Python Test

Question 1

Concurrency can be achieved in Python via:

1. Multithreading

    - Threads run on the same memory space.
    - Strengths:
        - Easier to communicate between threads.
    - Weaknesses:
        - Python's Global Interpreter Lock (GIL) allows only one thread to run at a time. Multiple threads are run in a time multiplexed manner.
        - Cannot take advantage of multiple CPU cores.
        - Thread scheduling is non deterministic. Race conditions and non-atomic accces to shared memory may cause data corruption.
        - If a thread leaks memory, it can affect other threads.

2. Multiprocessing

    - Processes run on different memory space.
    - Strengths:
        - Can take advantage by simulatenously running multiple processes on multiple CPU cores.
        - Bypasses GIL limitation.
    - Weaknesses:
        - Spawning processes generally require more memory and startup time.
        - Inter Process Communication (IPC) to share information is harder and requires more overhead.

Speedup gained from multithreading, compared to a single thread, in I/O-bound tasks is primarily due to the processor switching to another thread while a thread is waiting on an I/O operation. Multiprocessing module may be used for primarily CPU-bound tasks, to gain speedups with multiple CPU cores.

Question 2

The code prints the following:

```
[23]
[16, 1, 2]
[[10], 3, 4]
[[11], 23]
```

Function `fn(v, lst=[])` has input arguments `v` and `lst`. The `lst` argument has a default value of empty list `[]`. Defaut arguments in Python are evaluated when the function is first defined/executed at runtime. To simplify explanation, `print(hex(id(lst)))` was included in the code which prints the address of the `lst` variable. See `question2.py` file for the modified code. The code now prints the following:

```
[23]
0x297c148
[16, 1, 2]
0x2973b88
[[10], 3, 4]
0x2973b48
[[11], 23]
0x297c148
```

- Function call `fn(23)` inserts integer `23` into a default empty list to produce `[23]`. The address of default empty list is `0x297c148`.
- Function call `fn(16, [1,2])` inserts integer `16` into index 0 and right-shifts remaining list items to produce `[16, 1, 2]`
- Function call `fn([10], [3,4])` inserts list `[10]` into index 0 and right shifts remaining list items to produce `[[10], 3, 4]`
- Function call `fn([11])` inserts list `[11]` into the previously defined list at address `0x297c148` (from first function call above). Hence `[11]` appears at index 0 while the remaining list elements is shifted to the right producing `[[11], 23]`

Question 3

Please refer to `question3.py` file for the code.