Question 1: What is the difference between multithreading and multiprocessing?

Answer:

- **Multithreading** involves running multiple threads (smaller units of a process) concurrently within a single process. Threads share the same memory space, making communication between them easier but also increasing the risk of race conditions.

- **Multiprocessing**, on the other hand, involves running multiple processes concurrently. Each process has its own memory space, making it safer from conflicts but requiring more system resources and inter-process communication mechanisms.

Feature	Multithreading	Multiprocessing
Memory	Shared	Separate
Speed	Faster for I/O-bound tasks	Faster for CPU-bound tasks
Overhead	Low	High
Example Use	Web scraping, I/O tasks	Machine learning, heavy computation

---

Question 2: What are the challenges associated with memory management in Python?

Answer:

1. **Garbage Collection Overhead:** Python automatically manages memory using garbage collection, which can add overhead.

2. **Reference Cycles:** Objects referencing each other may not be freed automatically.

3. **Memory Fragmentation:** Frequent allocations/deallocations can fragment memory.

4. **Global Interpreter Lock (GIL):** Limits true parallelism in multi-threaded programs.

5. **High Memory Consumption:** Objects like lists or dictionaries consume more memory than lower-level data structures.

Question 3: Write a Python program that logs an error message to a log file when a division by zero exception occurs.

Answer:

```
import logging

logging.basicConfig(filename='error.log', level=logging.ERROR,
                    format='%(asctime)s:%(levelname)s:%(message)s')

try:
    num = 10 / 0
except ZeroDivisionError as e:
    logging.error("Division by zero error occurred: %s", e)
    print("Error logged to file.")
```

Question 4: Write a Python program that reads from one file and writes its content to another file.

Answer:

```
try:
    with open('source.txt', 'r') as src:
        data = src.read()

    with open('destination.txt', 'w') as dest:
        dest.write(data)

    print("File copied successfully.")
except FileNotFoundError:
    print("Source file not found.")
```

Question 5: Write a program that handles both IndexError and KeyError using a try-except block.

Answer:

```
try:
    my_list = [1, 2, 3]
    print(my_list[5])  # IndexError
```

```
    my_dict = {'a': 10, 'b': 20}
    print(my_dict['c'])  # KeyError

except IndexError:
    print("Index out of range!")
except KeyError:
    print("Key not found!")
```

Question 6: What are the differences between NumPy arrays and Python lists?

Answer:

Feature	NumPy Array	Python List
Data Type	Homogeneous	Heterogeneous
Performance	Faster (C-based)	Slower
Memory	Compact	Higher memory usage
Operations	Supports vectorized operations	Element-wise loops required
Example	`np.array([1,2,3])`	`[1,2,3]`

Question 7: Explain the difference between apply() and map() in Pandas.

Answer:

- `map()` is used with **Series** and applies a function element-wise.
- `apply()` can be used with both **Series and DataFrames** and can apply a function along rows or columns.

Example:

```
import pandas as pd

s = pd.Series([1, 2, 3])
print(s.map(lambda x: x * 2))  # Works on Series
```

```
df = pd.DataFrame({'A': [1, 2], 'B': [3, 4]})
print(df.apply(sum))  # Works along columns
```

Question 8: Create a histogram using Seaborn to visualize a distribution.

Answer:

```
import seaborn as sns
import matplotlib.pyplot as plt

data = sns.load_dataset("iris")
sns.histplot(data['sepal_length'], kde=True)
plt.title("Distribution of Sepal Length")
plt.show()
```

Question 9: Use Pandas to load a CSV file and display its first 5 rows.

Answer:

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.head())
```

Question 10: Calculate the correlation matrix using Seaborn and visualize it with a heatmap.

Answer:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')
corr = df.corr()

sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
```

```
plt.show()
```