Python Advance Assignment-4

Q1. Which two operator overloading methods can you use in your classes to support iteration?

Ans:- The __iter__ returns the iterator object and is implicitly called at the start of loops. The __next__ method returns the next value and is implicitly called at each loop increment. __next__ raises a StopIteration exception when there are no more value to return, which is implicitly captured by looping constructs to stop iterating.

```
class Counter:
    def __init__(self, low, high):
        self.current = low
        self.high = high

    def __iter__(self):
        return self

    def __next__(self):
        if self.current > self.high:
            raise StopIteration
        else:
            self.current += 1
            return self.current - 1


for num in Counter(5, 15):
    print(num)
```

Q2. In what contexts do the two operator overloading methods manage printing?

Ans:- Ans.

1) ' str '

2) ' repr '

str is used to print the string representation of an object in such a way that it is readable to the user. Only informatic part. We can also say that only inofficial string representation of the object is taken care by using this method. We cannot reuse it to build the object again for the respective class.

repr is used to print the official information about the object. Meaning, the representation of object in a string format in such a way that it is understand by the machine only, however user can also understand but its representation is not that much readable. It can be reuse for constructing the object again of respective class.

Q3. In a class, how do you intercept slice operations?

Ans.

In a class we use a dunder method ' getitem ' for slicing operation.

we use getitem (self,index) method with class object and this returns the value at that index position. It is use for list, tuple as well as dictionary also.

Q4. In a class, how do you capture in-place addition?

Ans.

It is possible that we can add something in a function in additional manner. We just need to pass the value inside the function.

```
class person:
    def __init__(self,name):
        self.name=name

    def detail(self,age):
        return f'name is {self.name} and age is {age}'

object=person(‘priyanka garach’)
object.detail(29)
'name is shivansh jayara and age is 29'
```

Q5. When is it appropriate to use operator overloading?

Ans.

method use : ' add__ '

o1=class(1)

o2=class(2)

o3=class('a')

o4=class('b')

print(o1+o2) : 3

print(o3+o4) : ab

When we want to get the different result based on data type by performing the similar operaton with the use of a same operator operator on different data type values which a class object stored in it then we use this operator overloading methods and with the use of the respective operator in between the objects of a class we can perform operations between the data.