



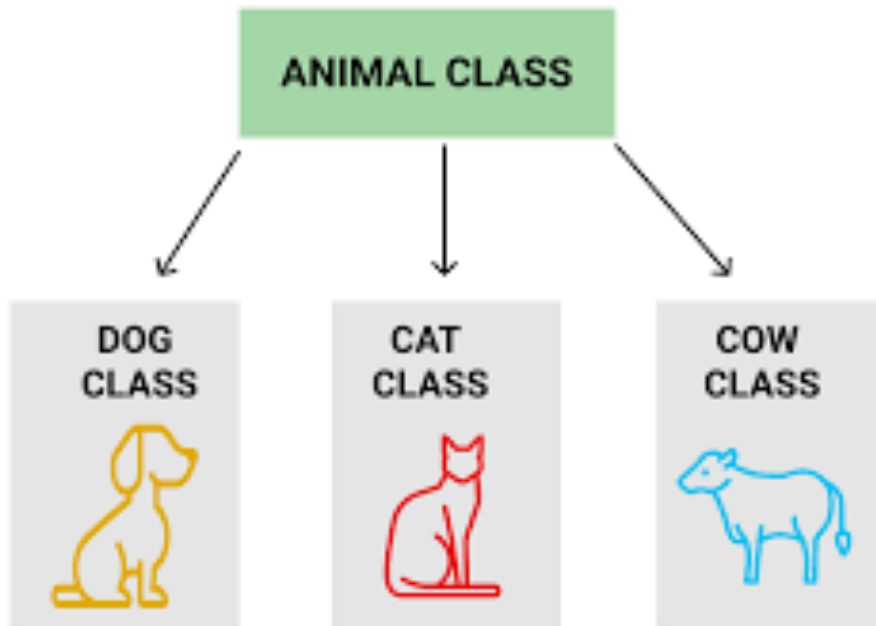
Inheritance

Week 2



Trinity Walton Club

What is Inheritance?



Inheritance is a hierarchy of classes within OOP that allows us to derive properties from one class to another

This represents real-world relationships well

Promotes the reusability of code

```

1  # A Python program to demonstrate inheritance
2  class Person(object):
3
4      # Constructor
5      def __init__(self, name, id):
6          self.name = name
7          self.id = id
8
9      # To check if this person is an employee
10     def Display(self):
11         print(self.name, self.id)
12
13
14     # Driver code
15     person_instance = Person("Emma", 102) # An Object of Person class
16     person_instance.Display()
17
18     class Student(Person):
19         def Print(self):
20             print("I am a student")
21
22     Student_details = Student("Emma", 103)
23
24     Student_details.Display()
25     Student_details.Print()
26

```

Example 1

```

Class BaseClass:
    {Body}
Class DerivedClass(BaseClass):
    {Body}

```

Notice how the the properties of the `Person` class are inherited in the `Student` class

Example 2

We can also create new methods within the child class, that is only accessible to the child class

```
1 class Person(object):
2
3     # Constructor
4     def __init__(self, name):
5         self.name = name
6
7     # To get name
8     def getName(self):
9         return self.name
10
11     # To check if this person is an employee
12     def isEmployee(self):
13         return False
14
15
16 # Inherited or Subclass (Note Person in bracket)
17 class Employee(Person):
18
19     # Here we return true
20     def isEmployee(self):
21         return True
22
23
24 # Driver code
25 emp = Person("Bob") # An Object of Person
26 print(emp.getName(), emp.isEmployee()) # returns False
27
28 emp = Employee("Stephen") # An Object of Employee
29 print(emp.getName(), emp.isEmployee()) # returns True
30
```

```
1 # parent class
2 class Person():
3     def __init__(self, name, age):
4         self.name = name
5         self.age = age
6
7     def display(self):
8         print(self.name, self.age)
9
10 # child class
11 class Student(Person):
12     def __init__(self, name, age):
13         self.Name = name
14         self.Age = age
15         # inheriting the properties of parent class
16         super().__init__("Cara", age)
17
18     def displayInfo(self):
19         print(self.Name, self.sAge)
20
21 obj = Student("Niamh", 23)
22 obj.display()
23 obj.displayInfo()
24
```

