Why do we need to know Object Oriented Programming?

Object Oriented Programming

Object Oriented Programming

Groups all the data and code within a single structure *(Class)*

Object Oriented Programming

Groups all the data and code within a single structure *(Class)*

Encapsulation

Polymorphism

Abstraction

Inheritance

Encapsulation in OOP

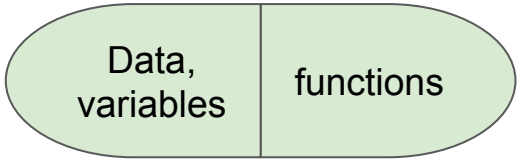
Encapsulation

Bundling of data with the functions that operate on that data into a single unit.

Encapsulation in OOP

Encapsulation

Bundling of data with the functions that operate on that data into a single unit.



Encapsulation in OOP

Encapsulation

Bundling of data with the functions that operate on that data into a single unit.

Benefits:

- Keeps the data and the code safe from external interference
- Prevents accidental modification of data
- Better for unit testing

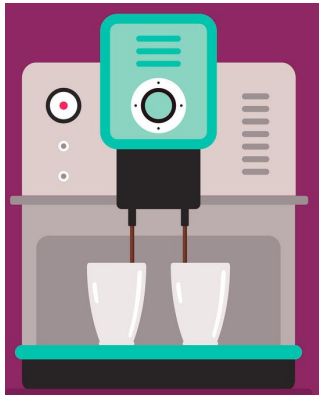
Abstraction in OOP

Abstraction	Hiding unwanted details while showing most essential information.

Abstraction in OOP

Abstraction

Hiding unwanted details while showing most essential information.



Analytics Vidhya

Abstraction in OOP

Abstraction

Hiding unwanted details while showing most essential information.

Benefits:

- Ease of use for the end user
- Reduce complexity of design
- Increases Security and confidentiality

Inheritance in OOP

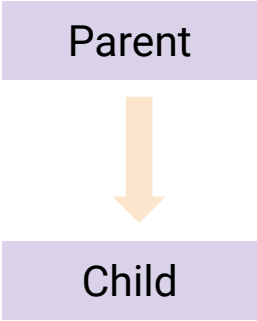
Inheritance

New class can inherit methods and attributes from existing classes.

Inheritance in OOP

Inheritance

New class can inherit methods and attributes from existing classes.



Analytics Vidhya

Inheritance in OOP

Inheritance

New class can inherit methods and attributes from existing classes.

Benefits:

- Code Reusability and Readability
- Avoid code repetition
- Program structure is short and concise

Polymorphism in OOP

Polymorphism

Refers to defining multiple functionalities under the same name.

Polymorphism in OOP

Polymorphism

Refers to defining multiple functionalities under the same name.

```
class triangle_operations()
    def area_calc()

class circle_operations()
    def area_calc()
```

Analytics Vidhya

Polymorphism in OOP

Polymorphism

Refers to defining multiple functionalities under the same name.

Benefits:

- Increases Code Reusability
- Single variable can be used to store multiple data types

