```
In [1]: # 1. Discuss the scenarios where multithreading is preferable to multiprocessing and scenarios where multiprocessing is a better choice.
```

# ANS Multithreading is quick to create and requires few resources. whereas multiprocessing requires a significant amount of time and specific resources to create, multiprocessing creates or executes many processes simantaniously, whereas multithreading executes executes many threads simantaniously. on a multiprocessor system, multiple threads can concurrently run on multiple CPUs. therefore,multithreading programs can run much faster than a multiprocessing system. they can also be faster than a program using multiple processing, because thraeds require few resources and generate less overhead.

```
In [2]: # 2. Describe what a process pool is and how it helps in managing multiple processes efficiently.
```

it is a tool which helps to improve a bussiness purpose or process. it helps in business to improve efficiency and productivity. reduce errors and rework, it can also improve compliance with audit assoicated costs and waste. Automation: process tolls can automotate routine tasks,which can reduce wasted time and resouces. Quality: process tools helps to improve products services compliance: it can help with audit trails to increase compliance and solve them. some ex of process tools: workflow management process mining and analytics business process management

```
In [ ]: # 3. Explain what multiprocessing is and why it is used in Python programs.
```

multiprocessing is the visualization of teo or more central processing unitss(cpu) in a single computer system. it generally refers to the sysytem's ability to support multiple cpu's and its capacity to distribute work among them. some essential parts of multiprocessing in python: process: the process class is used to create and mange independent processes. each process runs in its own memory space Queue: the queue c;ass is a shared job queue that allows process safe data exchange and coordinate between processes. it used for passing messages or results between processe instance. Pipe: pipes provides a way to istablish a communication chanel between processes.they are useful for bidirectional communication between two processes. lock: it is usde to insure that only one process is executng a certain section of code at a time. this prevents data corruption by sync acces to shared rsources.

4. Write a Python program using multithreading where one thread adds numbers to a list, and another

thread removes numbers from the list. Implement a mechanism to avoid race conditions using threading.Lock

```
In [ ]:
```

```
In [ ]: # 5. Describe the methods and tools available in Python for safely sharing data between threads and processes.
```

Sharing data between threads in Python with strategies due to the Global Interpreter Lock (GIL). various mechanisms ensure safe communication: Queue Module: Employ the queue module for thread-safe data sharing via queues. Thread-Safe Data Structures: Leverage collections module for thread-safe data structures like deque. Locks: Implement threading. Lock to synchronize access, preventing concurrent modifications. Event Objects: Use threading. Event for signaling between threads. Condition Objects: Employ threading. Condition to coordinate threads based on shared conditions. Thread-Safe Classes: Design custom classes with built-in thread safety using locks.

```
In [ ]:  # 6. Discuss why it’s crucial to handle exceptions in concurrent programs and the techniques available for doing so.
```

handling it in programes because it helps to prevent the programs from crashing and improves the progrmas reliability and experience. Avoids program crashing: Exception handling helps prevent a program from crashing due to errors like invalid user input, code errors, or device failure. Provides feedback: Exception handling allows developers to provide useful feedback to users about errors. Distinguishes normal and erroneous return values:Exception handling helps distinguish normal return values from erroneous ones. Improves reliability:Exception handling helps improve the reliability and user experience of a program.

```
In [ ]: #7. Create a program that uses a thread pool to calculate the factorial of numbers from 1 to 10 concurrently. Use concurrent.futures.ThreadPoolExecutor to manage the threads
```

```
In [8]: import threading

        def factorial_of_a_number(n):
            result = 1
            for i in range(1, n+1):
                result *= i
            return result

        def calculate_factorial(n):
            print(f"\calculating factorial of {n} in thread {thraeding.current_thread().name}")
            result = factorial_of_a_number(n)
            print(f"factorial of {n} is {result} i;n thread {threading.current_thread().name}")

        n = 12

        thread1 = threading.Thread(target=calculate_factorial, args=(n, ))
        thread2 = threading.Thread(target=calculate_factorial, args=(n, ))

        thread1.start()
        thread2.start()

        thread1.join()
        thread2.join()
```

```
<>:10: SyntaxWarning: invalid escape sequence '\c'
<>:10: SyntaxWarning: invalid escape sequence '\c'
C:\Users\24c ComputerS\AppData\Local\Temp\ipykernel_9780\859352298.py:10: SyntaxWarning: invalid escape sequence '\c'
  print(f"\calculating factorial of {n} in thread {thraeding.current_thread().name}")
Exception in thread Thread-9 (calculate_factorial):
Traceback (most recent call last):
  File "C:\Users\24c ComputerS\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1075, in _bootstrap_inner
Exception in thread Thread-10 (calculate_factorial):
Traceback (most recent call last):
  File "C:\Users\24c ComputerS\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1075, in _bootstrap_inner
    self.run()
  File "C:\Users\24c ComputerS\AppData\Roaming\Python\Python312\site-packages\ipykernel\ipkernel.py", line 766, in run_closure
    self.run()
  File "C:\Users\24c ComputerS\AppData\Roaming\Python\Python312\site-packages\ipykernel\ipkernel.py", line 766, in run_closure
    _threading_Thread_run(self)
  File "C:\Users\24c ComputerS\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1012, in run
    _threading_Thread_run(self)
  File "C:\Users\24c ComputerS\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1012, in run
    self._target(*self._args, **self._kwargs)
  File "C:\Users\24c ComputerS\AppData\Local\Temp\ipykernel_9780\859352298.py", line 10, in calculate_factorial
    self._target(*self._args, **self._kwargs)
  File "C:\Users\24c ComputerS\AppData\Local\Temp\ipykernel_9780\859352298.py", line 10, in calculate_factorial
NameError: name 'thraeding' is not defined. Did you mean: 'threading'?
NameError: name 'thraeding' is not defined. Did you mean: 'threading'?
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

In [ ]: