**Python Advance Assignment-10**

**1. How do *getattr* and *getattribute* differ?**

Ans.

***getattr***: It return the value of attributes which is constructed inside the class and having an class object which store its value.

If the attribute is not present inside the class and if default value is provided in the syntax only then it will return that value else it will show attribute error

**class** Person:

**def** \_\_init\_\_(self,name):

self**.**name**=**name

**def** \_\_getattr\_\_(self, i):

**return** 29

ob **=** Person('shivansh')

print(f'Name is: {ob**.**name}')

print(f'Age is: {ob**.**age}')

*# 'age' attributes is neither a instance variable nor a class variable but still it is printed because of \_\_getattr\_\_ method*

Name is: shivansh

Age is: 29

***getattribute***: This method will invoked before looking at the actual attributes on the object. Means, if ywe have *getattribute* method in our class, python invokes this method for every attribute regardless whether it exists or not. Lets see the following example.

**class** Person:

**def** \_\_init\_\_(self,name):

self**.**name**=**name

**def** \_\_getattribute\_\_(self, item):

**return** 29

*# def \_\_getattr\_\_(self, item):*

*# return 35*

ob **=** Person('shivansh')

print(ob**.**name)

print(f'Age is: {ob**.**age}')

*# 'age' attributes is neither a instance variable nor a class variable but still it is printed*

*# because of \_\_getattr\_\_ method*

29

Age is: 29

**2. How do properties and descriptors differ?**

Ans

**Properties:** We can bind getter, setter and delete functions with an attribute name, using the built-in property function. When we do this, each reference to an attribute looks like simple, direct access, but invokes the appropriate function of the object.

**Descriptors:** We can bind getter, setter (and deleter) functions into a separate class. We then assign an object of this class to the attribute name. When we do this, each reference to an attribute looks like simple, direct access, but invokes an appropriate function of the Descriptor object.

**3. How are properties and decorators related?**

Ans.

**Descriptors** are a low-level mechanism that lets you hook into an object's attributes being accessed. **Properties** are a high-level application of this; that is, properties are implemented using descriptors.

The property function gives us a handy way to implement a simple descriptor without defining a separate class. Rather than create a complete class definition, we can write getter and setter method functions, and then bind these functions to an attribute name.

**4. What are the main functional differences between getattr and getattribute and properties and descriptors?**

Ans.

***getattr***: It return the value of attributes which is constructed inside the class and having an class object which store its value.

If the attribute is not present inside the class and if default value is provided in the syntax only then it will return that value else it will show attribute error

**getattribute :** This method will invoked before looking at the actual attributes on the object. Means, if ywe have getattribute method in our class, python invokes this method for every attribute regardless whether it exists or not.

**Properties:** We can bind getter, setter and delete functions **with an attribute name, using the built-in property function.** When we do this, each reference to an attribute looks like simple, direct access, but invokes the appropriate function of the object.

**Descriptors:** We can bind getter, setter (and deleter) functions **into a separate class. We then assign an object of this class to the attribute name.** When we do this, each reference to an attribute looks like simple, direct access, but invokes an appropriate function of the Descriptor object.