Python Data Types (3): Turtle Graphics

- Objects and Classes
- Python Standard Library
- Turtle Graphics

Python Standard Library

- In addition to built-in data types (e.g., int, float, Boolean, string, list) Python has a large library of other data types – the Python Standard Library

- You can find a list of these 'modules' here
  `Help -> Python docs -> Global Module Index`

- Each module is a file of Python code that contains definitions of one (or more) data type(s), methods (functions) that operate on objects of that type, and possibly data (like the value of pi)

```
Quick task: Look
up three modules
that have names
that interest you
and see what data
types and
functions they
contain.

Hint: you might
try random,
urllib or pickle,
for example

Share the one you
like best with
the person
sitting next to
you.
```

Turtle Graphics Module

- Turtle graphics are a simple but powerful way to draw things on a coordinate plane
- Find and open the documentation for the turtle module
- Look up turtle info in this document as we discuss turtle graphics
- The Python Standard Library is contained in the standard distribution, but you must import any module that you want to use
- To get started, import the turtle module

```
>>> import turtle
```

Turtle Graphics Module

- The turtle module defines some new classes of graphical things
- Once you've imported the turtle module, you can create a graphics screen and a turtle (a whimsical name for a drawing pen), using their constructors
- The constructor syntax is

  `variableName = moduleName.ClassName()`
- A constructor is a method, which is a kind of function. Like every function it takes a parameter list enclosed in parentheses

```
Hint: notice that
the name of a
class (Screen,
Turtle) is upper
case, while the
name of a
variable or file
(shelly, turtle)
is lower case.

This is a Python
convention,
meaning that
Python does not
force you to do
it, but we always
follow this rule
to make our code
clear and
readable.
```

Screen constructor

Turtle constructor

```
>>> import turtle
>>> aScreen = turtle.Screen()
>>> shelly = turtle.Turtle()
```

Moving a Turtle

- A turtle has a position and an orientation on a graphics screen
- Change shelly's position with a forward (or back) statement
- Change shelly's orientation with a right or left statement
- Note: forward, back, right and left are all methods in the Turtle class, and when you invoke (call) them, you must include all of these:
    - an object (instance) of the class (for example, shelly)
    - the dot operator
    - the name of the method (e.g., forward)
    - parentheses for the parameter list (even if the list is empty)

```
>>> import turtle
>>> aScreen = turtle.Screen()
>>> shelly = turtle.Turtle()
>>> shelly.forward(100)
>>> shelly.right(90)
```

A Fancier Turtle

- A turtle also has color and width attributes that you can change
- A method (function) that applies to a particular object uses the dot operator, with the syntax

  `objectName.method(parameterList)`

```
>>> shelly.color('blue')
>>> shelly.width(10)
```

A Fat Blue Triangle

- Save this example as a Python file and run it

```
t_size = 100
blueT = turtle.Turtle()
blueT.color('blue')
blueT.width(10)
blueT.forward(t_size)
blueT.right(120)
blueT.forward(t_size)
blueT.right(120)
blueT.forward(t_size)
blueT.right(120)
```

- Engage in some turtle play by changing the size, color or width of the turtle
- Use turtle graphics to draw some other shapes

Some turtle methods

Usage	Explanation
`forward() | bk()`	move the turtle
`right() | left()`	rotate the turtle
`circle()`	draw a circle
`up() | down()`	raise/lower the pen
`goto()`	move to x, y coordinate
`setheading()`	set turtle orientation
`showturtle() | hideturtle()`	set turtle visibility
`color()`	set drawing color
`width()`	set line width

What we have learned

- A Python module is a file that contains Python code – usually defining one or more related new types of things (a class).
- Each class defines a type of object and a constructor method to create new objects of that type
- Each class defines methods (functions) for doing things with an object of that type. A method is invoked (or called) using the dot ('.') operator.
- By convention, a class name is capitalized; object and method names are lower case