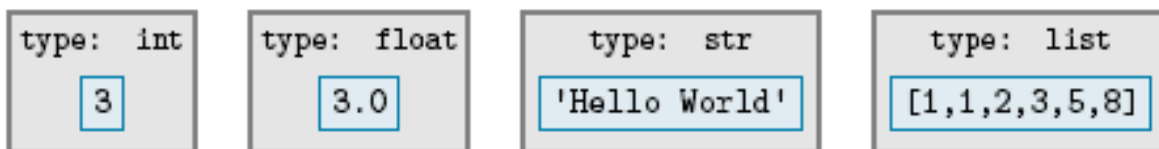


## CSSE 120 – Introduction to Software Development (Robotics section)

Concept: *Using Objects*Objects, Types and Values – and Classes

In Python, every “thing” (that is, every item of data) is called an *object*.

An *object* has a *type* and a *value*. For example:

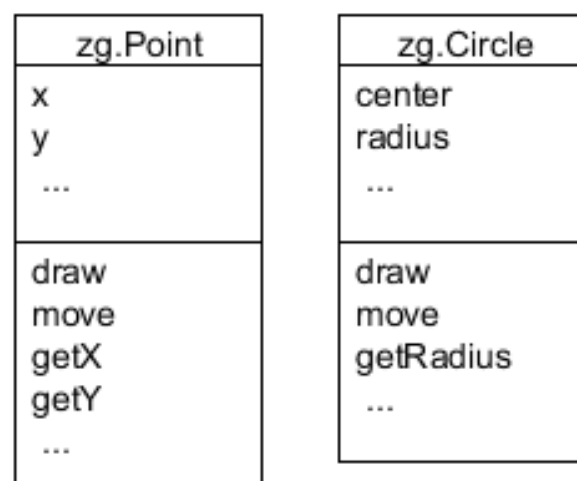


The type of an object determines:

- The *kind of thing* the object is
- What *operations* the object can do

You can create your own type, by writing a *class*. We'll see the insides of a class later, but for now all you need to know is that a class has:

- A *name*
- *Instance variables* (aka *fields*) – the data that instances of the class hold
- *Methods* – the operations (functions) that instances of the class can do



We describe these in a *UML Class Diagram*, where UML stands for Unified Modeling Language. See the examples to the right.

The 3 Key Ideas for Using Objects

- To *construct* an object:

```
win = zg.GraphWin()
point1 = zg.Point(500, 450)
line = zg.Line(point1, zg.Point(30, 40))
circle = zg.Circle(point1, 100)
```

- To *ask an object to do something*,

i.e. to apply its *methods* to it:

```
point1.draw(window)
line.move(45, -60)
x = point1.getX()
center = circle.getCenter()
```

- To reference what the object knows (its *instance variables*, aka *fields*):

```
point1.x   circle.p1   circle.p2
```

**Constructor:**

- Call it like a function, using the name of the *class*
- Style: Class names begin with an *uppercase* letter
- The constructor *allocates space* for the object and does whatever *initialization* the class specifies

**Method call:**

- Use the *dot notation*:

Who.Does\_What(With\_What)

Just like a function call, except that the method has access to the object invoking the method.

So the object is an *implicit argument* to the method call

**Instance variable (aka field) reference:**

- Use the *dot notation* but *without parentheses* Who.Has\_What