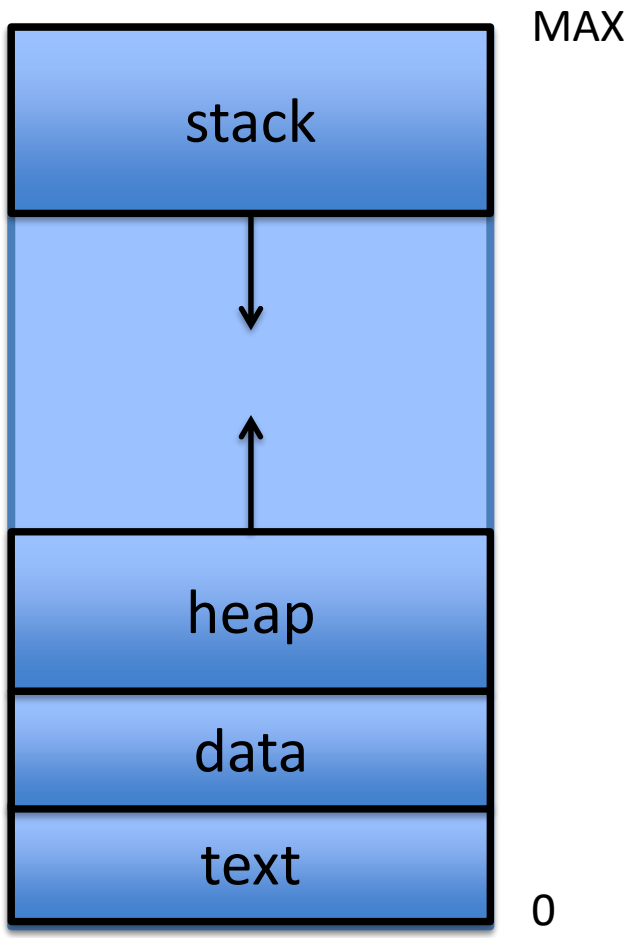
Processes and Threads

Process

- In simple terms, a process is “a program in execution”
  - MS word is a process and MS Excel is another process
- Each process is allocated some memory
- The process memory is divided into four sections

Process Memory

- Text
  - Compiled program codes
- Data
  - Global and static variables
- Heap
  - Dynamic memory for new objects created
- Stack
  - Local variables



Multitasking

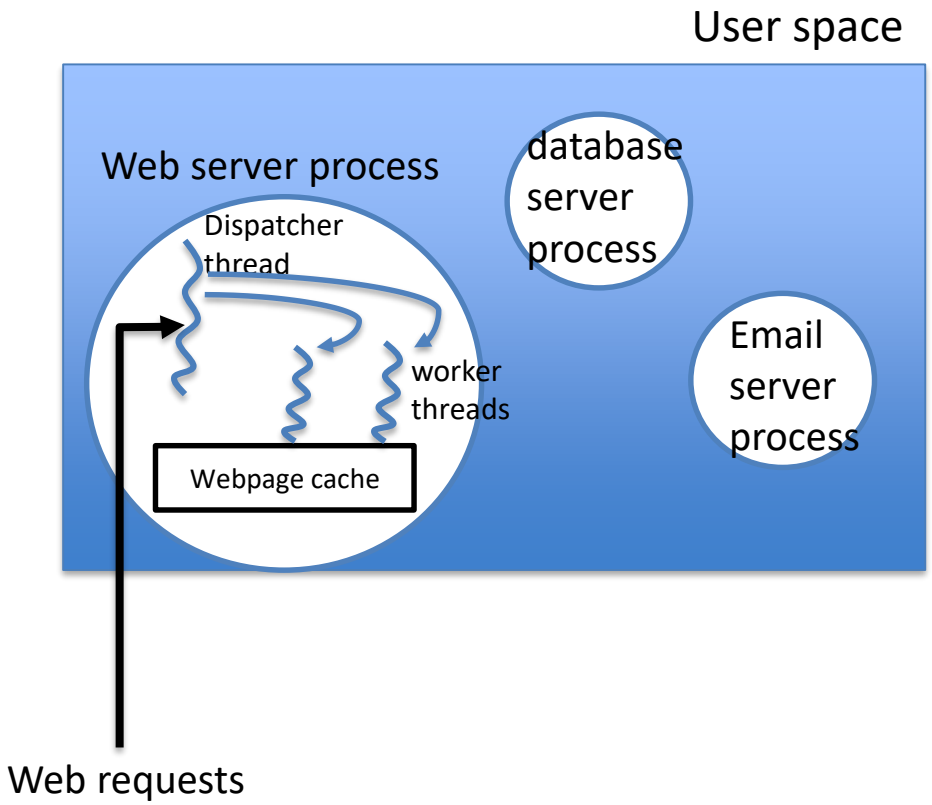
- Multitasking means multiple processes are active (not finished) at the same time
- When the CPU has multiple cores, different processes can be handled by different cores concurrently
- It is also possible to use multiple cores to execute instructions in the same program concurrently

Thread

- Thread is the basic unit that can be scheduled to a certain CPU core
- Thread is also called light-weight process
- A single process can consist of several threads
- By using threads appropriately, a program can be executed by several cores at the same time
  - A webserver can serve several users at the same time

Thread

- The threads created by the same process share the same data (global variables), heap, and text, but not stack (local variables)
  - Effect of modifying global variables can be seen by all threads in the same process



Python Threads (1)

- import threading library

```
import threading
```

- define your own thread class

```
class <new class name>(threading.Thread)
```

- define `__init__` method of your class as needed
  - usually define some thread ID information

```
def __init__(self, threadID, name):
    threading.Thread.__init__(self)
    self.threadID = threadID
    self.name = name
```

- define what this threads does in the `run(self)` method

Python Threads (2)

- To run a thread
  - first, create the thread object
  - then, call the `start()` method
- The thread objects will be executed simultaneously
- Execute `thread_demo.py` several times to observe the output
- Global variables are shared among threads. Changes made by one thread can be seen by others
  - execute `thread_shared_demo.py` to observe this property