**OOPS – Object oriented programming System language**

Oops is a concept or approach to build application

# **Class**

* Blue print/ objective / plan /datatype
* **E.g.:** class ExpenseTracker () - This class has a blueprint of expense tracking like variables or attribute, function, methods. Attributes like date, description, amount, credit, debit
* Class in python works fast since it is dynamical typed programming language
* Every class in python is inherited from object class

**Class format:**

1. Class
2. Attribute
3. Methods
4. Object creation

Eg: <https://github.com/bsdharshini/Oops-1/blob/main/a_class__init__example.py>

## def \_\_init\_\_(self,) method:

* when object calls the class at that time initializer function runs by default.
* This init method in inherited from object class
* This method takes values that is passed from outside in object creation and assign it to class.
* Should be used when using a class

Eg:

def \_\_init\_\_(self, rate, time, amount):

        self.rate = rate

        self.time = time

        self.amount = amount

    # left side assigning to class - right side getting value from obj creation

## Which operator used to access attribute, properties and methods of an object?

* full stop '.'

eg:

obj1.rate

obj1.calculate ()

## Attribute:

* Has value that is passed by user at the time of calling the class
* This attributes cannot be accessible without object outside the class

Type of attributes:

1. Class attributes:

* Attributes outside \_\_init\_\_ functions without self
* Will be same across all object
* Inside class this attributes are only accessible with class name

1. Instance/ object attribute:

* Defined inside \_\_init\_\_ with self keyword
* It may vary based on object

Eg: <https://github.com/bsdharshini/Oops-1/blob/main/c_type_of_attribute.py>

## What are refined methods

1. \_\_dict\_\_ : Returns dictionary of attributes and its value

Eg: print(obj1.\_\_dict\_\_)

o/p: {‘Rate’: 2, ‘Time’:1, ‘Amount’:1000}

1. getattr : Return asked attribute value

Eg: print(obj1, ‘Amount’)

o/p: 1000

1. hasattr : return true if attribute is present

Eg: hasattr(obj1, ‘ one’)

o/p: False

1. delattr : delete the specified attribute

Eg: delattr(obj1, ‘Amount’)

o/p: Does not return anything but deletes the attribute

## Type of methods:

1. **Instance method:**

* Instance method can be accessed only by object/ instance
* Method which have access to init method using self keyword or have access to instance attribute

Eg: <https://github.com/bsdharshini/Oops-1/blob/main/d_instance_method.py>

1. **Static method:**

* Should have decorator @staticmethod above method definition
* No need for self
* Can have arbitrary parameter. Since no self cannot access instance specific method or attribute
* It is different from OOPS static
* Mostly not used. Use only when needed

Eg: <https://github.com/bsdharshini/Oops-1/blob/main/e_static_method.py>

1. **Class method:**

* Class method should have ‘**cls’** keyword like self in instance method
* Using clas we can use static method inside the class
* Class method is not bounded by object
* Changes made in class method will reflect in all object
* Class method can also known as **Factory method**

Eg: <https://github.com/bsdharshini/Oops-1/blob/main/f_class_method.py>

# Access modifier:

In python we don’t have prefect access modifier like in other language

1. **Public**: Accessible inside and outside of class. By default all attribute and methods are public
2. **Private**: Access only inside class and subclass method. Private method or attribute should have two underscore before (\_\_getDetail()).

Eg: How to access private variable or method using object: <https://github.com/bsdharshini/Oops-1/blob/main/g_access_private_using%20object.py>

1. **Protected**: Accessible only inside class, its subclass and inherited child class . Protected method or attribute should have single underscore before (\_getDetail()).

Eg: <https://github.com/bsdharshini/Oops-1/blob/main/h_protected_methods.py>