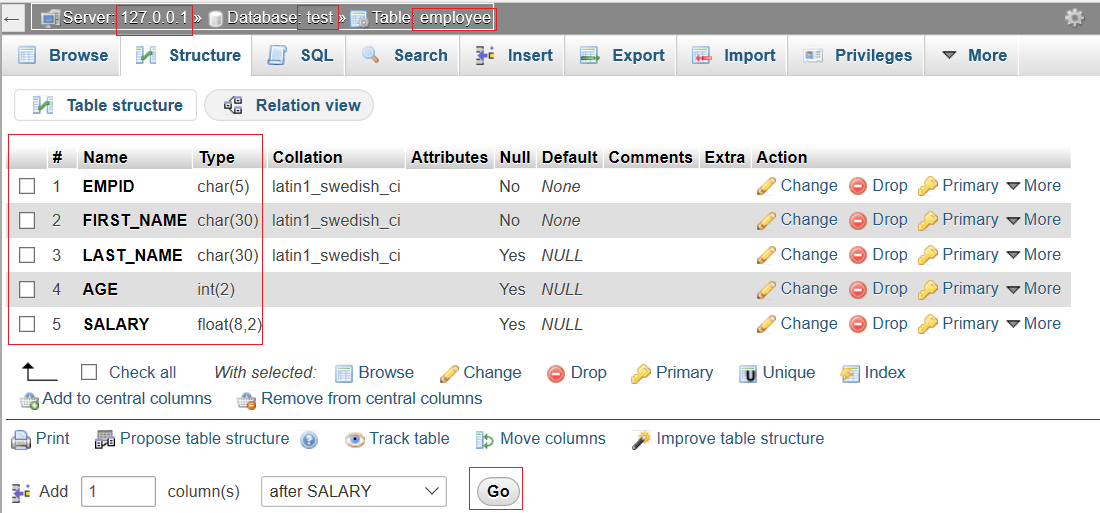
* Using the execute() method of cursor object, you can insert into the table EMPLOYEE shown above in the try – except block.
* The connection object method commit() , commits the current transaction and make the changes permanent.
* The connection object method rollback() , roll back the current transaction and cancels the changes.

*We can see these records in phpMyAdmin*

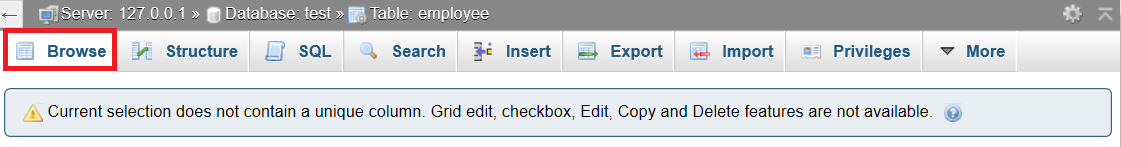
****

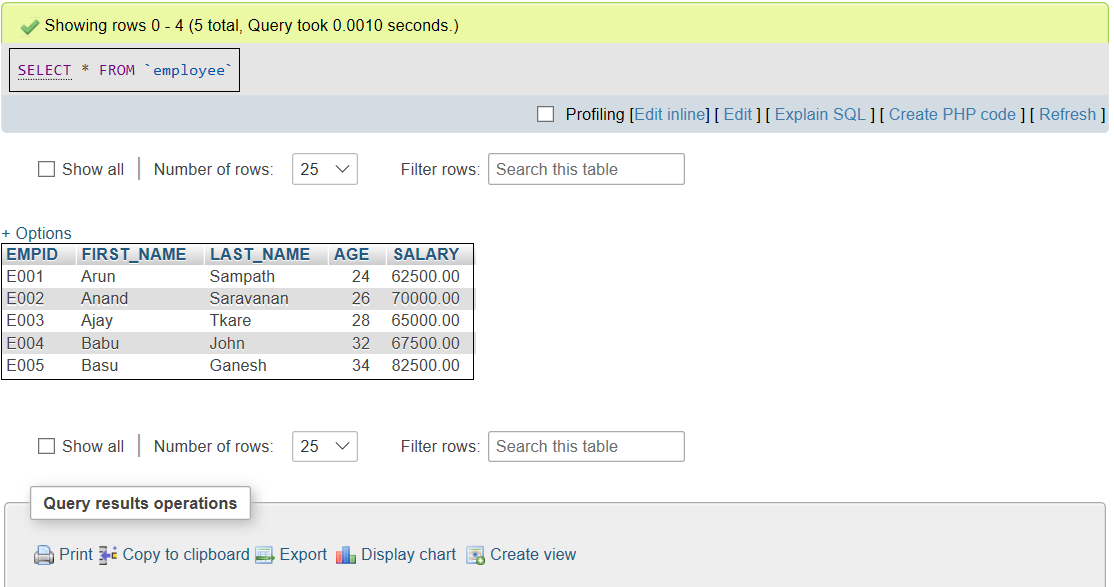
*We shall these pictures clearly in the next few pages.*

**

**

*We use ip address 127.0.0.1 for local hosting.*

**

**

Similarly insert records into LEAVES table.

