**Polymorphism** is one of the four fundamental pillars of Object-Oriented Programming (OOP), the other three being **Encapsulation**, **Abstraction**, and **Inheritance**. The term **Polymorphism** means "many forms." In , it allows a single function or method to operate on different types of objects, and the behavior of the function or method changes based on the type of the object.

**Types of Polymorphism in :**

1. **Method Overriding**: When a subclass provides a specific implementation of a method that is already defined in its superclass.
2. **Method Overloading**: doesn't support method overloading directly (like other languages such as Java or C++). However, we can simulate method overloading by using default arguments or variable-length arguments (\*args and \*\*kwargs).

**Key Concepts for Polymorphism in :**

1. **Inheritance**: Polymorphism is often used with inheritance where a subclass overrides a method of its parent class.
2. **Duck Typing**: In , polymorphism can be achieved without explicitly inheriting classes. If an object can respond to a method, it will be treated as an instance of the required type (this is called Duck Typing).
3. **Dynamic Typing**: 's dynamic typing allows polymorphism to work by checking the type of objects during runtime.

**Real-world Example: Payment Processing System**

Let’s build a simple payment processing system where different types of payments (like credit card, PayPal, and bank transfer) can be processed using polymorphism. Each payment method will have its own implementation of a process\_payment() method, but they will all share the same interface.

**Steps Needed for the Example:**

1. **Base Class** (PaymentMethod) that defines a common interface (method) for processing payments.
2. **Derived Classes** (CreditCardPayment, PayPalPayment, BankTransferPayment) that implement the process\_payment() method specific to each payment method.
3. Demonstrate **polymorphism** by processing payments of different types using a common interface.

**Code Example:**

# Base class: PaymentMethod (defines a common interface for all payment methods)

class PaymentMethod:

def process\_payment(self, amount):

raise NotImplementedError("This method should be overridden by subclasses.")

# Derived class 1: CreditCardPayment (overrides the process\_payment method)

class CreditCardPayment(PaymentMethod):

def \_\_init\_\_(self, card\_number, card\_holder):

self.card\_number = card\_number

self.card\_holder = card\_holder

def process\_payment(self, amount):

# Simulate processing credit card payment

print(f"Processing credit card payment of {amount} for card holder {self.card\_holder}.")

# Derived class 2: PayPalPayment (overrides the process\_payment method)

class PayPalPayment(PaymentMethod):

def \_\_init\_\_(self, paypal\_email):

self.paypal\_email = paypal\_email

def process\_payment(self, amount):

# Simulate processing PayPal payment

print(f"Processing PayPal payment of {amount} for email {self.paypal\_email}.")

# Derived class 3: BankTransferPayment (overrides the process\_payment method)

class BankTransferPayment(PaymentMethod):

def \_\_init\_\_(self, bank\_account\_number):

self.bank\_account\_number = bank\_account\_number

def process\_payment(self, amount):

# Simulate processing bank transfer payment

print(f"Processing bank transfer payment of {amount} to account {self.bank\_account\_number}.")

# Function that processes a payment using polymorphism

def process\_order(payment\_method, amount):

payment\_method.process\_payment(amount)

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

# Create objects for different payment methods

credit\_card\_payment = CreditCardPayment("1234-5678-9876-5432", "Alice Johnson")

paypal\_payment = PayPalPayment("alice\_johnson@example.com")

bank\_transfer\_payment = BankTransferPayment("001234567890")

# Process payments using polymorphism (different methods for different payment types)

print("Order 1: Credit Card Payment")

process\_order(credit\_card\_payment, 100)

print("\nOrder 2: PayPal Payment")

process\_order(paypal\_payment, 200)

print("\nOrder 3: Bank Transfer Payment")

process\_order(bank\_transfer\_payment, 300)

**Explanation:**

1. **Base Class (PaymentMethod)**:
   * The PaymentMethod class defines a method process\_payment() which is meant to be overridden by its subclasses. The method is abstract and raises a NotImplementedError to ensure that it is implemented by the subclasses.
2. **Derived Classes**:
   * **CreditCardPayment**: Implements process\_payment() for credit card transactions.
   * **PayPalPayment**: Implements process\_payment() for PayPal transactions.
   * **BankTransferPayment**: Implements process\_payment() for bank transfer transactions.

Each derived class provides its own specific behavior for processing payments, but they all share the same interface (process\_payment()).

1. **Polymorphism in Action**:
   * The process\_order() function accepts an object of PaymentMethod type (or any of its subclasses) and calls the process\_payment() method. Even though the objects are of different classes (CreditCardPayment, PayPalPayment, BankTransferPayment), they all provide their own version of the process\_payment() method, which is called dynamically at runtime based on the object type. This is polymorphism.

**Output:**

sql

Order 1: Credit Card Payment

Processing credit card payment of 100 for card holder Alice Johnson.

Order 2: PayPal Payment

Processing PayPal payment of 200 for email alice\_johnson@example.com.

Order 3: Bank Transfer Payment