Database and Parallel Programming

Database and Python

- In the current digital era, data are produced in many ways. Such massive production of data makes data processing a challenging task. To process the data effectively, the data should be organized effectively. In this context, Databases becomes very important.
- A database is the collection of data in an organized manner which is stored electronically.
- The system that manages databases is called Data Base Management System (DBMS).
- The database is divided into two types, namely,

1. Structural Data base
2. Unstructured database

ACID Property of Database

- For a database not to be a failure model, it should be designed and created properly.
- This could be ensured by checking whether the database sticks with the ACID property.
- The acronym ACID stands
- atomicity
- consistency
- Isolation
- durability.

Structured Query Language (SQL)

- SQL is the primary interface used to interact with relational databases. SQL is used to execute CRUD functionalities.
- The CRUD stands for create, read, update, and delete of database. The SQL commands are explained in the upcoming sections of the chapter.
- List of Structured Database Tools Some of the famous structured relational databases which are commonly used are,

1) MySQL
2) SQLite
3) PostgreSQL
4) Microsoft SQL Server
5) Oracle Database

Unstructured Database

- Unstructured database consists of data that does not follow any data model and is primarily unorganized. The conventional method of data processing is not useful in handling unstructured data.
- The word "query" is also suitable for an unstructured database. Non-relational SQL (NoSQL) and data lakes are used to manage unstructured data. Unstructured data is constantly increasing, and demand for managing the database is also increasing.
- List of Unstructured Database Tools Some of the popularly used unstructured database tools are as follows.

1) MongoDB

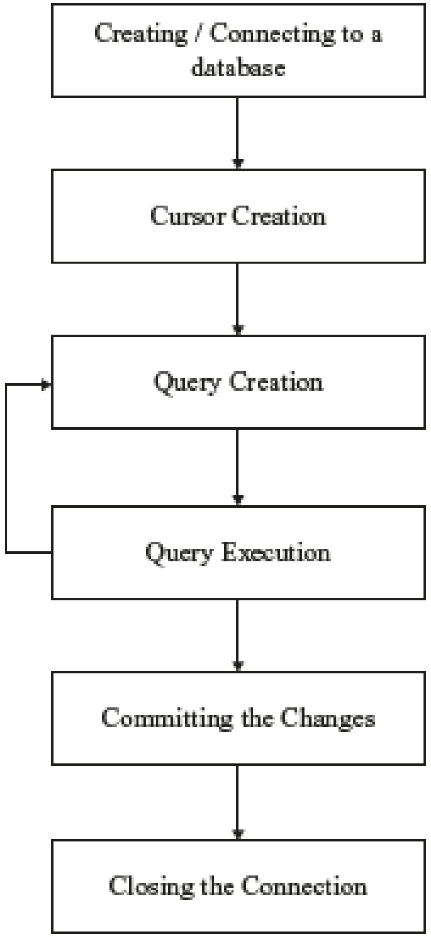
2) NoSQL

3) DynamoDB

4) Hadoop

5) Azure

The sequence of execution when working with Databases in Python



Creating / Connecting to a database

Cursor Creation

Query Creation

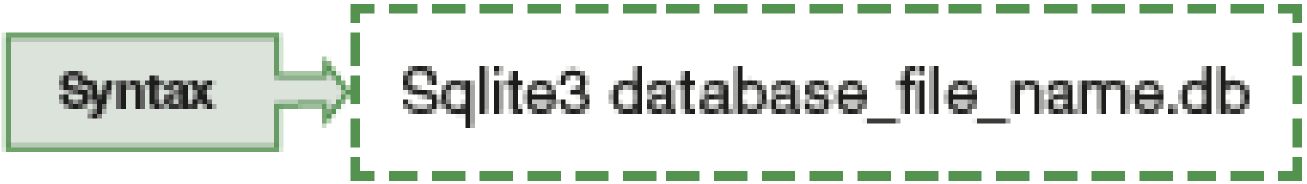
Query Execution

Committing the Changes

Closing the Connection

Connecting to Database

- Since SQLite is a relational database, a database refers to a file with a ".db" extension.
- While opening the command line for SQLite, the database name is given as the parameter.
- The syntax for the SQLite connection is shown below.



