**Python Advanced Assignment 19**

Q1. Define the relationship between a class and its instances. Is it a one-to-one or a one-to-many

partnership, for example?

Ans-) The relationship between a class and its instances is a one-to-many partnership, where a single class can have multiple instances created from it.

Q2. What kind of data is held only in an instance?

Ans-) Instance-specific data is held only in an instance.

Q3. What kind of knowledge is stored in a class?

Ans-) Class knowledge is stored in the form of class variables and methods.

Q4. What exactly is a method, and how is it different from a regular function?

Ans-) A method is a function that is defined inside a class, and it is called on instances of that class. The first parameter of a method is usually self, which refers to the instance on which the method is called. A regular function, on the other hand, is defined outside of a class and does not have a self parameter.

Q5. Is inheritance supported in Python, and if so, what is the syntax?

Ans-) Yes, inheritance is supported in Python, and it is implemented using the syntax:

class SubClassName(BaseClassName):

Q6. How much encapsulation (making instance or class variables private) does Python support?

Ans-) Python supports a limited form of encapsulation, where instance and class variables can be made private by prefixing their names with double underscores. However, these variables can still be accessed from outside the class using a slightly modified name.

Q7. How do you distinguish between a class variable and an instance variable?

Ans-) Class variables are shared across all instances of a class, while instance variables are unique to each instance.

Q8. When, if ever, can self be included in a class’s method definitions?

Ans-) self is included as the first parameter in a class's method definitions, and it refers to the instance on which the method is called. Therefore, it is necessary to include self in a method definition when you want to access or modify instance-specific data.

Q9. What is the difference between the \_ \_add\_ \_ and the \_ \_radd\_ \_ methods?

Ans-) The add method is called when an instance is the left operand of the "+" operator, while the radd method is called when an instance is the right operand of the "+" operator.

Q10. When is it necessary to use a reflection method? When do you not need it, even though you

support the operation in question?

Ans-) A reflection method is necessary when you want to inspect or modify an object's attributes or methods at runtime. It is not necessary when you already know the name of the attribute or method you want to access, or when you don't need to modify it.

Q11. What is the \_ \_iadd\_ \_ method called?

Ans-) The iadd method is called when an instance is the left operand of the "+=" operator.

Q12. Is the \_ \_init\_ \_ method inherited by subclasses?

Ans-) The init method is not inherited by subclasses, but it can be called explicitly from a subclass using the syntax super().\_\_init\_\_(args).