Task 1 Add a class of Triangle. Write testing code to verify the new class works as expected.

In [45]:
```
import math as mth
import random
```

A package is basically a directory with Python files and a file with the name init . py.

This means that every directory inside of the Python path, which contains a file named init . py, will be treated as a package by Python.

The init method is the Python equivalent of the C++ constructor in an object-oriented (OO) approach.

The init function is called every time an object is created from a class.

The init method lets the class initialize the object's attributes and serves no other purpose.

It is only used within classes.

In [46]:
```
# Here I will set up a Geom class for all geometric figures which all know their name
# Below is a standard class definition.

class Geom():
    geomType = 'Generic Geometry Type'
    def __init__(self):
        self.name = random.choice(['Bill','Sally','Tamica','Josh','Lammar','Hussain'])
        self.color = random.choice(['BLUE', 'RED', 'PURPLE'])
    def print_name(self):
        print('My name is ',self.name, 'and my color is ',self.color)
    def makeString(self):
        return f"Name: {self.name}, Color: {self.color}, Area: {self.area()}"
class Circle(Geom):
    def __init__ (self,radius):
        self.radius = radius
        super().__init__()

# Creating subclasses of the geometry types
# The area of a circle is pi times the radius squared (A = π r²), so we apply that for

    def area(self):
        return mth.pi * self.radius **2
class Square(Geom):
    def __init__ (self,side):
        self.side = side
        super().__init__()

# The area of a square is equal to (side) × (side) square units, so it will be:

    def area(self):
        return self.side **2

# As a part of my Task 1 I will define first my new class (Triangle).
# After that I will find a triangle's area, using an appropriate formula.
# The approach is similar to Circle and Square classes, but the formula is a bit more
# Since it was not specified, I have decided to use a triangle with all sides (and ang
# It means that I will use equilateral triangle formula.
# The area of an equilateral triangle is A = √3*a2 / 4.
# If we decide to use a scalene triangle, I should need to have a height of triangle w
# h = (side)**2 - (side/2)**2
# The area of scalene triangle can be calculated by using the formula:
# A = 0.5*(base self.side * h)

class Triangle(Geom):
    def _init_ ():
        self.side = side
        super().__init__()
    def area(self):
        return (sqrt(3) * (side)**2) / 4
```

In [47]:

```
# Above: I defined another class (Triangle) in the model, which is allowed.
# Geom module/package can contain multiple classes.
# Below: the side value (for multiple classes) is set on 8.
```

In [48]:

```
side = 8
my_square = Square(side)
my_square.print_name()
print('My area is ',my_square.area())
```

```
My name is  Tamica and my color is  PURPLE
My area is  64
```

Modules are collections of methods and constants. They cannot generate instances. Classes may generate instances (objects), and have per-instance state (instance variables). Everything in Python libraries and other Python applications is either a module or a package of modules. There is no limit on how many classes one can put in a module.

In [49]:

```
# Creating a list of circles with radius i (2,3)
```

In [50]:

```
circle_list = [Circle(i) for i in range(2,3)]
print(circle_list)
for x in circle_list:
    x.print_name()
print([x.makeString() for x in circle_list])
```

```
[<__main__.Circle object at 0x0000020EA90CCC70>]
My name is  Tamica and my color is  PURPLE
['Name: Tamica, Color: PURPLE, Area: 12.566370614359172']
```

Task 2

Reorganize these classes as a package with one or multiple modules. Import them into another Python Notebook or Script file to test them. The purpose of this task is to understand the concepts of methods or functions, classes, modules, and packages as well as their relationships and organization in Python. This would help us comprehend the real meaning of *(from ...) import ....*

More importantly, it is always good to organize your well-developed code as a package so that it can be extensively used by other members in the community.

It is always useful to define package, sub-package, module, and base class. Each package consist of modules and each module consist of classes.

We can also import specific functions, or classes, from a module or package, using the syntax from "our module" import "our function".

Module is a file that contains code to perform a specific task.