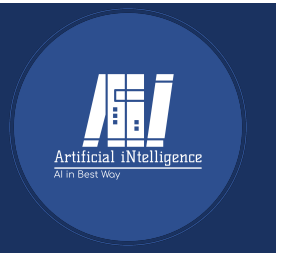

# OBJECT ORIENTED PROGRAMMING

PROBLEM SOLVING

KEVIN HARVEY

# EXAMPLE 1

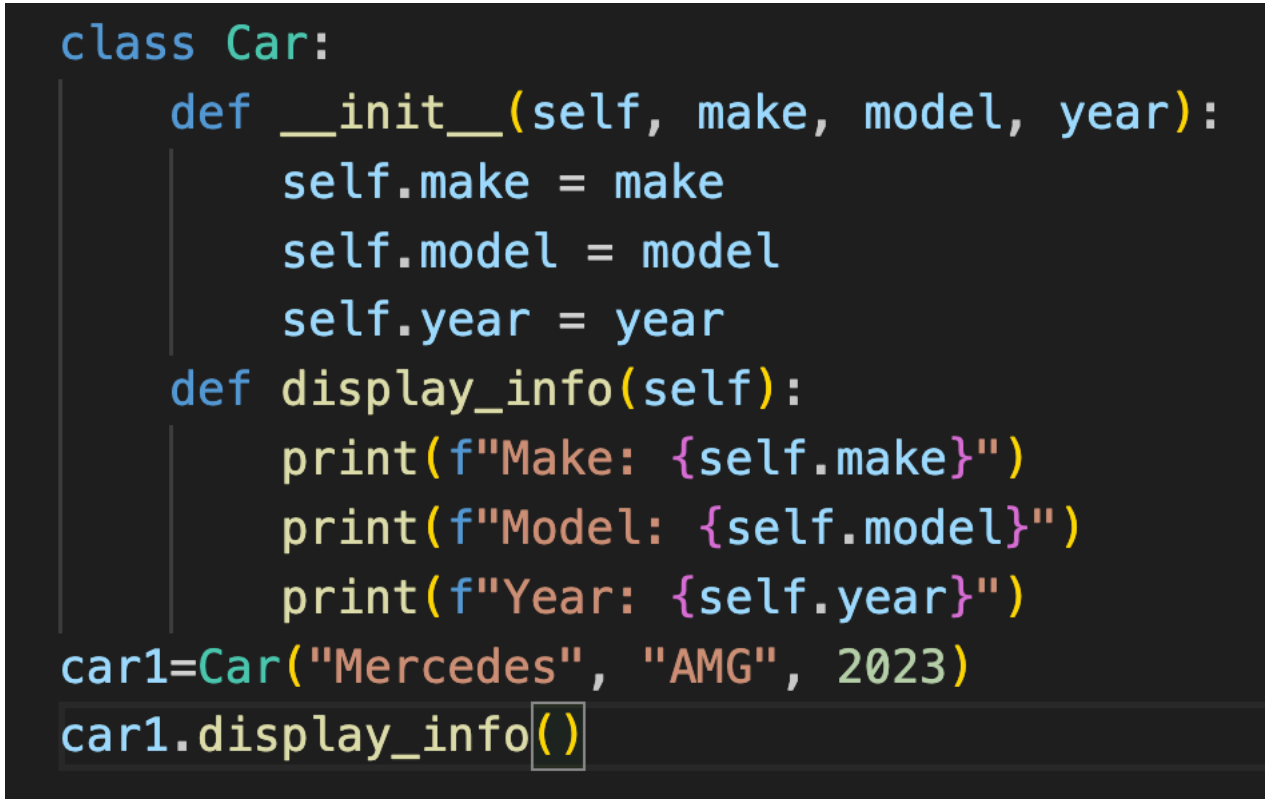



## 1. Basic Class & Object

**Task:**

Create a class `Car` with attributes `make`, `model`, and `year`. Write a method called `display_info` that prints all the car details.

# EXAMPLE 1 SOLUTION



# EXAMPLE 2

## 2. Encapsulation

**Task:**

Create a class `Student` with private attribute `__grade` (number).