**LEAVES table values**

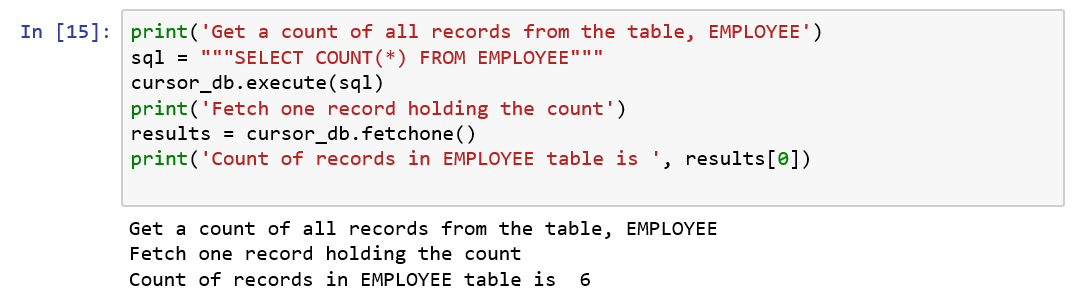
|  |  |  |
| --- | --- | --- |
| **#** | **Column** | **Values** |
| 1 | EMPID | [‘E001’, ‘E002’, ‘E003’, ‘E004’] |
| 2 | LEAVE\_DAYS | [1.5, 2.0, 3.5, 1.0] |



**SELECT**

1. **Simple listing with count of records from table, EMPLOYEE**

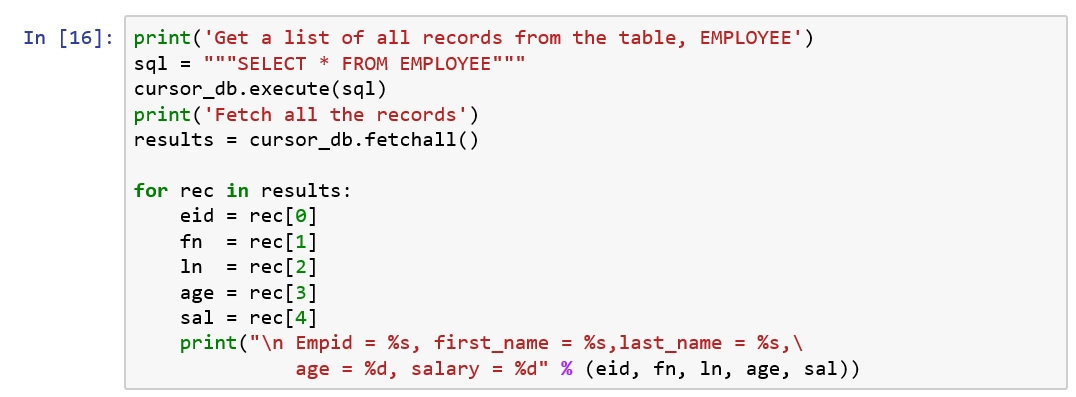
Having inserted records into the table, we need to get a list of employee records with count.

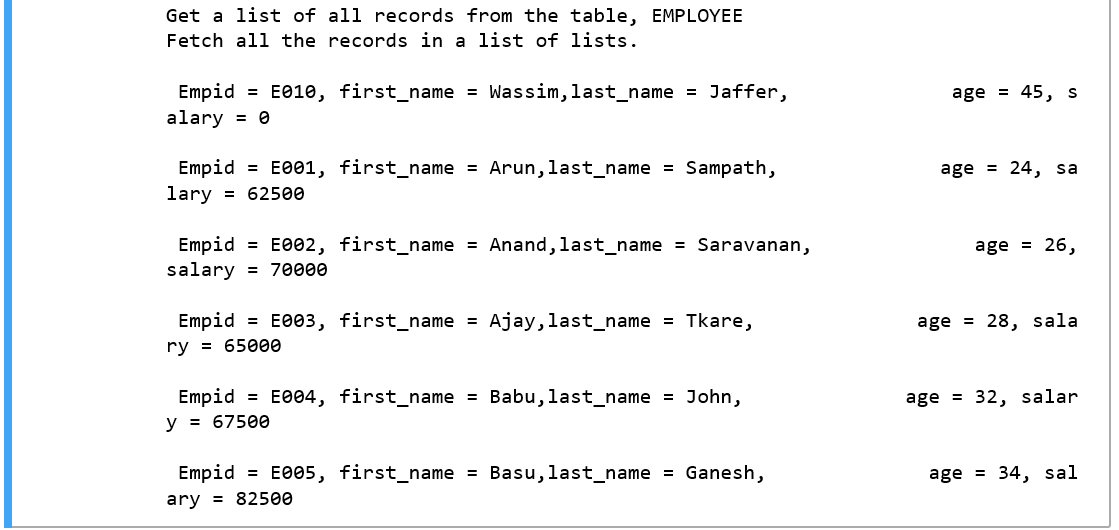


Note: By using the fetchone() method of cursor\_db, a cursor object, you get a single value as a tuple from the SQL query.