Syntax

```
Sqlite3 database_file_name.db
```

CRUD Database

- CRUD stands for Creation, Reading, Updation, and Deletion of a table and its data.
- Creation:

The creation is creating a new table in database and inserting new data into the table.

Creating a New table.

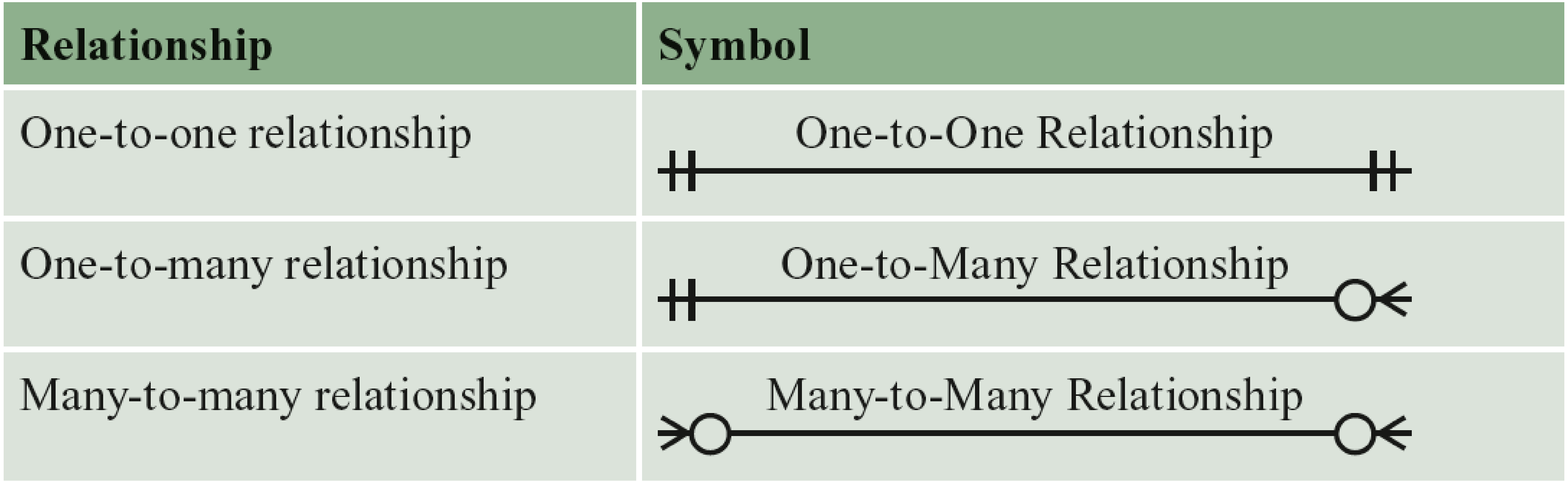
The following datatypes are widely used in SQLite.

1. INTEGER
2. TEXT
3. REAL
4. BLOB
5. NULL

Relationship between Tables.

- As seen above, a foreign key relates between any two tables. The relationship between the two tables falls into one of the following kinds. Let us consider that Table A and Table B, in which there is an attribute in Table B, point to the primary key of Table A.
- 1) One-to-one relationship: For one tuple in Table A, only one tuple exists in Table B.
- 2) One-to-many relationship: For one tuple in Table A, there exists one or more than one tuple in Table B.
- 3) Many-to-many relationship: For one tuple in Table A, there exist many tuples in Table B, and also, for one tuple in Table B, there exist many tuples in Table A

