# Writing a Letter of Application

**Salutation**: This is your polite greeting. The most common salutation is "Dear Mr./Ms." followed by the person's last name. Find out more about appropriate cover letter salutations, including what to do if you don't know the person's name, or are unsure of a contact's gender.

*Examples* of general salutations:

Dear Hiring Manager

To Whom It May Concern

Dear Human Resources Manager

Dear Sir or Madam

Dear Company Name Recruiter

Follow the salutation with a colon or comma before beginning your first paragraph on the following line. For example:

Dear XYZ Enterprises Recruiter,

First paragraph of letter.

**Body of email**: Think of this section as being three distinct parts.

In the first paragraph, you'll want to mention the job you are applying for and where you saw the job listing.

The next paragraph(s) are the most important part of your letter. Remember how you gathered all that information about what employers were seeking, and how you could meet their needs? This is where you'll share those relevant details on your experience and accomplishments.

The third and last part of the body of the email will be your thank you to the employer; you can also offer follow-up information.

**Complimentary Close**: Sign off your email with a polite close, such as "Best" or "Sincerely," followed by your name.

Subject Line of Email Message: Assistant Director Position - Your Name

Email Message:

Dear Hiring Manager,

It was with much interest that I read your April 8th [insert alternate date] job posting on Craigslist for an Assistant Communications Director. Your description of the work responsibilities incumbent upon your next Assistant Director closely match my experience, and so I am excited to submit my resume to you for your consideration.

In my position as an Assistant Communications Director for ABC Company, I wrote articles for the company website, managed the editing and posting of contributing articles, managed their social media presence, and wrote and sent out a weekly email newsletter to subscribers. I also implemented an automated email tool that grew their subscriber base by 40% within six months.

While Assistant Communications Director for Assemblyperson Janet Brown, I researched, drafted and amended legislation, wrote press releases, and was responsible for office communications and correspondence.

My resume is attached. If I can provide you with any further information on my background and qualifications, please let me know.

I look forward to hearing from you.

Thank you for your consideration.

# 1. Speak to your parents about your speciality. What are its advantages for the job placement?

the software development world is one of the most challenging and rewarding career paths that someone can take.

Since I lead the instruction for a coding bootcamp I meet daily with aspiring students from around the world. One pattern I see with new students is that when I ask the question:

“What kind of developer do you want to be?”

I’ll usually be met with a blank stare. One time I even had a student reply back with the response:

“A good one!”

However through the years I’ve discovered that it’s vital for coders to decide what their development focus will be.

How to Choose a Developer Speciality

The world of software development is so vast that it’s impossible for someone to master every aspect of the process. Consider if I approached a world class track and field coach and said that I wanted to train for the olympics and win a gold medal in track and field.

The coach would most likely give me a once over and chuckle to himself. But after that his first question would be:

“What event do you want to train in?”

He would ask this question because the training regime for the 100 yard dash is dramatically different from the high jump.

In the same way, as a development student, you need to narrow your focus on what type of developer you want to be in the long run. It’s perfectly fine to start out by learning the full stack of development skills at a high level. In fact, during our three month immersive coding bootcamp courses we teach students the full application stack.

However after you have been introduced to each of the programming components it’s important that you pick what your speciality is going to be.

I’m going to walk through each of the types of developer categories so you can see what they entail. After going through the list I hope that you will be able to have a better idea of what type of coder you want to be.

### Full Stack

In deciding how to choose a developer speciality I always like to start off with the full stack option. I start with this option because many new coding students students I’ve spoken with assume that all developers are full stack devs. And this is simply not true.

Full stack development means that you feel comfortable working with every stage of an application’s development. In referencing our track and field analogy, a full stack developer would be like a decathlete.

This is the category that I personally fall into. My focus on the full stack side of programming is due to a number of factors:

* In order to teach students and write development curriculum I need to be familiar with all of the key development types.
* I’ve spent years as a freelance developer. And in many cases freelance coders are asked to build an application from the ground up, create all of the features, design the system, and deploy it to the web or app store.

### Jack of All Trades…

Much like a decathlete, full stack developers usually are good at a number of technologies. However a common pattern you’ll see is that it’s very difficult to be world class at EVERY layer of the development stack. Programming is simply too complex, and languages/frameworks change versions so rapidly, it makes it nearly impossible to excel at every stage of the app dev lifecycle.

I’ve spoken at length on how repetition is a requirement for mastering any skill. Because of how time consuming each level of the development process is, full stack devs simply don’t have the time to become true masters at any one aspect. I augment this issue by focusing my time on the components that I excel in, such as server side development. And then working with other developers to help cover my weaker areas, such as UI/UX.

### Server Side

Next on the list of developer types is server side programming. This is probably my favorite layer of the dev stack. Server side specialists spend most of their time working on building and implementing algorithms that enable programs to work properly.

Additionally server side developers typically spend quite a bit of time building APIs. This is because most server based applications need to communicate with the outside world in some form or another.

This layer of the development stack will require you to specialize in a language such as: Ruby, Python, Java, or C++.

### Front End

When it comes to choosing a developer speciality, the third layer to choose from is the front end component. Not too long ago a front end developer was considered someone who spent all day working with HTML and CSS. Their main goal was to make applications look *pretty*.

However the definition of a front end developer has changed dramatically with the advent of client side frameworks. These frameworks, such as Angular and React, have made it possible for front end programmers to build complete apps with little server side interaction.

These applications are rendered completely in the browser because they’re written in JavaScript *(which is a programming language that browsers can understand)*. And whenever the app needs to get additional data it simply communicates with APIs.

A common pattern that I work with is building a number of server side Ruby applications and then having a single Angular front end app that renders the user interface in the browser.

So if love building applications that users will directly interact with and the idea of working with APIs doesn’t scare you off, front end development might be the right choice for you.

### Mobile

Next on the list of developer types is mobile. If the idea of building the next Angry Birds or Instagram excites you, the mobile development field may be a good fit.

Mobile programming used to be a very difficult field to enter. Only a few years ago you would have had to master multiple languages *(Objective C and Java)* in order to build smartphone apps. However JavaScript frameworks such as Ionic and React Native have made it possible to use JavaScript to build apps that behave like native smartphone applications.

You can still use languages such as Swift, Objective C, and Java to build truly native applications. And there will always be a great set of jobs for devs who specialize in these languages. However if you are a freelance or full stack developer, by leveraging a JavaScript framework you can build smartphone and tablet based apps for all platforms. And it’s been my experience that the learning curve for these JavaScript frameworks is quite a bit lower than the traditional mobile languages.

Additionally, you may have noticed that the tools used for JavaScript based mobile apps and front end programming are similar. Because of this synergy I have had a number of developer friends who have moved away from server side development and moved into front end coding because it allows them to tackle building applications for: desktops, tablets, and phones.

### Making the Decision

If you are new to development don’t feel pressured to pick out a specialty immediately. Instead my recommendation is to explore each type of development layer until you find a focus that you truly love.

In this guide I’ve provided a very high level view of the developer types. However in reality you will need to become even more specific with your development focus. For example, if you’re a server side developer, you may want to focus on building eCommerce applications or implementing accounting systems. If you are an aspiring front end developer you may want to become a world class security specialist.

A key that I’ve discovered helps quite a few students is to look at developer job boards. Job boards are great for listing out the specialities that companies are hiring for. And by going through a list of potential job descriptions it may help you figure out what you want to focus on.

# 2. Tell your fellow student about your scientific research and its significance for the science

Assembler language is a low-level programming language, which is the human-readable recording format of machine commands. Assembly language commands one to one match with processor’s commands and, in fact, are a convenient commands symbolic notation (mnemonic). Assembly language also provides basic programming abstraction: linking parts of the program and data through the labels with symbolic names and directives.

Assembler and disassembler are two sides of the same coin. Assembler program converts assembly language text into binary code and disassembler translates the module in binary code into a sequence of assembler commands. Each byte of your program code section participates in the formation of a particular machine instruction. To determine the start of the next instruction, you need to properly (as it did processor) disassemble the previous one. It is extremely important to know the machine commands, their binary format and presentation in assembly to analyze the disassembled code.

Assembly is an irreversible and unidirectional process in which a huge amount of extra information completely unnecessary to the processor gets lost. On the one hand the source code written in a high-level programming language is extremely redundant, but on the other hand it simplifies code understanding and helps the programmer to avoid some mistakes.