1. **Simple listing with list of records from table, EMPLOYEE**

**Now, we want to know, what values reside in our EMPLOYEE table!**

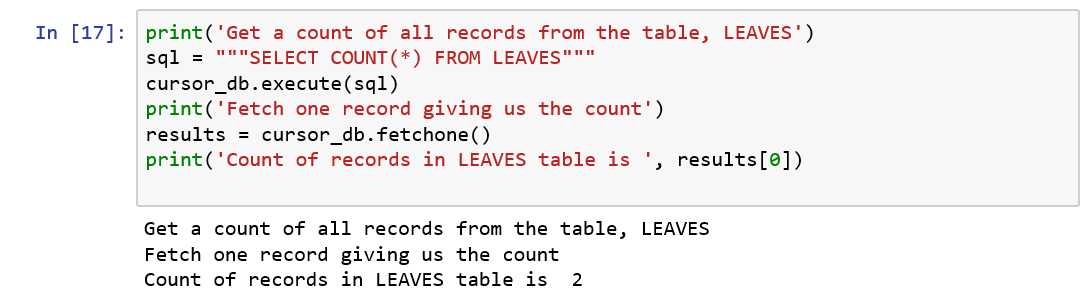
****



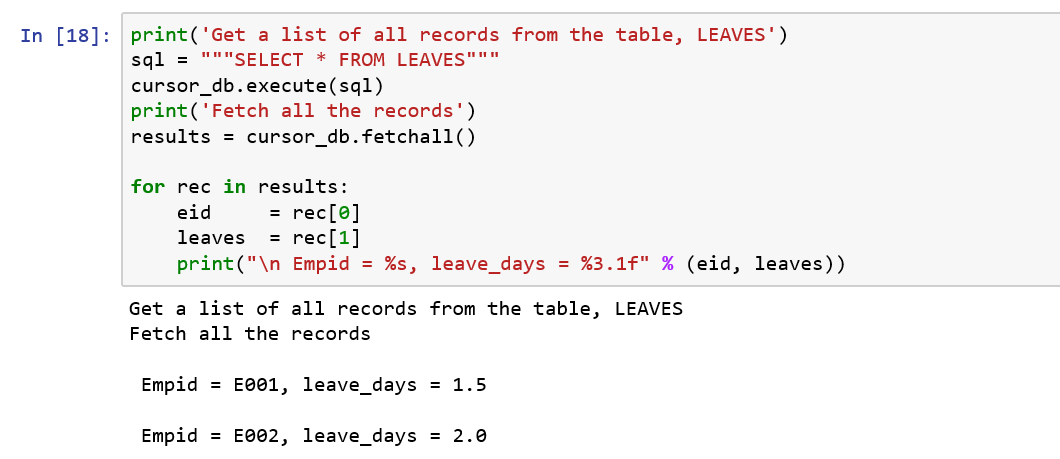
* The fetchall() method of cursor\_db object, returns multiple rows.
* You assign a variable, results to hold the output of fetchall() method.
* Each row in the output, will have values for EMPID, FIRST\_NAME, LAST\_NAME, AGE, SALARY.
* To access values in output, use the square brackets for slicing along with the index.
* For example, rec[0] will yield eid, rec[1] will yield fn and so on.

**Now, we want to know, what values reside in our LEAVES table!**

1. **Simple listing with count of records from table, LEAVES**

****

1. **Simple listing of records from table, LEAVES**

****

1. **Listing of records from two tables, EMPLOYEE and LEAVES**

How do you get a list of employees who have availed leave details? Please print the formatted output, easy on humans! For a decent looking report, there should be a title, heading line with column names and detailed lines for any report.

**A) Formatted string to a print function**

1) Create a formatted string

2) Assign to a variable to be passed to the print function

Some of the popular conversion symbols

d is for signed integer decimal

f is for floating point decimal format

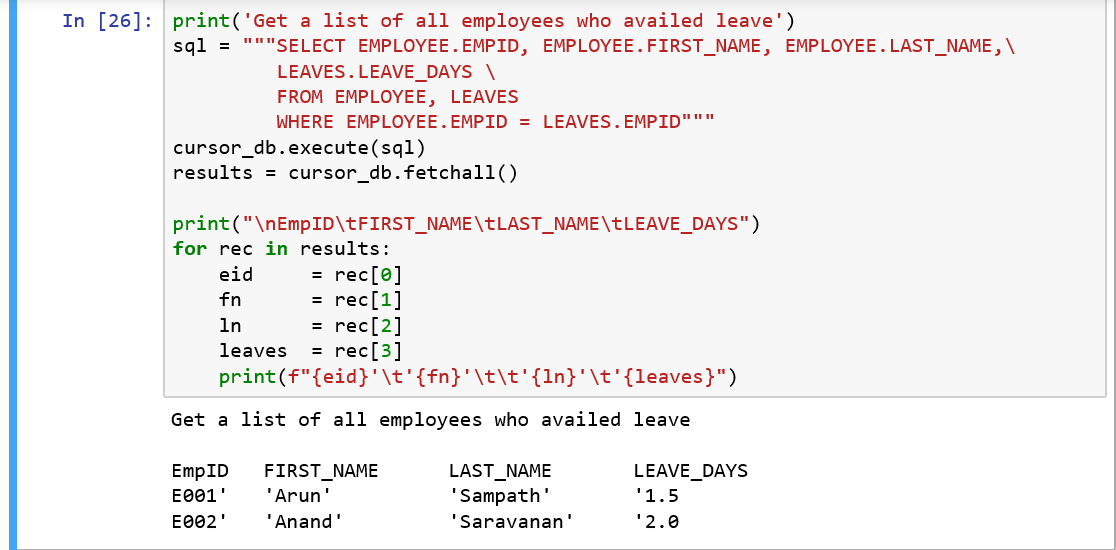
s is for string

We use /t to create a tab space between two strings and \n to create a line space (new line) between two strings

https://www.python-course.eu/python3\_formatted\_output.php

**B) Formatted string literals.**

They are prefixed with an f and they contain replacement fields formed with curly braces. The replacement fields are expressions, which are evaluated at run time, and then formatted using the format() protocol.

****

* **What we have done is we have obtained records that have matching values in both tables, EMPLOYEE and LEAVES.**

**What are the different types of JOINs in SQL?**

1. **INNER JOIN:** Returns records that have matching values in both tables.
2. **LEFT OUTER** **JOIN**: Returns all records from the left table, and the matched records from the right table.
3. **RIGHT OUTER JOIN**: Returns all records from the right table, and the matched records from the left table.
4. **FULL OUTER JOIN**: Return all records when there is a match in either left or right table.

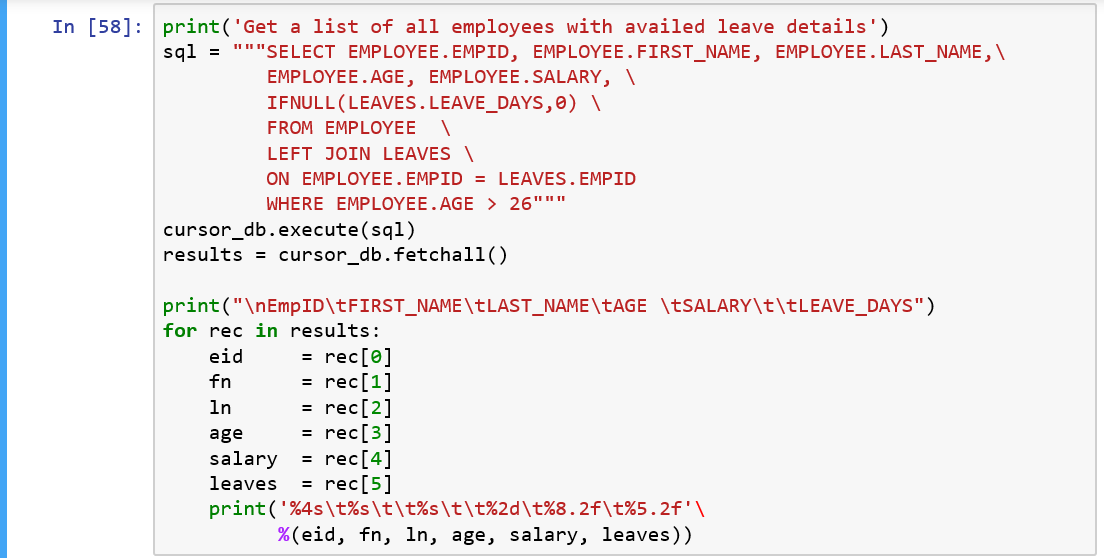
|  |  |
| --- | --- |
| Inner join returns records that have matching values in both tables, shaded green in the nearby picture.  We have used inner join in SQL statement for achieving the above purpose. |  |

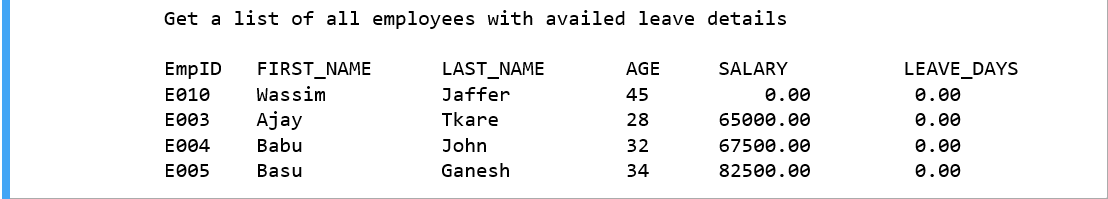
Suppose, you want to have a list of all employees with all details including leave availed. How do you achieve this? You need to use LEFT OUTER JOIN for this purpose.

|  |  |
| --- | --- |
| Left outer join returns all records from the left table, and the matched records from the right table.  We have used inner join in SQL statement for achieving the above purpose. |  |

Since there is no compulsion for any employee to avail leave, we are interested in all records contained in EMPLOYEE table but also interested in leave details of those employees appearing in LEAVES table with matching EMPIDs in EMPLOYEE table and whose age is above 26.

Refer: https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins





**UPDATE**

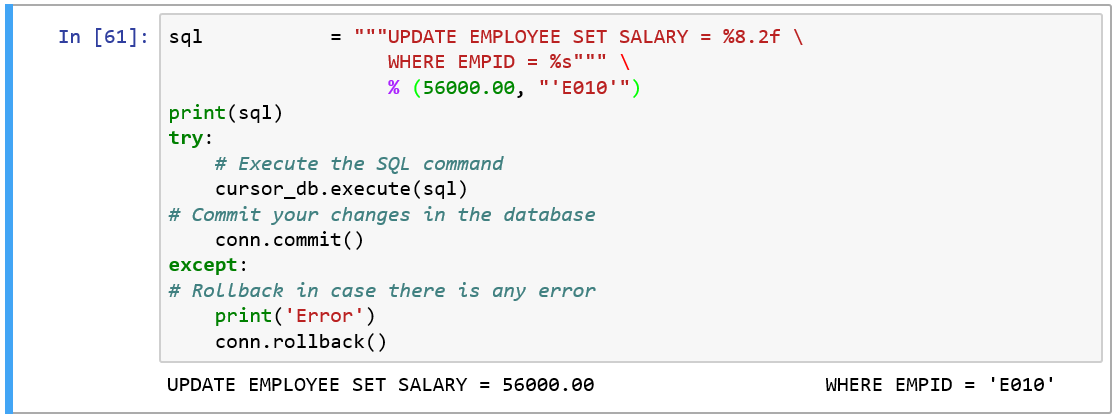
We now observe that for the employee, ‘E010’ – Wassim Jaffer the salary is 0 and he will not be happy about it, regardless of your efforts to convince him that zero plays a very important role in mathematics. Refer: <https://myessaypoint.com/importance-zero-mathematics>. We later found out that his salary is 56000.

How do you correct these typos?

