**Python Practice Questions on Magic Methods, Itertools, Map, Generators, Iterators**

**Magic Methods (4 Questions)**

1. What is the purpose of \_\_init\_\_() magic method in a Python class?

\_\_init\_\_() is a method that is called when object is created.

It initializes the instance variables.

class Person:

    def \_\_init\_\_(self, name):

        self.name = name

p = Person("Abhi")

print(p.name)

o/p:Abhi

2. How does \_\_str\_\_() differ from \_\_repr\_\_() in Python classes?

\_\_str\_\_() ->it is used for user friendly and normal representation of string

\_\_repr\_\_() -> it is used by developers for debugging ,it should return a valid python expression.

3. Write a simple example of overloading the \_\_add\_\_() magic method.

class Math:

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

    def \_\_add\_\_(self, num):

        return Math(self.x + num.x, self.y + num.y)

    def \_\_str\_\_(self):

        return f"({self.x}, {self.y})"

m1 = Math(2, 2)

m2 = Math(6, 4)

print(m1 + m2)

o/p:(8,6)

4. Which magic methods are required to make an object context manager?

\_\_exit\_\_

\_\_enter\_\_

class File:

    def \_\_enter\_\_(self):

        print("Opening resource")

        return self

    def \_\_exit\_\_(self, exc\_type, exc\_value, traceback):

        print("Closing resource")

with File():

    print("Using resource")

**Itertools (4 Questions)**

5. What is the use of itertools.product()? Give an example.

It gives the product of the input variables i.e multiplication value of input variables.

from itertools import product

for item in product(["a", "b"], [1, 2]):

       print(item)

6. How does itertools.permutations() differ from itertools.combinations()?

permutation()-order considered and does not allow repetation

combination()-order not considered and does not allow repetation

from itertools import permutations,combinations

print(list(itertools.permutations([1, 2], 2)))

print(list(itertools.combinations([1, 2], 2)))

o/p: [(1, 2), (2, 1)]

[(1,2)]

7. Explain the purpose of itertools.chain().

It combines multiple iterables/operations to a single one. Chaining or combining all together.

8. Write a code snippet using itertools.cycle().

From itertools import cycle

count=0

for x in cycle(["a","b"]):

   if count>3:

       break

   print(x,end=" ")

   count+=1

**Map Function (4 Questions)**

9. How does the map() function work in Python? What does it return?

map() ->this function applies it to all items

It returns a map() object.

10. Write a code snippet to add two lists element-wise using map().

a=[1,2,3]

b=[4,5,6]

result=map(lambda x,y:x+y ,a,b)

print(list(result))

o/p:[5, 7, 9]

11. What is the difference between map() and filter() functions?

map()->applies a given function all times in an iterable,it does not filer.

filer()-> filter the element based on condition true or false

12. Can map() work with lambda functions? Give an example.

Yes map() can work with lambda function

nums = [1, 2, 3, 4]

square = list(map(lambda x: x\*\*2, nums))

print(square)

o/p:[1,4,9,16]

**Generators (4 Questions)**

13. What is a generator function in Python? How is it defined?

Generator function is used for iteration one after the other. It uses yield keyword instead of return.

It is lazy

14. How does yield differ from return in a function?

yield->it does not end the function,it pauses and saves the function

return->ends the function,and gives the value.

Ex:

def simple\_generator():

    yield 1

    yield 2

    yield 3#yield is a return type-automatically updates state b/w calls

gen = simple\_generator()

print(next(gen))

print(next(gen))

print(next(gen))

def generator():

    return 1

   return 2

   return 3  #not understable,dont take return 2 and 3

print(generator())#we can add only one return type,when we use return

15. Write a simple generator to yield even numbers up to 10.

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. What happens if you call next() on a generator after it is exhausted?

It gives stop iteration error. example:

def gen():

    yield "Hi"

    yield "Abhi"

g = gen()

print(next(g))

print(next(g))

print(next(g))

o/p:

Hi

Abhi

Traceback (most recent call last):

  File "<main.py>", line 8, in <module>

StopIteration

**Iterators (4 Questions)**

17. What is an iterator in Python? How is it different from an iterable?

iterator()-> it uses generator, it iterates one after the other, looping is done without using for loop. It uses methods \_\_next\_\_, \_\_\_iter\_\_

iterable()-> it is an object that can looped,in (lists,tuples),can be converted to iterator using iter()

18. Which two magic methods must be implemented for a class to be an iterator?

\_\_iter\_\_() and \_\_next\_\_() methods

19. Write a simple iterator class that returns numbers from 1 to 5.

class Counter:

    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

c = Counter()

for i in c:

    print(i)

o/p:

1

2

3

4

5

20. How does the iter() function work on a list?

It returns iterator object.

list=[10,20,30,40,50]

it= iter(list)

print(next(it))

print(next(it))

print(next(it))

print(next(it))

o/p:10

20

30

40