DSA mock test 4

17.Write a function that takes a list of numbers as input and returns a new list containing only the even numbers from the input list. Use list comprehension to solve this problem.

Example:

Input: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Output: [2, 4, 6, 8, 10]

10 points

In [1]:

```
def get_even_numbers(numbers):
    """Returns a new list containing only the even numbers from the input list."""
    return [number for number in numbers if number % 2 == 0]


def main():
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    even_numbers = get_even_numbers(numbers)
    print(even_numbers)


if __name__ == "__main__":
    main()
```

```
[2, 4, 6, 8, 10]
```

18. Implement a decorator function called ‘timer’ that measures the execution time of a function. The ‘timer’ decorator should print the time taken by the decorated function to execute. Use the ‘time’ module in Python to calculate the execution time.

Example:

import time

@timer def my_function(): # Function code goes here time.sleep(2)

my_function()

Output: "Execution time: 2.00123 seconds" 10 points

In [2]:

```python
import time

def timer(func):
    """Decorator function that measures the execution time of a function."""

    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        execution_time = end_time - start_time
        print(f"Execution time: {execution_time:.3f} seconds")

        return result

    return wrapper


@timer
def my_function():
    """Function that sleeps for 2 seconds."""
    time.sleep(2)


if __name__ == "__main__":
    my_function()

```

```
Execution time: 2.010 seconds
```

19. Write a function called ‘calculate_mean’ that takes a list of numbers as input and returns the mean (average) of the numbers. The function should calculate the mean using the sum of the numbers divided by the total count.

Example:

def calculate_mean(numbers): total = sum(numbers) count = len(numbers) mean = total / count return mean

data = [10, 15, 20, 25, 30] mean_value = calculate_mean(data) print("Mean:", mean_value)

Output: Mean: 20.0

In [3]:

```
def calculate_mean(numbers):
    """Calculates the mean of a list of numbers."""
    total = sum(numbers)
    count = len(numbers)
    mean = total / count
    return mean


data = [10, 15, 20, 25, 30]
mean_value = calculate_mean(data)
print("Mean:", mean_value)
```

```
Mean: 20.0
```

20. Write a function called 'perform_hypothesis_test' that takes two lists of numbers as input, representing two samples. The function should perform a two-sample t-test and return the p-value. Use the 'scipy.stats' module in Python to calculate the t-test and p-value.

Example:

from scipy import stats

def perform_hypothesis_test(sample1, sample2): t_statistic, p_value = stats.ttest_ind(sample1, sample2) return p_value

sample1 = [5, 10, 15, 20, 25] sample2 = [10, 20, 30, 40, 50] p_value = perform_hypothesis_test(sample1, sample2) print("P-value:", p_value)

Output: P-value: 0.1064706396450037

In [4]:

```
import scipy.stats as stats

def perform_hypothesis_test(sample1, sample2):
    """Performs a two-sample t-test and returns the p-value."""
    t_statistic, p_value = stats.ttest_ind(sample1, sample2)
    return p_value


sample1 = [5, 10, 15, 20, 25]
sample2 = [10, 20, 30, 40, 50]
p_value = perform_hypothesis_test(sample1, sample2)
print("P-value:", p_value)
```

```
P-value: 0.09434977284243756
```

In [ ]:

```

```