Practical Python – Error Handling, Logging, and Data Manipulation | Assignment Answers

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

Multithreading allows multiple threads (smaller units of a process) to run concurrently within the same process, sharing memory and resources. It's ideal for I/O-bound tasks. Multiprocessing creates multiple processes, each with its own memory space, suitable for CPU-bound tasks.

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

1. Garbage collection overhead. 2. Reference cycles. 3. Fragmentation. 4. Large object handling. 5. Unreleased external resources.

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,
                    format='%(asctime)s - %(levelname)s - %(message)s')

try:
    x = 10 / 0
except ZeroDivisionError as e:
    logging.error("Division by zero error occurred: %s", e)
    print("An error has been logged.")
```

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

```
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.

```
data_list = [1, 2, 3]
```

```
data_dict = {"a": 10, "b": 20}

try:
    print(data_list[5])
    print(data_dict["c"])
except IndexError:
    print("IndexError: Invalid list index.")
except KeyError:
    print("KeyError: Invalid dictionary key.")
```

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

NumPy arrays are homogeneous, memory-efficient, and support vectorized operations, while Python lists are heterogeneous and slower for numerical computations.

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

```
import pandas as pd

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

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
print(df.apply(sum, axis=0))
```

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")["sepal_length"]
sns.histplot(data, kde=True, color="skyblue")
plt.title("Histogram of Sepal Length")
plt.xlabel("Sepal Length (cm)")
plt.ylabel("Frequency")
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 seaborn as sns
import pandas as pd
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 Matrix Heatmap")
plt.show()
```