1. While Python isn’t purely an object-oriented language, it’s flexible enough and powerful enough to allow you to build your applications using the object-oriented paradigm. One of the ways in which Python achieves this is by supporting **inheritance**, which it does with super().

**In this tutorial, you’ll learn about the following:**

* The concept of inheritance in Python
* Multiple inheritance in Python
* How the super() function works
* How the super() function in single inheritance works
* How the super() function in multiple inheritance works

**Free Bonus:** [**5 Thoughts On Python Mastery**](https://realpython.com/bonus/python-mastery-course/), a free course for Python developers that shows you the roadmap and the mindset you’ll need to take your Python skills to the next level.

**An Overview of Python’s super() Function**

If you have experience with object-oriented languages, you may already be familiar with the functionality of super().

If not, don’t fear! While the [official documentation](https://docs.python.org/3/library/functions.html#super) is fairly technical, at a high level super() gives you access to methods in a superclass from the subclass that inherits from it.

super() alone returns a temporary object of the superclass that then allows you to call that superclass’s methods.

Why would you want to do any of this? While the possibilities are limited by your imagination, a common use case is building classes that extend the functionality of previously built classes.

Calling the previously built methods with super() saves you from needing to rewrite those methods in your subclass, and allows you to swap out superclasses with minimal code changes.

2. Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but like other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let’s start with Reading and Writing files. 

### ****Working of open() function****

Before performing any operation on the file like read or write, first we have to open that file. For this, we should use Python’s inbuilt function open()

But at the time of opening, we have to specify the mode, which represents the purpose of the opening file.

f = open(filename, mode)

Where the following mode is supported:

1. **r:**open an existing file for a read operation.
2. **w:** open an existing file for a write operation. If the file already contains some data then it will be overridden.
3. **a:**open an existing file for append operation. It won’t override existing data.
4. **r+:**  To read and write data into the file. The previous data in the file will not be deleted.
5. **w+:** To write and read data. It will override existing data.
6. **a+:** To append and read data from the file. It won’t override existing data.

Take a look at the below example:

* Python3

|  |
| --- |
| # a file named "geek", will be opened with the reading mode.  file = open('geek.txt', 'r')  # This will print every line one by one in the file  for each in file:      print (each)  3. In this tutorial, you'll learn about multiple inheritance in Python and how to use it in your program. You'll also learn about multi-level inheritance and the method resolution order. Python Multiple Inheritance A [class](https://www.programiz.com/python-programming/class) can be derived from more than one base class in Python, similar to C++. This is called multiple inheritance.  In multiple inheritance, the features of all the base classes are inherited into the derived class. The syntax for multiple inheritance is similar to single [inheritance](https://www.programiz.com/python-programming/inheritance). Example class Base1:  pass  class Base2:  pass  class MultiDerived(Base1, Base2):  pass  Here, the MultiDerived class is derived from Base1 and Base2 classes.  Multiple Inheritance in PythonMultiple Inheritance in Python  The MultiDerived class inherits from both Base1 and Base2 classes. Python Multilevel Inheritance We can also inherit from a derived class. This is called multilevel inheritance. It can be of any depth in Python.  In multilevel inheritance, features of the base class and the derived class are inherited into the new derived class.  4. import mysql.connector  mydb = mysql.connector.connect(   host="localhost",   user="yourusername",   password="yourpassword",   database="mydatabase" )  mycursor = mydb.cursor()  sql = "INSERT INTO customers (name, address) VALUES (%s, %s)" val = [   ('Peter', 'Lowstreet 4'),   ('Amy', 'Apple st 652'),   ('Hannah', 'Mountain 21'),   ('Michael', 'Valley 345'),   ('Sandy', 'Ocean blvd 2'),   ('Betty', 'Green Grass 1'),   ('Richard', 'Sky st 331'),   ('Susan', 'One way 98'),   ('Vicky', 'Yellow Garden 2'),   ('Ben', 'Park Lane 38'),   ('William', 'Central st 954'),   ('Chuck', 'Main Road 989'),   ('Viola', 'Sideway 1633') ]  mycursor.executemany(sql, val)  mydb.commit()  print(mycursor.rowcount, "was inserted.")  [Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_mysql_insert_many)  ADVERTISEMENT Get Inserted ID You can get the id of the row you just inserted by asking the cursor object.  **Note:** If you insert more than one row, the id of the last inserted row is returned. Example Insert one row, and return the ID:  import mysql.connector  mydb = mysql.connector.connect(   host="localhost",   user="yourusername",   password="yourpassword",   database="mydatabase" )  mycursor = mydb.cursor() Prerequisite : [MongoDB : An introduction](https://www.geeksforgeeks.org/mongodb-an-introduction/) MongoDB is a cross-platform, document-oriented database that works on the concept of collections and documents. MongoDB offers high speed, high availability, and high scalability. The next question which arises in the mind of the people is “Why MongoDB”? **Reasons to opt for MongoDB :**   1. It supports hierarchical data structure (Please refer [docs](https://docs.mongodb.com/manual/tutorial/model-tree-structures/) for details) 2. It supports associate arrays like Dictionaries in Python. 3. Built-in Python drivers to connect python-application with Database. Example- PyMongo 4. It is designed for Big Data. 5. Deployment of MongoDB is very easy.   **MongoDB vs RDBMS**  https://media.geeksforgeeks.org/wp-content/cdn-uploads/MongoDB_and_Python_1.png  sql = "INSERT INTO customers (name, address) VALUES (%s, %s)" val = ("Michelle", "Blue Village") mycursor.execute(sql, val)  mydb.commit()  print("1 record inserted, ID:", mycursor.lastrowid)  Run |