1. Normally, if we want to access the attribute of a class, we access it with instance of the class like instance.attribute\_name. These attributes are already present in the class. Where as, to access the attribute of class which are not defined in the class, we use getattr() method.

But if the attribute does exist, getattr won’t be invoked

class Sample:

def \_\_init\_\_(self,address):

self.address = address

def \_\_getattr\_\_(self,name):

return name.lower()

s =Sample('Banglore')

print(s.address) # existing attribute

print(s.ABCD) # ABCD is non defined in class

Banglore

abcd

getattribute will look for every attribute, doesn’t matter if the attribute exists or not.

class Dummy():

def \_\_getattribute\_\_(self, attr):

return 'New Value'

d = Dummy()

d.value = "Old value"

print(d.value) # "YOU SEE ME?"

New Value

2. In Properties, We can bind getter, setter functions with an attribute name, using the built-in property function. In descriptor, We can bind getter, setter (and deleter) functions into a separate class. We then assign an object of this class to the attribute name.

3. To access the attribute of class which are not defined in the class, we use getattr() method. But if the attribute does exist, getattr won’t be invoked. getattribute will look for every attribute, doesn’t matter if the attribute exists or not.

In Properties, We can bind getter, setter functions with an attribute name, using the built-in property function. In descriptor, We can bind getter, setter (and deleter) functions into a separate class. We then assign an object of this class to the attribute name.