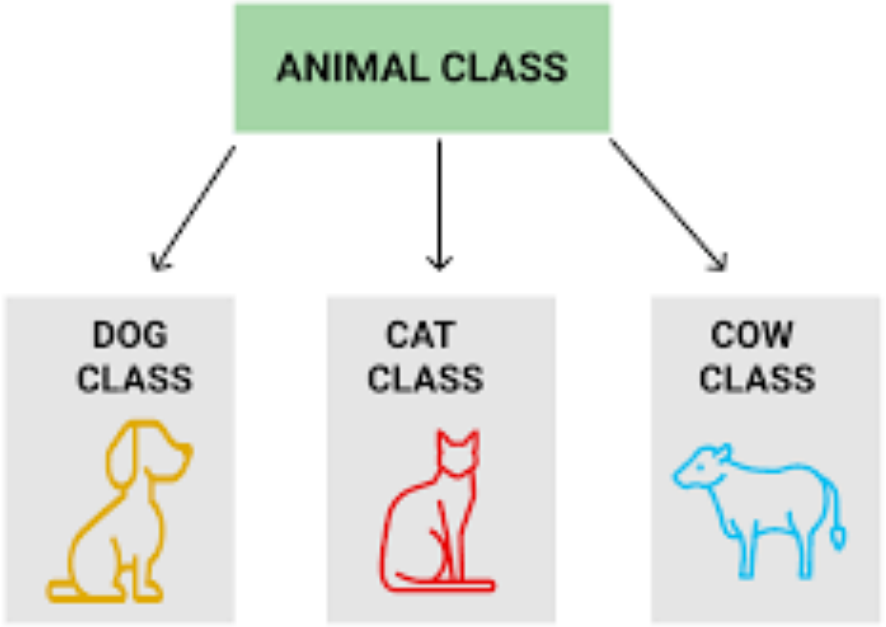
Inheritance

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

```python
# A Python program to demonstrate inheritance
class Person(object):

    # Constructor
    def __init__(self, name, id):
        self.name = name
        self.id = id

    # To check if this person is an employee
    def Display(self):
        print(self.name, self.id)


# Driver code
person_instance = Person("Emma", 102) # An Object c
person_instance.Display()

class Student(Person):
    def Print(self):
        print("I am a student")

Student_details = Student("Emma", 103)

Student_details.Display()
Student_details.Print()

```

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

```
class Person(object):

    # Constructor
    def __init__(self, name):
        self.name = name

    # To get name
    def getName(self):
        return self.name

    # To check if this person is an employee
    def isEmployee(self):
        return False


# Inherited or Subclass (Note Person in bracket)
class Employee(Person):

    # Here we return true
    def isEmployee(self):
        return True


# Driver code
emp = Person("Bob")  # An Object of Person
print(emp.getName(), emp.isEmployee()) # returns False

emp = Employee("Stephen")  # An Object of Employee
print(emp.getName(), emp.isEmployee()) # returns True
```

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

    def display(self):
        print(self.name, self.age)

# child class
class Student(Person):
    def __init__(self, name, age):
        self.Name = name
        self.Age = age
        # inheriting the properties of parent class
        super().__init__("Cara", age)

    def displayInfo(self):
        print(self.Name, self.sAge)

obj = Student("Niamh", 23)
obj.display()
obj.displayInfo()
```

Example 3

We can inherit the `__init__` from the parent class

Example 4

We can inherit from multiple classes at the same time

```python
class Vehicle:
    def start_engine(self):
        print("Engine started")

class Radio:
    def play_music(self):
        print("Music playing")

class Car(Vehicle, Radio):
    def honk_horn(self):
        print("Horn honked")

# Creating an instance of the Car class
car_instance = Car()

# Accessing methods from both base classes
car_instance.start_engine()
car_instance.play_music()
car_instance.honk_horn()
```

https://github.com/b00rg/WaltonProgramming

Exercise

1. 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)

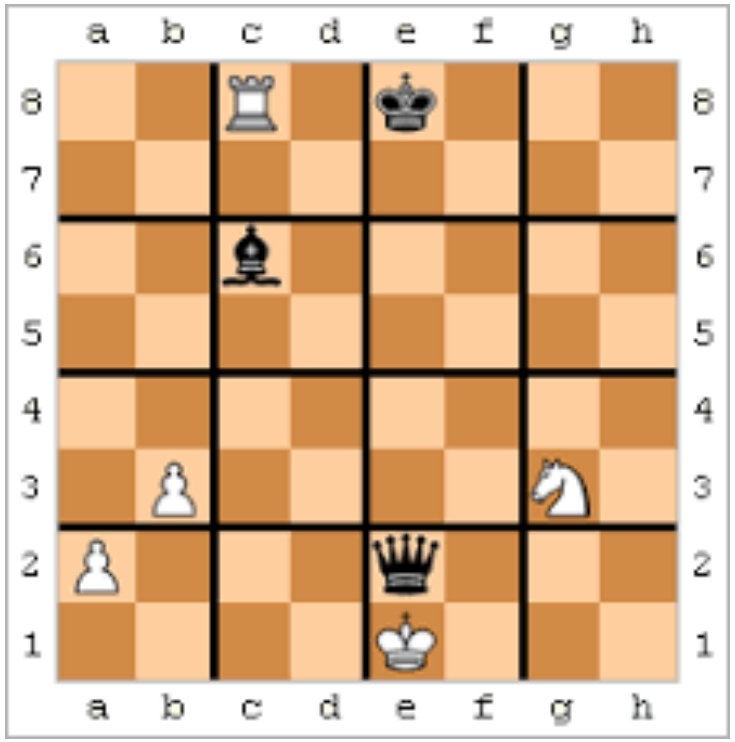
2. Create a `Pawn` child class that inherits the variables from the chess piece parent class. Create a function called 'IsLegalMove' that determines whether an inputted x and y variable is legal for the pawn to move to, based on its current x and y position (self.xpos, self.ypos)

    Note: this will have to be different depending on the colour of the chess piece

3. Create two instances of a pawn:

    a. black_pawn, with colour white, xpos 3 and ypos 2

    b. white_pawn, 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