Low-level processors handle basic types of information, i.e. byte, word, double word or lines are converted into a sequence of bytes, for which reason it is necessary to resort to heuristic algorithms to guess that this sequence of bytes is a string. The main stumbling block, however, is ambiguity of machine instructions correspondence to high-level language operators.

A disassembler is not expected to generate even so much as an approximate copy of the source code, but it is required to make an output listing readable and reassemble the text in a way that will not ruin the program and create an efficient module. Then we will be able to make any changes in the disassembled text, developing it in the required direction.

As a rule, the general principle of the disassembler is to look through memory

dump of the disassembled program, search for opcodes and data with their subsequent output in the form of assembler syntax structures.

There are single-pass and multi-pass disassemblers. The main computing difficulty for the disassembler is to distinguish data from the machine code. Thus, the first pass through the data collects information about the boundaries of procedures and functions, whereas on the last pass the final listing is formed.

Programs disassembly is one of the main tools of researching programs and their vulnerabilities. In order to secure software properly programmers must know hacking techniques. Programs research is also necessary to improve malware protection tools making the former an important area of computer science.

# 3. Data mining: background, process, uses

Data mining is simply filtering through large amounts of raw data for useful information that gives businesses a competitive edge. This information is made up of meaningful patterns and trends that are already in the data but were previously unseen.

The most popular tool used when mining is artificial intelligence (AI). AI technologies try to work the way the human brain works, by making intelligent guesses, learning by example, and using deductive reasoning.

Neural networks look at the rules of using data, which are based on the connections found or on a sample set of data. For example, you can set up mining software so that it looks at an investment database for patterns of stock prices in relation to other factors you dictate.

Clustering (also called family clusters and symbolic classifiers) divides data into groups based on similar features or limited data ranges. In a very simple example, mining might uncover a cluster of customers with low incomes. But when complex adjoining factors are filtered in and analyzed, mining might also reveal that young customers fall into this cluster, as well. Hence, the business’ customers who are younger tend to make less money.

Decision trees, like clusters, separate the data into subsets and then analyze the subsets to divide them into further subsets, and so on (for a few more levels). The final subsets are then small enough that the mining process can find interesting patterns and relationships within the data.

Once the data to be mined is identified, it should be cleansed. Cleansing data frees it from duplicate information and erroneous data. Next, the data should be stored in a uniform format within relevant categories or fields. Mining tools can work with all types of data storage, from large data warehouses to smaller desktop databases to flat files. Data warehouses and data marts are storage methods that involve archiving large amounts of data in a way that makes it easy to access when necessary.

When the process is complete, the mining software generates a report. An analyst goes over the report to see if further work needs to be done, such as refining mining parameters, using other data analysis tools to examine the data, or even scrapping the data if it’s unusable. If no further work is required, the report proceeds to the decision makers for appropriate action.

The power of data mining is being used for many other purposes, such as analyzing Supreme Court decisions, discovering patterns in health care, pulling stories about competitors from newswires, resolving bottlenecks in production processes, and analyzing sequences in the human genetic makeup. There really is no limit to the type of business or area of study where data mining can be beneficial.

# 4. Object-oriented programming

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another.[1][2] There is significant diversity of OOP languages, but the most popular ones are class-based, meaning that objects are instances of classes, which typically also determine their type.

Many of the most widely used programming languages (such as C++, Object Pascal, Java, Python etc.) are multi-paradigm programming languages that support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include Java, C++, C#, Python, PHP, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, and Smalltalk.

The prime purpose of C++ programming was to add object orientation to the C programming language, which is in itself one of the most powerful programming languages.

The core of the pure object-oriented programming is to create an object, in code, that has certain properties and methods. While designing C++ modules, we try to see whole world in the form of objects. For example a car is an object which has certain properties such as color, number of doors, and the like. It also has certain methods such as accelerate, brake, and so on.

There are a few principle concepts that form the foundation of object-oriented programming:

### Object

This is the basic unit of object oriented programming. That is both data and function that operate on data are bundled as a unit called as object.

### Class

When you define a class, you define a blueprint for an object. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

### Abstraction

Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

For example, a database system hides certain details of how data is stored and created and maintained. Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data.

### Encapsulation

Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object-oriented programming provides you framework to place the data and the relevant functions together in the same object.

### Inheritance

One of the most useful aspects of object-oriented programming is code reusability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

This is a very important concept of object-oriented programming since this feature helps to reduce the code size.

### Polymorphism

The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism. Poly refers to many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.

### Overloading

The concept of overloading is also a branch of polymorphism. When the exiting operator or function is made to operate on new data type, it is said to be overloaded.

# 5. Tell your fellow student about different operating systems. Give him a piece of advice on which one to use

When a brand new computer comes off the factory assembly line, it can do nothing. The hardware needs software to make it work because an applications software package does not communicate directly with the hardware. Between the applications software and the hardware is a software interface — an operating system. An operating system is a set of programs that lies between applications software and the computer hardware.

The most important program in the operating system, the program that manages the operating system, is the supervisor program, most of which remains in memory and is thus referred to as resident. It loads into memory other operating system programs (called nonresident) from disk storage only as needed.

An operating system has three main functions: (1) manage the computer's resources, (2) establish a user interface, and (3) execute and provide services for applications software. Keep in mind, however, that much of the work of an operating system is hidden from the user.

Today there are many operation systems: for desktop computers and mobile phones, with graphical user interface (GUI) or without it etc. But in the late I970s and early 80s users had to memorize and type a lot of commands to communicate with computer. In fact, it was only experts who used computers, so there was no need for a user-friendly interface.

Later, Apple produced the Macintosh, the first computer with a mouse and a (GUI). Macs were designed with one clear aim: to facilitate interaction with the computer.

A GUI makes use of a WIMP environment: windows, icons, menus and pointer. The background of the screen is called the desktop, which contains labeled pictures called icons. These icons represent files or folders. Double-clicking a folder opens a window which contains programs, documents, or more nested folders. When you run a program, your PC opens a window that lets you work with different tools. A modern OS also provides access to networks and allows multitasking, which means you can run several programs at the same time.

The most popular operating systems are:

■ The Windows family - designed by Microsoft and used on most PCs.

■ Windows Mobile - used on many PDAs and smart phones.

■ Mac OS - created by Apple.

■ Android OS

■ Unix - a multi-user system, found on mainframes and workstations in corporate installations.

■ Linux - open-source software developed under the GNU General Public License. This means 45 anybody can copy its source code, change it and distribute it. It is used in computers, appliances and small devices.

Linux has its roots in a student project. In 1992, an undergraduate called Linus Torvalds was studying computer science. Like most computer science courses, a big component of it was taught on Unix. Unix was the operating system which both a textbook example of the principles of operating system design, and robust to be the standard OS in engineering and scientific computing. But Unix was a commercial product, and cost more than a student could pay.

Annoyed by the shortcomings of Minix Linus set out to write his own 'kernel'. When he'd written a basic kernel, he released the source code to the Linux kernel on the Internet.

Source code is important. It's the original from which compiled programs are generated. If you don't have the source code to a program, you can't modify it to fix bugs or add new features. Most software companies won't sell you their source code, or will only do so for an eye-watering price.

Programmers began using Linux. They found that it didn't do things they wanted it to do —so they fixed it. And where they improved it, they sent the improvements to Linus, who rolled them into the kernel. And Linux began to grow.

As more and more people got to know about Linux, some of them began to port the Linux kernel to run on non-standard computers. Because it's free, Linux is now the most widely-ported operating system there is.

# 6. Cybercrime. Warn your parents on a possibility to infect the computer with viruses. Tell about their different types

The Internet is an amazing tool for communication, allowing users to connect instantly over great distances. Unfortunately, the reach and anonymity the network provides is also a great tool for criminals, who have taken advantage of the global network to ply their trade. There are many different types of cyber crime, and understanding the most common crimes and frauds can help you avoid becoming a victim.

### Identity Theft

One common form of cyber crime is identity theft. Hackers and scammers may use fake emails to trick victims into giving up passwords and account information, or they may use specialized programs called keyloggers to track what a user types when logging into bank or credit accounts. Once they have this personal information, they may be able to access existing accounts or make purchases with the victim’s credit cards. If a hacker can discover a user’s social security number and other identifying information, he can parlay that data into credit accounts in the victim’s name and cause considerable damage.

### Transaction Fraud

Simple financial fraud is another common crime in the online arena. A scammer may offer an item for sale through an auction site with no intention of delivering once he receives payment. Alternatively, a criminal might purchase an item for sale using a stolen credit card, or claim a fraudulent chargeback after receiving the goods.

### Advance Fee Fraud

One common crime is the advance fee fraud. These frauds, also known as 419 scams after the portion of the Nigerian criminal code relating to fraud, involve bilking victims out of money by promising them an eventual payoff. The scammer emails his victim with news of some financial windfall, often represented as the wealth of a distant relative or the remnants of some other illicit fortune. All the victim needs to do to claim this wealth is provide some identifying information and pay a few incidental expenses. The lure of easy wealth has found many victims for the perpetrators of these frauds, with some individual marks losing thousands, or even hundreds of thousands of dollars.

### Hacking

Another cyber crime is the practice of hacking, illegally circumventing security to access someone else’s computer system. Some hackers explore for sheer curiosity, finding their way into unfamiliar systems for love of the challenge, in some cases going so far as to alert system owners to security loopholes. Others hack for their own reasons, either to steal information, gain control over systems for their own purposes, or simply to cause as much damage and chaos as possible.

### Piracy

Piracy is the copying and distribution of programs, movies, music or other intellectual property without permission. Groups of dedicated pirates take the source material, remove any protection the data might have and then pass the unprotected results on to file sharing networks and distribution sites. The movie and recording industries in particular have fought the misuse of their intellectual property by filing extensive lawsuits against file sharers, while software companies fight piracy through expanded and intrusive copy protection schemes.

### Other Crimes

Other crimes which exist in the offline world may also take place online. Those who trade in child pornography, for instance, often take advantage of the anonymity provided by the Internet when interacting with their fellow criminals. The drug trade also has an online component, as dealers use alternative currencies and anonymous Web providers to peddle their wares. Even users looking for legal drugs may find a gray market on the Internet where they can purchase medications without a prescription from other countries, although in some cases these sellers may provide expired, incorrect, or even dangerous compounds to unwitting purchasers.

Knowledge is power. Cybercriminals know this and so, too, should anyone who accesses the Internet for personal enjoyment, work or to conduct personal business. With cybercrime rates and their associated costs continually on the rise, it’s imperative for Internet users to understand some of the most common threats in order to avoid them. There are several types of online safety risks that anyone – even those with highly sophisticated antiviral protections – can find themselves vulnerable to. Having an understanding of what they are and how they work can enable avoidance even if all other forms of protection fail.

### Online Scams

As they are in the offline worlds, scams are any type of scheme that is designed to part a person from their money or information that can lead others to the ability to access a target’s money. These schemes are often highly sophisticated and are meant to dupe victims into believing they are legitimate operations. Keep in mind, however, that not all Internet scams are sophisticated, but in many cases, they still work.According to the FBI, some of the more common Internet scams include:

* Email scams including the Nigerian Letter scam in which senders dupe people out of money by promising larger rewards in return;
* Advance fee scams;
* Ponzi schemes;
* Pyramid schemes;
* Online auction fraud;
* Credit card fraud;
* Investment and business fraud.

Protecting against scams can be as simple as heeding the old cliché: “If it sounds too good to be true, it probably is.” If an offer is unsolicited or it just smells fishy, it probably is. Do not respond to unsolicited emails. Do not follow through on “business-related” email links. Instead, go directly to the website in question. If applying for a credit card based on an offer, for example, go to the known banking company’s website directly to apply. Do not follow the link through.

### Phishing

Phishing scams are quite popular and don’t often have to be sophisticated to be effective. In this type of scheme, a bad guy solicits personal information, such as passwords, user names, bank account information and so on, by convincing the target the request comes from a reputable source. In essence, victims of phishing schemes willingly give up data about themselves because they believe they are dealing with a reputable source. Phishing schemes take place online and off. In the online arena, they are often introduced via emails that may profess to come from a bank, government agency, business or other trusted source. Phishing schemes often rely on fake websites that are designed to look identical to their legitimate counterparts to collect and capture the data victims unwittingly share. A phishing email, for example, might include a link to a fake bank website that has been set up to mirror the actual, legitimate bank’s website.

### Spoofing

This is a variation on the phishing scheme that is often more difficult to detect. In a spoofing scheme, the bad guys actually send out emails that appear to come from the trusted, legitimate source. If a person banks with ABC Money, for example, and email from that bank has an abcmoney.com identifier, the spoofed email will have the same. Essentially, the bad guys are sending out forgeries. Since spoofing can be especially difficult to spot and easy to fall for it’s important for anyone who receives “official” email correspondence to think twice before clicking through on links and sharing information. Beyond avoiding links in an email since they can lead to bogus mirror websites, if possible, call the entity directly to see if the email is valid before taking action online to respond to whatever the request asks.

### Hoaxes

Internet hoaxes are often benign stunts meant to gain publicity or just create an environment of excitement as a hoax goes viral. Many notable online hoaxes have caught a ton of press and attention without necessarily harming anyone. One of the more famous examples of online hoaxes include the BBC’s April Fool’s Day flying penguin video.

Online hoaxes, however, aren’t always quite so innocent. If a request for money or information is attached to a viral email, video or social media share, take heed. It is possible whatever is being promoted is less than legitimate. To safeguard against hoaxes, be sure to:

* Vet what is read, seen or heard by going to reliable, independent sources;
* Make sure to only share personal information, such as credit card numbers, with known, reliable sources and even then only via secured, encrypted websites.

Cybercriminals believe in diversifying to reel in new targets to their schemes. Attacks online can come in all forms from the very mundane to the overly sophisticated. Protecting against them requires a healthy dose of skepticism, good online security protection programs, and a willingness to follow through to make sure something is as it appears before information is shared.

# 7. Programming languages

Unfortunately for us, computers can’t understand spoken English or any other natural language. The only language they can understand directly is machine code, which consists of 1s and 0s. Machine code is too difficult to write. For this reason, we use symbolic languages to communicate instructions to the computer. Machine code and assembly languages are called low-level languages because they are closer to the hardware. They are quite complex and restricted to particular machines. To make the programs easier to write, and to overcome the problem of intercommunication between different types of computer, software developers designed high-level languages, which are closer to natural language. Here are some examples:

– FORTRAN stands for FORmula TRANslator it is used for scientific and engineering applications.

– COBOL (Common Business Oriented Language) is mainly used for business applications.

– Visual BASIC is a modern version of the old Basic (Beginners' All-purpose Symbolic Instruction Code). It is a simple-to-use language that has a graphical interface. It makes it easy for an inexperienced programmer to create programs.

– PASCAL. It is used in universities to teach the fundamentals of programming.

– C. It is used to write system software, graphics and commercial applications. C++ is a version of C which incorporates object-oriented programming.

– JAVA was designed to run on the Web. Java applets provide animation and interactive features on web pages.

Programs written in high-level languages must be translated into machine code by a compiler or an interpreter. A compiler translates the source code into object code – that is, it converts the entire program into machine code in one go. On the other hand, an interpreter translates the source code line by line as the program is running.

It is important not to confuse programming languages with markup languages, used to create web documents. Markup languages use instructions, known as markup tags, to format and link files. Examples are SGML, HTML and XML.

Standard Generalized Markup Language (SGML) is the language that spawned both HTML (HyperText Markup Language) and XML (extensible Markup Language). SGML is not a true language it is a metalanguage, which is a language from which you can create other languages.

HTML is a language that is all about the presentation of your information, not what the actual data is. You can, therefore, say that HTML is a presentation language. On the other hand XML is extensible because it lets website developers create their own set of customized tags for documents. This ability to define your own tags is the main feature of XML, and it is what gives developers more flexibility. In fact, because XML is an extensible language, you don't even have to have a browser to interpret the page. Applications can parse the XML document and read the information without any human intervention.

# 8. You take part in the conference on the systems analysis. Give a report on systems thinking and its use of different methodologies

For some, systems thinking is the cognitive process of studying and understanding systems of every kind. For others, the focus is integrating information from different sources and different types.[citation needed]

A system may be defined in general as a set of interrelated or interacting elements. This definition accommodates both passive structures (e.g. a necklace, or the Dewey Decimal System) and active structures. However, most system theorists focus on activity systems in which structures/components interact in behaviors/processes.

In biology, a living organism is seen as a set of organs, muscles etc. that interact in processes to sustain the organism. Each cell is seen as a collection of organelles that interact in processes to sustain both the cell and the wider organism. In business, the organization is seen as a set of people and machines that interact in processes to achieve business goals.

The term general system theory was coined by Ludwig von Bertalanffy in the middle of the 20th century. General systems theory is about broadly applicable concepts and principles, as opposed to concepts and principles applicable to one domain of knowledge. Bertalanffy's ideas were picked up by others, including Ross Ashby and Anatol Rapoport, working in the fields of mathematics, psychology, biology, game theory and social network analysis.

An early focus of general system theory was on homeostatic or self-regulating systems that maintain themselves in a consistent or viable state through input/output feedback loops.

Sociological systems thinking started much earlier, in the 19th century. It may be now seen as the specialism of general system theory focused on social and business systems. Such systems are often described in terms of inputs, transformations and outputs, and feedback loops that operate (in the light of goals, stakeholders, and external influences) to make an organization healthy or unhealthy.

The systems thinking approach incorporates several tenets:

Interdependence of objects and their attributes – independent elements can never constitute a system

Holism – emergent properties not possible to detect by analysis should be possible to define by a holistic approach

Goal seeking – systemic interaction must result in some goal or final state

Inputs and outputs – in a closed system inputs are determined once and constant; in an open system additional inputs are admitted from the environment

Transformation of inputs into outputs – the process by which the goals are obtained

Entropy – the amount of disorder or randomness present in any system

Regulation – a method of feedback is necessary for the system to operate predictably

Hierarchy – complex wholes are made up of smaller subsystems

Differentiation – specialized units perform specialized functions

Equifinality – alternative ways of attaining the same objectives (convergence)

Multifinality – attaining alternative objectives from the same inputs (divergence)

A treatise on systems thinking ought to address many issues including:

Encapsulation of a system in space and/or in time

Active and passive systems (or structures)

Transformation by an activity system of inputs into outputs

Persistent and transient systems

Evolution, the effects of time passing, the life histories of systems and their parts.

Design and designers.

Using the tenet of "multifinality", a supermarket could be considered a:

"Profit making system" from the perspective of management and owners

"Distribution system" from the perspective of the suppliers

"Employment system" from the perspective of employees

"Materials supply system" from the perspective of customers

"Entertainment system" from the perspective of loiterers

"Social system" from the perspective of local residents

"Dating system" from the perspective of single customers

As a result of such thinking, new insights may be gained into how the supermarket works, why it has problems, how it can be improved or how changes made to one component of the system affects other components.

### Introduction to Systems Thinking

Reductionism was the dominant mode of scientific thinking in the early 20th century. Reductionist thinking and methods were the basis for many of the well developed areas of modern science like physics, chemistry and biology (Wikipedia, 2008). Reductionism is an approach to building descriptions of systems out of the descriptions of the subsystems that a system is composed of, and ignoring the relationships between them (Bar-Yam, 2000). However reductionist thinking struggled to explain properties that systems exhibited as a whole which their constituent parts by themselves did not exhibit. Systems thinking emerged in the 20th century through a critique of reductionist thinking (Flood R. L., 2007). Systems thinking viewed the world as systemic and its foundational tenant was that phenomena is understood to be an emergent property of an interrelated whole. Emergence and interrelatedness are fundamental properties of systems thinking. An emergent property of a whole is said to arise where a phenomenon cannot be fully comprehended in terms only of properties of constituent parts. “The whole is greater than the sum of its parts” is the popularized phrase that explains emergence (Flood R. L., 2007).

### From Reductionism to Systems Thinking

Systems thinking came to the fore as a valid alternative in the 1920’s when research into living things encountered limitations to the concepts and principles of reductionism. The brilliant Ludwig Von Bertalanffy, an Austrian biologist, demonstrated that concepts of reductionism were helpless in appreciating the dynamics of organisms. Existence of an organism cannot be understood solely in terms of behavior of some fundamental parts (Flood R. L., 2007). In this regard, Von Bertalanffy developed the theory of “Open Systems”. Open Systems Theory employs functional and relational criteria to study the whole, rather than principles of reductionism to study the simple elements. The different parts of an organism exist together as a whole and it co-exists in relation to its environment. The various parts are interrelated through feedback loops – both positive and negative. An organism achieves a steady state of being i.e. normal condition, through the feedback loops. Von Bertalanffy generalized the open systems concept for other fields of study and named it as General Systems Theory (Bertalanffy, 1976). The lasting impact of his ideas collectively came to be known as Systems Thinking. Systems thinking was readily taken up as the basis of a new form of social theory (Flood R. L., 2007). Taken into the fields of organizational analysis, systems thinking observe organizations as complex systems made up of interrelated parts most usefully studied as an emergent whole. Thus a modern organization is seen as a system that comprises of people, processes and information systems that facilitates communication, business transactions and so on. Management action is taken to hold the organization in a steady state through management functions that control activities and information within the organization, and also between the organization and its environment (Flood R. L., 2007).

### Evolution of Systems Thinking

Cybernetic Theory, also based on systems thinking, came to the fore around the same time as Von Bertalanffy’s research in general systems theory. Cybernetics is traditionally defined as the science of communication and control in man and machine (Flood R. L., 2007). Cybernetics studied communication and control involving regulatory feedback in living organisms, machines and organizations, as well as their combinations. Cybernetics found its place in the management sciences in the guise of control theory, systems engineering and, information theory. As cybernetic theory grew, it began to influence practice and together with Bertalanffy’s open systems theory this endeavor became to be called as Applied Systems Thinking. The applied systems thinking methodology is an intervention that begins with problem identification and concludes with some final solution, with an expectation that things will attain a desirable condition (Flood R. L., 2007). The challenge was to find the most efficient means to get to this desired end. Jay Forrester, computer engineer and retired professor at MIT’s Sloan School of Management, advanced the field of systems thinking and created a new strand of thinking called System Dynamics. He defined it the following way: “System dynamics is a professional field that deals with the complexity of systems. System dynamics is the necessary foundation underlying effective thinking about systems. System dynamics deals with how things change through time, which covers most of what most people find important. System dynamics involves interpreting real life systems into computer simulation models that allow one to see how the structure and decision-making policies in a system create its behavior” (Forrester, 1999). Peter Senge, a student of Forrester at MIT, studied System Dynamics under him and through his book “The Fifth Discipline” popularized system dynamics for its contribution to organizational learning. Senge argues that five disciplines underpin learning organizations: personal mastery, mental models, shared vision, team learning and systems thinking. Systems thinking is the fifth discipline that provides substance to the other four disciplines and hence learning to the organization as a whole (Flood R. L., 2007).

### Systemic Thinking: Emergence of Soft Systems Thinking

As systems thinking evolved and began to grow as a field of enquiry, researchers faced the issue of how to view the social world. Was the social world “real and concrete” that comprised real social systems or was it to be seen as “emergent” through the meaning people ascribed to the world? Systemic thinking grew out of this debate and rejected the belief in a concrete social world. Researchers such as Peter B. Checkland began to argue that “human systems” are different and that they should be understood from the meaning people give to the world (Checkland P. B., Systems Thinking, Systems Practice, 1981). Soft Systems Thinking is a form of systems thinking that saw social reality as a construction of people’s interpretation of their experiences, thus linking itself to interpretive theory (Flood R. L., 2007).

# 9. Tell about the uses of Game Theory

Game theory is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers." Game theory is mainly used in economics, political science, and psychology, as well as logic, computer science and biology.[1] Originally, it addressed zero-sum games, in which one person's gains result in losses for the other participants. Today, game theory applies to a wide range of behavioral relations, and is now an umbrella term for the science of logical decision making in humans, animals, and computers.

Modern game theory began with the idea regarding the existence of mixed-strategy equilibria in two-person zero-sum games and its proof by John von Neumann. Von Neumann's original proof used the Brouwer fixed-point theorem on continuous mappings into compact convex sets, which became a standard method in game theory and mathematical economics. His paper was followed by the 1944 book Theory of Games and Economic Behavior, co-written with Oskar Morgenstern, which considered cooperative games of several players. The second edition of this book provided an axiomatic theory of expected utility, which allowed mathematical statisticians and economists to treat decision-making under uncertainty.