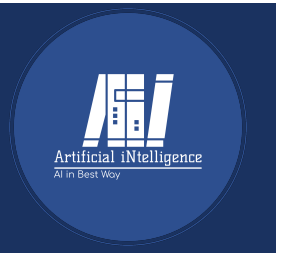
Write these methods:

- `set_grade(new_grade)` — sets the grade if between 0 and 100, else print "Invalid".
- `get_grade()` — returns the grade.
- **Try:** Create a student, set the grade, and print the grade.

# EXAMPLE 2 SOLUTION

```
class Student:
    def __init__(self, grade):
        self.__grade = grade
    def set_grade(self, newgrade):
        if 0 <=newgrade<= 100:
            self.__grade = newgrade
        else:
            print("Invalid grade")
    def return_grade(self):
        return self.__grade
student1= Student("a")
student1.set_grade(90)
print(student1.return_grade())
student2= Student("b")
student2.set_grade(120)
student2.return_grade()
```

# EXAMPLE 3




### 3. Inheritance

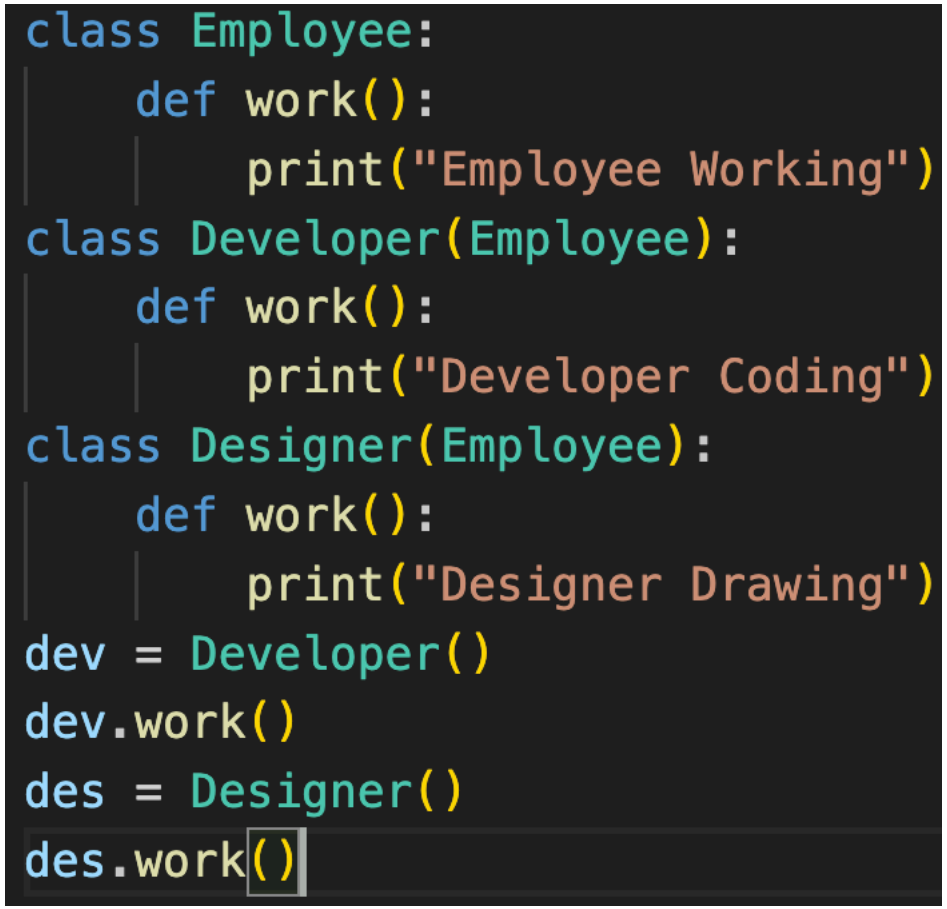
**Task:**

Make a base class `Employee` with a method `work()` that prints `"Employee working"`.

Make two child classes: `Developer` and `Designer`. Override `work()` in each to print `"Developer coding"` and `"Designer designing"` respectively.

- **Try:** Create one object of each child class and call their `work()` methods.