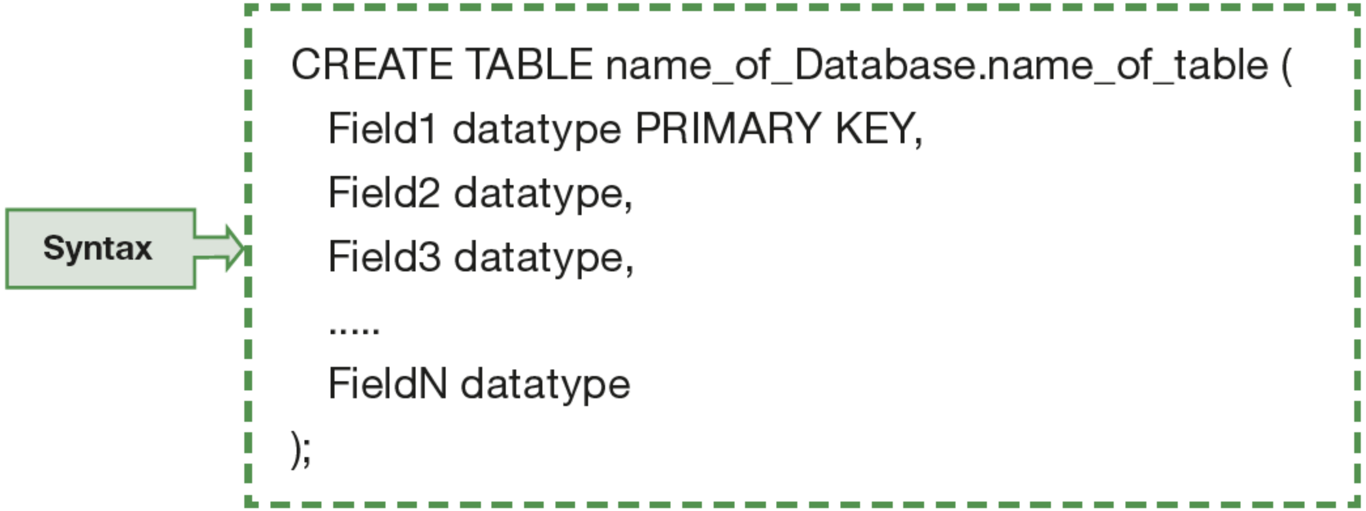
Relationship and Symbol



Relationship	Symbol
One-to-one relationship	One-to-One Relationship
One-to-many relationship	One-to-Many Relationship
Many-to-many relationship	Many-to-Many Relationship

