.

Assignment Answers

Question 1:

What is the difference between multithreading and multiprocessing? Answer:

- **Multithreading** uses multiple threads within the same process to perform tasks concurrently. All threads share the same memory space, which makes communication between them easier but can lead to race conditions.
- **Multiprocessing** uses multiple processes, each with its own memory space. It bypasses the Global Interpreter Lock (GIL) and allows true parallel execution, which is ideal for CPU-bound tasks.

Question 2:

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

- **Garbage Collection Overhead:** Automatic garbage collection can sometimes reduce performance.
- **Memory Leaks:** Caused by circular references or global variables.
- **Fragmentation:** Frequent allocation/deallocation can fragment memory.
- **Large Object Handling:** Managing large datasets can consume high memory.

Question 3:

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

```
import logging

logging.basicConfig(filename='error.log', level=logging.ERROR)

try:
    a = 10
    b = 0
    result = a / b
except ZeroDivisionError as e:
    logging.error("Division by zero error occurred: %s", e)
```

Question 4:

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

```
# Reading from one file and writing to another
with open('source.txt', 'r') as src:
    content = src.read()

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

Question 5:

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

```
data = [1, 2, 3]
dict_data = {'a': 10, 'b': 20}

try:
    print(data[5])         # May raise IndexError
    print(dict_data['c'])  # May raise 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?

Feature	NumPy Array	Python List
Data Type	Homogeneous	Heterogeneous
Performance	Faster (C-based)	Slower
Memory Efficiency	More efficient	Less efficient
Operations	Supports element-wise operations	Requires manual looping
Functionality	Vectorization & broadcasting	No built-in math operations

---

Question 7:

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

- **map():** Used for element-wise transformations on a Series. It applies a function, dictionary, or mapping to each element.
- **apply():** Used on both DataFrames and Series. It applies a function along an axis (rows or columns) and can handle complex logic.

```
import pandas as pd
s = pd.Series([1, 2, 3])
print(s.map(lambda x: x*2))     # Element-wise
print(s.apply(lambda x: x**2))  # Function applied to each element
```

Question 8:

Create a histogram using Seaborn to visualize a distribution.

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

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

Question 9:

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

```
 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.

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

df = sns.load_dataset('iris')
corr = df.corr(numeric_only=True)

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