**Day-6 Morning Assessment**

**Magic Methods**

1. The \_\_init\_\_() method is the constructor in Python. It is automatically called when a new object of a class is created, and is used to initialize the object's attributes.  
  
2. \_\_str\_\_() is meant to return a human-readable string representation of the object. It's used when you call str(obj) or print(obj).  
\_\_repr\_\_() is meant to return an unambiguous string, ideally one that could be used to recreate the object. It's used in the interactive interpreter and when you call repr(obj).  
  
3. You can use \_\_add\_\_() to define how the + operator behaves for your objects.  
Example:  
class Point:  
   def \_\_init\_\_(self, x, y):  
       self.x = x  
       self.y = y  
  
   def \_\_add\_\_(self, other):  
       return Point(self.x + other.x, self.y + other.y)  
  
   def \_\_repr\_\_(self):  
       return f"Point({self.x}, {self.y})"  
  
p1 = Point(1, 2)  
p2 = Point(3, 4)  
print(p1 + p2)

o/p: Point(4, 6)  
  
4.\_\_enter\_\_(self) – Called at the beginning of the with block.  
\_\_exit\_\_(self, exc\_type, exc\_value, traceback) – Called when exiting the with block.

**Itertools**

5. itertools.product() returns the Cartesian product of input iterables.

Example:  
import itertools  
result = itertools.product([1, 2], ['a', 'b'])  
print(list(result))  
o/p: [(1, 'a'), (1, 'b'), (2, 'a'), (2, 'b')]

6.In itertools.permutations() nPr the order matters whereas in itertools.combinations() nCr the order doesnot matter.

7. itertools.chain() is used to combine multiple iterables into a single sequence. It treats multiple iterables as one long iterable.  
  
8. from itertools import cycle  
import itertools  
colors = ['red', 'green', 'blue']  
cycler = cycle(colors)  
for i in range(6):  
   print(next(cycler))

o/p:

red

green

blue

red

green

blue

**Map Function**

9. The map() function applies a given function to each item of an iterable and returns a map object, which is an iterator.  
  
10. list1 = [1, 2, 3]  
list2 = [4, 5, 6]  
  
result = map(lambda x, y: x + y, list1, list2)  
print(list(result))  
o/p: [5, 7, 9]

11. The map() function transforms each item and it returns the transformed value and elements will be processed in the output whereas filter() function filters items based on condition and it returns the Boolean and it will get only elements where condition is true.

12. Yes. map() is often used with lambda functions for quick one-line transformations. Example:  
nums = [1, 2, 3, 4]  
squares = map(lambda x: x \* x, nums)  
print(list(squares))  
o/p:  
[1, 4, 9, 16]

**Generators**

13. A generator function in Python is a special kind of function that returns an iterator. Instead of using return, it uses the yield keyword to pause the function and save its state.  
  
14. -return is a regular function and yield is a generator function.

-return ends the function immediately and yield pauses the function, saving its state.

-return returns a single value and yield returns a generator.  
  
15. def even\_numbers():  
   for i in range(2, 11, 2):  
       yield i  
for num in even\_numbers():  
   print(num)  
o/p:  
2  
4  
6  
8  
10  
  
16. If you call next() after a generator is exhausted, it raises a StopIteration exception.

**Iterators**

17. An iterator is an object that implements the \_\_iter\_\_() and \_\_next\_\_() methods. It keeps track of its current position and produces the next value when next() is called, raising StopIteration when there are no more items.  
Iterable:  
An iterable is any Python object capable of returning its elements one at a time. It implements the \_\_iter\_\_()method, which returns an iterator.  
All iterators are iterables, but not all iterables are iterators.  
  
18. \_\_iter\_\_(self) → Should return the iterator object itself.  
\_\_next\_\_(self) → Should return the next value or raise StopIteration.  
  
19. class CountToFive:  
   def \_\_init\_\_(self):  
       self.num = 1  
  
   def \_\_iter\_\_(self):  
       return self  
  
   def \_\_next\_\_(self):  
       if self.num <= 5:  
           current = self.num  
           self.num += 1  
           return current  
       else:  
           raise StopIteration  
  
  
20. The iter() function is used to get an iterator from an iterable like a list.