https://data.geotab.com/intelligence-data

What Is An Electronic Logging Device (ELD)?

ELD’s are electronic devices that automatically record a driver’s hours of service. Such technology allows for accurate recordkeeping of electronic logs. An ELD device tracks a truck’s engine and captures data on its movement, whether the truck is in motion or idling.

All ELDs must store supporting documents, too. Your ELD must also be able to transfer this information through USB, Bluetooth, e-mail, or an app-based service. It is important to note that ELDs are not required to continuously track driver location in real time.

It is now official, trucks and trailers in the US must have electronic logging devices (ELD) installed. See the updated news about the ELD mandate for more information today.

**About this file**

Dataset Description:

GpsProvider - Vendor who provides GPS  
BookingID - Unique Identification for a trip  
Market/Regular - Type of trip. Regular - Vendors with whom we will have contract. Market - Vendor with whom we will not have contract  
BookingIDDate - Date when booking was created vehicleno - Truck Number  
OriginLocation - Trip start place DestinationLocation - Trip end place  
Orglatlon - Latitude/Longitude of start place  
Deslatlon - Latitude/Longitude of end place  
DataPingtime - Time when we receive GPS ping  
PlannedETA - Planned Estimated Time of Arrival CurrentLocation - Live location  
DestinationLocation - Repeat of destination location  
actualeta - Time when the truck arrived Currlat - current latitude - changes each time when we receive GPS ping  
Currlon - current longitude - changes each time when we receive GPS ping ontime - If the truck arrived on time - calculated based on Planned and Actual ETA delay - If the truck arrived with a delay - calculated based on Planned and Actual ETA OriginLocationCode - Origin code  
DestinationLocationCode - Destination code tripstartdate - Date/Time when trip started tripenddate Date/Time when trip ended - based on documentation (cant be considered for calculating delay)\ TRANSPORTATIONDISTANCEINKM - Total KM of travel  
vehicleType - Type of Truck  
Minimumkmstobecoveredinaday - Minimum KM the driver needs to cover in a day DriverName - Driver details  
Driver\_MobileNo - Driver details  
customerID - Customer details  
customerNameCode - Customer details  
supplierID - Supplier - Who provides the vehicle  
supplierNameCode - Supplier - Who provides the vehicle

**What is the name of the GPS provider?**

They are: Name of the GPS location provider. This provider determines location using satellites. Depending on conditions, this provider may take a while to return a location fix. Requires the permission android.permission.ACCESS\_FINE\_LOCATION.

**Identifying effective parameters and optimal trip**

[**Delivery truck trips data**](https://www.kaggle.com/ramakrishnanthiyagu/delivery-truck-trips-data).

**What ' s The difference between G delivery?**

We understand how important on-time delivery is for you and your customers. Whether it’s a high-value, time-sensitive, or just-in-time shipment, our Guaranteed (G!)

# Classes, Objects,Attributes, Methods

# Object Oriented Python

This section is brief over view of Object Oriented Python.

Here we will explore building blocks of OOP.

Python is a pure Object Oriented Programming language.

Everything in Python is an object.

That means even numbers, strings, functions, modules, classes etc are all objects.

Programming languages like C,C++, Java have primitive data types, which are not objects.

## What is an object ?

An object is a unit of data of a class or type.

Object have one or more attributes and functionality.

Attributes of an objects are also called properties,

and functionality are implemented in methods.

If any of the above lines making no sense, don’t worry,

You will get there, when we look at the, concrete examples of objects.

For the moment, understand objects in Python are no different than, real life objects,

like , Car, Person, Laptop, House, Office, etc.

## What every object has ?

Every object has,

1. a Type
2. an Id
3. Attribute(s), some of them are Method(s)
4. Value(s), of attributes.

If you have done at least a bit of Python programming,

then you have already used objects, without knowing them.

Let’s explore some built-in objects in Python and try to understand.

Example:

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| my\_int = 10 # my\_int variable referring to 10    # check type of object my\_int referring to    print(type(my\_int))    # get id of the object my\_int referring to    print(id(my\_int)) |

Here my\_int  is a variable, which is pointing to place in memory which holds value 10.

We can check what type of object my\_int is referring to by using built-in function **type()**

Also each object has a unique ID, that can be checked by another built-in function **id()**,

In CPython implementation id is nothing but memory address of the object.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| Output:    <class 'int'>    1488478656 |

   
“my\_int” is referencing an object of class int, int is short hand for integer.

it has value 10,

its id is 1488478656

and other properties of my\_int can be obtained by another built-in function **dir()**

[dir](https://www.justlearnpython.com/docs/python-101/built-in-functions/)(obj) built in function lists the properties of a given object.  
 

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| dir(my\_int) |

Check yourself these, by typing above code in Python interpreter.

The below table has few other built-in objects.

