Introduction to Computer Programming
Lecture 11.5:

Review :
Object Oriented Programming (OOP)

Hemma Philamore

Department of Engineering Mathematics

Why use classes?

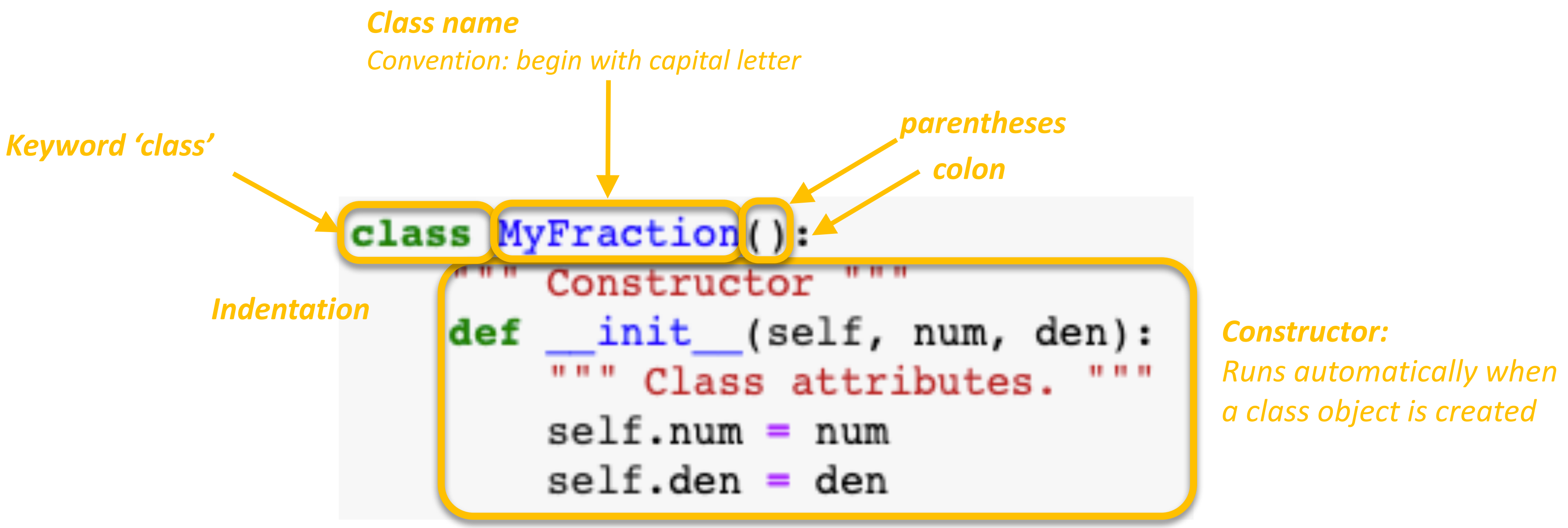
Modularity
Allows you to break down your code into smaller sections.

- Identify and fix problems
- Reuse in multiple programs

Reusability
More flexible reuse than standalone functions, modification via inheritance.

Polymorphism
Methods/attributes with the same name can have different functionality for different classes
e.g car.drive(), bike.drive()