Thank You

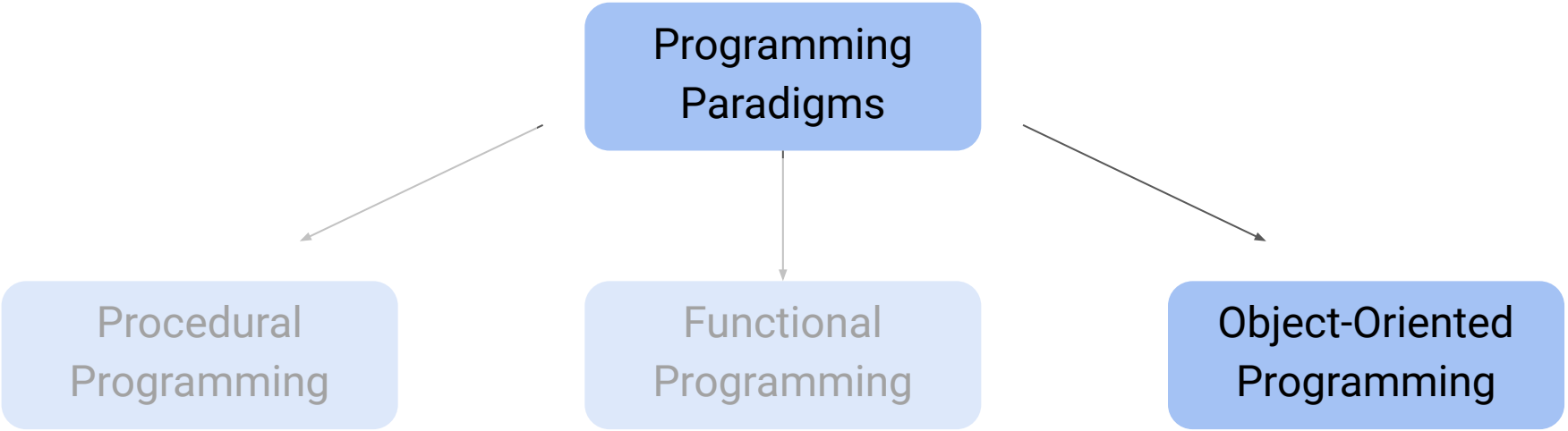
Analytics Vidhya

Introduction to Object Oriented Programming

Analytics Vidhya

Recap: Programming Paradigms

A programming paradigm is a style or way of programming



Analytics Vidhya

Object Oriented Programming

- Groups all the data and code within a single structure *(Class)*
- All data is stored as *Classes* and *objects*
- Modular and Organised code
- Easier to reuse code and reduces redundant code blocks
- Follows a Bottom-Up approach

Analytics Vidhya

Object Oriented Programming

Example:

Analytics Vidhya

Object Oriented Programming

Example:

All cars would have different

- Vehicle Number
- Fuel Type
- Fuel Capacity
- Torque

Object Oriented Programming

Example:

All cars would have different

- Vehicle Number
- Fuel Type
- Fuel Capacity
- Torque

Objects or Instances

Instance Variables

Analytics Vidhya

Object Oriented Programming

Example:

All cars would have different

- Vehicle Number
- Fuel Type
- Fuel Capacity
- Torque

✔ Cost of the car

✔ Power (kW)

Methods
(or functions)

Objects or
Instances

Instance Variables

Analytics Vidhya

Object Oriented Programming: Instance

Object/
Instance

Car 1:

Car Number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Analytics Vidhya

Object Oriented Programming: Instance Variables

Instance

Car 1:

Instance Variables

Car Number: 1922

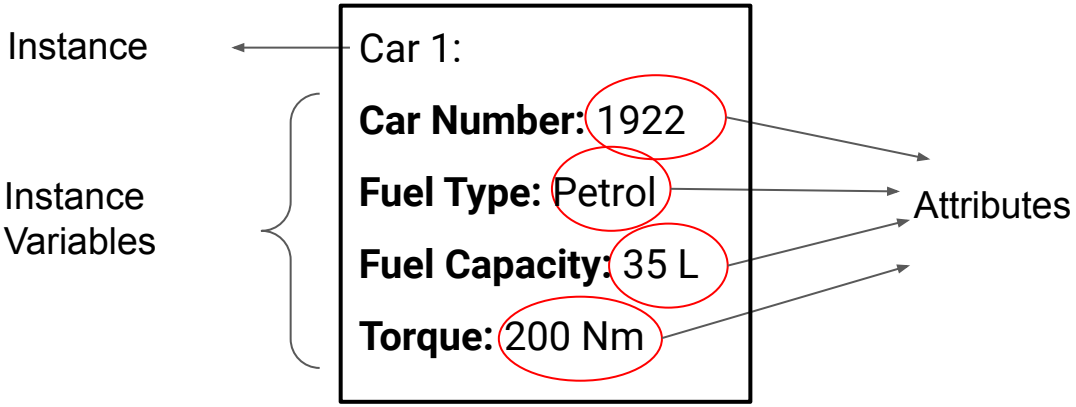
Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Analytics Vidhya

Object Oriented Programming: Attributes



Object Oriented Programming: Methods

Instance

Car 1:

Instance Variables

Car Number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Attributes

✔ Cost of the car

✔ Power (kW)

Methods
(or functions)

Note: Methods are functions defined within a class!

