Classes

Python

- Straightforward extension to what we've seen:

```
class MyClass:
   i=12345
   def __init__(self,x,y):
      self.x=x
      self.y=y
   def reset(self,x,y):
      self.x=x
      self.y=y
  def do_something(self,z)
     return MyClass.i+self.y-z
```

Self

- The first argument to the class methods must be `self`. This stands for the particular class instance for which the function is being invoked.
- The self argument is *not* used when we call the function. It is understood.

```
thing=MyClass(x,y)
thing.reset(w,z)
```

Constructor

- The __init__ function is the constructor. It is automatically called when a new variable of the class is instantiated.

```
aVar=MyClass(x,y)
```

Destructor

- The __del__ function isn't exactly a destructor; it is called when the garbage collector deletes the object (which your code doesn't do explicitly).

Fortran (2003)

- Similar to type

```
module mytype
type MyType
    integer   :: i,j
    real      :: x,y
     contains
       procedure :: init=>init_class
end type MyType

private init_class

contains
subroutine init_class(this,stuff1,stuff2)
    class(MyType) :: this
    real          :: stuff
    this%x=stuff1; this%y=stuff2
end subroutine init_class
end module mymod
…
type(mytype) :: thing
thing%init(x,y)
```

Inheritance (Single Only)

```
module mytype extends(OtherType)
type MyType
    integer   :: i,j
    real      :: x,y
    contains
       procedure :: init=>init_class
end type MyType
contains
subroutine init_class(this,stuff1, stuff2)
    class(MyType)
    real stuff
    this%x=stuff1; this%y=stuff2
end subroutine init_class
end module mymod

…

type(mytype) :: thing
thing%init(x,y)
```

No Constructor, Destructor

- We can define a FINAL procedure:

```
procedure::stuff
final:: free_up
end type etc.
contains
subroutine final(self)
```

- The final subroutine is invoked when the system deallocates the instance. It is not inherited by child classes.