Object-Oriented Scripting in Python

Overview

- Object-Oriented
- Object-Oriented Python
- Object-Oriented Scripting in Python
  - Unified Post Example
- Exception Handling

Object-Oriented Programming

Objects and Classes

- What is an object?
  - A logical grouping of functions and data.
- What is a class?
  - A class is a blueprint for making an object.

Object-Oriented Programming
A Square Example

- A Square:
  - Has a width.
  - Has a color.
- Functions:
  - Perimeter = 4*width
  - Area = width*width

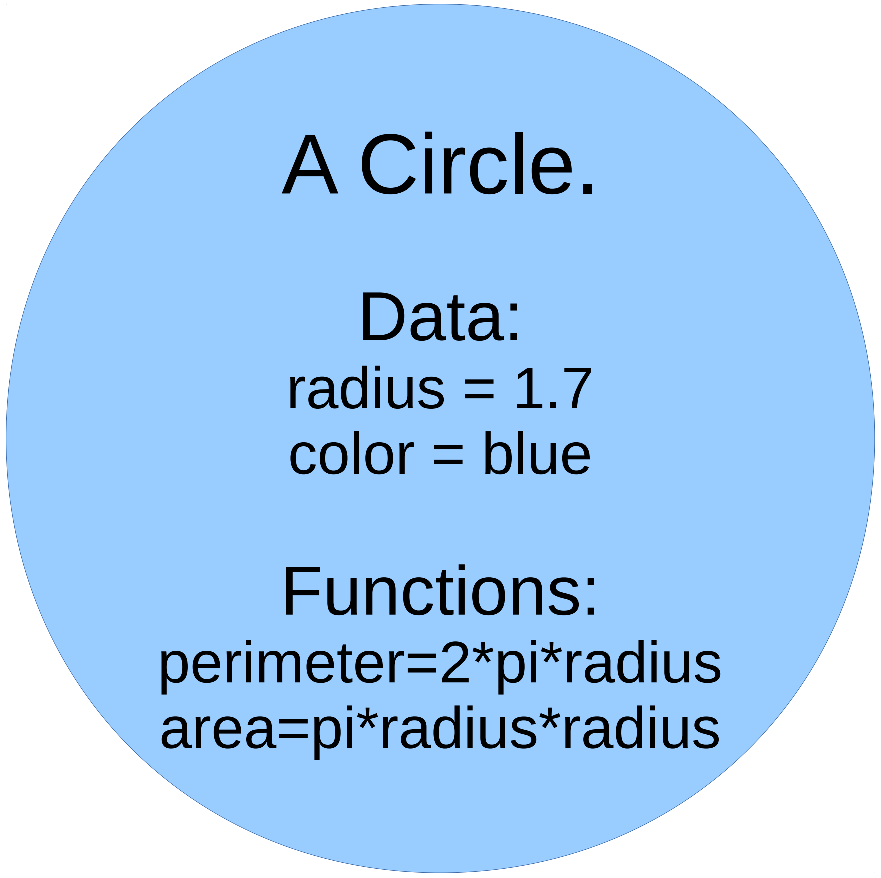
A Square.

Data:
width = 3
color = blue

Functions:
perimeter=4*width
area=width*width

Object-Oriented Programming
A Circular Example

- A Circle:
  - Has a radius.
  - Has a color.
- Functions:
  - Perimeter = 2*pi*radius
  - Area = pi*radius*radius