# EXAMPLE 3 SOLUTION



```
class Employee:
    def work():
        print("Employee Working")
class Developer(Employee):
    def work():
        print("Developer Coding")
class Designer(Employee):
    def work():
        print("Designer Drawing")
dev = Developer()
dev.work()
des = Designer()
des.work()
```

# EXAMPLE 4

## 8️⃣ Application: Library System

**Task:**

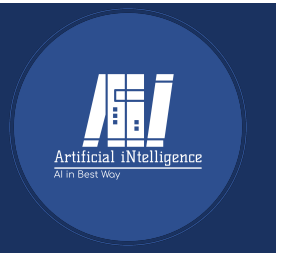
Create a `Book` class with:

- Attributes: `title`, `author`, `year`.
- Method: `display_info()` — shows all book details.

Then, make a `Library` class that:

- Holds a list of books.
- Methods:
  - `add_book(book)`
  - `remove_book(book_title)`
  - `display_all_books()`

# EXAMPLE 4 SOLUTION




```python
class Book:
    def __init__(self, title, author, year):
        self.title = title
        self.author = author
        self.year = year
    def display_info(self):
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")
        print(f"Year: {self.year}")
class Library(Book):
    def __init__(self, title, author, year):
        super().__init__(title, author, year)
        self.books = []
    def add_book(self, book):
        self.books.append(book)
    def remove_book(self, title):
        self.books.remove(title)
    def display_all_books(self):
        for book in self.books:
            book.display_info()
book1 = Book("The Great Gatsby", "F. Scott Fitzgerald", 1925)
book2 = Book("To Kill a Mockingbird", "Harper Lee", 1960)
library = Library("Classic Novels", "Various Authors", 2020)
library.add_book(book1)
library.add_book(book2)
library.display_all_books()
```

# EXAMPLE 5

Define a pure abstract base class called BasicShape . The BasicShape class should have the following members:
Private Member Variable: area, used to hold the shape's area.

Public Member Functions:
getArea. This function should return the value in the member variable area.
calcArea. This function should be a virtual function.

Next, define a class named Circle . It should be derived from the BasicShape class. It should have the following members:

Private Member Variables:
centerX, used to hold the x coordinate of the circle’s center.
centerY, used to hold the y coordinate of the circle’s center.
radius, used to hold the circle's radius.

Public Member Functions:
constructor—accepts values for centerX, centerY, and radius. Should call the overridden calcArea function.
calcArea—calculates the area of the circle (area = 3.14159 * radius * radius)
and stores the result in the inherited member area.
getCenterX—returns the value in centerX.
getCenterY—returns the value in centerY.

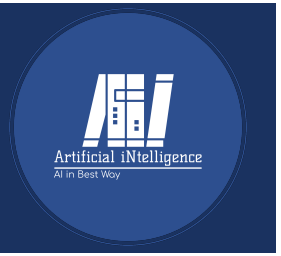
Next, define a class named Rectangle . It should be derived from the BasicShape class. It should have the following members:
Private Member Variables:
width, used to hold the width of the rectangle.
length, used to hold the length of the rectangle.

Public Member Functions:
constructor—accepts values for width and length. Should call the overridden calcArea function.
calcArea—calculates the area of the rectangle (area = length * width) and stores the result in the inherited member area.

getWidth—returns the value in width.
getLength—returns the value in length.

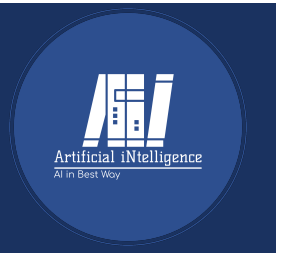
create a Circle object and a Rectangle object.
Demonstrate that each object properly calculates and reports its area.