Anatomy of a class

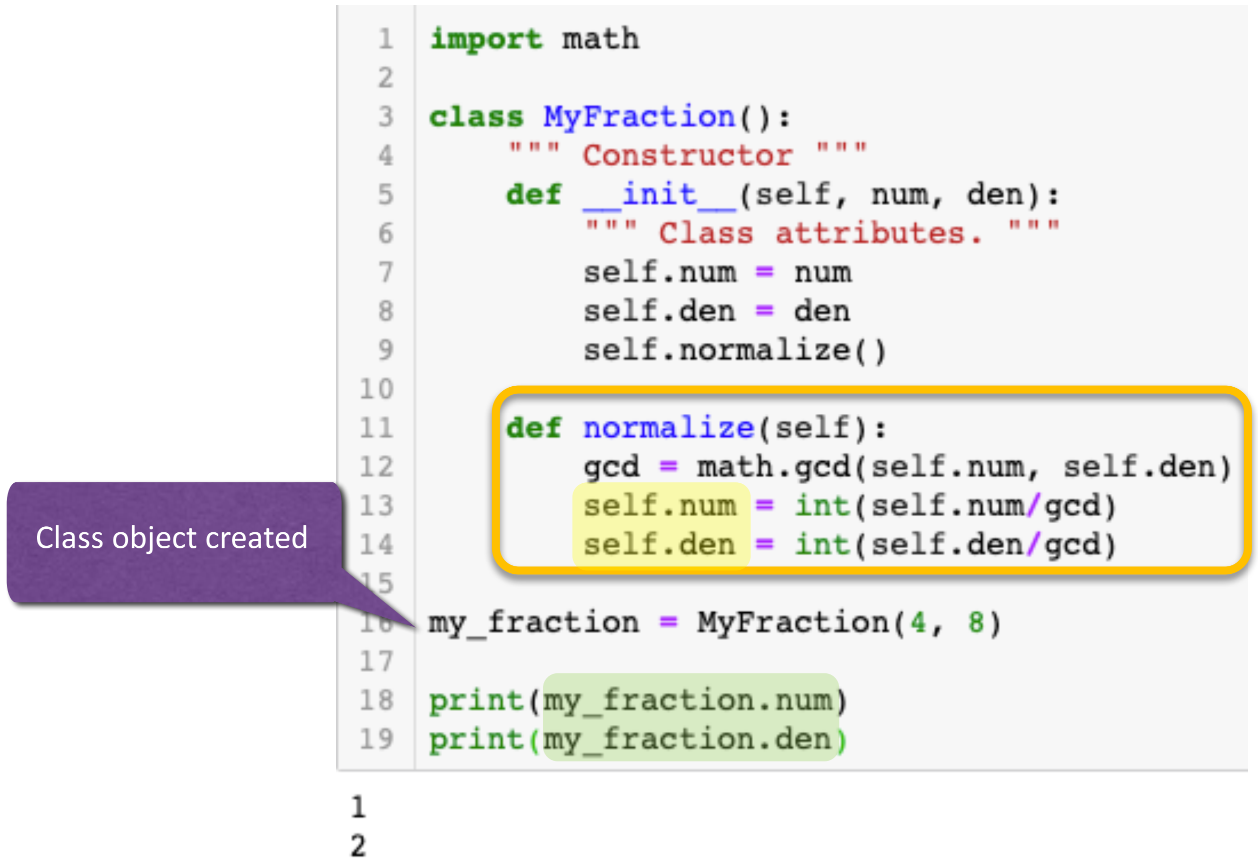

Class name
Convention: begin with capital letter
Keyword 'class'
parentheses
colon
Indentation
Constructor:
Runs automatically when a class object is created

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

self. behaves like the pronoun **'my'**

Inside the class, we are talking about **my** num , **my** den → **self.**num**, ,** **self.**den

Outside of the class → **my_fraction.num**
('my', means someone totally different when said by someone else.)



```
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)

my_fraction = MyFraction(4, 8)

print(my_fraction.num)
print(my_fraction.den)
```

```
1
2
```

Q.11.5.A
Write a Python class which has two methods:

