LESSON 4 – Inheritance, Errors & Exception Handling

Background: For this lesson, we will be reviewing how exception handling is implemented in Python. As well as, catching specific errors and using class inheritance to create parent/child class relationships.

SECTION 1 – Exception Handling

Just like in many other programming languages, exception handling is present in Python.

Observe the following code along with the produced output to get a better grasp of the syntax for performing exception handling in Python:

```
class MyClass:
# import module sys to get the type of exception ← important
import sys

randomList = ['a', 0, 2]

for entry in randomList:
    try:
        print("The entry is", entry)
        r = 1/int(entry)
        break
    except:
        print("Oops!",sys.exc_info()[0],"occured.")
        print("Next entry.")
        print()
print("The reciprocal of",entry,"is",r)

# output:
The entry is a
Oops! <class 'ValueError'> occured.
Next entry.

The entry is 0
Oops! <class 'ZeroDivisionError' > occured.
Next entry.

The entry is 2
The reciprocal of 2 is 0.5
```

SECTION 2 – Catching Specific Errors

This is not all though, you can also catch specific exceptions or errors and handle them accordingly.

Observe the code below:

```
try:
   # do something
   pass

except ValueError:
   # handle ValueError exception
   pass
```

```
except (TypeError, ZeroDivisionError):
   # handle multiple exceptions
   # TypeError and ZeroDivisionError
   pass

except:
   # handle all other exceptions
   pass

finally:
   # do something at the end
```

SECTION 3 – Class Inheritance

Python also supports class inheritance.

Study the code below to find a parent and child class.

Note: *There are a couple attributes which are inherited from the child class – try to observe what they are.*

```
class Polygon:
    def __init__(self, no_of_sides):
        self.n = no_of_sides
        self.sides = [0 for i in range(no_of_sides)]

    def inputSides(self):
        self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i in range(self.n)]

    def dispSides(self):
        for i in range(self.n):
            print("Side",i+1,"is",self.sides[i])
```

```
class Triangle(Polygon):
   def __init__(self):
      Polygon.__init__(self,3)

   def findArea(self):
      a, b, c = self.sides
      # calculate the semi-perimeter
      s = (a + b + c) / 2
      area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
      print('The area of the triangle is %0.2f' %area)


t = Triangle()

t.inputSides()
Enter side 1 : 3
Enter side 2 : 5
Enter side 3 : 4

t.dispSides()
Side 1 is 3.0
Side 2 is 5.0
Side 3 is 4.0

t.findArea()
The area of the triangle is 6.00
```

SECTION 4 – Exercise Your Python

1. **Create a Parent and Child class of Animal and Dog respectively.**
    a. The Dog child class should share two attributes with its parent class – age and weight.
        i. This means that these attributes should be stored within the Parent class and instantiated within the child class. (e.g. see above with the instantiation of the Triangle class.)
        ii. However, you will require the age and weight to be passed in as parameters upon instantiation.
        iii. Also, add an exception for a weight and age below 0.

Use the link below for a list of errors and exceptions:

https://www.tutorialspoint.com/python/standard_exceptions.htm

2. **Create a new function under the Dog child class which sorts a given array of dogs by age.**
    a. This function should return all the dogs returned in an array in sorted order from youngest to oldest.
        i. You must implement this function using the merge sort algorithm.

Note: See https://gist.github.com/jvashishtha/2720700 for an example of a Python implementation of merge sort. *Your implementation will be a little different since you're dealing with objects instead of numbers, but the principle remains.*