```python
#Inheritance, Polymorphism and Magic Methods
class Animal:
    #superclass
    def __init__(self,birthType="UNK",
                 appearance="UNK", blooded="UNK"):
                 self.birthType = birthType
                 self.appearance = appearance
                 self.blooded = blooded

    #getters == @property
    @property
    def birtType(self):
        return self.__birthType

    @property
    def appearance(self):
        return self.__appearance

    @property
    def blooded(self):
        return self.__blooded

    #setters @prop name
    @birthType.setter
    def birthType(self,birthType):
        self.__birthType = birthType

    @appearance.setter
    def appearance(self,appearance):
        self.__appearance = appearance
```

```
    @blooded.setter
    def blooded(self,blooded):
        self.__blooded = blooded

    #Magic Method
    def __str__(self):
        return "A {} is {} it is {} it is {}".format(type(__name__),self.birthType, self.appearan
ce, self.blooded)


################################################
class Mammal(Animal):
    def __init__(self,birthType="born alive", appearance="hair or fur", blooded="warm blooded", n
urseYoung=True):
        Animal.__init__(self,birthType,appearance,blooded)
        self.__nurseYoung = nurseYoung

     #getters == @property
    @property
    def birtType(self):
        return self.__birthType

    @property
    def appearance(self):
        return self.__appearance

    @property
    def blooded(self):
        return self.__blooded
```

```
    #setters @prop name
    @birthType.setter
    def birthType(self,birthType):
        self.__birthType = birthType

    @appearance.setter
    def appearance(self,appearance):
        self.__appearance = appearance

    @blooded.setter
    def blooded(self,blooded):
        self.__blooded = blooded

    #add the mammal properties
    @nurseYoung.setter
    def nurseYoung(self,nurseYoung):
        self.__nurseYoung = nurseYoung

    @property
    def nurseYoung(self):
        return self.__nurseYoung

## methods can be overwritten
    def __str__(self):
       return super().__str__ + " and it is {} they nurse"\
           " their young".format(self.nurseYoung)


class Reptile(Animal):
```

```
    def __init__(self, birthType="hatched", appearance = "scales", blooded = "cold blooded"):
        Animal.__init__(self,birthType,appearance,blooded)

    def sumAll(self, *args):  ##the * is called a splat no predefined args in a function. no over
loading of functions needed
        sum = 0
        for i in args:
            sum += i

        return sum


def main():
    animal1 = Animal("BornAlive")

main()
```

GAME PLAYED ################################3

bonnie@LAPTOP-##### #### ~/Downloads/Games

Maximus attacks Galaxon and deals 13 damage

Galaxon is down to 37 health

Galaxon attacks Maximus and deals 7 damage

Maximus is down to 43 health

Maximus attacks Galaxon and deals 19 damage

Galaxon is down to 18 health

Galaxon attacks Maximus and deals 12 damage

Maximus is down to 31 health

Maximus attacks Galaxon and deals -2 damage

Galaxon is down to 20 health

Galaxon attacks Maximus and deals -1 damage

Maximus is down to 32 health

Maximus attacks Galaxon and deals 22 damage

Galaxon is down to -2 health

Galaxon has died and Maximus is the winner

Game Over!