**OOPS:**

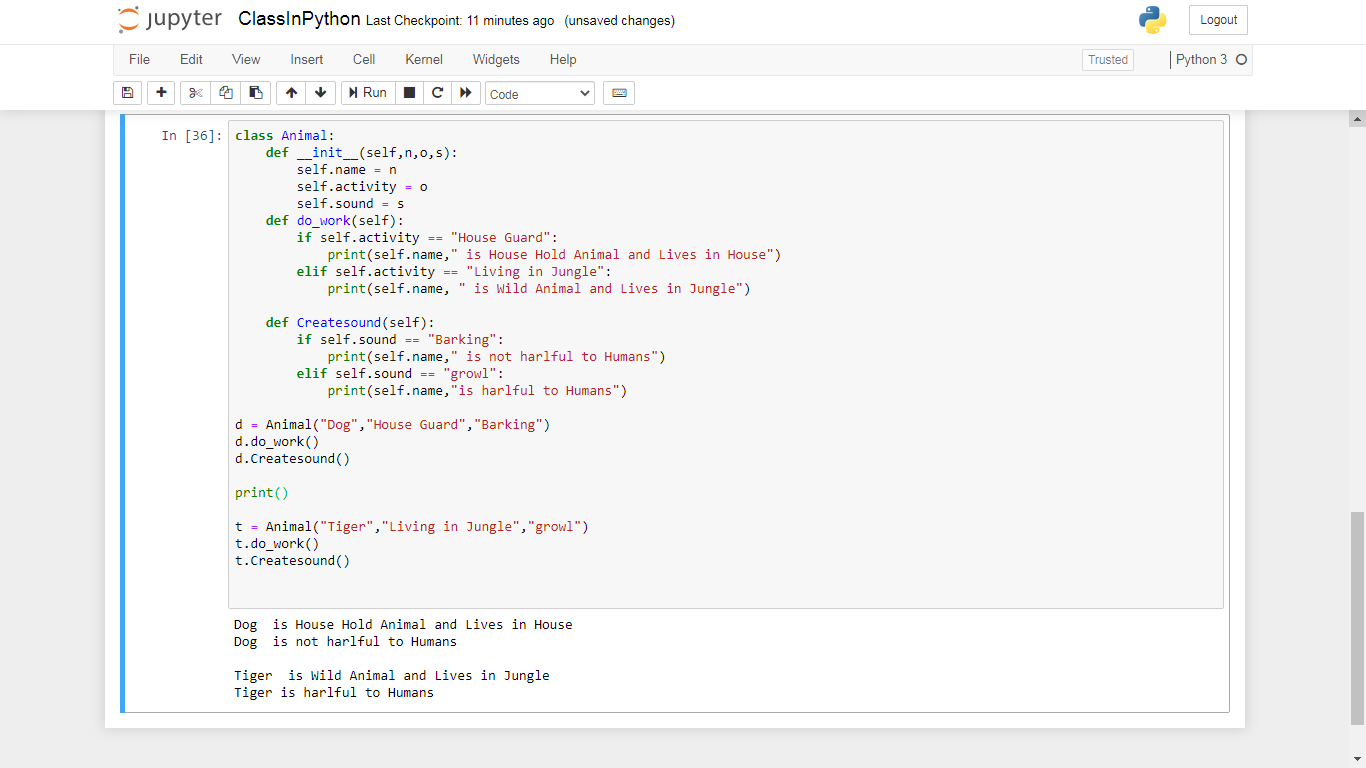
In Python, object-oriented Programming (OOPs) is a programming paradigm(set) that uses **objects and classes** in programming. It aims to implement real-world entities like **inheritance, polymorphisms, encapsulation,** etc. in the programming. The main concept of OOPs is to **bind** the data and the functions that work together as a single unit.