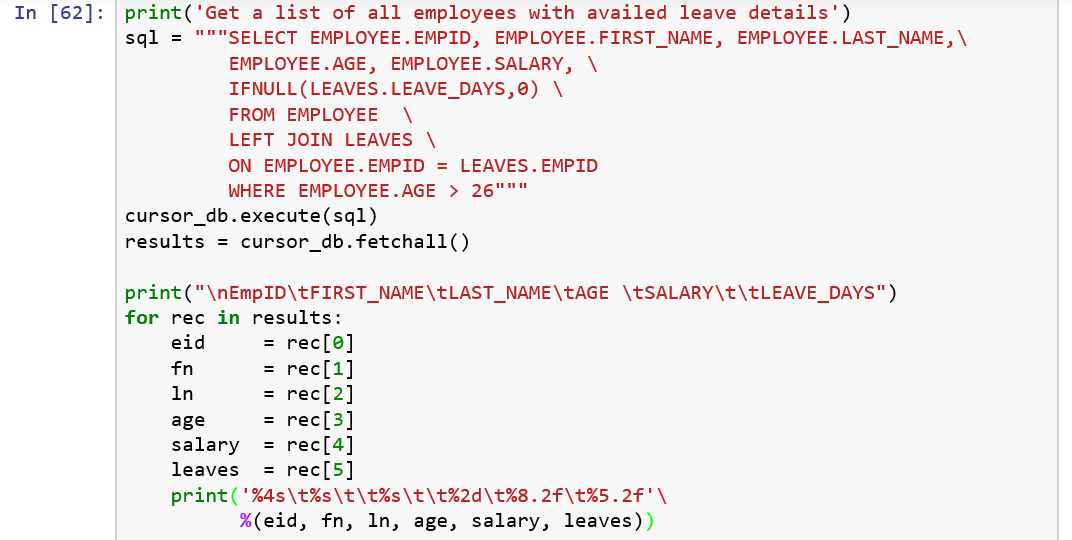
With the UPDATE command, we can

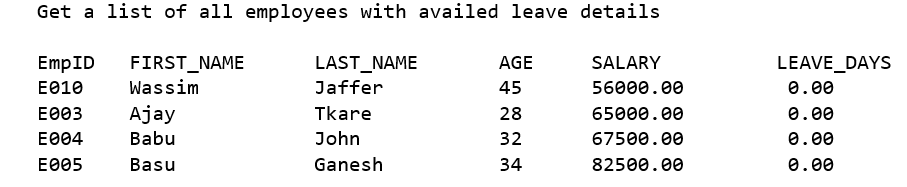
* Update one like SALARY field or more field altogether.
* Specify any condition such as EMPID = ‘E010’ using the WHERE clause.
* Update the values in a single table such as EMPLOYEE at a time.

*Note: The* ***WHERE*** *clause is very useful when you want to update the selected rows in a table.*

****

Let us check whether we have carried out the corrections correctly.

****

****

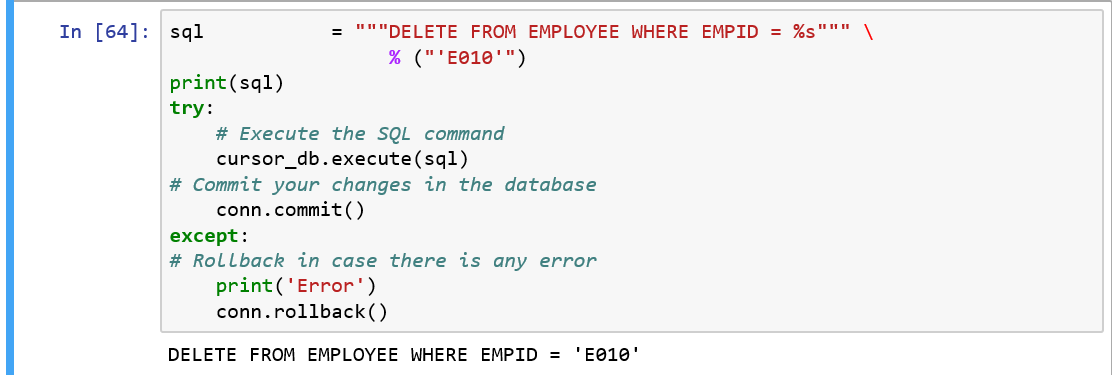
Good. It seems that we have fixed the problem.

**DELETE**

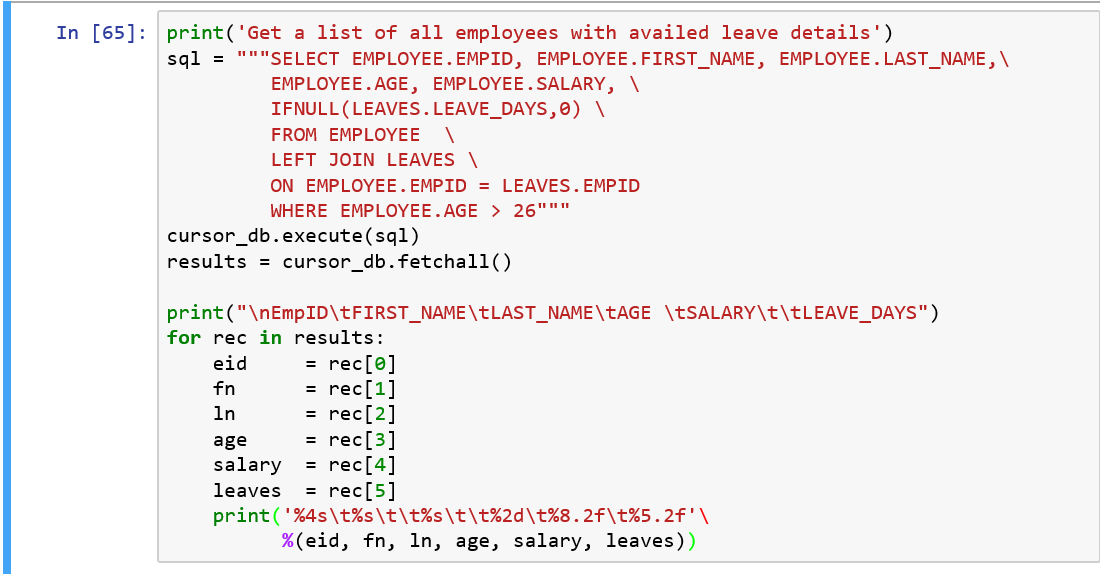
There are occasions when we want to delete a record in the table, EMPLOYEE owing to duplicate entry or data corruption etc.

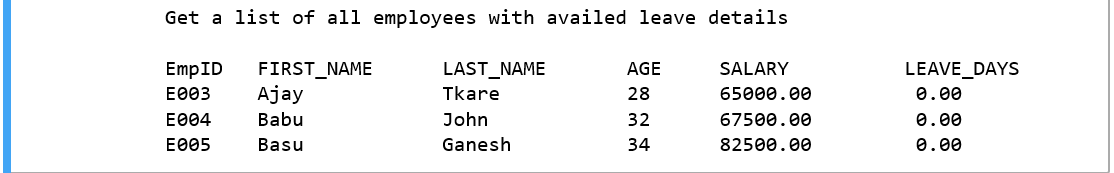
Note:

* Pay attention to the WHERE clause where you can specify any condition such as EMPID = ‘E010’. If you omit WHERE clause, then all the records will be deleted from the given table.
* At a time , you can delete records in a single table.