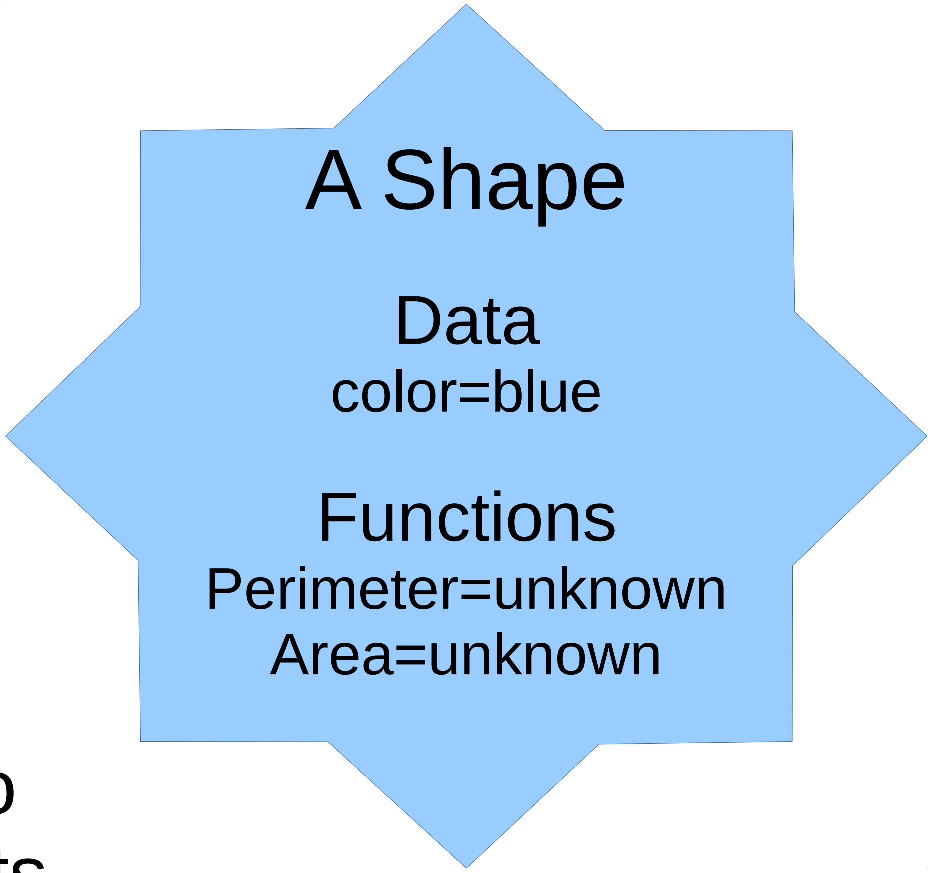


Object-Oriented Programming
Inheritance

- Squares and Circles both have colors, perimeters, and areas.
  - Why is there so much in common?
  - They are **Shapes**.
- Define a Shape class.

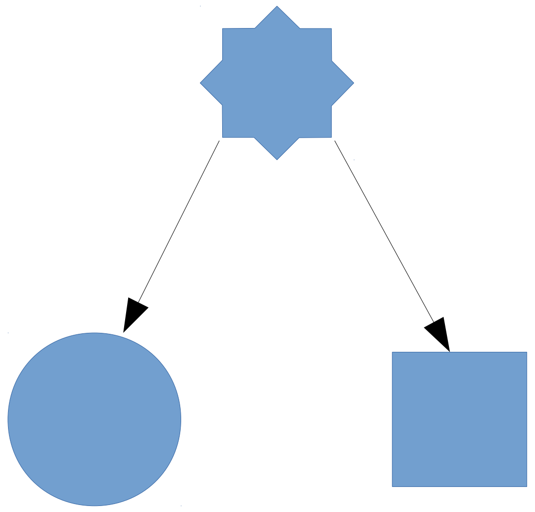
Object-Oriented Programming
A Shape Example

- A Shape:
  - Has a color.
  - Has a perimeter
  - Has an area
- Pure virtual functions:
  - perimeter
  - area
- Shape does not know how to determine its perimeter nor its area.