**Classes:** Classes are **blueprints** for creating objects. They define the **attributes (data)** and **methods (functions)** and these methods may contain **Attribute(data)**, **Logical Structure or both**

* **Methods or Function:** Methods are functions that belong to objects. They can access and manipulate object attributes.
  + **Attribute(data)** Attributes are described by data variables for example name, age, height, etc.
  + **Logical Structure:** Logical structure that incorporates conditional statements (like if, else, elif) and loops (such as for loops, while loops) within a class, you can certainly do so. Here's an example illustrating this:

**Object:** Objects are instances of classes. They are created using the class as a template. With the help of an object we can access the **Attribute(data)** or **Methods(Function)** of a Class

**Example of Class:**



**Different features of oops**

1. **Inheritance:**
2. **Encapsulation:**
3. **Abstraction:**
4. **Polymorphism:**

**Encapsulation:**

Binding DATA and Function into a Single Entity and Encapsulation refers to bundling the data (attributes) and methods that operate on the data together within a class. It helps in hiding the internal state of an object and only exposes the necessary functionalities.

**Abstraction:**

Hiding the implemented details and showing the essential details

**Access Specifier**

Access modifier set the limit of accessibility.

**Public Variable**

public = 10

* Can be Accessible by any function in any class.
* In this example, public\_variable is accessible outside the class because it's directly assigned to self within the \_\_init\_\_ method. It behaves like a public variable.

public = 10

abc = 30

myVarible = 40

**Protected Variable**

* Can should be accessed by the same class or Inherited class.
* since \_private\_variable is marked as "private" by convention, it's still accessible outside the class.
* The underscore convention is a signal to users of the class that they should not access it directly, but Python does not enforce this.
* Examle:

\_abc\_variable =10

\_abc = 30

\_myVarible = 40

**Private Variable**

* The members of a class that are declared private are accessible within the class only.
* the private access modifier is the most secure.
* Data members of a class are declared private by adding a double underscore ‘**\_ \_**’ symbol before the data member of that class.

Examle:

\_\_privateVariable = 20

\_\_abc = 30

\_\_myVarible = 40



**Inheritance:**

The derived class(Child Class) inherits the properties of the Base class(Parent Class)

In the Inheritance Derived class(Child Class) can access all the properties (Attributes and Functions) of the Base class(Parent Class).

There are different types of Inheritance

1. **Single Inheritance**
2. **Multiple Inheritance**
3. **Multilevel Inheritance**

|  |
| --- |
| Base class(Parent Class) |
| Attributes |
| Function |

|  |
| --- |
| Derived class(Child Class) |
| Attributes |
| Function |

**Poly morphism:**

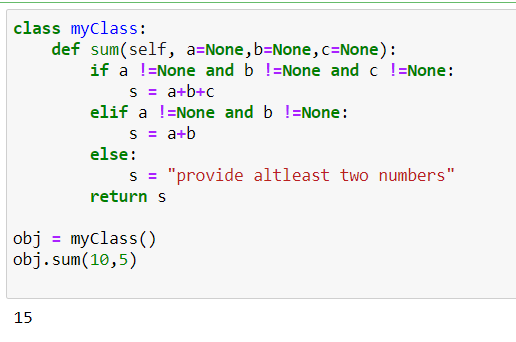
Polymorphism means creating multiple forms, either by passing multiple argument in a method or use of same method but with different datatype, arguments or operation.

There are two polymorphisms.

1. Method overloading,
2. Method overriding,

**Method Overloading**

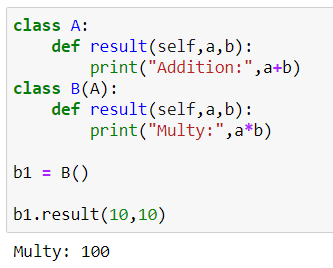
If a method is written in such a way that it can perform more than one task, it is called method overloading. We can achieve method overloading by writing the same method with several parameters.



**Method Overriding:**

In inheritance, if we write **the** **same method** in both classes, I.e. parent class as well as child class then the parent class method is not available in the child class. In this case child class method replaces the parent class Method. Method overriding is used when the programmer wants to modify the existing behaviour of a method.

**Method Overriding:**



**Method Overriding with super keyword:**

However, if a programmer still wants to use the Parent class method then, the same can be achieved by using the “**super” keyword.** As shown in example two.