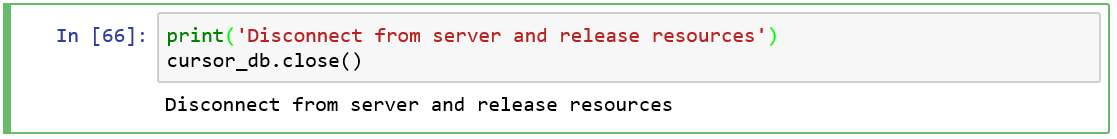


**Let us check whether we have carried out the deletion correctly.**





**Step 4: Close the server connection**

****

Since our mission is accomplished, it is time to disconnect from the server and we disconnect using the close() function for the same.

In this section, you have learnt how to

a) use MySQL connector / Python to access MySQL databases.

b) perform database operations such as SELECT, INSERT, UPDATE and DELETE.

<https://stackoverflow.com/questions/372885/how-do-i-connect-to-a-mysql-database-in-python>

<http://dev.mysql.com/downloads/connector/python/2.0.html>

## **How to access, create JSON files?**

**What is JSON?**

Refer <http://json.org/> for details. Since its inception, JavaScript Object Notation (JSON) is a very popular standard for information exchange and it was specified by **Douglas Crockford**.

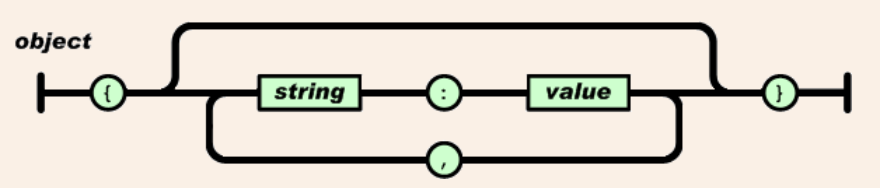
* **JSON is light weight data interchange format.** It is used for serializing and transmitting structured data between over a network connection such as a server and web applications. Web services and Application Programming Interfaces (API) use JSON format to provide public data.
* **JSON is easy to read and write than other data interchange format such as Extensible Markup Language (XML).**
* **JSON is a generic data format with a minimal number of value types: strings, numbers, bools, lists, objects, and null.** These data types are represented in many programming languages such as C, C++, Java, Python, Perl making it language independent.
* **The file name extension of a JSON file is .json.**

**Why JSON?**

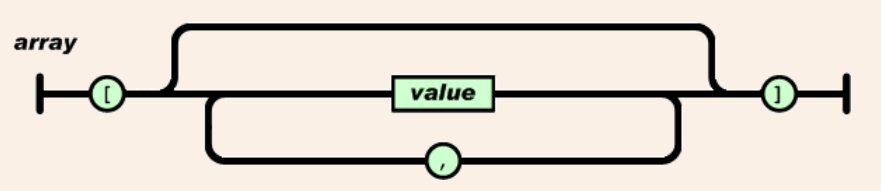
* **JSON requires less tags than its competitor, XML**. XML items must be wrapped in open and close tags whereas JSON names the tag once. Since it is light-weighted, it becomes easier to get and load the requested data quickly.
* **JSON is transportation independent.** JSON exists as a sequence of bytes which is very useful when we need to transmit a stream of data over a network**.**
* **JSON is easy to read and comprehend.**

**JSON is built on two structures: All images from** <http://json.org/>

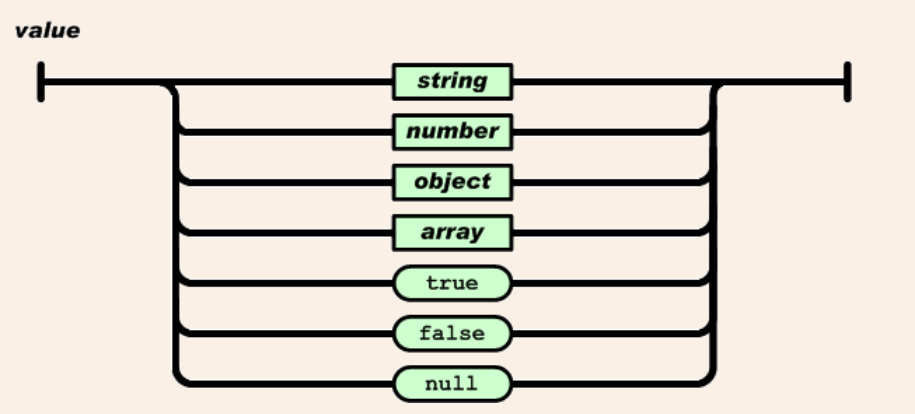
1. A collection of name / value pairs known as an object. For example, {"Name" : "Peter", "Age" : 27}. An object begins with { and ends with }. Each name is followed by : (colon) and the name / value pairs separated by , (comma).



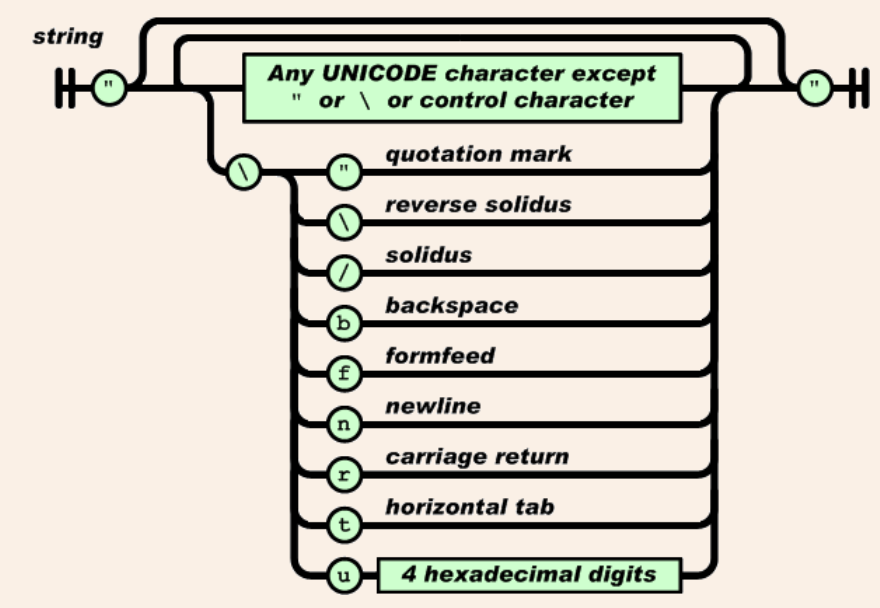
1. An array is an ordered collection of values such as ["Peter", "John", "Susan"], [ 27, 31, 25]



1. An array begins with [ and ends with ]. Values are separated by , (comma).
2. A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.