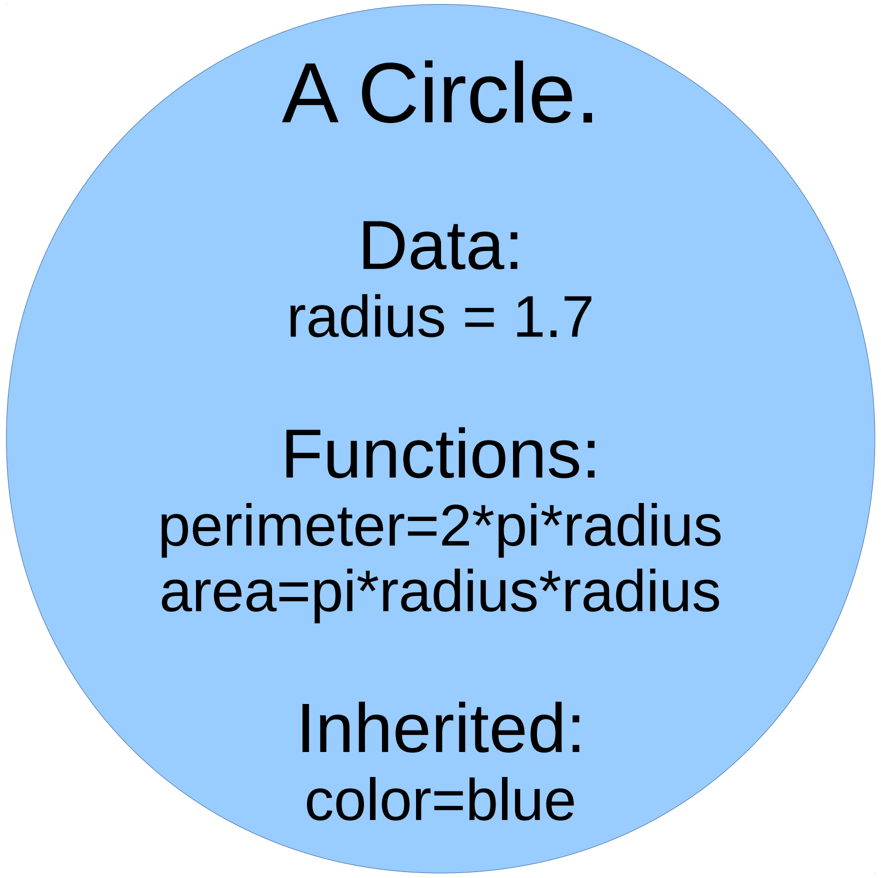
Object-Oriented Programming
A Shape Example

- Square and Circle are subclasses of Shape.
  - Shape implements the color.
