CLASS 06-07 - FUNCTIONS, MODULES AND OBJECTS

COMP130 – INTRODUCTION TO COMPUTING

DICKINSON COLLEGE

MODULES, OBJECTS & CLASSES

- A Python ***module*** is a collection of related functions, variables **and/or classes/objects**.
  - We've seen functions and variables…
    - `math.sqrt(x), math.cos(y)`
    - `math.pi`
    - `random.randint(a, b)`
  - Today and next class we'll learn about objects (and loosely classes).
  - Example:
    - "A Turtle Object" in the TurtleExample.ipynb notebook

OBJECT STATE

- An ***object*** represents a *thing* in a program that has ***state*** and ***methods***.
  - An **object's state** is the collection of internal variables and values that represent the object.
    - What state might be needed to represent:
      - A `Turtle`?
      - An `Account` for a banking application?

OBJECT METHODS

- An ***object*** represents a *thing* in a program that has ***state*** and ***methods***.
  - An ***object's methods*** are functions that operate on the state of the object.
  - ***Methods come in two general types:***
    - A ***constructor*** is a special method that *creates a new object* and *initializes its state*.
    - ***Instance Methods*** (usually just called ***methods***) are functions that *operate on a specified object* (i.e. use or manipulate its state.)

OBJECT CONSTRUCTORS

- A ***constructor*** is a special method that *creates a new object* and *initializes its state.*
  - `sue=turtle.Turtle()`
    - The expression `turtle.Turtle()` creates the `Turtle` and invokes the constructor.
    - The constructor name always matches the type of the object (e.g. `Turtle`).
    - The assignment (=) causes sue to refer to the new `Turtle` object.
      - Because `sue` refers to a `Turtle` we can say:
        - `sue.forward(100)`, or
        - `sue.right(270)`.
  - How is the state of a `Turtle` initialized by its constructor?

OBJECT INSTANCE METHODS

- ***Instance Methods*** (or usually just ***methods***): are functions that *operate on the state* of a specific object.
  - Instance methods are called (invoked) using ***dot notation.***
    - `sue.forward(100)`
    - `print(sue.xcor())`
      - The `forward` and `xcor` will operate on the state of the `Turtle` *referred* to by `sue`.

OBJECT STATE DIAGRAMS

- An ***Object State Diagram*** is a state diagram that shows the state of a specific object.

ANOTHER EXAMPLE

- Imagine a system for managing a bank…
  - This system might have an object named `Account` that represents a customer's account.
    - What state might be needed to represent an account?
    - What methods might an `Account` object have?
    - Would the constructor need any information to initialize a new `Account` object?

CLASSES

- A Python ***module*** is a collection of related functions, variables **and/or classes/objects.**
  - A ***class*** is a collection of code that defines a type of object.
  - It specifies:
    - What state the object has.
    - How the state is initialized (i.e. the constructor).
    - What methods the object has.
    - How those methods operate on the state.

End of Class 06 material.

HAVING MULTIPLE OBJECTS

- We can construct more than one object of any given type.
  - Example:
    - "Multiple Turtle Objects" in the TurtleExample.ipynb notebook.

INDEPENDENT STATE

- Every object has its own state that is independent of any others.
  - Two Turtles… Two States
  - Dot notation specifies which object's state a method operates on.
    - `sue.forward(50)`
      - Operates on the state of the `Turtle` object referred to by `sue`.
    - `joe.left(90)`
      - Operates on the state of the `Turtle` object referred to by `joe`.

FRUITFUL AND FRUITLESS FUNCTIONS

- Fruitless Functions/Methods perform some operation and do not return a value.
  - `print(txt + '!')`
  - `sue.forward(100)`
  - `joe.pencolor('red')`
- Fruitful Functions/Methods return a value that can be used in further computations.
  - `txt_msg='miss U ' + str(2)`
  - `sueHeading=sue.heading()`
  - `joeDist=math.sqrt(joe.xcor()**2 + joe.ycor()**2)`