class Car:

def \_\_init\_\_(self, make, model):

self.make = make

self.model = model

car1 = Car("Toyota", "Camry")

print(car1.make)

print(car1.model)

Constructor with Parameters:

o Define a class Book with a constructor that takes title and author as parameters

and initializes these attributes. Create an instance of the Book class and print its

attributes.

class Book:

def \_\_init\_\_(self, title, author):

self.title = title

self.author = author

book1 = Book("The Great Gatsby", "F. Scott Fitzgerald")

print(book1.title)

print(book1.author)

Constructor with Default Parameters:

o Define a class Person with a constructor that takes name and age as parameters.

Set default values for the parameters to &quot;Unknown&quot; and 0, respectively. Create

instances of the Person class with and without passing arguments, and print their

attributes.

class Person:

def \_\_init\_\_(self, name="Unknown", age=0):

self.name = name

self.age = age

person1 = Person()

print(person1.name)

print(person1.age)

Instance and Class Variables:

o Define a class Employee with instance variables name and salary, and a class

variable company\_name. Create instances of the Employee class, modify the class

variable, and observe the changes.

class Employee:

company\_name = "ABC Company"

def \_\_init\_\_(self, name, salary):

self.name = name

self.salary = salary

self.company\_name = "spc"

employee1 = Employee("John Doe", 50000)

employee2 = Employee("Jane Smith", 60000)

print(employee1.name)

print(employee1.salary)

print(employee1.company\_name)

Instance Methods:

o Define a class Circle with an instance variable radius. Implement an instance

method area that calculates and returns the area of the circle. Create an instance

of the Circle class and call the area method.

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return 3.14 \* self.radius \*\* 2

circle1 = Circle(5)

print(circle1.area())

Class Methods:

o Define a class Student with an instance variable name and a class variable

school\_name. Implement a class method set\_school\_name to change the class

variable school\_name. Create instances of the Student class and use the class

method to change the school name.

class Student:

school\_name = "ABC School"

def \_\_init\_\_(self, name):

self.name = name

@classmethod

def set\_school\_name(cls, new\_name):

cls.school\_name = new\_name

return cls.school\_name

student1 = Student("John Doe")

student2 = Student("Jane Smith")

print(student1.school\_name)

print(student2.school\_name)

Student.set\_school\_name("XYZ School")

print(student1.school\_name)

print(student2.school\_name)

7. Static Methods:

o Define a class MathUtils with a static method add that takes two numbers and

returns their sum. Call the static method without creating an instance of the class.

class MathUtils:

@staticmethod

def add(a, b):

return a + b

print(MathUtils.add(5, 3))

Passing Members of One Class to Another Class:

o Define a class Address with attributes street and city. Define a class Person

with attributes name and address. Pass an instance of the Address class as an

argument to the Person class and print the person&#39;s details.

class Address:

def \_\_init\_\_(self, street, city):

self.street = street

self.city = city

class Person:

def \_\_init\_\_(self, name, address):

self.name = name

self.address = address

address1 = Address("123 Main St", "New York")

person1 = Person("John Doe", address1)

print(person1.name)

print(person1.address.street)

print(person1.address.city)

Combination of Methods:

o Define a class BankAccount with instance variables account\_number and

balance, a class variable bank\_name, an instance method deposit, a class

method set\_bank\_name, and a static method interest\_rate that returns a fixed

interest rate. Create an instance of the BankAccount class and demonstrate the use

of these methods.

class BankAccount:

bank\_name = "ABC Bank"

def \_\_init\_\_(self, account\_number, balance):

self.account\_number = account\_number

self.balance = balance

def deposit(self, amount):

self.balance += amount

@classmethod

def set\_bank\_name(cls, new\_name):

cls.bank\_name = new\_name

@staticmethod

def interest\_rate():

return 0.3

account1 = BankAccount("123456789", 1000)

account1.deposit(500)

print(account1.balance)

BankAccount.set\_bank\_name("XYZ Bank")

print(BankAccount.bank\_name)

print(account1.bank\_name)

print(BankAccount.interest\_rate())