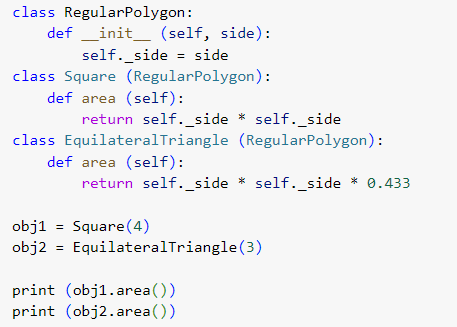
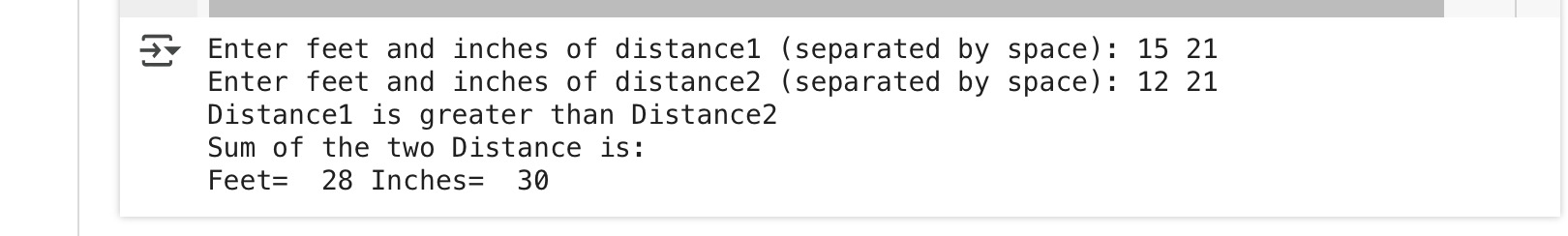
|  |  |
| --- | --- |
| **Laboratory Activity No. 7** | |
| **Polymorphism** | |
| **Course Code:** CPE103 | **Program:** BSCPE |
| **Course Title:** Object-Oriented Programming | **Date Performed:** 02/22/25 |
| **Section:** BSCpE 1A | **Date Submitted:** 02/25/25 |
| **Name:** Nerio, Hannah Grace A. | **Instructor:** Engr. Maria Rizette Sayo |
| **1. Objective(s):** | |
| This activity aims to familiarize students with the concepts of Polymorphism in Object-Oriented Programming | |
| **2. Intended Learning Outcomes (ILOs):** | |
| The students should be able to:   * 1. Identify the use of Polymorphism in Object-Oriented Programming   2. Implement an Object-Oriented Program that applies Polymorphism | |
| **3. Discussion:** | |
| Polymorphism is a core principle of Object-Oriented that is also called “method overriding”. Simply stated the principles says  that a method can be redefined to have a different behavior in different derived classees.  For an example, consider a base file reader/writer class then three derived classes Text file reader/writer, CSV file reader/ writer, and JSON file reader/writer. The base file reader/writer class has the methods: read(filepath=””) , write(filepath=””). The three derived classes (classes that would inherit from the base class) should have behave differently when their read, write methods are invoked.  Operator Overloading:  Operator overloading is an important concept in object oriented programming. It is a type of polymorphism in which a user defined meaning can be given to an operator in addition to the predefined meaning for the operator.  Operator overloading allow us to redefine the way operator works for user-defined types such as objects. It cannot be used for built-in types such as int, float, char etc., For example, '+' operator can be overloaded to perform addition of two objects of distance class.  Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator. For example, when we use + operator on objects, the magic  method add () is automatically invoked in which the meaning/operation for + operator is defined for user defined objects. | |
| **4. Materials and Equipment:** | |
| Windows Operating System Google Colab | |
|  | |

