OOP Fundamentals in Python

1. OOP Fundamentals (Objects & Classes)

Object-Oriented Programming (OOP) is a paradigm that uses objects and classes to structure programs. A class is a blueprint for creating objects (instances). Objects have attributes (data) and methods (functions).

Syntax Example:

```
class ClassName:
    def __init__(self, attribute):
        self.attribute = attribute

# Creating an object
obj = ClassName("value")
print(obj.attribute)
```

Assignment 1:

Create a class `Car` with attributes `brand` and `model`. Instantiate 2 objects and print their details.

Solution:

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

car1 = Car("Toyota", "Corolla")
car2 = Car("Honda", "Civic")

print(car1.brand, car1.model)
print(car2.brand, car2.model)
```

2. Attributes and Methods

Attributes are variables that belong to a class or object. Methods are functions defined inside a class that operate on its objects.

Syntax Example:

```
class Student:
```

```
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print(f"Name: {self.name}, Age: {self.age}")

student1 = Student("Alice", 20)
student1.display()
```

Assignment 2:

Create a class `Book` with attributes `title` and `author`. Add a method to display book details.

Solution:

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def display(self):
        print(f"Book: {self.title}, Author: {self.author}")

book1 = Book("1984", "George Orwell")
book1.display()
```

3. Instance Methods vs Class Methods vs Static Methods

- Instance methods work with instance data using 'self'.
- Class methods work with class data using 'cls'.
- Static methods do not use self or cls; they are utility functions inside a class.

Syntax Example:

```
class Demo:
    class_var = "Class Level Variable"

    def __init__(self, value):
        self.value = value

    # Instance Method
    def show_value(self):
```

```
        print("Instance Value:", self.value)

    # Class Method
    @classmethod
    def show_class_var(cls):
        print("Class Variable:", cls.class_var)

    # Static Method
    @staticmethod
    def greet():
        print("Hello from Static Method!")

# Usage
obj = Demo("Instance Data")
obj.show_value()
Demo.show_class_var()
Demo.greet()
```

Assignment 3:

Create a class `MathOps` with:
- An instance method to add two numbers.
- A class method to return the class name.
- A static method to return the square of a number.

Solution:

```
class MathOps:
    def add(self, a, b):
        return a + b

    @classmethod
    def get_class_name(cls):
        return cls.__name__

    @staticmethod
    def square(num):
        return num ** 2

math = MathOps()
print("Addition:", math.add(5, 3))
print("Class Name:", MathOps.get_class_name())
print("Square:", MathOps.square(4))
```