Analytics Vidhya

Thank You

Analytics Vidhya

Defining Classes in Python

Quick Recap

Instance

Car 1:

Instance Variables

Car Number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Attributes

- ✔ Cost of the car
- ✔ Power (kW)

Methods
(or functions)

Analytics Vidhya

Defining Classes in Python

Storing the following information for cars

- Vehicle Number
- Type of fuel
- Engine Capacity
- Torque

Defining Classes in Python

class **Cars**: ← Class Name

Defining Classes in Python

```
class Cars:
        def __init__
```

Initializing Attributes

Defining Classes in Python

```
class Cars:
    def __init__(self, number, fuel_type, capacity, torque):
```

Initializing instance variables

Initializing Instance

Analytics Vidhya

Defining Classes in Python

```
class Cars:
    def __init__(self, number, fuel_type, capacity, torque):
```

Car 1:

Car Number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Analytics Vidhya

Defining Classes in Python

```
class Cars:
    def __init__(self, number, fuel_type, capacity, torque):
```

Car 1:

Car Number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Analytics Vidhya

Defining Classes in Python

```
class Cars:
    def __init__(self, number, fuel_type, capacity, torque):
        self.number = number
        self.fuel_type = fuel_type
        self.capacity = capacity
        self.torque = torque
```

Defining Instance Variables

Thank You

Analytics Vidhya

Attributes and its Types

Quick Recap

Instance

Car 1:

Instance Variables

Car Number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Attributes

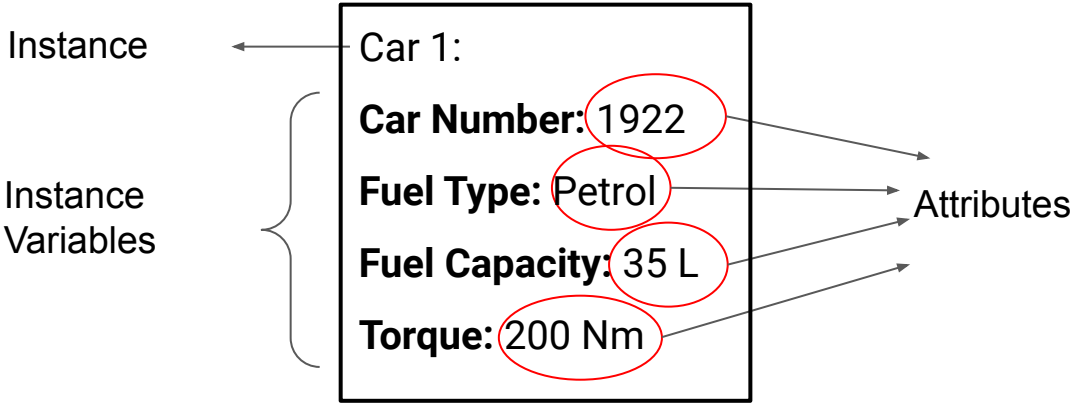
- ✔ Cost of the car
- ✔ Power (kW)

Methods
(or functions)

Analytics Vidhya

Object Oriented Programming: Attributes

- Attributes are the data stored inside the class or instance



Analytics Vidhya

Types of Attributes

- Attributes are the data stored inside the class or instance
- In Object Oriented Programming, we have
  - Class Attributes
  - Instance Attributes

Types of Attributes

- Attributes are the data stored inside the class or instance
- In Object Oriented Programming, we have
  - **Class Attributes:** Shared between all objects of this class
  - Instance Attributes

Analytics Vidhya

Types of Attributes

- Attributes are the data stored inside the class or instance
- In Object Oriented Programming, we have
  - **Class Attributes:** Shared between all objects of this class
  - **Instance Attributes:** Belongs to one and only one object

Analytics Vidhya

Types of Attributes

Car 1:

Car number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Car 2:

Car number: 1719

Fuel Type: Petrol

Fuel Capacity: 30 L

Torque: 150 Nm

Car 3:

