OOPS (Object Oriented Programming System) is a base of programming languages like Java and Python.

# OOPS Core Principles

* [Encapsulation](https://www.google.com/search?sca_esv=25c2551b4f45e1f2&cs=0&q=Encapsulation&sa=X&ved=2ahUKEwj_2L6J3oeQAxUAWEEAHRltGpAQxccNegQIIhAB&mstk=AUtExfBnSQvgvATJN_kCSc8zedU-qoLt0XZeNt1D0KKxHtr_FL8PT_Zc--TdkZH_W0AhVheX5SC3ozmc48JrJH1r1EBNs56hGi4njhMBdVFpFSuc9okWQ3nqKI8CXyKNz3FIkpH7QWyrdO3ySnUUT6C5wq2xGCjUH0vUmyF4gr1rwOUoknpyhMYBYzoYBC-2Dn9Q0P1n&csui=3)
* [Inheritance](https://www.google.com/search?sca_esv=25c2551b4f45e1f2&cs=0&q=Inheritance&sa=X&ved=2ahUKEwj_2L6J3oeQAxUAWEEAHRltGpAQxccNegQILBAB&mstk=AUtExfBnSQvgvATJN_kCSc8zedU-qoLt0XZeNt1D0KKxHtr_FL8PT_Zc--TdkZH_W0AhVheX5SC3ozmc48JrJH1r1EBNs56hGi4njhMBdVFpFSuc9okWQ3nqKI8CXyKNz3FIkpH7QWyrdO3ySnUUT6C5wq2xGCjUH0vUmyF4gr1rwOUoknpyhMYBYzoYBC-2Dn9Q0P1n&csui=3)
* [Polymorphism](https://www.google.com/search?sca_esv=25c2551b4f45e1f2&cs=0&q=Polymorphism&sa=X&ved=2ahUKEwj_2L6J3oeQAxUAWEEAHRltGpAQxccNegQILhAB&mstk=AUtExfBnSQvgvATJN_kCSc8zedU-qoLt0XZeNt1D0KKxHtr_FL8PT_Zc--TdkZH_W0AhVheX5SC3ozmc48JrJH1r1EBNs56hGi4njhMBdVFpFSuc9okWQ3nqKI8CXyKNz3FIkpH7QWyrdO3ySnUUT6C5wq2xGCjUH0vUmyF4gr1rwOUoknpyhMYBYzoYBC-2Dn9Q0P1n&csui=3)
* [Abstraction](https://www.google.com/search?sca_esv=25c2551b4f45e1f2&cs=0&q=Abstraction&sa=X&ved=2ahUKEwj_2L6J3oeQAxUAWEEAHRltGpAQxccNegQILRAB&mstk=AUtExfBnSQvgvATJN_kCSc8zedU-qoLt0XZeNt1D0KKxHtr_FL8PT_Zc--TdkZH_W0AhVheX5SC3ozmc48JrJH1r1EBNs56hGi4njhMBdVFpFSuc9okWQ3nqKI8CXyKNz3FIkpH7QWyrdO3ySnUUT6C5wq2xGCjUH0vUmyF4gr1rwOUoknpyhMYBYzoYBC-2Dn9Q0P1n&csui=3)

# Object-Oriented Programming (OOP) in Finance

# 1)Encapsulation

class Bank\_Ac:

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

self.\_\_balance = balance # Private attribute

self.\_\_method = method # Private attribute

def get\_method(self):

return self.\_\_method

def get\_balance(self):

return self.\_\_balance

my\_account = Bank\_Ac(“500 gbp”, “withdraw”)

print(my\_account.get\_balance())

# Output: 500 gbp

# 2)Abstraction

When a client has to use your object, the client need not import your class or have your class definition in his jar, he/she can just import the abstract class or interface and accept your object as an argument. So client will still be able to use your object with parent reference but can't actually see (or need) the functional implementation of the class.

Client/User will only see below code

abstract class Bank\_Ac {

abstract get\_method(); }

# 3)Inheritance

# create an object of the subclass and it inherits from original class Bank\_Ac

Class Savings\_Ac(savings\_ac):

def get\_balance(self):

print("My balance is ", self.get\_balance)

# create an object of the subclass

Wamika\_Ac = Savings\_Ac()

# access superclass attribute and method

Wamika\_Ac.balance = "200 gbp"

# call subclass method

Wamika\_Ac.get\_balance()

# Output: 200 gbp

# 4) Polymorphism

Polymorphism allows methods to do different things based on the object they are acting upon. For example, the deposit method can behave differently in a Bank\_Ac and a Savings\_Ac

Eg. Interest rate formula for savings account is different than standard Bank account but Savings\_Ac is object of class Bank\_Ac