Q1. What is the relationship between classes and modules?
Answer : Classes can be saved into separate file, that file is called module. This files can be imported in any number of programs you want to use it in.

Q2. How do you make instances and classes?
Answer : Creating instances :
For creating a class instance, we call the class by its class name and pass the arguments it's init method accepts.
E.g. class_instance=class1("abc",123). Here class_isntance is class instance of class1 with attributes "abc" and 123.

Creating classes:
"class" keyword creates a class. class keyword is followed by classname followed by a collon.
E.g. class Person : , this creates a class named "Person".

Q3. Where and how should be class attributes created?
Answer : Class attributes belong to the class itself they will be shared by all the instances. Such attributes are defined in the class body parts usually at the top.

```
class Person:
        kind= 'homo_sapiens'   #this is  class attribute
        def __init__(self, name, surname, year_of_birth):
                self.name = name
                self.surname = surname
                self.year_of_birth = year_of_birth
```

Q4. Where and how are instance attributes created?
Answer : Instance attributes are not shared by objects. Every object has its own copy of the instance attribute (In case of class attributes all object refer to single copy).

```
class Person:
        kind= 'homo_sapiens'   #this is  class attribute
        def __init__(self, name, surname, year_of_birth):
                self.name = name
                self.surname = surname
                self.year_of_birth = year_of_birth

alec = Person("Alec", "Baldwin", 1958)

joy= Person("joy','Hobs", 1962)
```

alec and joy (both are instances of class Person) have there different attributes as ("Alec", "Baldwin", 1958) and ("joy','Hobs", 1962) called instance attributes.

Q5. What does the term "self" in a Python class mean?
Answer : self is first attribute passed in __init__(self,..) methos and it refers to object itself.

Q6. How does a Python class handle operator overloading?
Answer : Python class handle operator overloading by use of special functions. This special function begin with doulble underscore (__).

Q7. When do you consider allowing operator overloading of your classes?
Answer : When we want to have different meaning for the same operator according to the context, we use operator overloading.

Q8. What is the most popular form of operator overloading?
Answer : + operator is the most popular form of operator overloading,

Q9. What are the two most important concepts to grasp in order to comprehend Python OOP code?
Answer : classes and objects are two concepts to comprehend Python OOP as More formally objects are entities that represent instances of a general abstract concept called class