|  |
| --- |
| **5. Procedure:** |
| **Creating the Classes**   1. Create a folder named oopfa1<lastname>\_lab8 2. Open your IDE in that folder. 3. Create the base polymorphism\_a.ipynb file and Class using the code below:   Coding:  # distance is a class. Distance is measured in terms of feet and inches class distance:  def init (self, f,i):  self.feet=f self.inches=i  # overloading of binary operator > to compare two distances def gt (self,d):  if(self.feet>d.feet):  return(True)  elif((self.feet==d.feet) and (self.inches>d.inches)): return(True)  else:  return(False)  # overloading of binary operator + to add two distances def add (self, d):  i=self.inches + d.inches f=self.feet + d.feet if(i>=12):  i=i-12 f=f+1  return distance(f,i)  # displaying the distance def show(self):  print("Feet= ", self.feet, "Inches= ",self.inches)  a,b= (input("Enter feet and inches of distance1: ")).split() a,b =[int(a),int(b)]  c,d= (input("Enter feet and inches of distance2: ")).split() c,d =[int(c),int(d)]  d1 = distance(a,b) d2 = distance(c,d)  if(d1>d2):  print("Distance1 is greater than Distance2") else:  print("Distance2 is greater or equal to Distance1") d3=d1+d2  print("Sum of the two Distance is:") d3.show() |

4. Screenshot of the program output:





# Testing and Observing Polymorphism

1. Create a code that displays the program below:
2. Save the program as polymorphism\_b.ipynb and paste the screenshot below:

A screenshot of a computer program

AI-generated content may be incorrect.

1. Run the program and observe the output.
2. Observation:

This Python program defines a base class called RegularPolygon and two subclasses, Square and

Equilatera lTriangle. The RegularPolygon class has an initializer (*\_init\_*) that takes the side length as an argument and stores it. The Square class inherits from RegularPolygon and defines an area() method that calculates the

area using the formula side × side. Similarly, the EquilateralTriangle class also inherits from RegularPolygon and

has an area() method that estimates the area using the formula side × side × 0.433, which is an approximation

of (√3)/4, the actual formula for the area of an equilateral triangle. In the program, an instance of Square is

