**PW SKILLS**

**MULTIPROCESSING**

Q1. \*\*What is multiprocessing in Python? Why is it useful?\*\*

- Multiprocessing in Python refers to the ability of a program to create and manage multiple processes to execute tasks concurrently.

- It is useful for parallelizing tasks that can be run independently, thereby utilizing multiple CPU cores and improving performance.

- Multiprocessing is particularly effective for CPU-bound tasks, as it allows them to be executed concurrently, taking advantage of the full processing power of the system.

Q2. \*\*Differences between multiprocessing and multithreading\*\*:

- \*\*Multiprocessing\*\*:

- In multiprocessing, each process has its own memory space.

- Processes are isolated from each other and communicate via inter-process communication mechanisms like pipes, queues, and shared memory.

- Multiprocessing is well-suited for CPU-bound tasks and can take advantage of multiple CPU cores.

- \*\*Multithreading\*\*:

- In multithreading, multiple threads share the same memory space within a single process.

- Threads within the same process share resources such as memory and file descriptors.

- Multithreading is suitable for I/O-bound tasks, as threads can overlap I/O operations and maintain responsiveness.

Q3. \*\*Python code to create a process using the multiprocessing module\*\*:

```python

import multiprocessing

def worker():

print("Worker process")

if \_\_name\_\_ == "\_\_main\_\_":

process = multiprocessing.Process(target=worker)

process.start()

process.join()

```

Q4. \*\*What is a multiprocessing pool in Python? Why is it used?\*\*

- A multiprocessing pool is a feature provided by the multiprocessing module for managing a pool of worker processes.

- It is used to parallelize the execution of a function across multiple input values, distributing the workload among multiple processes in the pool.

- The pool manages the creation, execution, and termination of worker processes, making it easier to perform parallel processing tasks.

Q5. \*\*Creating a pool of worker processes in Python using the multiprocessing module\*\*:

```python

import multiprocessing

def worker(x):

return x \*\* 2

if \_\_name\_\_ == "\_\_main\_\_":

with multiprocessing.Pool(processes=4) as pool:

results = pool.map(worker, range(10))

print(results)

```

Q6. \*\*Python program to create 4 processes\*\*:

```python

import multiprocessing

def worker(process\_num):

print(f"Process {process\_num}")

if \_\_name\_\_ == "\_\_main\_\_":

processes = []

for i in range(4):

p = multiprocessing.Process(target=worker, args=(i,))

processes.append(p)

p.start()

for p in processes:

p.join()

```