Magic Methods (4 Questions)

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

The __init__() method is called automatically when a new object of the class is created. It initializes instance variables.

```
class Color:
    def __init__(self, name):
        self.name = name
p = Color("Black")
```

2. How does __str__() differ from __repr__() in Python classes?

__str__() is used to return a user-friendly string representation (used by print()).

__repr__() returns an formal string representation.

```
class Welcome:
    def __init__(self, name):
        self.name = name
    def __str__(self):
        return f"Welcome: {self.name}"
    def __repr__(self):
        return f"Welcome('{self.name}')"
```

3. Write a simple example of overloading the __add__() magic method.

```
class Number:
    def __init__(self, value):
        self.value = value
    def __add__(self, other):
        return self.value + other.value
a = Number(500)
b = Number(100)
print(a + b)
```

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

__enter__() and __exit__()

Itertools (4 Questions)

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

It gives you all combinations of items from two or more lists.

```
import itertools
for p in itertools.product([1, 2], ['a', 'b']):
    print(p)
```

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

- permutations() considers order of elements.

- combinations() does not consider order.

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

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

It joins multiple iterables into a single iterable.

```
import itertools

for x in itertools.chain([1, 2], ['a', 'b']):
    print(x)
```

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

```
import itertools

count = 0

for item in itertools.cycle(['A', 'B']):
    print(item)
    count += 1
    if count == 3:
        break
```

Map Function (4 Questions)

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

map() applies a function to each item in an iterable and returns a map object.

```
def square(x):
    return x * x

result = map(square, [1, 2, 3])

print(list(result))
```

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))
```

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

- map() applies a function to transform each item.

- filter() applies a function to select/filter items.

```
print(list(map(lambda x: x*2, [1, 2, 3])))

print(list(filter(lambda x: x%2==0, [1, 2, 3])))
```

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

Yes map() can work with lambda function.

```
nums = [1, 2, 3]

squared = map(lambda x: x ** 2, nums)

print(list(squared))
```

Generators (4 Questions)

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

A generator function yields values one at a time using the yield keyword. It does not return all values at once.

```
def num():
    yield 1
    yield 2
h = num()
print(next(h))
```

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

yield pauses the function and keeps its state, so it can continue later.

return just ends the function once.

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

```
def even_gen():
    for i in range(2, 11, 2):
        yield i
for num in even_gen():
    print(num)
```

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

A StopIteration exception is raised.

Iterators (4 Questions)

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

- Iterator: Object with __next__() and __iter__() methods.

- Iterable: Can be converted to an iterator using iter().

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

__iter__() and __next__()

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

```
class MyIterator:
    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
for i in MyIterator():
    print(i)
```

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

It turns the list into an iterator.

```
lst = [10, 20, 30]
it = iter(lst)
print(next(it))
```