Car number: 2349

Fuel Type: Diesel

Fuel Capacity: 40 L

Torque: 250 Nm

Instance Attributes

Types of Attributes

Num_wheels = 4
Year_manufacture = 2020

Class Attributes

Car 1:

Car number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Car 2:

Car number: 1719

Fuel Type: Petrol

Fuel Capacity: 30 L

Torque: 150 Nm

Car 3:

Car number: 2349

Fuel Type: Diesel

Fuel Capacity: 40 L

Torque: 250 Nm

Instance Attributes

Instance Variables and Class Variables

Num_wheels = 4
Year_manufacture = 2020

Class Variables

Car 1:

Car number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Car 2:

Car number: 1719

Fuel Type: Petrol

Fuel Capacity: 30 L

Torque: 150 Nm

Car 3:

Car number: 2349

Fuel Type: Diesel

Fuel Capacity: 40 L

Torque: 250 Nm

Instance Variables

Notebook

Thank You

Analytics Vidhya

Methods and its Types

Quick Recap

Instance

Car 1:

Instance Variables

Car Number: 1922

Fuel Type: Petrol

Fuel Capacity: 35 L

Torque: 200 Nm

Attributes

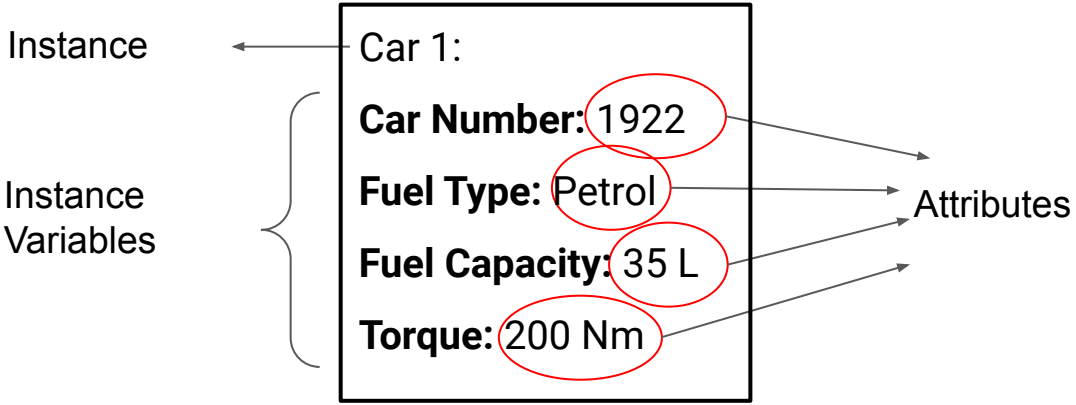
- ✔ Cost of the car
- ✔ Power (kW)

Methods
(or functions)

Analytics Vidhya

Object Oriented Programming: Methods

- Methods are functions associated with classes



- ✔ Cost of the car
- ✔ Power (kW)

Methods

Analytics Vidhya

Object Oriented Programming: Methods

- Methods are functions associated with classes

Power (KW) $= \frac{\text{Torque}_{\text{Nm}} * \text{Speed}_{\text{RPM}}}{9550}$

```
# defining a method to calculate power
def power_calc(self):
    return (self.torque *self.speed_rpm)/9550
```

- ✔ Cost of the car
- ✔ Power (kW)

Methods

Analytics Vidhya

Notebook

Types of Methods

- Methods are functions associated with classes
- Methods can also be of four types:
  - Instance Method
  - Class Method
  - Static Method
  - Special Methods

Types of Methods

- Methods are functions associated with classes
- Methods can also be of four types:
  - **Instance Method:** Take instance as an input

```
# defining a method
def power_calc(self):
    return (self.torque *self.speed_rmp)/9550
```

Analytics Vidhya

Types of Methods

- Methods are functions associated with classes
- Methods can also be of four types:
  - **Instance Method:** Take instance as an input
  - **Class Method:** Take class as an input

Types of Methods

- Methods are functions associated with classes
- Methods can also be of four types:
  - **Instance Method:** Take instance as an input
  - **Class Method:** Take class as an input
  - **Static Method:** Neither instance nor class as input