  - Square calculates the perimeter and area from the width.
  - Circle calculates the perimeter and area fro the radius.



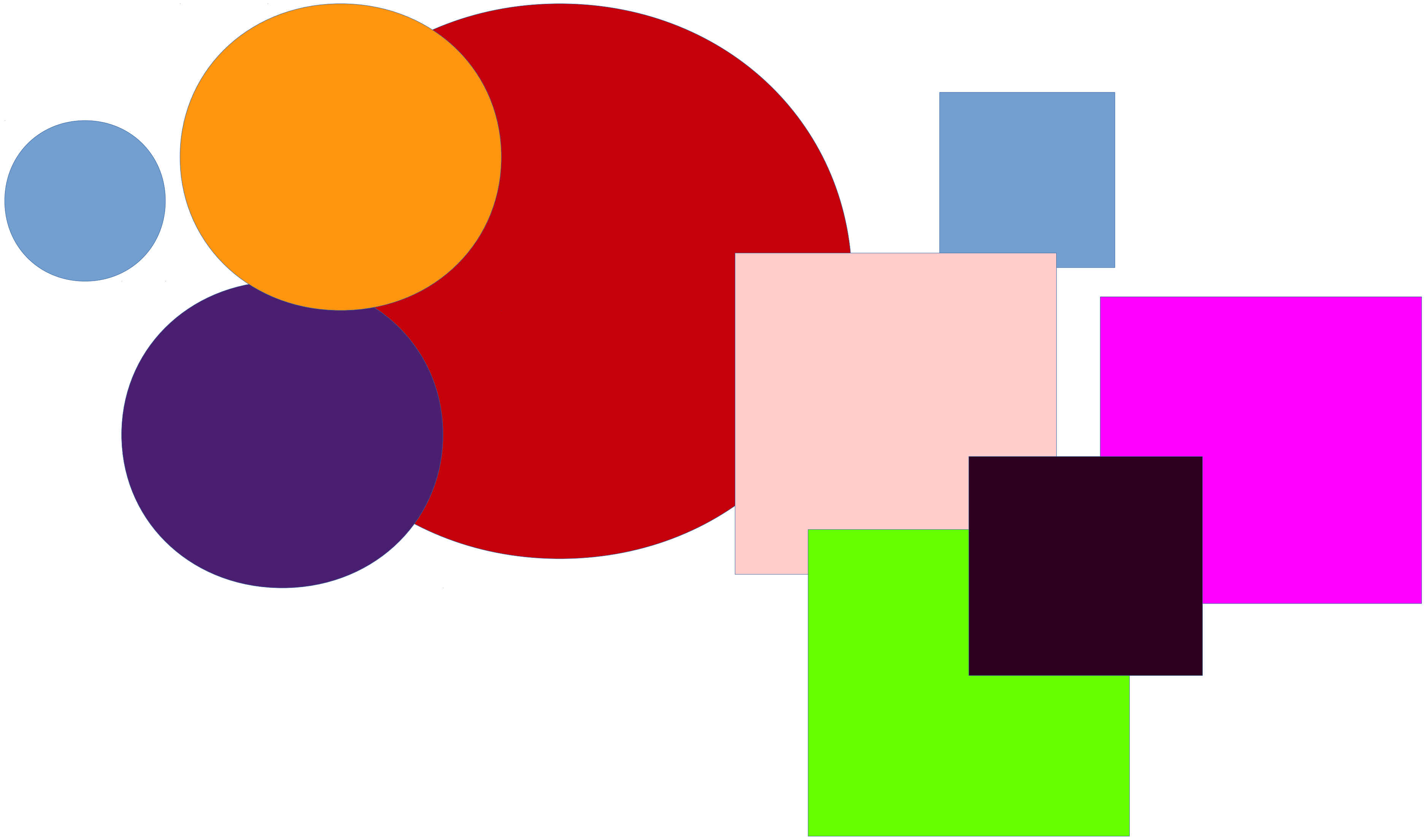
Object-Oriented Programming
A Circular Example

