# Q1. Define the relationship between a class and its instances. Is it a one-to-one or a one-to-many partnership, for example?

```
Class is defined as bluprint of the instance.Instance is one of the entitiy or the image of the class.
It is one to many relationship.One object may have many instances.
class of animal may have many instances like snake,lion etc....
```

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

```
an instnce store the datta related to th atributrs fo class.
An instance can hold the related to it any change itsdata does not effects the data of other classes
```

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

```
The lists of its atriburtes
the list of its methods']
and the references for all of its instances to know weather the instance belongs to the class or not
```

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

```
a method is a function that is defined within a class and is associated with instances of that class.
a function may or may not bound to the class or instance.
method will have self keyword as its first argument bur function do not have it
```

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

```
Yes inheritence is suppirted by python

syntax:

class A:
    pass
class B(A):
    pass
class C(A):
```

```
    pass
```

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

```
Python support encapsulaton upto a level.
because the use of private and protected variables have some ways to access them
```

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

```
class variable is defined upside the __init__() method
andthe instance variables defines inside the __init__() meeethod with self keyword
the value class variables is same for all the instances of that class
the value of instance variables will warry from one instance to other
```

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

```
when we want to bind the method to the instance we need to add self key word

when we want to declare it as a static method no need of self keyword
```

# Q9. What is the difference between the _ *add* _ and the _ *radd* _ methods?

```
 _ _add_ _: This method is called when the addition operation is performed with the object on the left-hand
        side of the operator

 _ _radd_ _:his method is called when the addition operation is performed, but the object on the left-hand
    side does not support the addition operation.
```

# Q10. When is it necessary to use a reflection method? When do you not need it, even though you support the operation in question?

```
Reflection methods, such as __getattr__, __setattr__, __getattribute__, and __setattr__, are used
to customize attribute access and modify the default behavior of attribute lookup and assignment in Python classes.


```

```
When you have static attributes,When you don't require custom behavior we dont
have need to use them.
```

# Q11. What is the _ *iadd* _ method called?

```
The __iadd__ method is called the "in-place addition" method.
```

# Q12. Is the _ *init* _ method inherited by subclasses? What do you do if you need to customize its behavior within a subclass?

```
Yes, the __init__ method is inherited by subclasses in Python.
If you need to customize the behavior of the __init__ method within a subclass,
you can override
the method by defining a new __init__ method in the subclass
```