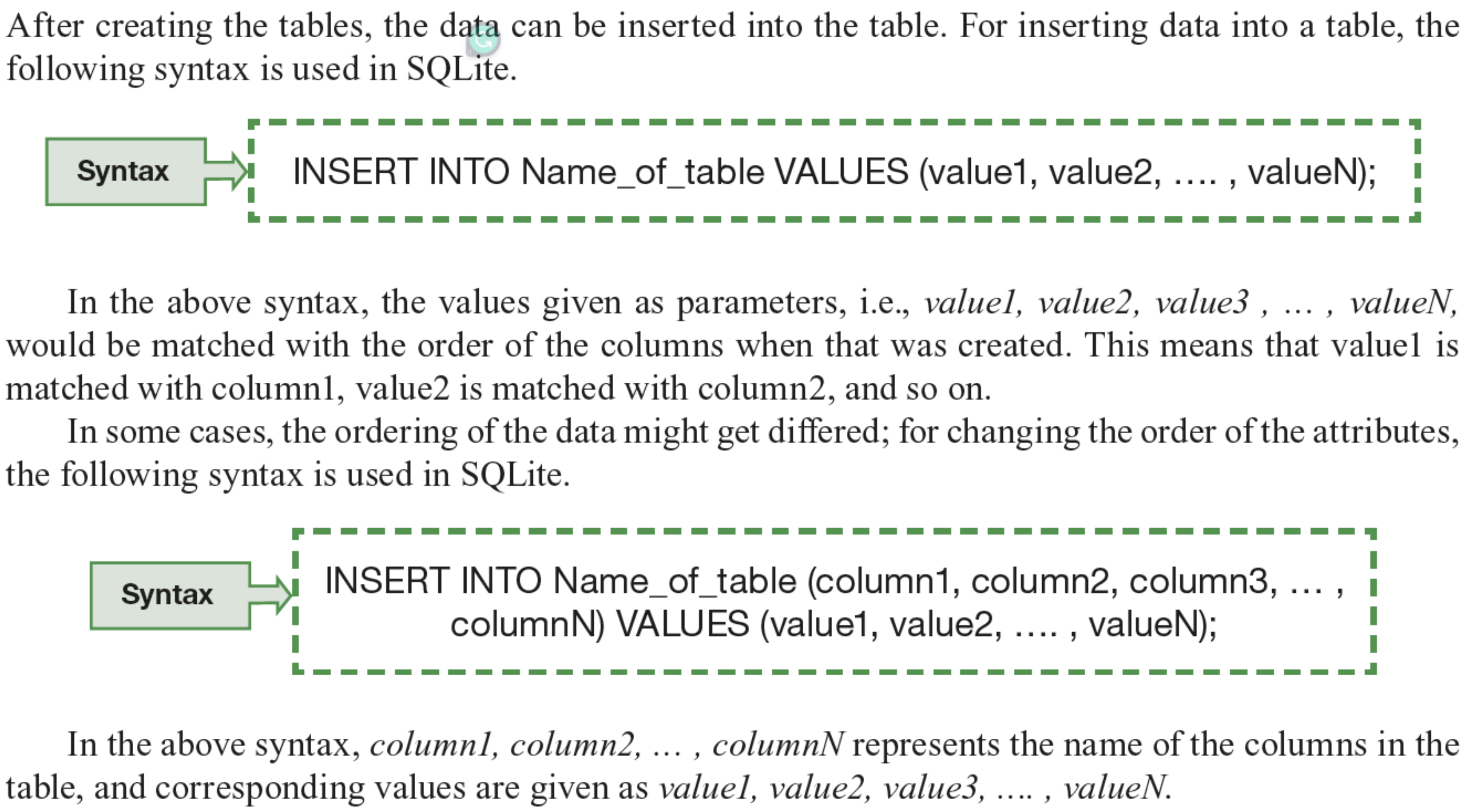
Creating a Table in a Database

- After creating a database, a table is needed for storing the data in a structured format. The designing table and its attributes involve a complete understanding of the purpose or the problem that a programmer is working on.



```
CREATE TABLE name_of_Database.name_of_table (
   Field1 datatype PRIMARY KEY,
   Field2 datatype,
   Field3 datatype,
   .....
   FieldN datatype
);
```

Inserting Data to the Tables.



After creating the tables, the data can be inserted into the table. For inserting data into a table, the following syntax is used in SQLite.

Syntax

```
INSERT INTO Name_of_table VALUES (value1, value2, …. , valueN);
```

In the above syntax, the values given as parameters, i.e., *value1, value2, value3 , … , valueN,* would be matched with the order of the columns when that was created. This means that value1 is matched with column1, value2 is matched with column2, and so on.

In some cases, the ordering of the data might get differed; for changing the order of the attributes, the following syntax is used in SQLite.

Syntax

```
INSERT INTO Name_of_table (column1, column2, column3, … ,
columnN) VALUES (value1, value2, …. , valueN);
```

In the above syntax, *column1, column2, … , columnN* represents the name of the columns in the table, and corresponding values are given as *value1, value2, value3, …. , valueN.*

Reading

- After creating the table and inserting the data, the data in the tables of the databases should be extracted. The process of extraction is called as read the database.
- The syntax for extracting the data from a table is as follows.

```
SELECT * FROM Teacher WHERE Teacher.position = "HoD";
```

- In the above syntax, the following understandings are needed.