- A Circle:
  - Has a radius.
- Functions:
  - Perimeter = 2*pi*radius
  - Area = pi*radius*radius
- Is a Shape:
  - This gives us the color.



Object-Oriented Programming

Objects are Instances of Classes



Object-Oriented Python

class Shape

```
class Shape:
    def __init__(self,color):
        self.__color=color
    @property
    def color(self):
        return self.__color
    @property
    def perimeter(self):
        return NotImplemented
    @property
    def area(self):
        return NotImplemented
```

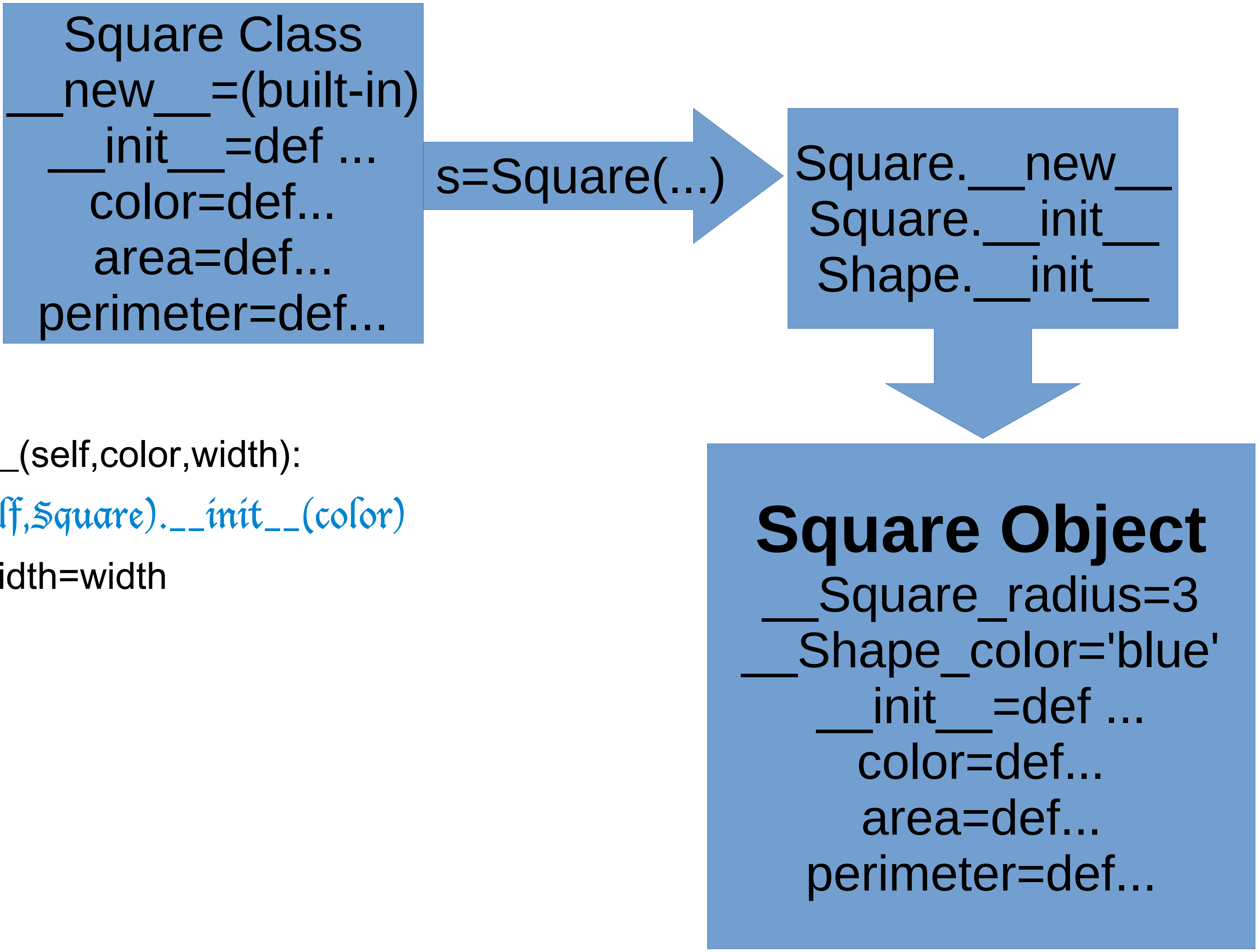
Object-Oriented Python

class Circle

```
class Circle(Shape):
    def __init__(self,color,radius):
        super(self,Circle).__init__(color)
        self.__radius=radius
    @property
    def perimeter(self):
        return math.pi*self.__radius*2
    @property
    def area(self):
        return math.pi*self.__radius**2
```

Python OO Mechanics

Constructing



```
def __init__(self,color,width):
   super(self,Square).__init__(color)
   self.__width=width
```

Object-Oriented Scripting

class UnifiedPost

```
class UnifiedPost:
  def __init__(self,infile,fixd,postexec,when):
    (self.infile,self.fixd.self.postexec,self.when)=\
        infile,    fixd,    postexec,    when
  def run_post(self):
    self.link_fix()
    self.make_itag()
    make_symlink(self.infile,”INFILE”,
             logger=self.log(),force=True)
    cmd=mpirun(mpi(self.postexec)<”itag”)
    checkrun(cmd,all_ranks=true,logger=self.log())
  def link_fix(self):
    fixes=[f for f in glob.glob(fixd+”/*”)]
    make_symlinks_in(fixes,”.”,logger=self.log())
```

Object-Oriented Scripting

HWRFPost, NEMSPost

```
class HWRFPost(UnifiedPost):
  def make_itag    (self):
    with open(“itag”,”wt”) as f:
      itagdata=self.when.strftime(
        “INFILE\nnetcdf\n%Y-%m-%d_%H:%M:%S”  “\nNMM NEST\n”)
      f.write(itagdata)


class NEMSPost(UnifiedPost):
  def make_itag    (self):
    with open(“itag”,”wt”) as f:
      itagdata=self.when.strftime(
        “INFILE\nnetcdf\n%Y-%m-%d_%H:%M:%S” “\nNEMS\n”)
      f.write(itagdata)
```

Object-Oriented Exception Handling

What if Something Fails? try/except/finally

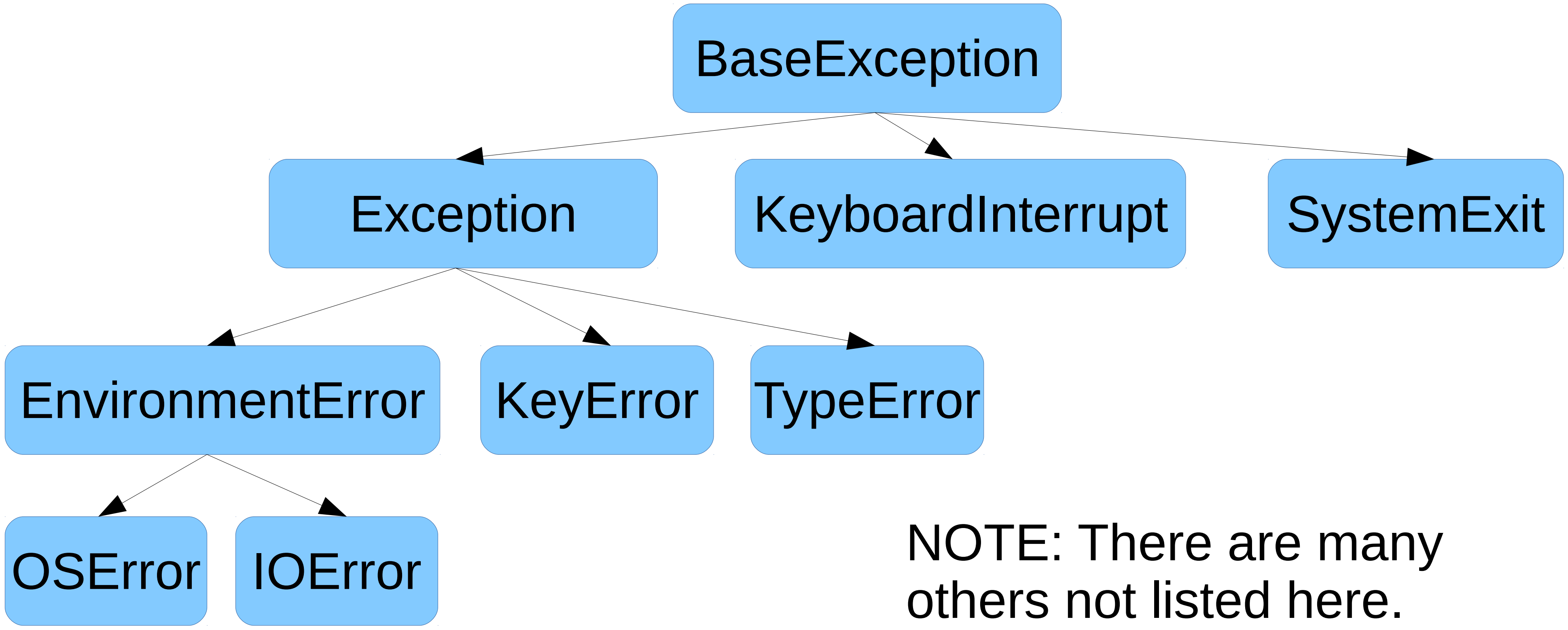
```
try:
  ... code that may break ...
except ExceptionClass as e:
  print 'Something broke!'
except AnotherExceptionClass as a:
  print 'Something else broke!'
finally:
  print 'This line is always run.'
```

- NOTE: finally and except are optional; only one of them must be present

Object-Oriented Exception Handling

Exception Classes

- Exceptions are objects.
- Python has pre-defined classes of exceptions.
  - Never raise BaseException; raise subclasses of Exception if possible.



NOTE: There are many others not listed here.

Object-Oriented Scripting

Workflow Object Structure

- ush/hwrf_expt.py:

```
post=HWRFPost('/path/to/infile',
   '/path/to/fixd', '/path/to/hwrf_post',
   to_datetime('2015081818'))
```

- scripts/exhwrf_run_post.py:

```
import hwrf_expt
hwrf_expt.init_module()
hwrf_expt.post.run_post()
```

- Rocoto/ecFlow would be configured to run the new ex-script.

?