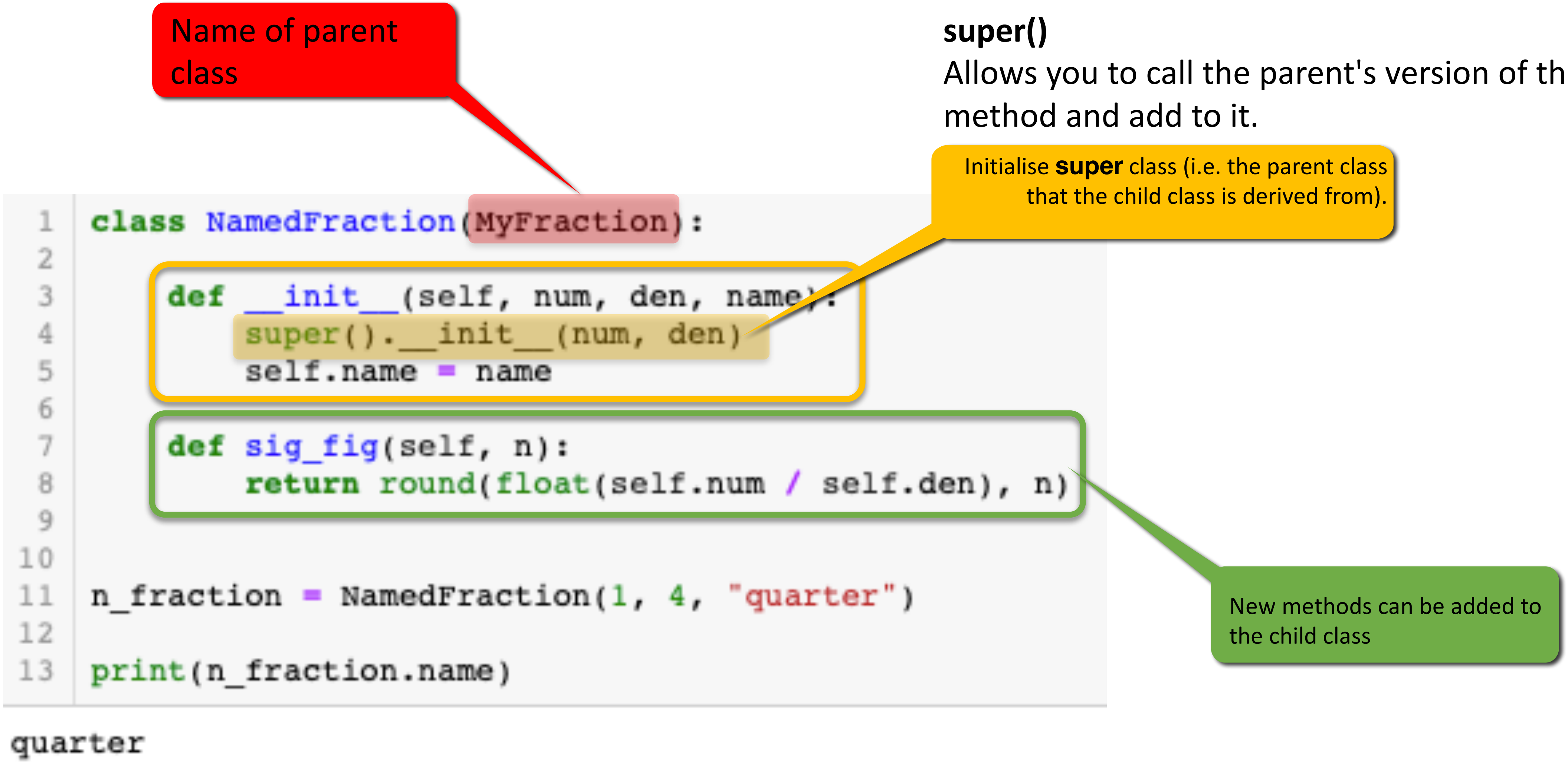
- **get_String** : request string from user and assign value to class attribute
- **print_String** : print the string in upper case

Q.11.5.B
Write a Python class, **Square_analyser** which:

- is constructed using a single input argument, **h** (length of one side).
- has two methods:
    - **area** : prints the area of the square
    - **perimeter** : print the perimeter of the square

Inheritance

super()
Allows you to call the parent's version of the method and add to it.


Name of parent class
Initialise **super** class (i.e. the parent class that the child class is derived from).
New methods can be added to the child class

```python
class NamedFraction(MyFraction):

    def __init__(self, num, den, name):
        super().__init__(num, den)
        self.name = name

    def sig_fig(self, n):
        return round(float(self.num / self.den), n)


n_fraction = NamedFraction(1, 4, "quarter")

print(n_fraction.name)
```

```
quarter
```

Q.11.5.C

Write a child class, **Rectangle_analyser**, inherited from **Square_analyser** (Q.11.5.B), that:

- is constructed using two input variables:
    - **h** - height
    - **w** - width
- has two methods:
    - **area** : prints the area of the rectangle
    - **perimeter** : prints the perimeter of the rectangle