1) After the keyword SELECT, the columns needed could be given, so the data from the specified columns would be extracted. The "*" can be used instead of the column names, which extract the data from all the columns.

2) The Conditions after the keyword WHERE are used to extract the subset of the data from the specified table.

3) The result of the query is a table that contains the required data, which is the result of the given query.

Deletion Table and Data

The concept of deletion is of two types. One is deleting a particular set of data from a table, and the second is deleting an entire table. Both are discussed in detail below.

Deleting Entries in the Tables

For deleting entries or a set of data from a Table is executed using the DELETE query, whose syntax is as follows.

Python Syntax

```
DELETE FROM Table_Name WHERE Conditions;
```

Threading Module

- A process in an Operating System is defined as an entity that is a collection of works to be implemented in the system.
- Conventionally, a process is a heavy weighted execution downsized into shorter units of execution called threads.
- A thread is a single unit of a process that is executed sequentially.
- A thread consists of a stack, a set of registers, a program counter, and a thread ID.
- A thread works in shared memory, whereas a process works in an isolated memory allocated for the execution of the process.
- Individual tasks in a process are allocated to separate threads that are independent of each other, and they are executed parallelly.
- Thus parallelism is achieved using threads.

The threading mechanism offers the following benefits.

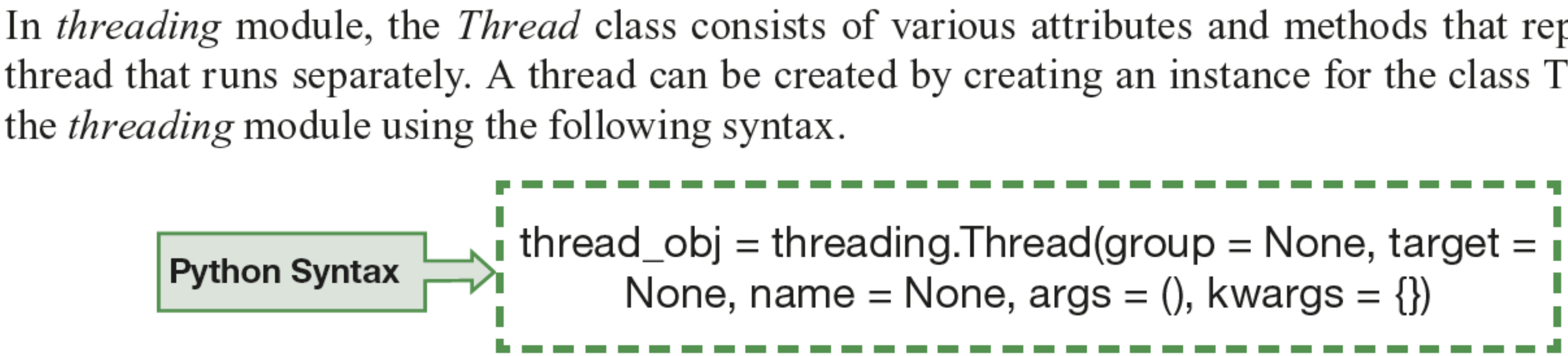
a) Economy

b) Resource Sharing

c) Responsiveness

d) Scalability

Thread Objects:



In *threading* module, the *Thread* class consists of various attributes and methods that represent a thread that runs separately. A thread can be created by creating an instance for the class Thread in the *threading* module using the following syntax.

Python Syntax

```
thread_obj = threading.Thread(group = None, target =
    None, name = None, args = (), kwargs = {})
```

start() Method

After declaring a thread with the above syntax, the thread should be executed by the *start()* method associated with a thread object. The syntax for using the *start()* method is as follows.

