Assignment: Python Intermediate Topics

(if you know = better) topics, not compulsory as beginners

Note: For theory questions, please do online research, read more than one forum/website for better perspectives

---

1. Decorator

Theory:

- Q1. What is a decorator in Python? Explain its use with a real-world analogy.
- Q2. What is the role of *args and **kwargs in decorator functions?

Practical:

- Q3. Create a decorator called @log_call that prints "Function called" before any function runs.
- Q4. Write a decorator @only_even_args that allows a function to run **only** if all arguments are even numbers.
- Q5. Create a decorator that measures the **execution time** of a function.

---

2. Generators (Basics)

Theory:

- Q6. What is a generator? How is it different from a regular function?
- Q7. What are the advantages of using generators?

Practical:

- Q8. Write a generator function that yields numbers from 1 to 100.
- Q9. Create a generator that yields the **square of even numbers** from a given list.
- Q10. Implement a generator that returns Fibonacci numbers up to n terms.

---

3. Map and Filter

Theory:

- Q11. What does map() do in Python? Give an example.
- Q12. How is filter() different from map()?

Practical:

- Q13. Given a list of numbers, use map() to return a new list with their squares.
- Q14. Using filter(), return only prime numbers from a list [2, 3, 4, 5, 6, 7, 8, 9, 10].
- Q15. Use map() and filter() together to first double the numbers and then filter out the ones greater than 10.

---

4. List and Dictionary Comprehension

Theory:

- Q16. Write the syntax and one-line example of list comprehension.
- Q17. How can dictionary comprehension be used to reverse a dictionary?

Practical:

Q18. Given a list of strings, use list comprehension to convert them to uppercase.

```
names = ['alice', 'bob', 'charlie']
```

-
- Q19. Create a list of tuples like (x, x**2) for even numbers from 1 to 10 using list comprehension.
- Q20. Given a dictionary of students and marks, create a new dictionary with only those students who passed (marks ≥ 40).

Bonus Challenge:

Write a program using:

- a **generator** to generate random numbers,
- a **decorator** to log when the generator is used,
- and **filter/map/list comprehension** to get only the odd numbers squared from the result.