Introduction to Computer Programming Lecture 5.2:

Classes in Python: Inheritance

Hemma Philamore

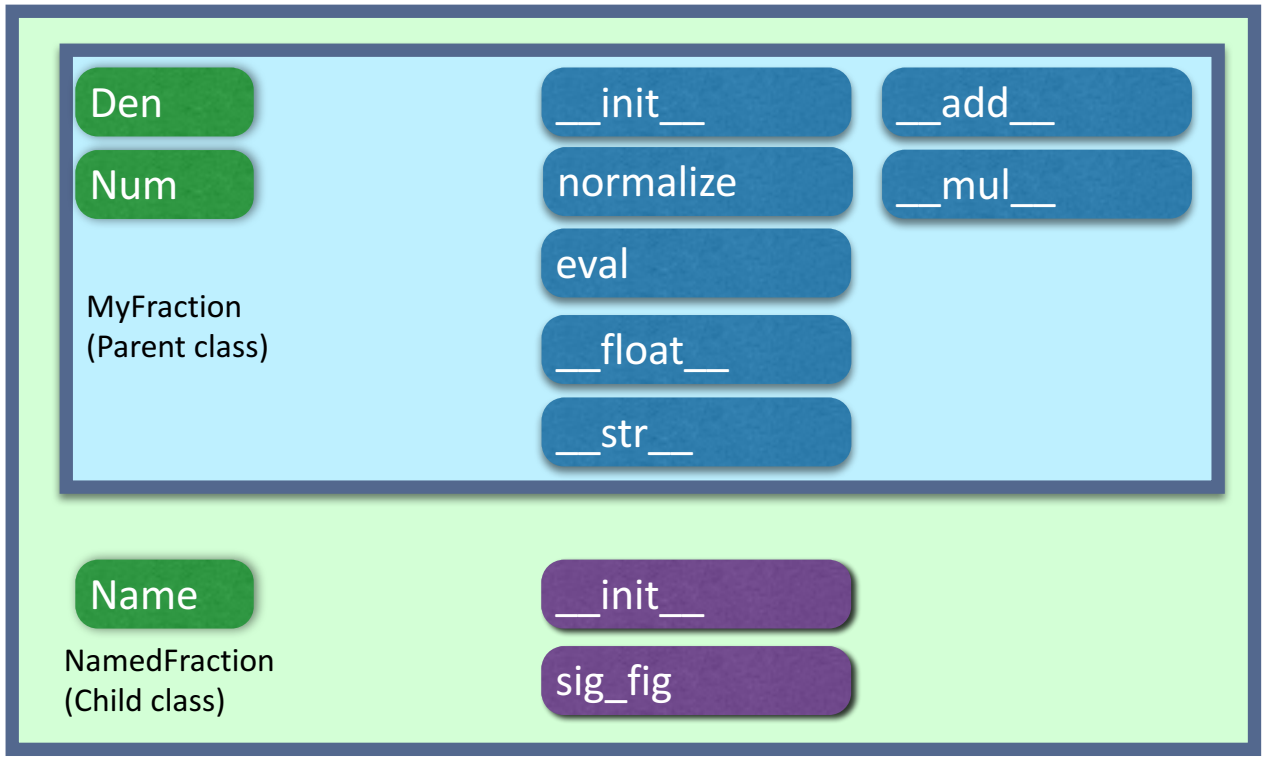
Department of Engineering Mathematics

Inheritance

A child class **inherits** all attributes and methods of the parent class.

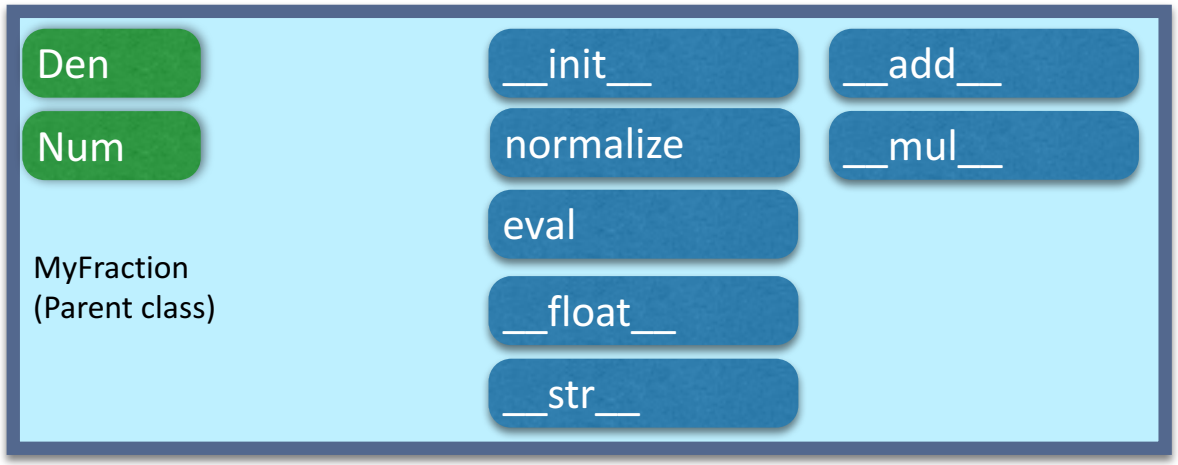
We can add/remove attributes and methods that are specific to the child class.

