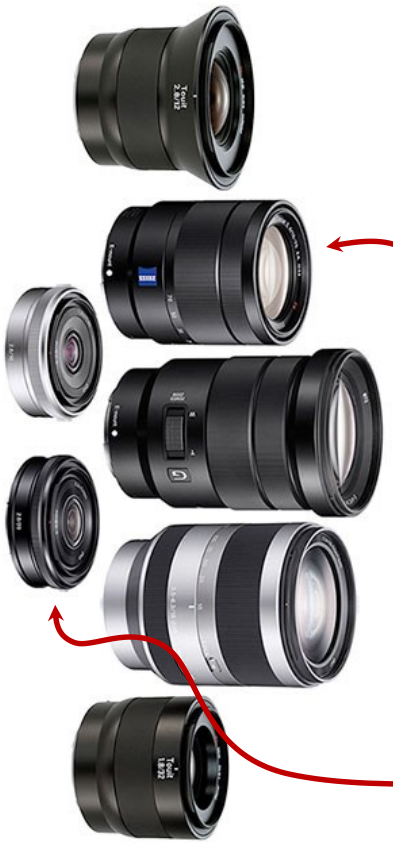
Thinking in Object-Orientation

- Known functional coding, now create it in a class and its class members (of Object-oriented programming)
  - class
  - objects
- Why OOP?



Powerful representation

Instead of repeating the same definitions, each for each lens

```
class Lens {
FocusLength
Aperture }
```

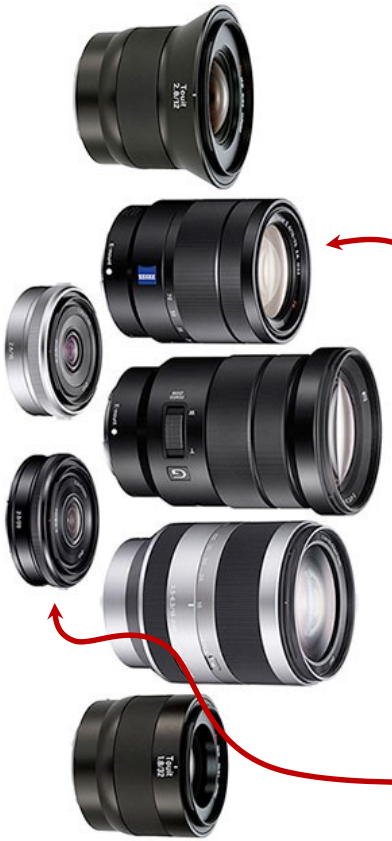
ONE Definition

Construct an object based on the class

- (1-105, 4-8)
- (55, 1.4-8)
- (zoom,, 4-7)

Multiple INSTANTIATION

Coding in Object-Orientation



Powerful representation

DESIGN

```
class Lens:
  def __init__ (self, focus, apert):
    self.focus = focus
    self.apert = apert
```

ONE Definition

Add more functions:

```
def buy(self, owner):

def display(self):
```

INSTANTIATION

```
lens1 = Lens("1-105", "4-8")
lens2 = Lens(55, "1.4-8")
lens3 = Lens("zoom", 1.4)
```

Multiple Objects
(Class members)

Put It Together

```
class Lens:
  def __init__ (self, focus, apert):
    self.focus = focus
    self.apert = apert

  def buy(self,owner):
    self.owner = owner

  def display(self):
    print("{} owns a lens of focus length: {}; aperture:
{}".format(self.owner, self.focus, self.apert))
```

Constructor

- These two boxes of coding can be in one file.

```
lens1 = Lens("1-105", "4-8")
lens2 = Lens(55, "1.4-8")
lens3 = Lens("zoom", 1.4)
```

An object is constructed by calling the __init__() constructor.

Try to Invoke More Functions

```
class Lens:
  def __init__ (self, focus, apert):
    self.focus = focus
    self.apert = apert

  def buy(self,owner):
    self.owner = owner

  def display(self):
    print("{} owns a lens of focus length: {}; aperture:
{}".format(self.owner, self.focus, self.apert))
```

```
lens1 = Lens("1-105", "4-8")
lens2 = Lens(55, "1.4-8")
lens3 = Lens("zoom", 1.4)
```

An object can invoke additional functions by using a dot notation:
`object.function(arguments)`

Example of Output:

John owns a lens of focus length: 1-105; aperture: 4-8

Homework (OOP)

```
from tkinter import *

root = Tk() # constructor: an object of Tk is constructed and labeled by "root"
root.title("Mercy GUI #1")
can = Canvas(root, width=700, height=500, bg="#aa88ff")
# constructor of Canvas: construct an object of Canvas
can.create_rectangle(10,10, 100,100, fill="#ff7711")
# x,y coordinators of the upper left cornder and lower right
can.create_rectangle(100,100, 500,250)
can.create_oval(70,70, 130,130)
can.create_oval(320,220, 380,280)
can.pack()
```

- Rewrite the above to define a function and call the function