DAY-6 MORNING ASSESSMENT

# Magic Methods

1.The \_\_init\_\_() method is a special method in Python that gets automatically called when an object is created from a class.

Used to initialize the object’s attributes.

2. class Book:  
   def \_\_init\_\_(self, title, author):  
       self.title = title  
       self.author = author  
  
   def \_\_str\_\_(self):  
       return f"{self.title} by {self.author}"  
  
   def \_\_repr\_\_(self):  
       return f"Book(title='{self.title}', author='{self.author}')"

b = Book("1984", "George Orwell")  
  
print(b)          # Uses \_\_str\_\_() #Output: 1984 by George Orwell

print(repr(b))    # Uses \_\_repr\_\_() #Output:Book(title='1984',author='GeorgeOrwell')

3. 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 \_\_str\_\_(self):  
 return f'({self.x}, {self.y})'  
  
p1=Point(1,2)  
p2=Point(3,4)  
print(p1 + p2)

o/p: (4, 6)

4.\_\_enter\_\_(self) –called when entering the block

\_\_exit\_\_(self,exc\_type,exc\_val ,exc\_tb) – called when exiting the with block(even if an exception occurs)

# Itertools

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

Its like nested for loops – all possible pairs from the given sequences.

6. from itertools import permutations,combinations  
print(list(permutations(['a', 'b', 'c'],2))) #order matters  
print(list(combinations(['a', 'b', 'c'],2))) #order doesn't matters

o/p: [('a', 'b'), ('a', 'c'), ('b', 'a'), ('b', 'c'), ('c', 'a'), ('c', 'b')]

[('a', 'b'), ('a', 'c'), ('b', 'c')]

7.itertools.chain() is used to combine multiple iterables into a single sequence(iterator) ,one after the other.

8. from itertools import cycle  
import itertools  
colors=cycle(['red','green','blue'])  
  
for i in range(5):  
 print(next(colors))

o/p: red

green

blue

red

green

# Map Functions

9.map(function,iterable)

Map applies a given function to each item of an iterable and returns a map object.

10. list1=[1,2,3]  
list2=[4,5,6]  
result=list(map(lambda x,y:x+y,list1,list2))  
print(result)

o/p: [5, 7, 9]

11. map transforms each item and returns a transformed value while filter selects the items that passes a test and returns iterator.

12.yes map works with lambda function

# Generators

13.A generator function is a special kind of function that returns a generator iterator, which can be used to iterate over a sequence of values.defined using yield keywords.

14.yield produces a value and pauses and resumes from where it left off where as return exits the function immediately and ends the function entirely.

15. def even\_numbers():  
 for i in range(2,11,2):  
 yield i  
  
for number in even\_numbers():  
 print(number)

o/p: 2

4

6

8

10

16.It raises a StopIteration exception.

# Iterators

17.An object that keeps state and yields next value is called Iterator where as an object that can return an iterator.

18.\_\_iter\_\_(self)—returns the iterator object.

\_\_next\_\_(self)—returns the next value or raises stopiteration

19. class OneToFive:  
 def \_\_init\_\_(self):  
 self.num=1  
  
 def \_\_iter\_\_(self):  
 return self  
 def \_\_next\_\_(self):  
 if self.num<=5:  
 val = self.num  
 self.num+=1  
 return val  
 else:  
 raise StopIteration  
nums=OneToFive()  
for num in nums:  
 print(num)

o/p:

1

2

3

4

5