**Name:** Mohamed Mohamed Alsayed Hassan

**How super Function Handle Multiple Inheritance**

The **super()** function is used to give access to methods and properties of a parent or sibling class. Let’s look at some examples to see how super() function handle multiple inheritance.

**A screen shot of a computer program

AI-generated content may be incorrect.Example 1:**

When it comes to multiple inheritance, super() function is very helpful. In the Mybook class, you might think Book.\_\_init\_\_() will be called twice for both Ebook and PrintedBook, but super() function is smart enough to call it just once.

Let’s look at another example.

**Example 2:**

A computer screen shot of a program

AI-generated content may be incorrect.

Here, it follows the method resolution order, it looks for \_\_init\_\_ at Employee class first. There, it will find the Person.\_\_init\_\_ function and execute it, then finishes the Employee.\_\_init\_\_ function. It does not execute the Person\_\_init\_\_ function again for the inherited Person class since it is already executed in the Employee.\_\_init\_\_ function.

A screen shot of a computer program

AI-generated content may be incorrect.**Example 3:** If Human and Mammal Have the same method like eat but with different Implementation. When Child[Employee] calls eat method how python handle this case.

To explain what happened here, let’s first define the **MRO.** The **MRO** stands for Method Resolution Order. It is the order in which Python looks for a method in a hierarchy of classes. In multiple inheritance, methods are executed based on the order specified while inheriting the classes. In our example, it searches for the ‘eat’ function from the left to the right class. First, it looks for an ‘eat’ function in the Human class. Since Human class has ‘eat’ function, it will be executed. If we reverse the classes and put Mammal first, its ‘eat’ function will be executed instead.

A computer screen shot of a program

AI-generated content may be incorrect.