1. A string is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. Example: “John”.

****

**JSON using Python**

* JSON can store dictionaries, lists, bools, numbers.
* After reducing these structures to a string, it can be saved into a file.
* The json, a module in Python will convert the data structures to JSON strings.

**Basic terminologies:**

1. **Serialization:** is a process of encoding a complex object such as dictionary into a JSON representation for easy transmission or streaming.

2. **Deserialization**: is a process of decoding data received in JSON format as native data types for further manipulation.

**By default, the JSON encoder deals with native Python data types such as list, tuple, dict, str, int, float and bool.**

**Conversion table**

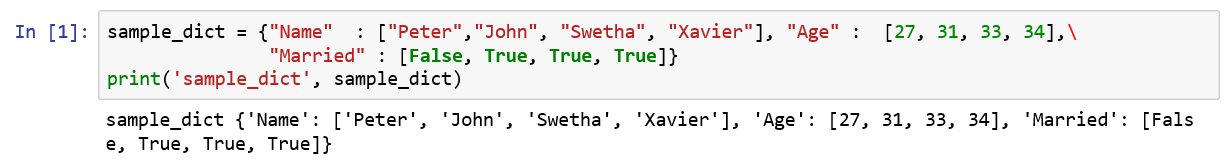
|  |  |
| --- | --- |
| **Python data type** | **JSON data type** |
| dict | object |
| list, tuple | array |
| str | string |
| int, float | Number |
| True | true |
| False | False |
| None | null |

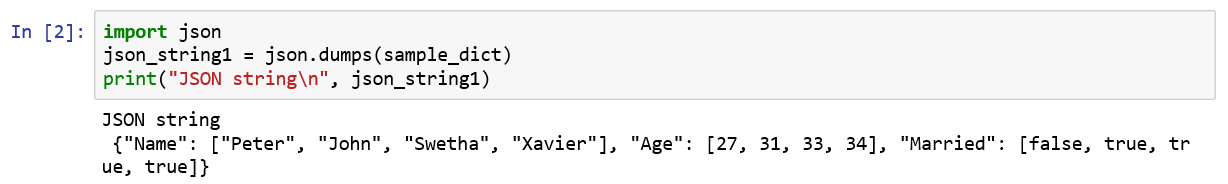
**Two methods of serialization**

1. ***dumps() to serialize an object to a JSON formatted string***
2. ***dump() to serialize an object to a JSON formatted stream (which facilitates writing to a file)***

**Example: Serializing native python datatypes**

The JSON function, json.dumps() is used to convert the python dictionary into a JSON string that can be written to a file.





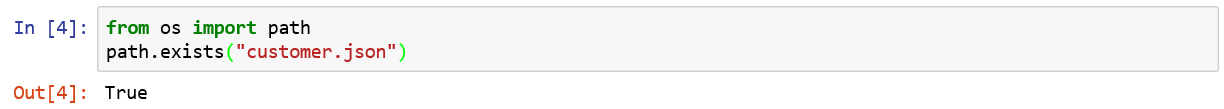
***Did you notice the Bool data type for Python is True or False and for JSON is true or false?***

The function, json.dump() is used to dump a JSON string into a file.

A with() function simplifies the process of writing and closing the file.



Let us check if the file, customer.json really exists in the working directory.



**The JSON decoder converts JSON decoded data into Python data types such as list, tuple, dict, str, int, float and bool.**

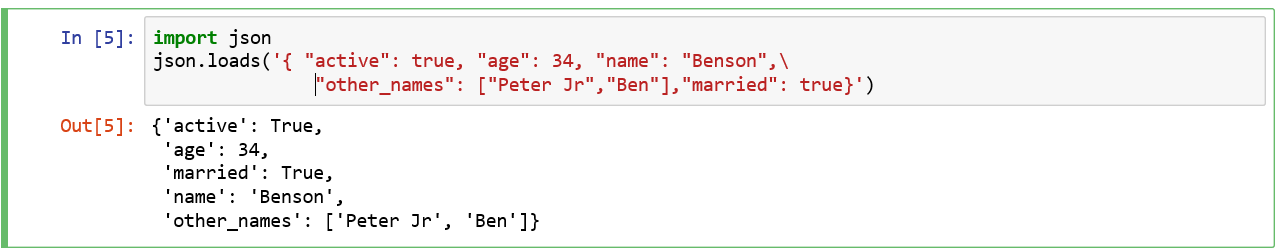
**Two modules for deserialization**

1) loads() to deserialize a JSON document to a Python object

2) load() to deserialize a JSON formatted stream (example from a file) to a Python object