Example 3

We can inherit the `__init__` from the parent class

Example 4

We can inherit from multiple classes at the same time

```
1  class Vehicle:
2      |      def start_engine(self):
3      |          print("Engine started")
4
5  class Radio:
6      |      def play_music(self):
7      |          print("Music playing")
8
9  class Car(Vehicle, Radio):
10     |      def honk_horn(self):
11     |          print("Horn honked")
12
13     # Creating an instance of the Car class
14     car_instance = Car()
15
16     # Accessing methods from both base classes
17     car_instance.start_engine()
18     car_instance.play_music()
19     car_instance.honk_horn()
20
```

```

1 class ChessPiece:
2     def __init__(self, colour, xpos, ypos):
3         self.xpos = xpos
4         self.ypos = ypos
5         self.colour = colour
6
7 class Pawn(ChessPiece):
8     def __init__(self, colour, xpos, ypos):
9         super().__init__(colour, xpos, ypos)
10        self.firstMove = True
11
12    def legalMove(self, movingxpos, movingypos):
13        if self.colour == "black":
14            if self.firstMove and movingypos - self.ypos == 2 and movingxpos == self.xpos:
15                self.firstMove = False
16                return True
17            elif movingypos - self.ypos == 1 and movingxpos == self.xpos:
18                return True
19
20            elif self.colour == "white":
21                if self.firstMove and self.ypos - movingypos == 2 and movingxpos == self.xpos:
22                    self.firstMove = False
23                    return True
24                elif self.ypos - movingypos == 1 and movingxpos == self.xpos:
25                    return True
26        return False

```

```

27
28 pawn_instance = Pawn("black", 2, 4)
29 print(pawn_instance.legalMove(2, 6)) # True
30 print(pawn_instance.legalMove(3, 6)) # False
31

```

```

class ChessPiece:
    def __init__(self, colour, xpos, ypos):
        self.xpos = xpos
        self.ypos = ypos
        self.colour = colour

class Pawn(ChessPiece):
    def __init__(self, colour, xpos, ypos):
        super().__init__(colour, xpos, ypos)
        self.firstMove = True

    def legalMove(self, movingxpos, movingypos):
        if self.colour == "black":
            if self.firstMove and movingypos - self.ypos == 2 and movingxpos == self.xpos:
                self.firstMove = False
                return True
            elif movingypos - self.ypos == 1 and movingxpos == self.xpos:
                return True

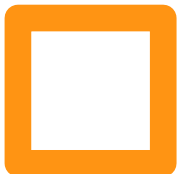
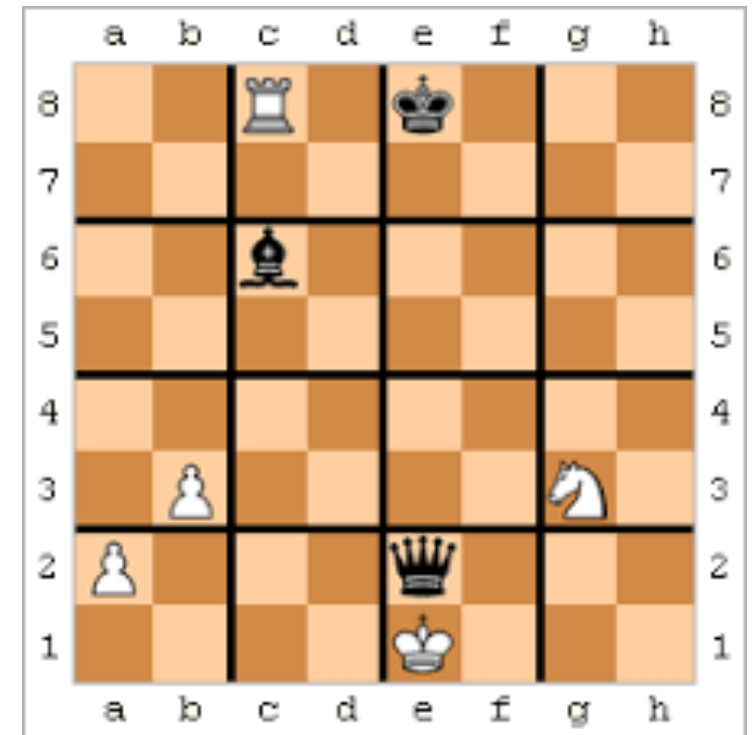
            elif self.colour == "white":
                if self.firstMove and self.ypos - movingypos == 2 and movingxpos == self.xpos:
                    self.firstMove = False
                    return True
                elif self.ypos - movingypos == 1 and movingxpos == self.xpos:
                    return True
        return False

pawn_instance = Pawn("black", 2, 4)
print(pawn_instance.legalMove(2, 6)) # True
print(pawn_instance.legalMove(3, 6)) # False

```

Exercise

1. Get into groups of 4-5 people; this will be the team for the group project in this semester
2. Create a `ChessPiece` parent class with the variables initialized of xpos, ypos and colour (with the xpos and ypos being the position of the piece in a 2D array of the chess board)
3. Create a chess piece child class that inherits the variables from the chess piece parent class. Create a function called 'IsLegalMove' that determines whether an inputted x and y variable is legal for the chess piece to move to, based on its current x and y position (self.xpos, self.ypos) Note: each member of the team should work on one chess piece.
4. Create two instances of the chess piece, and test whether the 'IsLegalMove' function works:
 - a. black_piece, with colour black, xpos 3 and ypos 2
 - b. white_piece, with colour white, xpos 5 and ypos 7



References

- <https://akshayraut.medium.com/inheritance-in-object-oriented-programming-8c61b93ca5a8>
- <https://www.geeksforgeeks.org/inheritance-in-python/>
- https://en.wikipedia.org/wiki/Grid_chess