If the parent class is changed, the child class will automatically inherit the changes.



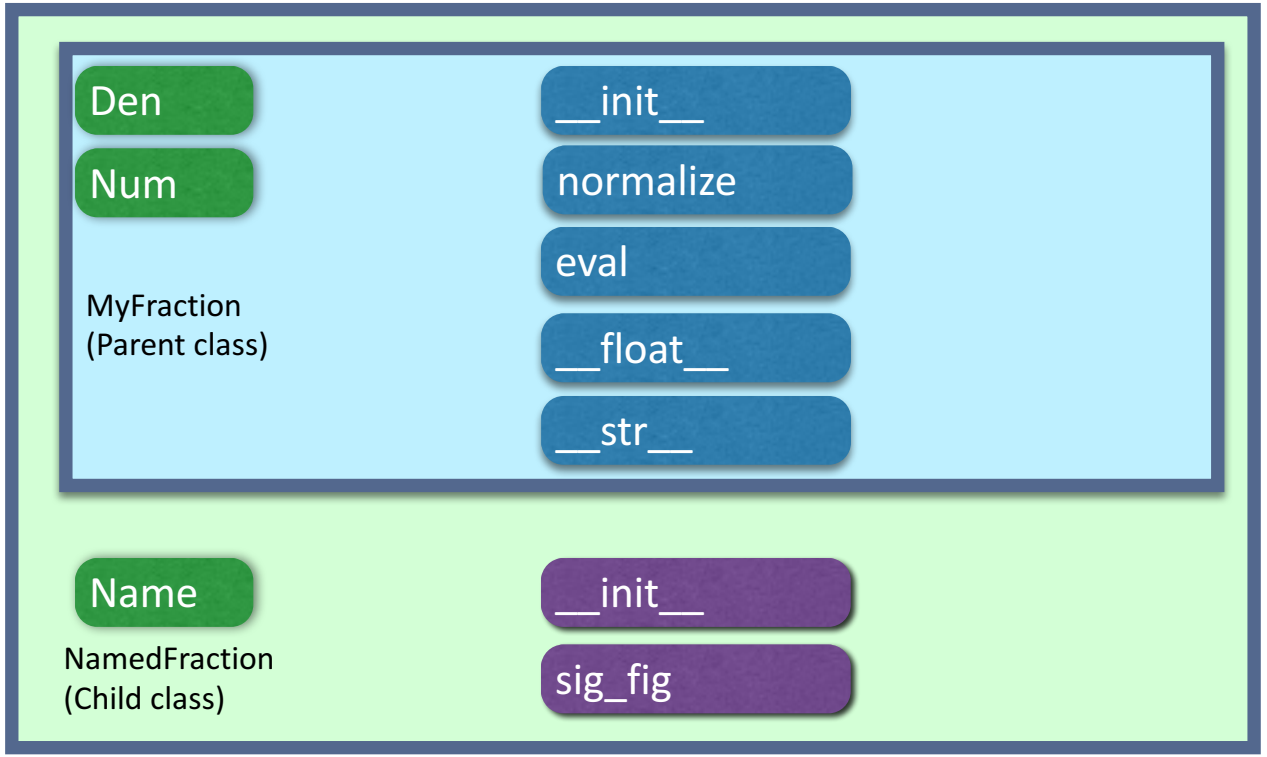
Parent class

Remember the **MyFraction** class

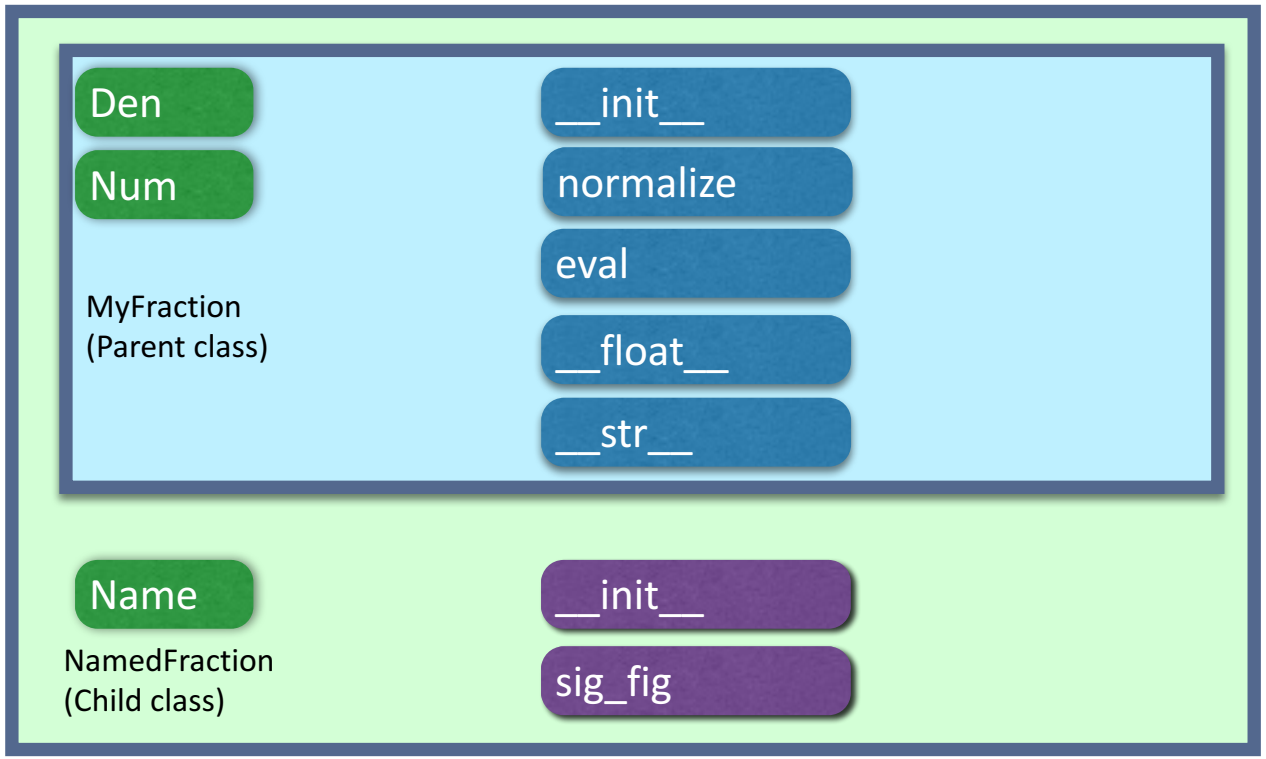



```python
import math

class MyFraction():
    """ Constructor """
    def __init__(self, num, den):
        """ Class attributes. """
        self.num = num
        self.den = den
        self.normalize()

    def normalize(self):
        gcd = math.gcd(self.num, self.den)
        self.num = int(self.num/gcd)
        self.den = int(self.den/gcd)

    def eval(self):
        return(self.num / self.den)

    def __float__(self):
        return(self.num / self.den)

    def __str__(self):
        return (" " + str(self.num) + "\n---\n" + " " + str(self.den) + "\n")

    def __add__(self, other):
        CommonDen = self.den * other.den
        CommonNum = self.num*other.den + other.num*self.den
        return MyFraction(CommonNum, CommonDen)

    def __mul__(self, other):
        return MyFraction(self.num*other.num , self.den*other.den)

```

Derive a class




```python
import math

class MyFraction():
    """ Constructor """
    def __init__(self, num, den):
        """ Class attributes. """
        self.num = num
        self.den = den
        self.normalize()

    def normalize(self):
        gcd = math.gcd(self.num, self.den)
        self.num = int(self.num/gcd)
        self.den = int(self.den/gcd)

    def eval(self):
        return(self.num / self.den)

    def __float__(self):
        return(self.num / self.den)

    def __str__(self):
        return (" " + str(self.num) + "\n---\n" + " " + str(self.den) + "\n")

```



Derive from class "MyFraction"

```
1  class NamedFraction(MyFraction):
2
3      def sig_fig(self, n):