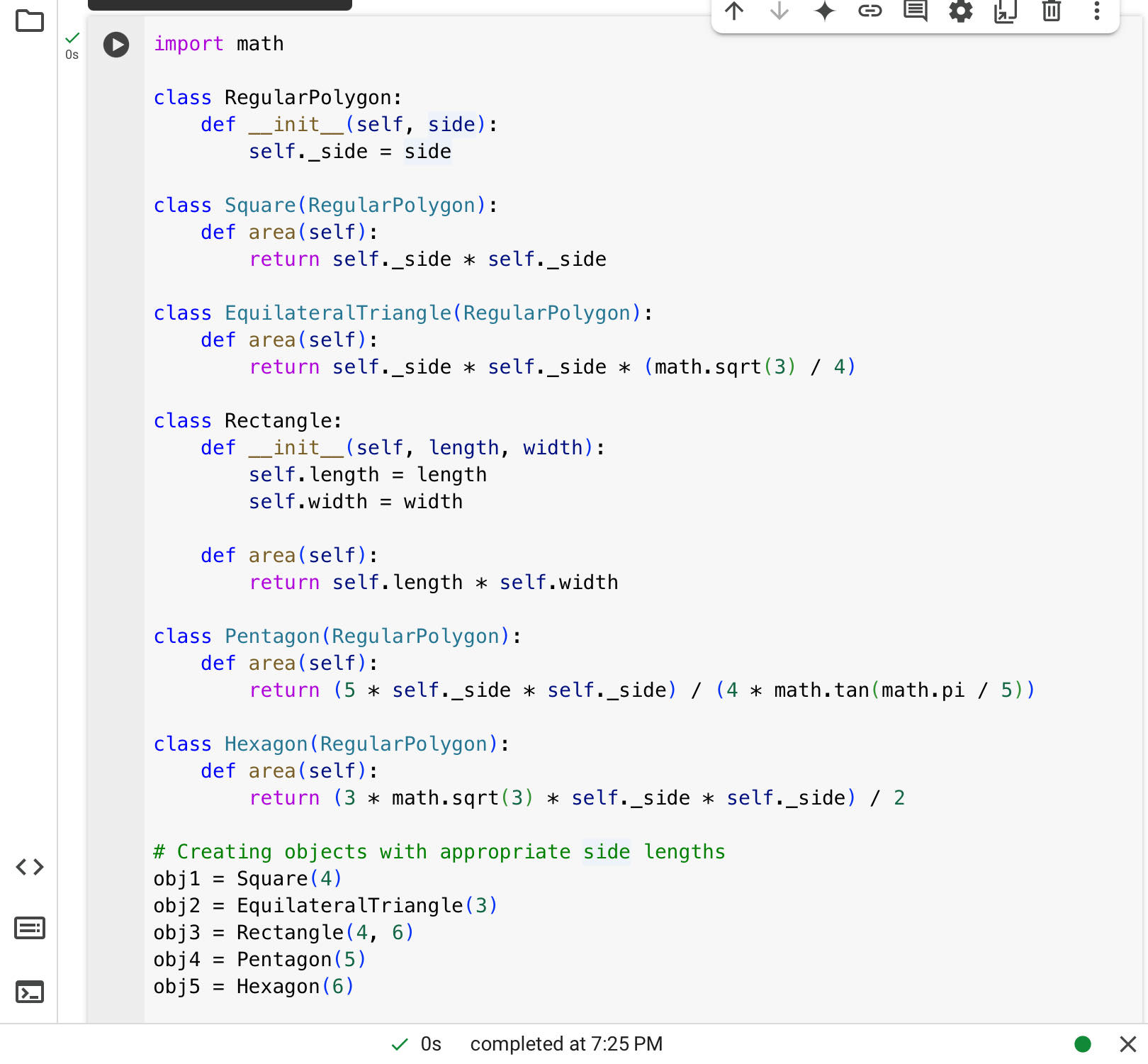
created with a side length of 4, and an instance of EquilateralTriangle is created with a side length of 3. When

their area() methods are called, the program prints 16 for the square (since 4 × 4 = 16) and approximately 3.897

for the triangle (since 3 × 3 × 0.433 ≈ 3.897)

6. Supplementary Activity:

In the above program of a Regular polygon, add three more shapes and solve for their area using each proper formula. Take a screenshot of each output and describe each by typing your proper labeling.



# Rectangle

Unlike the other shapes, the Rectangle class does not inherit from RegularPolygon because it requires

two attributes: length and width. The area is computed using the formula Area = length × width, making it

straightforward yet distinct from regular polygons. The class is properly structured with an *\_init\_* method

to initialize its dimensions, ensuring flexibility for different rectangle sizes. For a 4 × 6 rectangle, the

output is 24, confirming the correctness of its implementation. Keeping it separate from regular polygons

improves code clarity and maintains proper object-oriented structure.

#Pentagon

The Pentagon class extends RegularPolygon and computes its area using the formula Area = (5 × side²)

/ (4 × tan(π/5)), leveraging trigonometric functions to ensure precision. This formula accounts for the

interior angles of a pentagon, making it accurate for any given side length. The class benefits from

inheritance by reusing the \_side attribute, simplifying its design. For side length 5, the calculated area is

43.01, demonstrating the effectiveness of using mathematical functions instead of approximations,

ensuring reliability and extensibility for other polygonal shapes.