Python Syntax

```
thread_obj.start()
```

run() Method

The run() method associated with a thread object represents the activity of the Thread. A thread can be created by overriding this method of the Thread class. The run() method invokes the callable object passed to the object's constructor. The arguments present in args and kwargs of the method are considered as the positional arguments and keyword arguments respectively for the created thread.

join() Method

When the join() method associated with a thread object is invoked, the control waits till the thread terminates. A thread may terminate for the following three reasons.

1) The execution of the run() method of the thread object finishes.
2) The execution of the thread finishes.
3) The thread raises an error or an exception.

The syntax for the join() method is as follows.

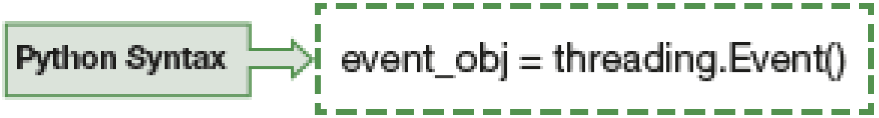
Python Syntax

```
thread_object.join()
```

Event Objects

- The event objects are used to set the lock based on the events and enable the interactions between the threads.
- An event flag is an internal flag, one of the process synchronization primitives, consisting of either a True or False value.
- There are three functions, namely, set(), clear(), and wait(), which are used to alter the event flag.
- Creating an Event Object

The following syntax is used to create an event object


Python Syntax

```
event_obj = threading.Event()
```

Set Method

Clear Method

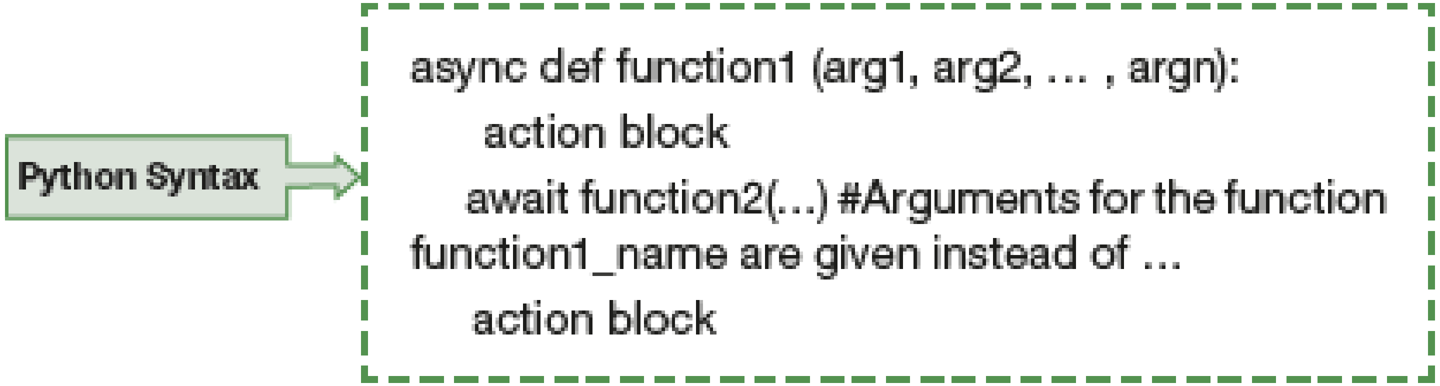
Wait Method

Asynchronous Programming

- In synchronous programming, tasks are executed sequentially, and the results are obtained sequentially. In other words, a task waits for the execution of previous tasks irrespective of their interdependency. Asynchronous programming is a type of parallel programming where a set of tasks is allowed to execute separately from the primary executions.
- This kind of parallel execution improves the performance of the application.

async/await

- A coroutine in Python is created using the async keyword, and to pause the execution of a coroutine until it executes and the results are obtained, the await is used. The await keyword is used inside the function, defined with the async keyword.
- The syntax for using async and await keywords are as follows.



