
# OBJECT ORIENTED PROGRAMMING

LECTURES 5,6 & 7

BY: KEVIN HARVEY

# CLASS HIERARCHY

Same concept as inheritance but with a father for the father

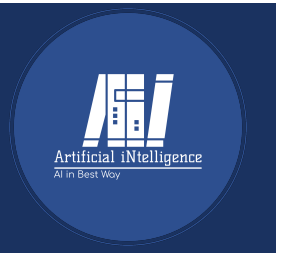
```python
import math
class Shape:
    def __init__(self,name):
        self.name = name
    def area(self):
        return "area formula not defined for this shape"
    def describe(self):
        return f"this is a {self.name}"
class Triangle(Shape):
    def __init__(self,base,height):
        super().__init__("triangle")
        self.base = base
        self.height = height
    def area(self):
        return (self.base*self.height)/2
class righttriangle(Triangle):
    def __init__(self,base,height):
        super().__init__(base,height)
        print(self.name)
        self.name = "right triangle"
        self.hypotenuse = math.sqrt(base**2+height**2)
right_triangle = righttriangle(3,4)
print(right_triangle.describe(), "Hypotenuse is", right_triangle.hypotenuse)
```

في الحالة دي:

- ال Shape اب لل Triangle class و جد ال right triangle class
- ال Triangle ابن ال Shape Class و اب ال right triangle class

ال right triangle class هو ابن ال Triangle class و حفيد ال Shape class

# MULTIPLE INHERITANCE




**Inherits A only and ignores B**

```
1    class A:
2        def greet(self):
3            print("hello")
4    class B:
5        def greet(self):
6            print("hi")
7    class C(A,B):
8        pass
9    D = C()
10   print(D.greet())
```

Output:
hello
None

**Here, it inherits from both classes**

```
1    class A:
2        def greet(self):
3            print("hello")
4    class B:
5        def greet(self):
6            print("hi")
7    class C(A,B):
8        def greet(self):
9            print(f"{A.greet(self)} {B.greet(self)}")
10   D = C()
11   print(D.greet())
```

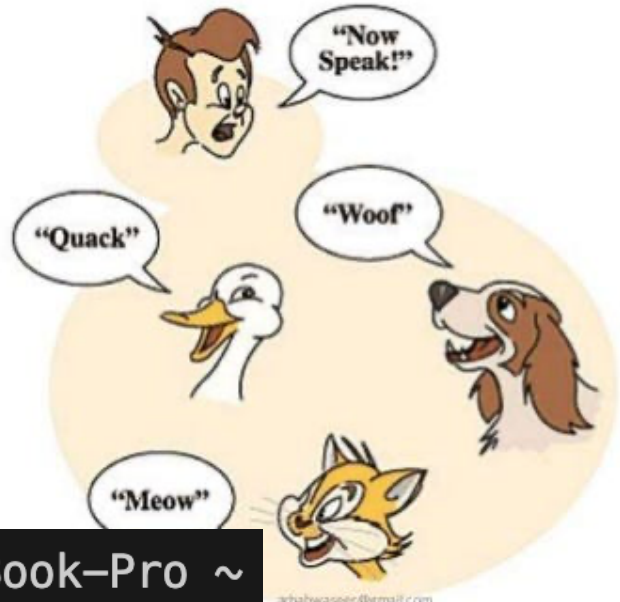
Output:
hello hi

# POLYMORPHISM

- It's basically a function across classes that has the same name but acts differently
- Polymorphism is a programming concept that allows different classes to be treated through a common interface

```
class Animal:
    def __init__(self, name):
        self.name = name
    def speak(self):
        pass
class Dog(Animal):
    def speak(self):
        print("woof-woof!")
class Cat(Animal):
    def speak(self):
        print("meow-meow!")
class duck(Animal):
    def speak(self):
        print("quack-quack!")
dog = Dog("Snoopy")
cat = Cat("Garfield")
Duck = duck("Donald Duck")
dog.speak()
cat.speak()
Duck.speak()
```

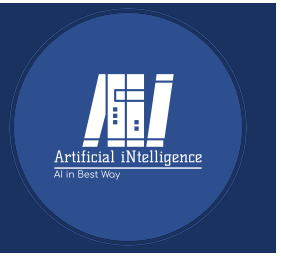
All of these classes have a function with the same name but performs differently




**Output:**

```
kevinharvey@Kevins-MacBook-Pro ~
woof-woof!
meow-meow!
quack-quack!
```

# POLYMORPHISM NOTES:




1. **Polymorphism cannot exist without functions**
2. **Polymorphism is about how methods are called on different classes**

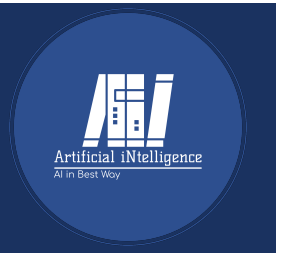
# VIRTUAL FUNCTIONS (THE DEFAULT)

- **A virtual function is a method that is made in the parent class and is meant to be overridden in child classes**
- **Virtual functions are key to polymorphism**
- **All methods are virtual by default**

```
1   class Animal:
2       def make_sound(self):
3           print("Some generic animal sound")
4
5   class Dog(Animal):
6       def make_sound(self):
7           print("Woof!")
8       def animal_sound(Animal):
9           Animal.make_sound()
10  doggo = Dog()
11  doggo.animal_sound()
```

Output: Woof!

# ENCAPSULATION




- It's basically referring to hiding internal details and exposing only what's necessary using public methods
- It refers to the limiting of direct access to SOME of the data

```python
class bankaccount:
    def __init__(self, balance):
        self.__balance = balance
    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount
    def get_balance(self):
        return self.__balance
```

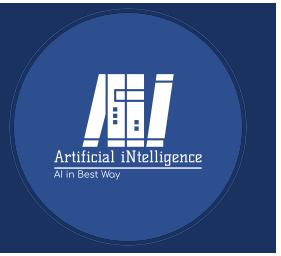
# ABSTRACT CLASSES

- **Abstract classes are only meant for inheritance and making sure it's methods are implemented in other classes.**
- **هي بس بتاكد ان شرط Polymorphism اتحقق و ان كل الفانكشنز اتطبقت في الكود**

في حالتي انا، انا عملت فانكشن واحدة بس abstract بس لارم تتطبق لكل ال classes اللى هتورث من ال class ده

```python
from abc import ABC , abstractmethod
class Shape(ABC):
    @abstractmethod
    def area(self):
        pass
class Circle(Shape):
    def __init__(self,radius):
        self.radius=radius
    def area(self):
        return 3.14*self.radius*self.radius
class Rectangle(Shape):
    def __init__(self,length,width):
        self.length=length
        self.width = width
    def area(self):
        return self.length*self.width
```

# MAGIC METHODS




- بكل بساطة الميثودز دي بتساعدنا نعمل مقارنات او عمليات حسابية على الاوبجيكت بتاعك او ما بين اتنين او اكتر من الاوبجيكتس

```
# for subtraction
def __sub__(self, other):
    return self - other
```

```
# for multiplication
def __mul__(self, other):
    return self * other
```

```
# for addition
def __add__(self, other):
    return self + other
```

```
# for division
def __truediv__(self, other):
    return self / other
```

```
# for equality check (boolean)
def __eq__(self, other):
    return self == other
```

```
# for checking if one number is less than the other (boolean)
def __lt__(self, other):
    return self < other
```

```
# for checking if one number is greater than the other (boolean)
def __gt__(self, other):
    return self > other
```

# TABLE

| | | |
|---|---|---|
| | | |
| | | |
| | | |

# Thank you