Here are additional Python interview questions across various difficulty levels, covering a broader range of concepts:

**Basic Python Questions**

**Q1:** What is the difference between a list and a tuple in Python?  
**A:**

* **List**: Mutable, can be changed after creation. Defined using [].
* **Tuple**: Immutable, cannot be changed after creation. Defined using ().

**Q2:** What are Python's built-in data types?  
**A:** Common built-in data types include:

* Numeric: int, float, complex
* Sequence: list, tuple, range
* Text: str
* Set types: set, frozenset
* Mapping: dict
* Boolean: bool

**Q3:** What are Python’s logical operators?  
**A:** and, or, and not.

**Intermediate Python Questions**

**Q4:** What are Python decorators? Provide an example.  
**A:** Decorators are functions that modify the behavior of other functions. They are used with the @ symbol.

Example:

def decorator(func):

def wrapper():

print("Before function call")

func()

print("After function call")

return wrapper

@decorator

def my\_function():

print("Inside the function")

my\_function()

Output:

Before function call

Inside the function

After function call

**Q5:** Explain list comprehensions in Python.  
**A:** List comprehensions provide a concise way to create lists.

Example:

squares = [x\*\*2 for x in range(5)]

print(squares) # Output: [0, 1, 4, 9, 16]

**Q6:** What is the difference between shallow copy and deep copy?  
**A:**

* **Shallow Copy**: Copies the reference to objects. Changes in nested objects affect the copy. Use copy.copy().
* **Deep Copy**: Creates independent copies of all objects. Changes in nested objects don't affect the copy. Use copy.deepcopy().

Example:

import copy

original = [[1, 2], [3, 4]]

shallow = copy.copy(original)

deep = copy.deepcopy(original)

**Advanced Python Questions**

**Q7:** What is metaprogramming in Python?  
**A:** Metaprogramming allows modification or generation of code during runtime using metaclasses. Classes themselves are instances of metaclasses.

Example of a metaclass:

class Meta(type):

def \_\_new\_\_(cls, name, bases, dct):

dct['added\_attr'] = 42

return super().\_\_new\_\_(cls, name, bases, dct)

class MyClass(metaclass=Meta):

pass

print(MyClass.added\_attr) # Output: 42

**Q8:** What are Python's garbage collection mechanisms?  
**A:** Python uses automatic garbage collection, which includes:

1. **Reference counting**: An object is deallocated when its reference count reaches zero.
2. **Cycle detection**: Detects and collects circular references.

**Q9:** What is the difference between @staticmethod and @classmethod?  
**A:**

* **@staticmethod**: A method that does not depend on the class or instance. Does not take self or cls as arguments.
* **@classmethod**: A method that operates on the class itself, taking cls as its first argument.

Example:

class MyClass:

@staticmethod

def static\_method():

return "Called static method"

@classmethod

def class\_method(cls):

return f"Called class method of {cls}"

print(MyClass.static\_method()) # Output: Called static method

print(MyClass.class\_method()) # Output: Called class method of <class '\_\_main\_\_.MyClass'>

**Q10:** How are generators different from iterators?  
**A:**

* **Iterator**: Implements \_\_iter\_\_() and \_\_next\_\_() methods to iterate through elements.
* **Generator**: Simplifies the creation of iterators using the yield keyword. Each yield statement pauses the generator's state.

Example of a generator:

def my\_generator():

yield 1

yield 2

yield 3

gen = my\_generator()

print(next(gen)) # Output: 1

print(next(gen)) # Output: 2

**Coding Challenges**

**Q11:** Write a function to check if a string is a palindrome.

def is\_palindrome(s):

s = s.lower().replace(" ", "") # Ignore spaces and case

return s == s[::-1]

print(is\_palindrome("A man a plan a canal Panama")) # Output: True

**Q12:** Write a program to find the second largest element in a list.

def second\_largest(nums):

nums = list(set(nums)) # Remove duplicates

nums.sort(reverse=True)

return nums[1] if len(nums) > 1 else None

print(second\_largest([4, 1, 7, 3, 7, 4])) # Output: 4

**Q13:** Explain and demonstrate context managers in Python.  
**A:** Context managers manage resources like file streams or database connections. They are implemented using the with statement and the \_\_enter\_\_ and \_\_exit\_\_ methods.

Example:

with open('example.txt', 'w') as file:

file.write('Hello, world!')

# File is automatically closed here

Let me know if you'd like more in-depth coverage or help on a specific Python topic!