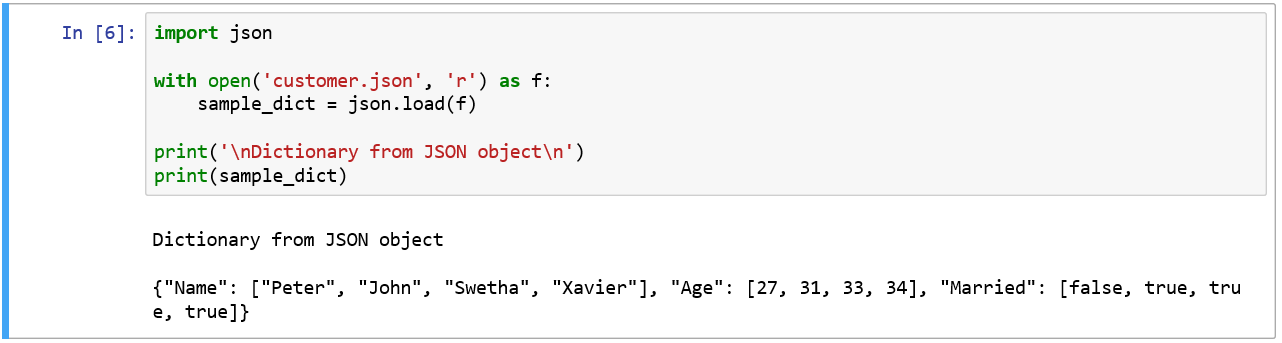
**Example: Deserializing native python datatypes**

The function, json.loads() is used to convert a json string to a python dictionary.



We passed a JSON string to the json.loads() function and obtained a dictionary as the output.

By using the json.load methods, you can convert the JSON into a dictionary as shown below:



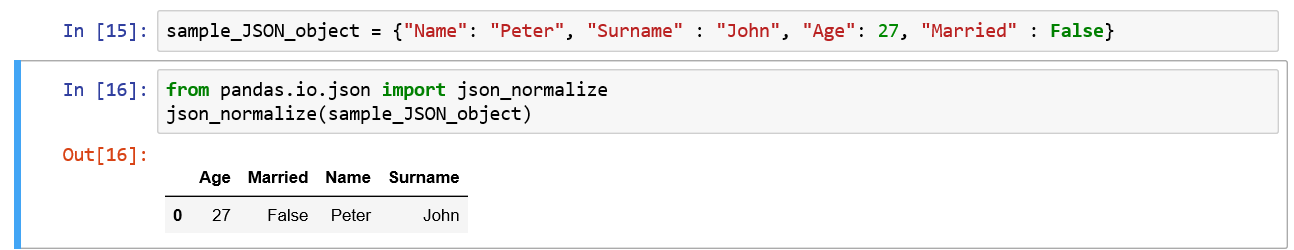
**Flattening a JSON file**

There are occasions when you need to flatten the complex JSON object with many hierarchy to force them into a pandas data frame.

1. **Simple JSON object**

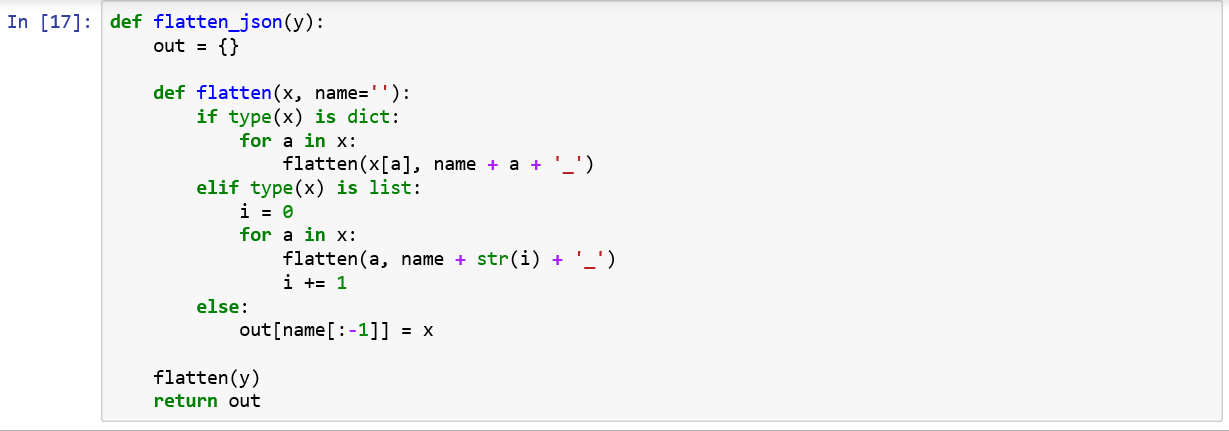
Pandas provides a function json\_normalize() for flattening semi-structured JSON objects.

For example:

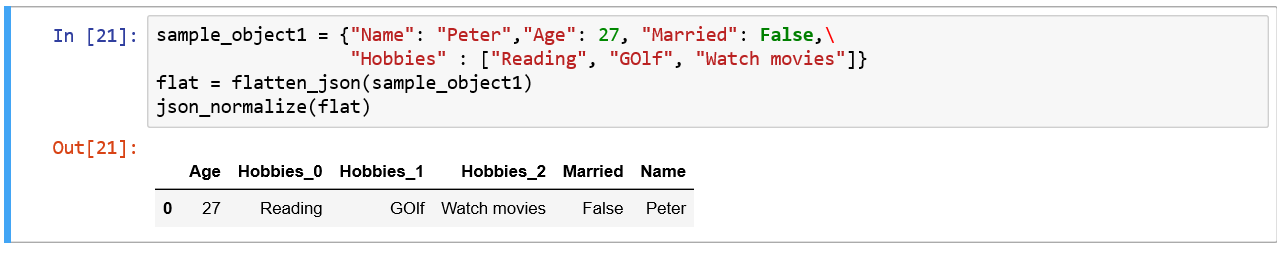


1. **Complex JSON object with hierarchy**

Write a function to recursively extracts values out of the input JSON object into a flattened dictionary.



Apply json\_normalize to the output of flatten\_object to produce a python dataframe as shown below:



Refer:

<https://realpython.com/python-json/>

<http://borkweb.com/story/the-case-for-json-what-is-it-and-why-use-it>

<https://www.tutorialspoint.com/json/json_overview.htm>

<https://www.w3schools.com/js/js_json_datatypes.asp>

<https://developer.rhino3d.com/guides/rhinopython/python-xml-json/>