#Hexagon

The Hexagon class also extends RegularPolygon and calculates its area using the formula Area = (3 ×

√3 × side²) / 2, a widely used formula for regular hexagons. The implementation correctly applies

math.sqrt(3) for better accuracy, avoiding manual approximations. The use of inheritance keeps the code

clean and modular, allowing easy extension to other polygons with similar properties. For a hexagon with

side length 6, the computed area is 93.53, verifying the correctness of the formula. This approach makes

the code both scalable and mathematically precise.

|  |
| --- |
| **Questions**   1. Why is Polymorphism important?   Polymorphism is a key idea in object-oriented programming that lets objects of different classes be treated as if they are objects of a common superclass. This is important because it makes code more flexible and reusable. Instead of writing multiple versions of a function for different types, you can write one function that works with objects of different classes. This simplifies the code and makes it easier to maintain and extend. When new classes are added, the existing code can work with these new classes without needing changes.   1. Explain the advantages and disadvantages of using applying Polymorphism in an Object-Oriented Program.   Polymorphism in object-oriented programming has its pros and cons. On the positive side, it makes code more flexible and reusable. You can write functions that work with different types of objects, which simplifies the code and makes it easier to maintain. When new classes are added, the existing code can handle them without needing changes. Polymorphism also allows for dynamic method binding, meaning the method that gets called is decided at runtime based on the object's actual type, adding flexibility to the program.  However, there are some downsides. Polymorphism can introduce a performance overhead because the system needs to figure out which method to call at runtime, which can be slower. Debugging polymorphic code can be tricky since the actual method called is determined at runtime, making it harder to trace issues. While it simplifies certain aspects of code, it can also add complexity, especially for those not familiar with the concept. If used incorrectly, polymorphism can lead to code that is harder to understand and maintain.   1. What maybe the advantage and disadvantage of the program we wrote to read and write csv and json files?   Writing a program to read and write CSV and JSON files has its benefits and drawbacks. On the plus side, such a program can make data handling much easier. CSV files are great for handling tabular data and are easy to read and write. JSON files are excellent for storing structured data and are widely used in web applications. Having a program that can handle both formats means you can work with different types of data seamlessly, making your code more versatile and useful.  However, there are some downsides. One disadvantage is that handling different file formats can add complexity to your code. You need to ensure that your program correctly reads and writes both CSV and JSON files, which can be tricky and time-consuming. Additionally, if the data is large, reading and writing these files can be slow and may require more memory. Debugging issues related to file handling can also be challenging, especially if the data is not formatted correctly.   1. What maybe considered if Polymorphism is to be implemented in an Object-Oriented Program?   When implementing polymorphism in an object-oriented program, there are several important things to consider. First, ensure that your classes are properly designed with a common superclass or interface that defines the shared methods. This allows different classes to be treated as instances of the superclass or interface. Second, think about how you will use dynamic method binding, where the method that gets called is determined at runtime based on the object's actual type.Third, consider the performance impact, as polymorphism can sometimes slow down your program due to the overhead of dynamic method binding. Finally, make sure your code is well-documented and easy to understand, as polymorphism can make debugging and maintenance more challenging.   1. How do you think Polymorphism is used in an actual programs that we use today?   Polymorphism is used in many programs we use today to make them more flexible and efficient. For example, in a drawing application, you might have different shapes like circles, squares, and triangles. Polymorphism allows the program to treat all these shapes as a generic "shape" type, so it can call the same method to draw any shape without knowing the specific type. This makes the code simpler and easier to maintain. Similarly, in a payment system, polymorphism lets the program handle different payment methods (like credit cards, PayPal, and bank transfers) using the same interface, making it easier to add new payment options in the future. |
| **7. Conclusion:** |
| In this activity, I learned how polymorphism works in object-oriented programming and how it makes coding easier and more flexible. Polymorphism allows different classes to share the same method names but have their own unique behavior. This makes the code more reusable and easier to update in the future. One of the things I found interesting was operator overloading, where the same operator can work differently depending on the type of object. I also saw how polymorphism is used in real-life applications, like reading and writing different file formats or calculating areas of different shapes. While polymorphism has many benefits, I also realized that it can make debugging harder and might slow down a program if not used properly. |
| **8. Assessment Rubric:** |