# EXAMPLE 5 SOLUTION




```python
from abc import ABC , abstractmethod
class BasicShape(ABC):
    def __init__(self):
        self.__area = 0
    def getArea (self):
        return self.__area
    @abstractmethod
    def calcArea (self):
        pass
class Circle(BasicShape):
    def __init__(self,centerX,centerY,radius):
        self.__centerX = centerX
        self.__centerY = centerY
        self.__radius = radius
    def calcArea(self):
        self.__area = 3.14 * self.__radius * self.__radius
        return self.__area
    def getcenterX(self):
        return self.__centerX
    def getcenterY(self):
        return self.__centerY
class Rectangle (BasicShape):
    def __init__(self , length , width):
        self.__length = length
        self.__width = width
    def calcArea(self):
        self.__area = self.__length * self.__width
        return self.__area
    def getwidth(self):
        return self.__width
    def getlength(self):
        return self.__length
circle = Circle(centerX=5 , centerY=10 , radius= 5)
rectangle = Rectangle(width=5 , length= 10)
print(circle.calcArea())
```

# BREAK BETWEEN THE EXAMPLES




- **What is Object-Oriented Programming (OOP), and what are its main principles?**
- **Explain the difference Between a class and an object?**
- **What does __init__ do?**

# EXAMPLE 6

**1) What will happen if we run this code?**

```
class Rectangle:
    width = 2
    length = 3
    def getArea():
        Rectangle.width * Rectangle.length

result = Rectangle.getArea()
print(result)
```

# EXAMPLE 6 SOLUTION

**1) What will happen if we run this code?**

```
class Rectangle:
    width = 2
    length = 3
    def getArea():
        Rectangle.width * Rectangle.length

result = Rectangle.getArea()
print(result)
```

الاجابة None

ليه؟ عشان هنا مافيش return او print

# EXAMPLE 7

What will happen if we run this code?

```
class Rectangle:
    def __init__(self, width, length):
        self.width = width
        self.length = length

    def get_area():
        return self.width * self.length

rect = Rectangle(5, 10)
print("Area of rectangle:",
rect.get_area())
```

# EXAMPLE 7 SOLUTION

الاجابة Error

ليه؟ عشان هنا مافيش self فا مش هيتعامل مع ال objects

```
class Rectangle:
    def __init__(self, width, length):
        self.width = width
        self.length = length

    def get_area():
        return self.width * self.length

rect = Rectangle(5, 10)
print("Area of rectangle:",
rect.get_area())
```

# EXAMPLE 8

**1) What will happen if we run this code?**

```
class car:
    color = "Red"
car1 = car()
car2 = car()
car1.color = "Yellow"
car.color = "Blue"
print(car1.color)
print(car2.color)
```

# EXAMPLE 8 SOLUTION

**1) What will happen if we run this code?**

```
class car:
    color = "Red"
car1 = car()
car2 = car()
car1.color = "Yellow"
car.color = "Blue"
print(car1.color)
print(car2.color)
```

الاجابة هتبقى yellow و blue

ليه؟ عشان زي ما قولنا، ال class variable بت assume ان هي الصح لحد ما الاوبجيكت يثبت عكس كده.

طب ليه هما الاتنين مابيقاش Blue ؟

عشان car1 اثبتت ال class variable ان هي مش red و car.color دي بتعدل على الكلاس variable فا مش هتوصل ل car1

# EXAMPLE 9

What will happen if we run this code?

```
class Dog:
    def __init__(self, name):
        self.name = name
dog1 = Dog("Buddy")
dog2 = dog1
dog2.name = "Charlie"
print(dog1.name)
```

# EXAMPLE 9 SOLUTION

الاجابة Charlie

ليه؟ عشان احنا ساوينا dog2 ب

dog1 فا في الاخر هتبقى Charlie

```
class Dog:
    def __init__(self, name):
        self.name = name
dog1 = Dog("Buddy")
dog2 = dog1
dog2.name = "Charlie"
print(dog1.name)
```

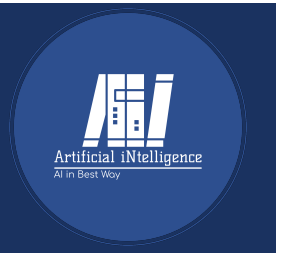
# EXAMPLE 10

What will happen if we run this code?

```
class Rectangle:
    def __init__(self):
        self.__width = 0
    def get_width(self):
        return self.__width

rect = Rectangle(20)
print("Width:", rect.__width)
```

