Python Inheritance https://www.w3schools.com/python/python_inheritance.asp

a class can inherit methods from another class

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def printname(self):
        print(self.firstname, self.lastname)

#Use the Person class to create an object, and then execute the
printname method:

x = Person("John", "Doe")
x.printname()
```

```
John Doe
```

'Person' acts as the parent class and 'Student' is the child class.

Student(Person) is telling the Student class to inherit all the methods and properties of the Person class

```
class Student(Person):
      pass

x = Student("Mike", "Olsen")
x.printname()
```

```
Mike Olsen
```

```
class Student(Person):
    def __init__(self, fname, lname):
    #add properties etc.
```

```
  Cell In[14], line 3
    #add properties etc.
                        ^
SyntaxError: unexpected EOF while parsing
```

Running the above ends in an error because it is expecting items to be there.

Since it is no longer being passed, it expects more contents

Because **init** is being defined in the Student/Child class, it overrides the inherited **init** function properties from the Person/Parent class.

```
class Student(Person):
    def __init__(self, fname, lname):
        # using the super() function lets the class also inherit all
methods and properties from the parent
        super().__init__(fname, lname)
        # properties can be still be inherited from the parent while
adding new properties. example:
        self.graduationyear = 2019
        # but this property is hardcoding 2019 into the class, which
is not useful for the class purpose

class Student(Person):
    # to add the property the proper way, first 'year' attribute has
to be added within __init__
    def __init__(self, fname, lname, year):
        # we still want to inherit the other properties from the
parent class
        super().__init__(fname, lname)
        # instead of hardcode 2019, pass in a variable
        self.graduationyear = year

    # add a method to display the new property
    def printname(self):
        print(self.firstname, self.lastname, self.graduationyear)

# now when the class gets called, adding in a year will display that
in the print output
x = Student("Mike", "Olsen", 2019)

x.printname()

Mike Olsen 2019

class Student(Person):
    def __init__(self, fname, lname, year):
        super().__init__(fname, lname)
        self.graduationyear = year

    def welcome(self):
        print("Welcome", self.firstname, self.lastname, "to the class
of", self.graduationyear)

x = Student("Mike", "Olsen", 2019)
x.welcome()

Welcome Mike Olsen to the class of 2019
```