Multithreading and Multiprocessing in Python

1. Multithreading for I/O-bound Tasks

Web scraping often involves making numerous network requests to fetch web pages. These tasks are I/O-bound because they spend a lot of time waiting for responses from servers. Multithreading can significantly improve the performance by allowing multiple web pages to be fetched concurrently.

Code Snippet:

```
import threading  # Importing threading module for thread management
import requests  # Importing requests library for making HTTP requests
from bs4 import BeautifulSoup  # Importing BeautifulSoup for HTML parsing

urls = [  # List of URLs to scrape
    'https://python.langchain.com/v0.2/docs/introduction/',
    'https://python.langchain.com/v0.2/docs/concepts/',
    'https://python.langchain.com/v0.2/docs/tutorials/'
]

def fetch_content(url):  # Function to fetch content from a URL
    response = requests.get(url)  # Making an HTTP GET request
    soup = BeautifulSoup(response.content, 'html.parser')  # Parsing the HTML content
    print(f"Fetched {len(soup.text)} characters from {url}")  # Printing the number of ch

threads = []  # List to hold thread objects

for url in urls:  # Iterating over each URL
    thread = threading.Thread(target=fetch_content, args=(url,))  # Creating a new thread
    threads.append(thread)  # Adding thread to the list
    thread.start()  # Starting the thread

for thread in threads:  # Joining all threads
    thread.join()  # Waiting for each thread to finish

print('All Web Pages Fetched')  # Indicating that all web pages have been fetched
```

2. Multiprocessing for CPU-bound Tasks

Factorial calculations, especially for large numbers, involve significant computational work. Multiprocessing can be used to distribute the workload across multiple CPU cores, improving performance.

Code Snippet:

```
import multiprocessing  # Importing multiprocessing module for process management
import math  # Importing math module for mathematical functions
import sys  # Importing sys module for system-specific parameters
import time  # Importing time module for time-related functions
```

```
sys.set_int_max_str_digits(100000)  # Setting maximum digits for integer conversion to av

def compute_factorial(number):  # Function to compute factorial of a number
    print(f"Computing factorial of {number}")  # Indicating which factorial is being comp
    result = math.factorial(number)  # Calculating factorial
    print(f"Factorial of {number} is {result}")  # Printing the result
    return result  # Returning the computed factorial

if __name__ == "__main__":  # Checking if the script is run directly
    numbers = [5000, 6000, 7000, 8000]  # List of numbers for which factorial will be com

    start_time = time.time()  # Recording start time

    with multiprocessing.Pool() as pool:  # Creating a pool of worker processes
        results = pool.map(compute_factorial, numbers)  # Distributing the computation ac

    end_time = time.time()  # Recording end time

    print(f"Results: {results}")  # Printing the results of the computations
    print(f"Time taken: {end_time - start_time} seconds")  # Printing the time taken for
```