This theory was developed extensively in the 1950s by many scholars. Game theory was later explicitly applied to biology in the 1970s, although similar developments go back at least as far as the 1930s. Game theory has been widely recognized as an important tool in many fields. With the Nobel Memorial Prize in Economic Sciences going to game theorist Jean Tirole in 2014, eleven game-theorists have now won the economics Nobel Prize. John Maynard Smith was awarded the Crafoord Prize for his application of game theory to biology.

## Game types

### Cooperative / Non-cooperative

A game is *cooperative* if the players are able to form binding commitments externally enforced (e.g. through [contract law](https://en.wikipedia.org/wiki/Contract_law)). A game is *non-cooperative* if players cannot form alliances or if all agreements need to be [self-enforcing](https://en.wikipedia.org/wiki/Self-enforcing_agreement) (e.g. through [credible threats](https://en.wikipedia.org/wiki/Credible_threat)).

Cooperative games are often analysed through the framework of *cooperative game theory***,** which focuses on predicting which coalitions will form, the joint actions that groups take and the resulting collective payoffs. It is opposed to the traditional *non-cooperative game theory* which focuses on predicting individual players' actions and payoffs and analyzing [Nash equilibria](https://en.wikipedia.org/wiki/Nash_equilibria).

Cooperative game theory provides a high-level approach as it only describes the structure, strategies and payoffs of coalitions, whereas non-cooperative game theory also looks at how bargaining procedures will affect the distribution of payoffs within each coalition. As non-cooperative game theory is more general, cooperative games can be analyzed through the approach of non-cooperative game theory (the converse does not hold) provided that sufficient assumptions are made to encompass all the possible strategies available to players due to the possibility of external enforcement of cooperation. While it would thus be optimal to have all games expressed under a non-cooperative framework, in many instances insufficient information is available to accurately model the formal procedures available to the players during the strategic bargaining process, or the resulting model would be of too high complexity to offer a practical tool in the real world. In such cases, cooperative game theory provides a simplified approach that allows to analyze the game at large without having to make any assumption about bargaining powers.

### Symmetric / Asymmetric

|  |  |  |
| --- | --- | --- |
|  | E | F |
| E | 1, 2 | 0, 0 |
| F | 0, 0 | 1, 2 |
| *An asymmetric game* | | |

A symmetric game is a game where the payoffs for playing a particular strategy depend only on the other strategies employed, not on who is playing them. If the identities of the players can be changed without changing the payoff to the strategies, then a game is symmetric. Many of the commonly studied 2×2 games are symmetric. The standard representations of [chicken](https://en.wikipedia.org/wiki/Game_of_chicken), the [prisoner's dilemma](https://en.wikipedia.org/wiki/Prisoner%27s_dilemma), and the [stag hunt](https://en.wikipedia.org/wiki/Stag_hunt) are all symmetric games. Some[[*who?*](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Words_to_watch#Unsupported_attributions)] scholars would consider certain asymmetric games as examples of these games as well. However, the most common payoffs for each of these games are symmetric.

Most commonly studied asymmetric games are games where there are not identical strategy sets for both players. For instance, the [ultimatum game](https://en.wikipedia.org/wiki/Ultimatum_game) and similarly the [dictator game](https://en.wikipedia.org/wiki/Dictator_game) have different strategies for each player. It is possible, however, for a game to have identical strategies for both players, yet be asymmetric. For example, the game pictured to the right is asymmetric despite having identical strategy sets for both players.

### Zero-sum / Non-zero-sum

|  |  |  |
| --- | --- | --- |
|  | A | B |
| A | –1, 1 | 3, –3 |
| B | 0, 0 | –2, 2 |
| *A zero-sum game* | | |

Zero-sum games are a special case of constant-sum games, in which choices by players can neither increase nor decrease the available resources. In zero-sum games the total benefit to all players in the game, for every combination of strategies, always adds to zero (more informally, a player benefits only at the equal expense of others). [Poker](https://en.wikipedia.org/wiki/Poker) exemplifies a zero-sum game (ignoring the possibility of the house's cut), because one wins exactly the amount one's opponents lose. Other zero-sum games include [matching pennies](https://en.wikipedia.org/wiki/Matching_pennies) and most classical board games including [Go](https://en.wikipedia.org/wiki/Go_(board_game)) and [chess](https://en.wikipedia.org/wiki/Chess).

Many games studied by game theorists (including the famed [prisoner's dilemma](https://en.wikipedia.org/wiki/Prisoner%27s_dilemma)) are non-zero-sum games, because the [outcome](https://en.wikipedia.org/wiki/Outcome_(game_theory)) has net results greater or less than zero. Informally, in non-zero-sum games, a gain by one player does not necessarily correspond with a loss by another.

Constant-sum games correspond to activities like theft and gambling, but not to the fundamental economic situation in which there are potential [gains from trade](https://en.wikipedia.org/wiki/Gains_from_trade). It is possible to transform any game into a (possibly asymmetric) zero-sum game by adding a dummy player (often called "the board") whose losses compensate the players' net winnings.

### Simultaneous / Sequential

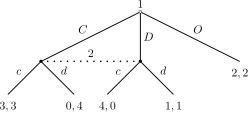
[Simultaneous games](https://en.wikipedia.org/wiki/Simultaneous_game) are games where both players move simultaneously, or if they do not move simultaneously, the later players are unaware of the earlier players' actions (making them *effectively* simultaneous). [Sequential games](https://en.wikipedia.org/wiki/Sequential_game) (or dynamic games) are games where later players have some knowledge about earlier actions. This need not be [perfect information](https://en.wikipedia.org/wiki/Perfect_information) about every action of earlier players; it might be very little knowledge. For instance, a player may know that an earlier player did not perform one particular action, while he does not know which of the other available actions the first player actually performed.

The difference between simultaneous and sequential games is captured in the different representations discussed above. Often, [normal form](https://en.wikipedia.org/wiki/Normal_form_game) is used to represent simultaneous games, while [extensive form](https://en.wikipedia.org/wiki/Extensive_form_game) is used to represent sequential ones. The transformation of extensive to normal form is one way, meaning that multiple extensive form games correspond to the same normal form. Consequently, notions of equilibrium for simultaneous games are insufficient for reasoning about sequential games; see [subgame perfection](https://en.wikipedia.org/wiki/Subgame_perfection).

In short, the differences between sequential and simultaneous games are as follows:

|  |  |  |
| --- | --- | --- |
|  | **Sequential** | **Simultaneous** |
| Normally denoted by | [Decision trees](https://en.wikipedia.org/wiki/Decision_tree) | [Payoff matrices](https://en.wikipedia.org/wiki/Payoff_matrix) |
| Prior knowledge of opponent's move? | Yes | No |
| Time axis? | Yes | No |
| Also known as | [Extensive-form game](https://en.wikipedia.org/wiki/Extensive-form_game) Extensive game | [Strategy game](https://en.wikipedia.org/wiki/Strategy_game) Strategic game |

**Perfect information and imperfect informatio**

[](https://en.wikipedia.org/wiki/File:PD_with_outside_option.svg)

A game of imperfect information (the dotted line represents ignorance on the part of player 2, formally called an [information set](https://en.wikipedia.org/wiki/Information_set_(game_theory)))

An important subset of sequential games consists of games of [perfect information](https://en.wikipedia.org/wiki/Perfect_information). A game is one of perfect information if, in extensive form, all players know the moves previously made by all other players. Simultaneous games can not be games of perfect information, because the conversion to extensive form converts simultaneous moves into a sequence of moves with earlier moves being unknown. Most games studied in game theory are imperfect-information games. Interesting examples of perfect-information games include the [ultimatum game](https://en.wikipedia.org/wiki/Ultimatum_game) and [centipede game](https://en.wikipedia.org/wiki/Centipede_game). Recreational games of perfect information games include [chess](https://en.wikipedia.org/wiki/Chess) and [checkers](https://en.wikipedia.org/wiki/Draughts). Many card games are games of imperfect information, such as [poker](https://en.wikipedia.org/wiki/Poker) or [contract bridge](https://en.wikipedia.org/wiki/Contract_bridge).[[13]](https://en.wikipedia.org/wiki/Game_theory#cite_note-13)

Perfect information is often confused with [complete information](https://en.wikipedia.org/wiki/Complete_information), which is a similar concept. Complete information requires that every player know the strategies and payoffs available to the other players but not necessarily the actions taken. Games of incomplete information can be reduced, however, to games of imperfect information by introducing "[moves by nature](https://en.wikipedia.org/wiki/Move_by_nature)".

### Combinatorial games

Games in which the difficulty of finding an optimal strategy stems from the multiplicity of possible moves are called combinatorial games. Examples include chess and go. Games that involve imperfect or incomplete information may also have a strong combinatorial character, for instance [backgammon](https://en.wikipedia.org/wiki/Backgammon). There is no unified theory addressing combinatorial elements in games. There are, however, mathematical tools that can solve particular problems and answer general questions.

Games of perfect information have been studied in [combinatorial game theory](https://en.wikipedia.org/wiki/Combinatorial_game_theory), which has developed novel representations, e.g. [surreal numbers](https://en.wikipedia.org/wiki/Surreal_numbers), as well as [combinatorial](https://en.wikipedia.org/wiki/Combinatorics) and [algebraic](https://en.wikipedia.org/wiki/Abstract_algebra) (and [sometimes non-constructive](https://en.wikipedia.org/wiki/Strategy_stealing_argument)) proof methods to [solve games](https://en.wikipedia.org/wiki/Solved_game) of certain types, including "loopy" games that may result in infinitely long sequences of moves. These methods address games with higher combinatorial complexity than those usually considered in traditional (or "economic") game theory. A typical game that has been solved this way is [hex](https://en.wikipedia.org/wiki/Hex_(board_game)). A related field of study, drawing from [computational complexity theory](https://en.wikipedia.org/wiki/Computational_complexity_theory), is [game complexity](https://en.wikipedia.org/wiki/Game_complexity), which is concerned with estimating the computational difficulty of finding optimal strategies.

Research in [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) has addressed both perfect and imperfect (or incomplete) information games that have very complex combinatorial structures (like chess, go, or backgammon) for which no provable optimal strategies have been found. The practical solutions involve computational heuristics, like [alpha-beta pruning](https://en.wikipedia.org/wiki/Alpha-beta_pruning) or use of [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network) trained by [reinforcement learning](https://en.wikipedia.org/wiki/Reinforcement_learning), which make games more tractable in computing practice.

### Infinitely long games

Games, as studied by economists and real-world game players, are generally finished in finitely many moves. Pure mathematicians are not so constrained, and [set theorists](https://en.wikipedia.org/wiki/Set_theory) in particular study games that last for infinitely many moves, with the winner (or other payoff) not known until *after* all those moves are completed.

The focus of attention is usually not so much on the best way to play such a game, but whether one player has a [winning strategy](https://en.wikipedia.org/wiki/Determinacy#Basic_notions). (It can be proven, using the [axiom of choice](https://en.wikipedia.org/wiki/Axiom_of_choice), that there are games – even with perfect information and where the only outcomes are "win" or "lose" – for which *neither* player has a winning strategy.) The existence of such strategies, for cleverly designed games, has important consequences in [descriptive set theory](https://en.wikipedia.org/wiki/Descriptive_set_theory).

### Discrete and continuous games

Much of game theory is concerned with finite, discrete games, that have a finite number of players, moves, events, outcomes, etc. Many concepts can be extended, however. [Continuous games](https://en.wikipedia.org/wiki/Continuous_game) allow players to choose a strategy from a continuous strategy set. For instance, [Cournot competition](https://en.wikipedia.org/wiki/Cournot_competition) is typically modeled with players' strategies being any non-negative quantities, including fractional quantities.

### Differential games

[Differential games](https://en.wikipedia.org/wiki/Differential_game) such as the continuous [pursuit and evasion game](https://en.wikipedia.org/wiki/Pursuit-evasion) are continuous games where the evolution of the players' state variables is governed by [differential equations](https://en.wikipedia.org/wiki/Differential_equation). The problem of finding an optimal strategy in a differential game is closely related to the [optimal control](https://en.wikipedia.org/wiki/Optimal_control) theory. In particular, there are two types of strategies: the open-loop strategies are found using the [Pontryagin maximum principle](https://en.wikipedia.org/wiki/Pontryagin%27s_Minimum_Principle) while the closed-loop strategies are found using [Bellman's Dynamic Programming](https://en.wikipedia.org/wiki/Hamilton%E2%80%93Jacobi%E2%80%93Bellman_equation) method.

A particular case of differential games are the games with a random [time horizon](https://en.wikipedia.org/wiki/Time_horizon). In such games, the terminal time is a random variable with a given [probability distribution](https://en.wikipedia.org/wiki/Probability_distribution)function. Therefore, the players maximize the [mathematical expectation](https://en.wikipedia.org/wiki/Mathematical_expectation) of the cost function. It was shown that the modified optimization problem can be reformulated as a discounted differential game over an infinite time interval.

### Many-player and population games

Games with an arbitrary, but finite, number of players are often called n-person games.[[21]](https://en.wikipedia.org/wiki/Game_theory#cite_note-FOOTNOTELuceRaiffa1957-21) [Evolutionary game theory](https://en.wikipedia.org/wiki/Evolutionary_game_theory) considers games involving a [population](https://en.wikipedia.org/wiki/Population) of decision makers, where the frequency with which a particular decision is made can change over time in response to the decisions made by all individuals in the population. In biology, this is intended to model (biological) [evolution](https://en.wikipedia.org/wiki/Evolution), where genetically programmed organisms pass along some of their strategy programming to their offspring. In economics, the same theory is intended to capture population changes because people play the game many times within their lifetime, and consciously (and perhaps rationally) switch strategies.

### Stochastic outcomes (and relation to other fields)

Individual decision problems with stochastic outcomes are sometimes considered "one-player games". These situations are not considered game theoretical by some authors.[[*by whom?*](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Words_to_watch#Unsupported_attributions)] They may be modeled using similar tools within the related disciplines of [decision theory](https://en.wikipedia.org/wiki/Decision_theory), [operations research](https://en.wikipedia.org/wiki/Operations_research), and areas of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence), particularly [AI planning](https://en.wikipedia.org/wiki/AI_planning) (with uncertainty) and [multi-agent system](https://en.wikipedia.org/wiki/Multi-agent_system). Although these fields may have different motivators, the mathematics involved are substantially the same, e.g. using [Markov decision processes](https://en.wikipedia.org/wiki/Markov_decision_process) (MDP).[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

Stochastic outcomes can also be modeled in terms of game theory by adding a randomly acting player who makes "chance moves" ("[moves by nature](https://en.wikipedia.org/wiki/Move_by_nature)").[[23]](https://en.wikipedia.org/wiki/Game_theory#cite_note-FOOTNOTEOsborneRubinstein1994-23) This player is not typically considered a third player in what is otherwise a two-player game, but merely serves to provide a roll of the dice where required by the game.

For some problems, different approaches to modeling stochastic outcomes may lead to different solutions. For example, the difference in approach between MDPs and the [minimax solution](https://en.wikipedia.org/wiki/Minimax) is that the latter considers the worst-case over a set of adversarial moves, rather than reasoning in expectation about these moves given a fixed probability distribution. The minimax approach may be advantageous where stochastic models of uncertainty are not available, but may also be overestimating extremely unlikely (but costly) events, dramatically swaying the strategy in such scenarios if it is assumed that an adversary can force such an event to happen.[[24]](https://en.wikipedia.org/wiki/Game_theory#cite_note-McMahan-24) (See [Black swan theory](https://en.wikipedia.org/wiki/Black_swan_theory) for more discussion on this kind of modeling issue, particularly as it relates to predicting and limiting losses in investment banking.)

General models that include all elements of stochastic outcomes, adversaries, and partial or noisy observability (of moves by other players) have also been studied. The "[gold standard](https://en.wikipedia.org/wiki/Gold_standard)" is considered to be partially observable [stochastic game](https://en.wikipedia.org/wiki/Stochastic_game) (POSG), but few realistic problems are computationally feasible in POSG representation.

### Metagames

These are games the play of which is the development of the rules for another game, the target or subject game. [Metagames](https://en.wikipedia.org/wiki/Metagame) seek to maximize the utility value of the rule set developed. The theory of metagames is related to [mechanism design](https://en.wikipedia.org/wiki/Mechanism_design) theory.

