Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code..  
Normally the first step in debugging is to attempt to reproduce the problem.  
Scripting and breakpointing is also part of this process.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
There exist a lot of different approaches for each of those tasks.  
 Following a consistent programming style often helps readability.  
While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se.  
For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
 Programmable devices have existed for centuries.  
 Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA).  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference.