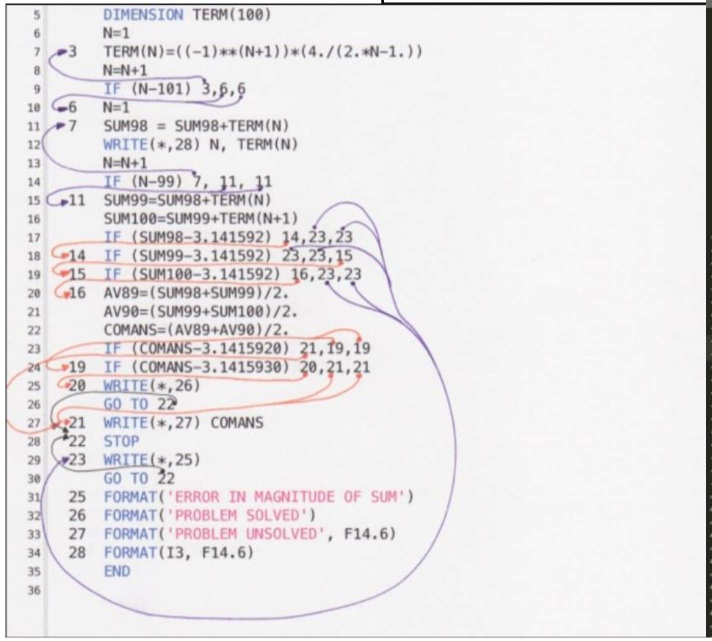
CRM	ERP
Customer Relationship Management (CRM) is a strategy and tool that helps companies improve their relationships with customers and increase sales and profits. CRM can be defined as follows: CRM is an approach that helps companies improve relationships with current customers and acquire new customers more quickly. It uses software to collect, organize, and analyze customer data and interactions, providing valuable insights for developing customer interaction strategies and creating better relationships.	It is a system that helps companies manage their day-to-day business activities, such as accounting, procurement, inventory, production, sales, and others. Enterprise resource planning (ERP) can be defined as follows: Enterprise resource planning (ERP) is an integrated software system that connects all aspects of business processes in a single database, specific applications, and user interface. Collects, organizes and analyzes customer and supplier data, products and services, and provides valuable insights to improve business efficiency and effectiveness. It also helps in simplifying and automating many office functions related to technology, services, and human resources.

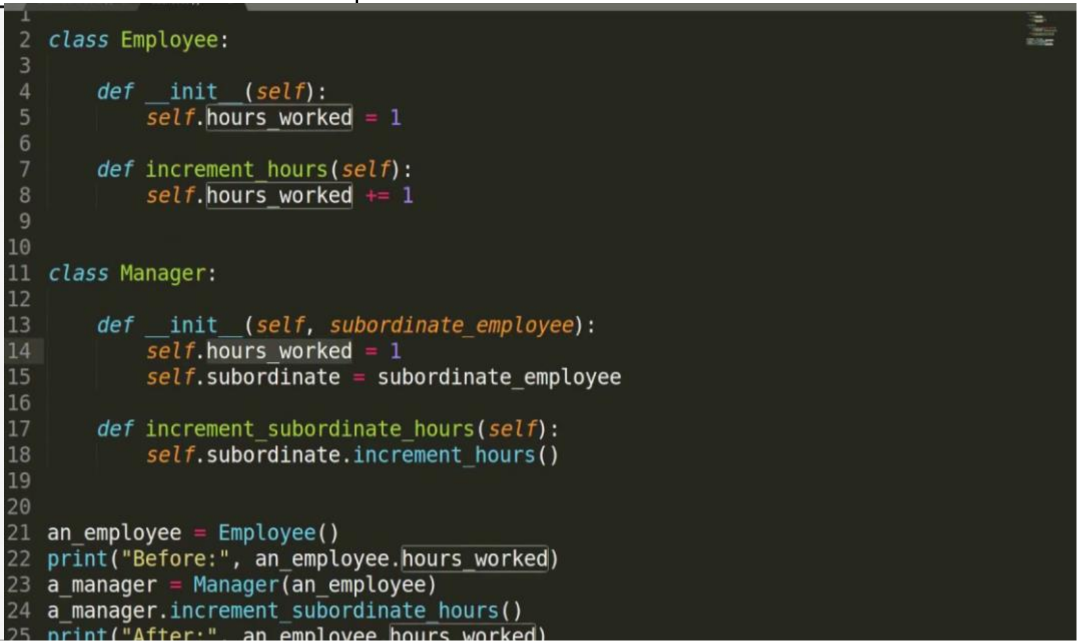
cython

```
def print_hello_world():
 cdef int i
for i in range(10):
print("Hello World")
```

Spaghetti code vs clean code



```
      DIMENSION TERM(100)
      N=1
 3    TERM(N)=((-1)**(N+1))*(4./(2.*N-1.))
      N=N+1
      IF (N-101) 3,6,6
 6    N=1
 7    SUM98 = SUM98+TERM(N)
      WRITE(*,28) N, TERM(N)
      N=N+1
      IF (N-99) 7, 11, 11
 11   SUM99=SUM98+TERM(N)
      SUM100=SUM99+TERM(N+1)
      IF (SUM98-3.141592) 14,23,23
 14   IF (SUM99-3.141592) 23,23,15
 15   IF (SUM100-3.141592) 16,23,23
 16   AV89=(SUM98+SUM99)/2.
      AV90=(SUM99+SUM100)/2.
      COMANS=(AV89+AV90)/2.
      IF (COMANS-3.1415920) 21,19,19
 19   IF (COMANS-3.1415930) 20,21,21
 20   WRITE(*,26)
      GO TO 22
 21   WRITE(*,27) COMANS
 22   STOP
 23   WRITE(*,25)
      GO TO 22
 25   FORMAT('ERROR IN MAGNITUDE OF SUM')
 26   FORMAT('PROBLEM SOLVED')
 27   FORMAT('PROBLEM UNSOLVED', F14.6)
 28   FORMAT(I3, F14.6)
      END
```



```

class Employee:

    def __init__(self):
        self.hours_worked = 1

    def increment_hours(self):
        self.hours_worked += 1


class Manager:

    def __init__(self, subordinate_employee):
        self.hours_worked = 1
        self.subordinate = subordinate_employee

    def increment_subordinate_hours(self):
        self.subordinate.increment_hours()


an_employee = Employee()
print("Before:", an_employee.hours_worked)
a_manager = Manager(an_employee)
a_manager.increment_subordinate_hours()
print("After:", an_employee.hours_worked)
```

1)Clean code is less likely to contain bugs and errors, which can save time and resources in debugging and testing
2)Clean code can help developers work more efficiently in a team environment by allowing for easier communication and collaboration. This can lead to more productive teamwork and better results..
3) Clean code is easier to read and understand, making it simpler for developers to maintain and modify the code as needed. This can save time and resources in the long run.

Virtual RAM

The virtual RAM is a space on the hard disk that is used as temporary RAM when all of the actual RAM is being used. The operating system uses virtual RAM to enable applications to continue working and executing necessary tasks, but the speed of virtual RAM is much slower than that of actual RAM.
In fact, the speed of virtual RAM depends on the speed of the hard disk and the data transfer rate to and from the hard disk. Typically, the hard disk is much slower than the RAM, and thus virtual RAM is much slower than actual RAM. However, the speed of virtual RAM can be improved by increasing the speed of the hard disk or using a high-speed hard disk, such as modern SSDs, which have much higher data transfer rates than traditional hard disks. Increasing the actual available RAM in the system can also reduce the use of virtual RAM and improve performance speed.

Inverse priority queue

In Python, you can invert the priority queue by simply multiplying the priority value by -1 while inserting elements into the queue.

```
import heapq
q = []
heapq.heappush(q, (-priority, item))
```

In C++, you can invert the priority queue by defining a custom comparison function that compares the second element of the pair instead of the first.

```
#include <queue>
#include <utility>
using namespace std;
struct Compare {
  bool operator() (const pair<int, int>& lhs, const pair<int, int>& rhs) const {
    return lhs.second > rhs.second;
  }
};
int main() {
priority_queue<pair<int, int>, vector<pair<int, int>>, Compare> q;
  q.push(make_pair(item, -priority));
}
```

Hashmap

A HashMap is a data structure used to store and retrieve values by keys. It is part of the set of data structures known as hash tables.A HashMap works by converting the key to a specific value in an array using a hash function. The value associated with this key is stored at the location in the array. When retrieving the value, the hash function is used again to convert the key to the same location in the array to retrieve the associated value. The HashMap has a high speed in adding, deleting, and retrieving elements. It is used in many applications and programs, such as databases, file storage, text analysis software, and many other applications that require storing and retrieving values by keys.The complexity of a HashMap depends on the size of the array and the hash function used. In the average case, adding, deleting, and retrieving an element in a HashMap has a complexity of O(1). However, in the worst case, the complexity can be O(n), where n is the number of elements in the HashMap.

Paging and fragmintation

Paging is a memory storage technique used in operating systems that allows the main memory to be divided into small-sized pages, rather than storing information in one large block. The necessary pages are loaded into the main memory when needed and the unnecessary ones are unloaded.
The main memory space is divided into equal-sized pages, and the pages are stored in a page file on the hard disk. The required pages are loaded into the main memory, and when the program finishes using them, they are removed from the main memory and saved in the page file on the hard disk.
Fragmentation occurs when pages are loaded and unloaded frequently, leaving small gaps in the main memory. When a new page needs to be loaded, there may not be enough space in the main memory, even if its size is small. This increases the number of pages that need to be stored in the page file, thus affecting the system's performance.
Disk fragmentation occurs when files are stored on the hard disk and are deleted and modified frequently, causing data fragmentation on the hard disk and scattering data across the disk.