The term [metagame analysis](https://en.wikipedia.org/wiki/Metagame_analysis) is also used to refer to a practical approach developed by Nigel Howard.[[25]](https://en.wikipedia.org/wiki/Game_theory#cite_note-FOOTNOTEHoward1971-25) whereby a situation is framed as a strategic game in which stakeholders try to realise their objectives by means of the options available to them. Subsequent developments have led to the formulation of [confrontation analysis](https://en.wikipedia.org/wiki/Confrontation_analysis).

### Pooling games

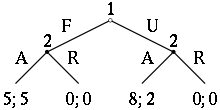
These are games prevailing over all forms of society. Pooling games are repeated plays with changing payoff table in general over an experienced path and their equilibrium strategies usually take a form of evolutionary social convention and economic convention. Pooling game theory emerges to formally recognize the interaction between optimal choice in one play and the emergence of forthcoming payoff table update path, identify the invariance existence and robustness, and predict variance over time. The theory is based upon topological transformation classification of payoff table update over time to predict variance and invariance, and is also within the jurisdiction of the computational law of reachable optimality for ordered system.[[26]](https://en.wikipedia.org/wiki/Game_theory#cite_note-26)

## Representation of games

The games studied in game theory are well-defined mathematical objects. To be fully defined, a game must specify the following elements: the [*players* of the game](https://en.wikipedia.org/wiki/Player_(game)), the *information*and *actions* available to each player at each decision point, and the [*payoffs*](https://en.wikipedia.org/wiki/Utility) for each outcome. (Eric Rasmusen refers to these four "essential elements" by the acronym "PAPI".)[[27]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r7-27) A game theorist typically uses these elements, along with a [solution concept](https://en.wikipedia.org/wiki/Solution_concept) of their choosing, to deduce a set of equilibrium [strategies](https://en.wikipedia.org/wiki/Strategy_(game_theory)) for each player such that, when these strategies are employed, no player can profit by unilaterally deviating from their strategy. These equilibrium strategies determine an [equilibrium](https://en.wikipedia.org/wiki/Economic_equilibrium) to the game—a stable state in which either one outcome occurs or a set of outcomes occur with known probability.

Most cooperative games are presented in the characteristic function form, while the extensive and the normal forms are used to define noncooperative games.

### Extensive form

[](https://en.wikipedia.org/wiki/File:Ultimatum_Game_Extensive_Form.svg)

An extensive form game

The extensive form can be used to formalize games with a time sequencing of moves. Games here are played on [trees](https://en.wikipedia.org/wiki/Tree_(graph_theory)) (as pictured here). Here each [vertex](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) (or node) represents a point of choice for a player. The player is specified by a number listed by the vertex. The lines out of the vertex represent a possible action for that player. The payoffs are specified at the bottom of the tree. The extensive form can be viewed as a multi-player generalization of a [decision tree](https://en.wikipedia.org/wiki/Decision_tree).[[28]](https://en.wikipedia.org/wiki/Game_theory#cite_note-FOOTNOTEFudenbergTirole199167-28) To solve any extensive form game, [backward induction](https://en.wikipedia.org/wiki/Backward_induction) must be used. It involves working backwards up the game tree to determine what a rational player would do at the last vertex of the tree, what the player with the previous move would do given that the player with the last move is rational, and so on until the first vertex of the tree is reached.

The game pictured consists of two players. The way this particular game is structured (i.e., with sequential decision making and perfect information), *Player 1* "moves" first by choosing either *F* or *U* (Fair or Unfair). Next in the sequence, *Player 2*, who has now seen *Player 1'*s move, chooses to play either *A* or *R*. Once *Player 2* has made his/ her choice, the game is considered finished and each player gets their respective payoff. Suppose that *Player 1*chooses *U* and then *Player 2* chooses *A*: *Player 1* then gets a payoff of "eight" (which in real-world terms can be interpreted in many ways, the simplest of which is in terms of money but could mean things such as eight days of vacation or eight countries conquered or even eight more opportunities to play the same game against other players) and *Player 2* gets a payoff of "two".

The extensive form can also capture simultaneous-move games and games with imperfect information. To represent it, either a dotted line connects different vertices to represent them as being part of the same information set (i.e. the players do not know at which point they are), or a closed line is drawn around them. (See example in the [imperfect information section](https://en.wikipedia.org/wiki/Game_theory#Perfect_information_and_imperfect_information).)

### Normal form

|  |  |  |
| --- | --- | --- |
|  | Player 2 chooses *Left* | Player 2 chooses *Right* |
| Player 1 chooses *Up* | **4**, **3** | **–1**, **–1** |
| Player 1 chooses *Down* | **0**, **0** | **3**, **4** |
| *Normal form or payoff matrix of a 2-player, 2-strategy game* | | |

The normal (or strategic form) game is usually represented by a [matrix](https://en.wikipedia.org/wiki/Matrix_(mathematics)) which shows the players, strategies, and payoffs (see the example to the right). More generally it can be represented by any function that associates a payoff for each player with every possible combination of actions. In the accompanying example there are two players; one chooses the row and the other chooses the column. Each player has two strategies, which are specified by the number of rows and the number of columns. The payoffs are provided in the interior. The first number is the payoff received by the row player (Player 1 in our example); the second is the payoff for the column player (Player 2 in our example). Suppose that Player 1 plays *Up* and that Player 2 plays *Left*. Then Player 1 gets a payoff of 4, and Player 2 gets 3.

When a game is presented in normal form, it is presumed that each player acts simultaneously or, at least, without knowing the actions of the other. If players have some information about the choices of other players, the game is usually presented in extensive form.

Every extensive-form game has an equivalent normal-form game, however the transformation to normal form may result in an exponential blowup in the size of the representation, making it computationally impractical.

**Prescriptive or normative analysis**[[edit](https://en.wikipedia.org/w/index.php?title=Game_theory&action=edit&section=23)]

|  |  |  |
| --- | --- | --- |
|  | Cooperate | Defect |
| Cooperate | -1, -1 | -10, 0 |
| Defect | 0, -10 | -5, -5 |
| *The*[*Prisoner's Dilemma*](https://en.wikipedia.org/wiki/Prisoner%27s_Dilemma) | | |

Some scholars, like [Leonard Savage](https://en.wikipedia.org/wiki/Leonard_Savage),[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] see game theory not as a predictive tool for the behavior of human beings, but as a suggestion for how people ought to behave. Since a strategy, corresponding to a [Nash equilibrium](https://en.wikipedia.org/wiki/Nash_equilibrium) of a game constitutes one's [best response](https://en.wikipedia.org/wiki/Best_response) to the actions of the other players – provided they are in (the same) Nash equilibrium – playing a strategy that is part of a Nash equilibrium seems appropriate. This normative use of game theory has also come under criticism.

**Economics and business**[[edit](https://en.wikipedia.org/w/index.php?title=Game_theory&action=edit&section=24)]

Game theory is a major method used in [mathematical economics](https://en.wikipedia.org/wiki/Mathematical_economics) and business for [modeling](https://en.wikipedia.org/wiki/Economic_model) competing behaviors of interacting [agents](https://en.wikipedia.org/wiki/Agent_(economics)).[[34]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r1-34)Applications include a wide array of economic phenomena and approaches, such as [auctions](https://en.wikipedia.org/wiki/Auction), [bargaining](https://en.wikipedia.org/wiki/Bargaining), [mergers & acquisitions](https://en.wikipedia.org/wiki/Mergers_%26_acquisitions) pricing,[[35]](https://en.wikipedia.org/wiki/Game_theory#cite_note-GT-A-E-00-35) [fair division](https://en.wikipedia.org/wiki/Fair_division), [duopolies](https://en.wikipedia.org/wiki/Duopoly), [oligopolies](https://en.wikipedia.org/wiki/Oligopoly), [social network](https://en.wikipedia.org/wiki/Social_network) formation, [agent-based computational economics](https://en.wikipedia.org/wiki/Agent-based_computational_economics),[[36]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r2-36) [general equilibrium](https://en.wikipedia.org/wiki/General_equilibrium), [mechanism design](https://en.wikipedia.org/wiki/Mechanism_design),[[37]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r3-37) and [voting systems](https://en.wikipedia.org/wiki/Voting_system);[[38]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r4-38) and across such broad areas as [experimental economics](https://en.wikipedia.org/wiki/Experimental_economics),[[39]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r5-39) [behavioral economics](https://en.wikipedia.org/wiki/Behavioral_game_theory),[[40]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r6-40) [information economics](https://en.wikipedia.org/wiki/Information_economics),[[27]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r7-27) [industrial organization](https://en.wikipedia.org/wiki/Industrial_organization),[[41]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r8-41) and [political economy](https://en.wikipedia.org/wiki/Political_economy).[[42]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r9-42)[[43]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r10-43)

This research usually focuses on particular sets of strategies known as ["solution concepts" or "equilibria"](https://en.wikipedia.org/wiki/Solution_concept). A common assumption is that players act rationally. In non-cooperative games, the most famous of these is the [Nash equilibrium](https://en.wikipedia.org/wiki/Nash_equilibrium). A set of strategies is a Nash equilibrium if each represents a best response to the other strategies. If all the players are playing the strategies in a Nash equilibrium, they have no unilateral incentive to deviate, since their strategy is the best they can do given what others are doing.[[44]](https://en.wikipedia.org/wiki/Game_theory#cite_note-GT-F-R-09-44)[[45]](https://en.wikipedia.org/wiki/Game_theory#cite_note-GT-F-R-10-45)

The payoffs of the game are generally taken to represent the [utility](https://en.wikipedia.org/wiki/Utility) of individual players.