# EXAMPLE 10 SOLUTION




الاجابة هتبقى Error
ليه؟ عشان احنا ماستعديناش ال
get_width فانكشن فا بالتالي انت
كده يعتبر بت access a private
variable natively او بمعنى اصح
من غير وسيط فا هتطلع Error

```
class Rectangle:
    def __init__(self):
        self.__width = 0
    def get_width(self):
        return self.__width

rect = Rectangle(20)
print("Width:", rect.__width)
```

# EXAMPLE 11

What will happen if we run this code?

```
class Rectangle:
    count = 0
    def __init__(self, width, height):
        self.width = width
        self.height = height
        Rectangle.count += 1
    @staticmethod
    def get_object_count():
        return Rectangle.count
rect1 = Rectangle(10, 20)
print(rect1.get_object_count())
rect2 = Rectangle(15, 25)
print(rect2.get_object_count())
```

a) 1 , 2

b) 2, 2

c) 1,1

d) Error

# EXAMPLE 11 SOLUTION

اللعبة كلها في دول

a) 1 , 2

الاجابة هتبقى

عشان عند rect1 كان العدد ب 1 و

بعدين عند rect2 العدد ب 2

```
class Rectangle:
    count = 0
    def __init__(self, width, height):
        self.width = width
        self.height = height
        Rectangle.count += 1
    @staticmethod
    def get_object_count():
        return Rectangle.count
rect1 = Rectangle(10, 20)
print(rect1.get_object_count())
rect2 = Rectangle(15, 25)
print(rect2.get_object_count())
```

# EXAMPLE 12

**1) A common use of …….. is to free memory that was dynamically allocated by the class object.**

a) Constructor

b) Destructor

c) Destroyer

d) Duplicator

# EXAMPLE 12 SOLUTION

**1) A common use of …….. is to free memory that was dynamically allocated by the class object.**

a) Constructor

b) Destructor

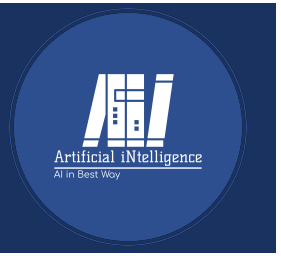
c) Destroyer

d) Duplicator

# EXAMPLE 13

```
class Animal:
    def __init__(self, name, speed):
        self.name = name
        self.speed = speed
    def move(self):
        return f"{self.name} is moving at {self.speed} m/s!"
class Turtle(Animal): ……..
class Rabbit(Animal):……..
turtle = Turtle("Turtle", 2)
rabbit = Rabbit("Rabbit", 10)
print(turtle.move())
print(rabbit.move())
```

# EXAMPLE 13 SOLUTION




```python
class Animal:
    def __init__(self, name, speed):
        self.name = name
        self.speed = speed
    def move(self):
        return f"{self.name} is moving at {self.speed} m/s!"
class Turtle(Animal):
    def __init__(self, name, speed):
        super().__init__(name, speed)
class Rabbit(Animal):
    def __init__(self, name, speed):
        super().__init__(name, speed)
turtle = Turtle("Turtle", 2)
rabbit = Rabbit("Rabbit", 10)
print(turtle.move())
print(rabbit.move())
```

# EXAMPLE 14

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

    def __eq__(self, other):
        return self.name == other.name and self.age == other.age

person1 = Person("Alice", 25)
person2 = Person("Alice", 25)
person3 = Person("Bob", 30)

print(person1 == person2)
print(person1 == person3)
```

# EXAMPLE 14 SOLUTION

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

    def __eq__(self, other):
        return self.name == other.name and self.age == other.age

person1 = Person("Alice", 25)
person2 = Person("Alice", 25)
person3 = Person("Bob", 30)

print(person1 == person2)
print(person1 == person3)
```

**Output:**

True

False

# EXAMPLE 15

```
class Demo:
    def __init__(self):
        self.__value = 10

    def show(self):
        print(self.__value)

def friend(obj):
    obj.__value = 50
    print(obj.__value)

d = Demo()
friend(d)
```

# EXAMPLE 15 SOLUTION

```
class Demo:
    def __init__(self):
        self.__value = 10

    def show(self):
        print(self.__value)

def friend(obj):
    obj.__value = 50
    print(obj.__value)

d = Demo()
friend(d)
```

# Thank you