4          return round(self.eval(), n)
5
6  n_fraction = NamedFraction(5,6)
7
8  print(n_fraction.eval())
9
10 print(n_fraction.sig_fig(3))
```

```
0.8333333333333334
0.833
```

Same functionality as "**MyFraction**", **eval** from **MyFraction** class

New method

Super

Using super() followed by a dot . and then a method name allows you to call the parent's version of the method and add to it.

```
1  class NamedFraction(MyFraction):
2
3      def __init__(self, num, den, name):
4          super().__init__(num, den)
5          self.name = name
6
7      def sig_fig(self, n):
8          return round(float(self.num / self.den), n)
9
10
11 n_fraction = NamedFraction(1, 4, "quarter")
12
13 print(n_fraction.name)
```

Initialise **super** class (i.e. the one that you have derived your new class from)

```
quarter
```

Override

Methods can be overridden by a child class to achieve different functionality.

```
1  class NamedFraction(MyFraction):
2
3      def __init__(self, num, den, name):
4          super().__init__(num, den)
5          self.name = name
6
7      def sig_fig(self, n):
8          return round(float(self.num / self.den), n)
9
10     def __str__(self):
11         return "This is: " + self.name + "\n" + \
12                "{:^5}".format(self.num) + "\n" + \
13                "-----" + "\n" + \
14                "{:^5}".format(self.den)
15
16
17 fraction = MyFraction(1, 4)
18 n_fraction = NamedFraction(1, 4, "quarter")
19
20
21 print(fraction)
22 print(n_fraction)
```

You can override functions for the new class

Uses the ***__str__*** from **"MyFraction"**

Uses the ***__str__*** from **"NamedFraction"** instead of **"MyFraction"**

```
 1
---
 4

This is: quarter
  1
-----
  4
```

NamedFraction is derived from MyFraction