Ans (i) **Polymorphism** :- It defines the ability to take different forms. In python it allows us to define methods in the child class with the same name as defined in their parent class.

It is taken from the Greek words Poly (many) & morphism (forms.) It means that the same function name can be used for different types. This makes programming more intuitive and easier.

- Static Polymorphism or Compile time Polymorphism Polymorphism that is resolved during compiler time is known as static polymorphism. Method overloading is an example of compile time polymorphism.
  Method Overloading : This allows us to have more than one method having the same name, if the parameters of methods are different in number, sequence and data types of parameters.

- Dynamic Polymorphism or Runtime Polymorphism :- It is also known as Dynamic Method Dispatch. It is a process in which a call to an overridden method is resolved at runtime, that's which it is called runtime polymorphism. Same method is overridden with same

signature in different classes.

- Tuple :— In python, a tuple is similar to List except that the object in tuple are immutable which means we cannot change the element of a tuple once assigned. On the other hand, we can change the elements of a list.

To create a tuple in python, place all the elements in a () parenthesis, seperated by commas. A tuple can have heterogenous data items, a tuple can have string and list as data items as well.

Exp. :—

```
# tuple of strings
my_data = ("hi", "hello", "Bye")
Print (my_data)

# tuple of int, float, string
my_data 2 = (1, 2.8, "Hello World")
Print (my_data 2)

# tuple of string and list
my_data 3 = ("Book", [1, 2, 3])
Print (my_data 3)
```

tuple inside another tuple
nested tuple

```
my_data 4 = ((2,3,4),(1,2,"hi"))
Print (my_data 4)
```

Output :—

```
('hi', 'hello', 'bye')
(1, 2.8, 'Hello World')
('Book', [1,2,3])
((2,3,4),(1,2,'hi'))
```