Difference between Public, Protected and Private attributes/methods in Python.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **PUBLIC** | **PROTECTED** | **PRIVATE** |
| **Access** | Can be accessed from anywhere, both inside and outside the class | As per convention, can be accessed only within class and subclasses. But the Python doesn’t enforce strict access restrictions | Are meant to be accessed only within the class where they are defined. |
| **Declaration** | An identifier is used to declare | Single underscore is used to declare | Double underscore is used to declare |
| **Example** | class MyClass:  def \_\_init\_\_(self):  self.public\_attribute = "I am public"    def public\_method(self):  return "This is a public method"  obj = MyClass()  print(obj.public\_attribute) # Accessible  print(obj.public\_method()) # Accessible | class MyClass:  def \_\_init\_\_(self):  self.\_protected\_attribute = "I am protected"    def \_protected\_method(self):  return "This is a protected method"  class SubClass(MyClass):  def access\_protected(self):  return self.\_protected\_attribute # Accessible within subclass  obj = SubClass()  print(obj.access\_protected()) # Accessible within subclass | class MyClass:  def \_\_init\_\_(self):  self.\_\_private\_attribute = "I am private"    def \_\_private\_method(self):  return "This is a private method"  def access\_private(self):  return self.\_\_private\_method() # Accessible within the class  obj = MyClass()  print(obj.access\_private()) # Accessible within the class  # print(obj.\_\_private\_attribute) # Raises AttributeError |