For simplicity, only type and Id are shown.

Like above check,  type and id of these objects in Python interpreter.

| **Object** | **Type** | **Id** |
| --- | --- | --- |
| my\_int = 10 | class 'int' | 1488478656 |
| my\_string = "Welcome" | class 'str' | 1488772652424 |
| my\_list = ['Apple', 'Orange','Grapes'] | class 'list' | 1488772571848 |
| my\_bool = True | class 'bool' | 1487986848 |
| my\_tuple = c | class 'tuple' | 1488772335560 |
| my\_dict ={"name" :"User"} | class 'dict' | 1488772640704 |
| my\_type = type(type(10)) | class 'type' | 1488040944 |

   
Please note even “type” is an object,

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| my\_type = type(type(10))    print(type(my\_type))    print(id(my\_type))    dir(my\_type) |

which is of class ‘type’ with a unique id.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| type => <class 'type'>    id => 1488040944 |

Until now, we discussed about objects which are built-in Python.

Let’s look at, how to create user defined class or types.

## Classes

The user defined objects are created by using the **‘class’** keyword.

The ‘class’ is like a blueprint,

that defines the nature of a future object.

By using classes we construct instances.

In Python we have two types of syntax for class creation,

1. Old or classic style

2. New style

Below you will see examples for both styles,

but for the rest of the course, we will fallow New style syntax.

Syntax:

**Old Style:**

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| class ClassName():          statements    or    Parentheses are not mandatory after class name.  class ClassName:          statements |

**New style:**

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| class ClassName(BaseClasses):          statements |

Now lets create a new user defined object type called MyClass,

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| class MyClass(object):        pass  #pass is keyword, equals to do nothing |

By convention class names starts with a capital letter.

In the above example **MyClass**is class name,

the class statement binds the classname and class body.

Currently the class body, we don’t have anything. So MyClass is of no use.

A class can have arbitrary named attributes.

## Body of a class

Body of a class where you normally specify the attributes of a class.

## Attributes

Inside MyClass, we currently just have **pass**.

Which means, this class don’t have any implementation, and is useless.

Let’s add some implementation by adding attributes.

An attribute is a characteristic of an object.

Attributes can be divided into states and behaviors.

Functions can be an attributes of a class, these are called methods.

Behaviors are implemented by using methods,

and states are defined using variables .

The terms attributes and  properties are interchangeable.

## Method

Methods, are functions which are defined inside a class.

An operation, we can perform with the object are implemented as methods..

Method is also know as “callable attribute” of the class.

Let’s get a better understanding of attributes through an example.

The syntax for adding an attribute  to class is:

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| class MyClass(object):        x = 10 |

**x** is an attribute of a class, which is defined,  
   
inside class body and it refers value 10.

We can access the attribute of a class by using dot notation.

## Dot notation

A Python attribute can be accessed by using ( . ) dot notation as below.

The ( . ) dot notation can be used to set and get the attributes.

Dot is used after the class name or object name fallowed by attribute name.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| MyClass.x |

we can print and check what is x holding.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| print(MyClass.x)    output:    10 |

In the above example we have defined attributes inside class body.

But in Python attributes can also be added dynamically,

outside the class body, by using dot notation.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| MyClass.y = 20      print(MyClass.y)    output:    20 |

There is no difference between defining attributes inside or outside of class body.

### Function as attribute

We can also define inside as class, which is called as method.

Lets define a say\_hello() method inside MyClass.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| class MyClass(object):      x = 10        def say\_hello():          print("Hello, There!") |

As function is also an attribute, we call it by using dot notation.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| MyClass.say\_hello()    output:    Hello, There! |

When we use class statement, Python implicitly adds some useful attributes to class,

like **\_\_name\_\_** and **\_\_bases\_\_**

We can check these on our MyClass class.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| print(MyClass.\_\_name\_\_)    print(MyClass.\_\_bases\_\_)      output:    MyClass    (<class 'object'>,) |

Notice \_\_name\_\_ returns name of the class,

and bases returns the inherited class name.

You will learn more about inheritance in later section.

class also has another useful attribute called \_\_dict\_\_

the dict object holds other attributes of the class.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| print(MyClass.\_\_dict\_\_)      {'\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'MyClass' objects>,                '\_\_doc\_\_': None,                '\_\_module\_\_': '\_\_main\_\_',                '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'MyClass' objects>,                'say\_hello': <function \_\_main\_\_.MyClass.say\_hello>,                'x': 10,                'y': 20} |

## Private attributes.

An identifier starting with two underscores inside a class, is private member.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| like \_\_num\_elements = 10 |

We cannot access private members directly by using a dot notation.

