Volume of 3D Shapes

Write a program that accepts the dimensions of a 3D shape and prints its corresponding volume.

PROGRAM DESIGN
Create a class named Shape. Only this class should exist. Within this class, are the following methods: the constructor method and the printVolume method**. Only this class should exist and should be coded within your file.** Name your file as Volume3d.py

Shape
- *attribs
+ printVolume()

The constructor should accept *any number* of **arbitrary arguments**. These arguments are the ***diameter*** and ***height*** of a given right-cylinder, **in that order**. However, if only 1 argument was passed to the constructor, it is assumed that this is the ***diameter*** of a sphere.

The printVolume method, gets and prints the volume of the shape given its dimensions.

Only the following shapes are entertained in the printVolume method: **cylinder** (*right-angle*), and **sphere**. However, if the user entered any number of keyword arguments, other than 2 (cylinder) or 1 (sphere) - then, the error "Shape Volume Error!" should be printed instead.

Shape	Volume
Right-Cylinder	$\pi r^2 h$
Sphere	$\frac{4}{3}\pi r^3$

Note: Please use **3.14159** as *the constant π value.*

Name this exercise as *Volume3d.py*

INPUT
There will be *no inputs* for this activity, as you are only tasked to create the required class. However, if you want to try the test cases below, you can download the inputs.py file – and run it along with your Volume3d.py file. Note: you are *not* allowed to edit the contents of inputs.py or place the contents of inputs.py in Volume3d.py!

OUTPUT
Once the user enters the input, the program should be able to calculate and print the given shape's area, rounded off to 1 floating point (use %.1f). However, if the user decides to enter 0 or negative values in the diam or hei values, then the program should output "Shape Volume Error!" instead.

SAMPLE INPUT

```
0 5
5 5
3 5
5
5 2
-3 -5
```

SAMPLE OUTPUT

```
Shape Volume Error!
98.2
35.3
65.4
39.3
Shape Volume Error!
```

GRADING CRITERIA

Implementation of programming structures – 35%

- This entails that loops and conditionals are implemented judiciously on the appropriate python sequences for the program specifications. Judicious implementation means that redundancy in control structures (conditions and loops) should be avoided when these can be simplified. Classes and methods should be properly coded, without any redundancies. Use of Object-Oriented functionality (classes and methods) should be strictly observed.

Functionality of the program – 30%

- This entails that the program should have the proper methods that would process the data inputs and output of the project correctly (as stated in the sample inputs and outputs). No unhandled errors should be encountered. Brute force solutions (predefined outputs for specific inputs) are not allowed.

Proper documentation of the program – 25%

- This entails that proper documentation was made on each functionality / structure done in the code through proper comments within the code.