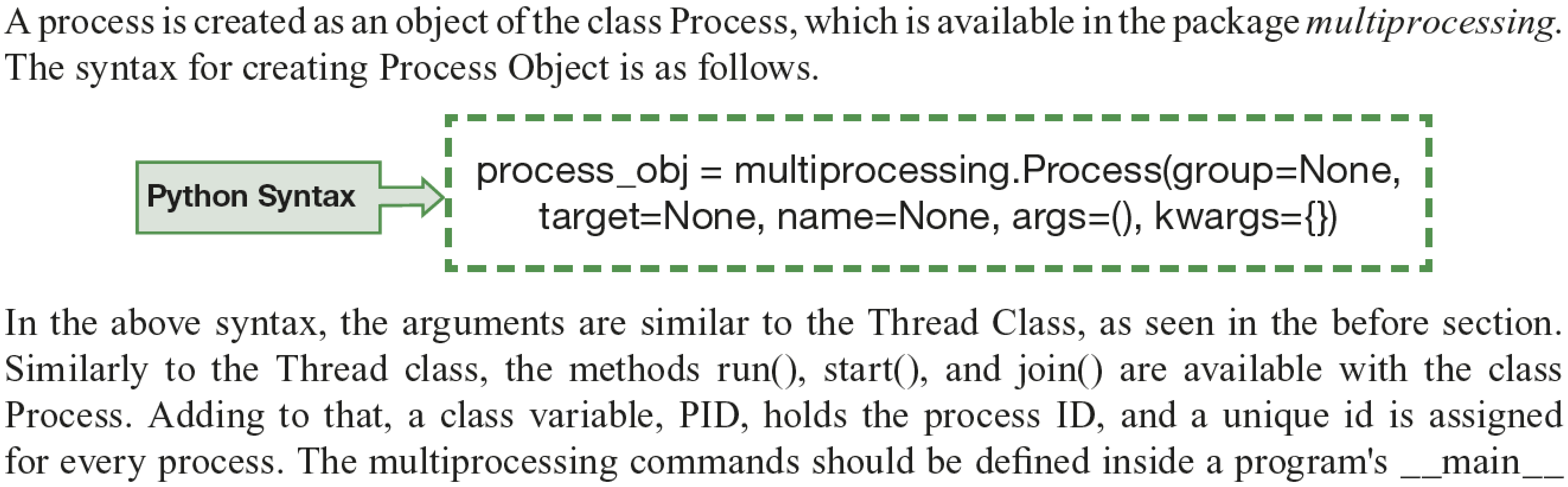
```
async def function1 (arg1, arg2, … , argn):
     action block
    await function2(…) #Arguments for the function
function1_name are given instead of …
    action block
```

Multiprocessing

- Multiprocessing is a type of parallelism in which multiple processes are executed parallelly with the help of hardware.
- The Global Interpreter Lock (GIL) of Python prevents the execution of more than one thread at a time. So the concept of the effectiveness of multithreading cannot be achieved in the interpreter.
- This can be compensated by using multiprocessing in Python.
- Having multiple processes in Python is equivalent to having individual interpreters for each process.
- In Python, the multiprocessing is handled by the package multiprocessing, which provides various functions and classes that enables the multiprocessing to be an easy job.

Creating a Process

- A process is created as an object of the class Process, which is available in the package multiprocessing.
- The syntax for creating Process Object is as follows.



A process is created as an object of the class Process, which is available in the package *multiprocessing*. The syntax for creating Process Object is as follows.

Python Syntax

```
process_obj = multiprocessing.Process(group=None,
    target=None, name=None, args=(), kwargs={})
```

In the above syntax, the arguments are similar to the Thread Class, as seen in the before section. Similarly to the Thread class, the methods run(), start(), and join() are available with the class Process. Adding to that, a class variable, PID, holds the process ID, and a unique id is assigned for every process. The multiprocessing commands should be defined inside a program's __main__