Analytics Vidhya

Types of Methods

- Methods are functions associated with classes
- Methods can also be of four types:
  - **Instance Method:** Take instance as an input
  - **Class Method:** Take class as an input
  - **Static Method:** Neither instance nor class as input
  - **Special Method:** Providing existing operators with user defined meaning

Analytics Vidhya

Notebook

Thank You

Analytics Vidhya

Special Method

Types of Methods

- Methods are functions associated with classes
- Methods can also be of four types:
  - **Instance Method:** Take instance as an input
  - **Class Method:** Take class as an input
  - **Static Method:** Neither instance nor class as input
  - **Special Method:** Providing existing operators with user defined meaning

Analytics Vidhya

Special Methods

- Special Methods are set of predefined methods
- Surrounded by a double underscores (Example: __init__)

Special Methods

Performing Addition

5 + 3

Operator **+**

Length of string

len('data science')

Function **len**

Analytics Vidhya

Special Methods

Performing Addition

5 + 3

↕

int.**__add__**(5,3)

Length of string

len(‘data science’)

↕

str.**__len__(‘data science’)**

Special Methods

Performing Addition on Integers

5 + 3

8

int.**__add__**(5,3)

Performing Addition on Strings

‘a’ + ‘b’

ab

str.**__add__(‘a’, ‘b’)**

Special Methods

- Special Methods are set of predefined methods
- Surrounded by a double underscores (Example: __init__)
- Special Methods are used for ‘Operator Overloading’

Special Methods

- Special Methods are set of predefined methods
- Surrounded by a double underscores (Example: __init__)
- Special Methods are used for ‘Operator Overloading’

What is **‘Operator Overloading’?**

Analytics Vidhya

Special Methods

- Special Methods are set of predefined methods
- Surrounded by a double underscores (Example: __init__)
- Special Methods are used for ‘Operator Overloading’

What is **‘Operator Overloading’?**

- Providing Existing operators with user defined meanings

Analytics Vidhya

Working Example: Operator Overloading

Cost: 5L

Cost: 5.5L

Cost: 8.5L

Notebook

Commonly used Special Methods

- **Mathematical Operators:** __add__ , __sub__ , __mul__

Commonly used Special Methods

- **Mathematical Operators:** __add__ , __sub__ , __mul__
- **Emulating container types:** __len__ , __getitem__ , __iter__

Analytics Vidhya

Commonly used Special Methods

- **Mathematical Operators:** __add__ , __sub__ , __mul__
- **Emulating callable objects:** __call__

Analytics Vidhya

Commonly used Special Methods

- **Mathematical Operators:** __add__ , __sub__ , __mul__
- **Emulating callable objects:** __call__
-

Commonly used Special Methods

- **Mathematical Operators:** __add__ , __sub__ , __mul__
- **Emulating callable objects:** __call__
- **Emulating container types:** __len__ , __getitem__ , __iter__
- **Basic customisation:** __init__, __str__ , __repr__

Analytics Vidhya

Commonly used Special Methods

- **Mathematical Operators:** __add__ , __sub__ , __mul__
- **Emulating callable objects:** __call__
- **Emulating container types:** __len__ , __getitem__ , __iter__
- **Basic customisation:** __str__ , __repr__
- **Customising attribute access:** __getattribute__

Analytics Vidhya

Special Methods: Operator Overloading

Performing Addition

5 + 3

↕

int.**__add__**(5,3)

Length of string

len('data science')

↕

str.**__len__('data science')**

```
# special method to add costs
def __add__(first, second):
  total_cost = first.cost + second.cost
  return print('Total cost will be', total_cost, 'Rs')
```

Analytics Vidhya

Special Methods

Performing Addition on Integers

5 + 3

↑

8

Performing Addition on Strings

‘a’ + ‘b’

↑

ab

Special Methods

Performing Addition on Integers

5 + 3

8

int.**__add__**(5,3)

Performing Addition on Strings

‘a’ + ‘b’

ab

str.**__add__(‘a’, ‘b’)**