See in the below example, x is not a private member,

**\_\_num\_elements**is private member. only x can be accessed with dot notation.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| class Myclass(object):      x = 10      \_\_num\_elements = 10      print(MyClass.x)    output:      print(MyClass.\_\_num\_elements)    output:    ---------------------------------------------------------------------------  AttributeError                            Traceback (most recent call last)  <ipython-input-31-f6eb2d6af7d8> in <module>()        7        8  ----> 9 print(MyClass.\_\_num\_elements)       10       11    AttributeError: type object 'MyClass' has no attribute '\_\_num\_elements' |

But private attributes can be accessed by using methods.

## Creating Instance from user defined class

To create an instance, use class as if it is function.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| obj\_1 = MyClass()    obj\_2 = MyClass() |

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| # creates an object, returns its reference    obj\_1 = MyClass()    # Address of the first object    print("Address of first object :", obj\_1)    output:    Address of first object : <\_\_main\_\_.MyClass object at 0x0000026F6DD9DB00> |

Crating your second object.

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| obj\_2 = MyClass()    # Address of the second object    print("Address of second object :",obj\_2)      Output:    Address of second object : <\_\_main\_\_.MyClass object at 0x0000026F6DDA85C0> |

In above example MyClass(),  
   
each time constructs an object and returns a reference to that object.

obj\_1 points to an object at  0x0000026F6DD9DB00 in memory location.

obj\_2 points to an object at  0x0000026F6DDA85C0 in memory location.

## Module Vs Class

We will explore more about attributes in the next lesson,

But before that lets understand the difference between classes and modules.

Let’s understand.

## What is the difference between modules and classes ?

Python modules are files containing Python code.

Modules can be executed individually, or imported in some other module and get executed.

A module may contain, classes, properties or functions any of these or all of these.

Where as a class is Python code inside a module.

Let’s look at an example of accessing a class from a module.

Example:

[?](https://www.justlearnpython.com/docs/object-oriented-programming/classes-objects-attributes-methods/)

|  |
| --- |
| from decimal import Decimal  # Decimal is a class name starts with 'D'  num1 = Decimal('1.5')    num2 = Decimal('2.5')    print(num1 + num2) |

Output:

4.0

Here **Decimal**is class inside **decimal**module,

which is part of standard Python installation.

in the above example, Decimal class is imported from decimal module.

In the next section we will look at some more examples of class and objects.

In regular English, a query means a **request for information**. What is a query in computer programming then? Simple, it’s the same – except the information is retrieved from a database. This is handy for data manipulation – adding, removing, and changing data.

**What is the purpose of a query?**

Primarily, queries are used to **find specific data by filtering specific criteria**. Queries can also calculate or summarize data, as well as automate data management tasks. Other queries include parameter, totals, crosstab, make table, append, update and delete.

**What does it mean to query data?**

Data querying is the process of **asking questions of data in search of a specific answer**.

**What is query and query language?**

Query languages or data query languages (DQLs) are **computer languages used to make queries in databases and information systems**.

**What is the use of query by example?**

Query by example is a query language used in **relational databases** that allows users to search for information in tables and fields by providing a simple user interface where the user will be able to input an example of the data that he or she wants to access. The principle of QBE is that it is merely an abstraction between the user and the real query that the database system will receive.

In regular English, a query means a request for information. What is a query in computer programming then? Simple, it’s the same – except the information is retrieved from a database. This is handy for data manipulation – adding, removing, and changing data.  That’s how we’ll use this word in the article.

However, you don’t just type in a random ‘request.’ You write your query based on a set of pre-defined code, so your database understands the instruction. We refer to this code as the query language.

The standard for database management is Structured Query Language (SQL). Remember, [**SQL is different from MySQL**](https://www.hostinger.com/tutorials/what-is-mysql): the former is the query language, the latter is the software **which uses** the language. While it is true that SQL is the most popular choice among database software, it is definitely not the only one. Several others are AQL, Datalog, and DMX.

Regardless, these languages make database communication easy.

## ****How Does Query Work****?

Let’s say that you want to order an Americano at Starbucks. You make a request by saying “Can I have an Americano?”. The Barista will understand **the meaning** of your request and give you the ordered item.

A query works the same way. It gives meaning to the code used in any query language. Be it SQL or anything else, both the user and the database can always exchange information as long as they ‘speak’ the same language.

You may now think that placing a query is the only way to request data. In fact, quite a few database software options let you use other methods. The most popular ones are:

* **Using available parameters**  
  The software, by default, has lists of parameters on their menu. Users can choose one, and the system will then guide you to produce the desired output. It’s easy, but not flexible and offers limited operations.
* **Query by example**  
  The system will show you a set of code with some blank areas, in which you can write and specify the fields and values of your data.
* **Query language**  
  This is what we’ve been talking about. You have to write the queries from scratch whenever you want to manipulate data. This method requires understanding the query language used by your database software. Although it is complex, it gives you full control over your data.