A prototypical paper on game theory in economics begins by presenting a game that is an abstraction of a particular economic situation. One or more solution concepts are chosen, and the author demonstrates which strategy sets in the presented game are equilibria of the appropriate type. Naturally one might wonder to what use this information should be put. Economists and business professors suggest two primary uses (noted above): *descriptive* and [*prescriptive*](https://en.wikipedia.org/wiki/Decision_theory#Normative_and_descriptive_decision_theory).

**Biology**

|  |  |  |
| --- | --- | --- |
|  | Hawk | Dove |
| Hawk | 20, 20 | 80, 40 |
| Dove | 40, 80 | 60, 60 |
| *The*[*hawk-dove*](https://en.wikipedia.org/wiki/Chicken_(game)#Hawk-Dove)*game* | | |

Unlike those in economics, the payoffs for games in [biology](https://en.wikipedia.org/wiki/Biology) are often interpreted as corresponding to [fitness](https://en.wikipedia.org/wiki/Fitness_(biology)). In addition, the focus has been less on [equilibria](https://en.wikipedia.org/wiki/Solution_concept) that correspond to a notion of rationality and more on ones that would be maintained by [evolutionary](https://en.wikipedia.org/wiki/Evolution) forces. The best known equilibrium in biology is known as the [*evolutionarily stable strategy*](https://en.wikipedia.org/wiki/Evolutionarily_stable_strategy) (ESS), first introduced in ([Smith & Price 1973](https://en.wikipedia.org/wiki/Game_theory#CITEREFSmithPrice1973)). Although its initial motivation did not involve any of the mental requirements of the [Nash equilibrium](https://en.wikipedia.org/wiki/Nash_equilibrium), every ESS is a Nash equilibrium.

In biology, game theory has been used as a model to understand many different phenomena. It was first used to explain the evolution (and stability) of the approximate 1:1 [sex ratios](https://en.wikipedia.org/wiki/Sex_ratio). ([Fisher 1930](https://en.wikipedia.org/wiki/Game_theory#CITEREFFisher1930)) suggested that the 1:1 sex ratios are a result of evolutionary forces acting on individuals who could be seen as trying to maximize their number of grandchildren.

Additionally, biologists have used [evolutionary game theory](https://en.wikipedia.org/wiki/Evolutionary_game_theory) and the ESS to explain the emergence of [animal communication](https://en.wikipedia.org/wiki/Animal_communication).[[52]](https://en.wikipedia.org/wiki/Game_theory#cite_note-FOOTNOTEHarperMaynard_Smith2003-52) The analysis of [signaling games](https://en.wikipedia.org/wiki/Signaling_games) and [other communication games](https://en.wikipedia.org/wiki/Cheap_talk) has provided insight into the evolution of communication among animals. For example, the [mobbing behavior](https://en.wikipedia.org/wiki/Mobbing_behavior) of many species, in which a large number of prey animals attack a larger predator, seems to be an example of spontaneous emergent organization. Ants have also been shown to exhibit feed-forward behavior akin to fashion (see [Paul Ormerod](https://en.wikipedia.org/wiki/Paul_Ormerod)'s [*Butterfly Economics*](https://en.wikipedia.org/wiki/Butterfly_Economics)).

Biologists have used the [game of chicken](https://en.wikipedia.org/wiki/Chicken_(game)) to analyze fighting behavior and territoriality.[[53]](https://en.wikipedia.org/wiki/Game_theory#cite_note-53)

According to Maynard Smith, in the preface to *Evolution and the Theory of Games*, "paradoxically, it has turned out that game theory is more readily applied to biology than to the field of economic behaviour for which it was originally designed". Evolutionary game theory has been used to explain many seemingly incongruous phenomena in nature.[[54]](https://en.wikipedia.org/wiki/Game_theory#cite_note-stan-egt-54)

One such phenomenon is known as [biological altruism](https://en.wikipedia.org/wiki/Altruism_in_animals). This is a situation in which an organism appears to act in a way that benefits other organisms and is detrimental to itself. This is distinct from traditional notions of altruism because such actions are not conscious, but appear to be evolutionary adaptations to increase overall fitness. Examples can be found in species ranging from vampire bats that regurgitate blood they have obtained from a night's hunting and give it to group members who have failed to feed, to worker bees that care for the queen bee for their entire lives and never mate, to [vervet monkeys](https://en.wikipedia.org/wiki/Vervet_monkey) that warn group members of a predator's approach, even when it endangers that individual's chance of survival.[[55]](https://en.wikipedia.org/wiki/Game_theory#cite_note-qudzyh-55) All of these actions increase the overall fitness of a group, but occur at a cost to the individual.

Evolutionary game theory explains this altruism with the idea of [kin selection](https://en.wikipedia.org/wiki/Kin_selection). Altruists discriminate between the individuals they help and favor relatives. [Hamilton's rule](https://en.wikipedia.org/wiki/Hamilton%27s_rule) explains the evolutionary rationale behind this selection with the equation c<b\*r where the cost (c) to the altruist must be less than the benefit (b) to the recipient multiplied by the coefficient of relatedness (r). The more closely related two organisms are causes the incidences of altruism to increase because they share many of the same alleles. This means that the altruistic individual, by ensuring that the alleles of its close relative are passed on, (through survival of its offspring) can forgo the option of having offspring itself because the same number of alleles are passed on. Helping a sibling for example (in diploid animals), has a coefficient of ½, because (on average) an individual shares ½ of the alleles in its sibling's offspring. Ensuring that enough of a sibling’s offspring survive to adulthood precludes the necessity of the altruistic individual producing offspring.[[55]](https://en.wikipedia.org/wiki/Game_theory#cite_note-qudzyh-55) The coefficient values depend heavily on the scope of the playing field; for example if the choice of whom to favor includes all genetic living things, not just all relatives, we assume the discrepancy between all humans only accounts for approximately 1% of the diversity in the playing field, a co-efficient that was ½ in the smaller field becomes 0.995. Similarly if it is considered that information other than that of a genetic nature (e.g. epigenetics, religion, science, etc.) persisted through time the playing field becomes larger still, and the discrepancies smaller.

### Computer science and logic

Game theory has come to play an increasingly important role in [logic](https://en.wikipedia.org/wiki/Logic) and in [computer science](https://en.wikipedia.org/wiki/Computer_science). Several logical theories have a basis in [game semantics](https://en.wikipedia.org/wiki/Game_semantics). In addition, computer scientists have used games to model [interactive computations](https://en.wikipedia.org/wiki/Interactive_computation). Also, game theory provides a theoretical basis to the field of [multi-agent systems](https://en.wikipedia.org/wiki/Multi-agent_system).

Separately, game theory has played a role in [online algorithms](https://en.wikipedia.org/wiki/Online_algorithm); in particular, the [k-server problem](https://en.wikipedia.org/wiki/K-server_problem), which has in the past been referred to as *games with moving costs* and *request-answer games*.[[56]](https://en.wikipedia.org/wiki/Game_theory#cite_note-56) [Yao's principle](https://en.wikipedia.org/wiki/Yao%27s_principle) is a game-theoretic technique for proving [lower bounds](https://en.wikipedia.org/wiki/Upper_and_lower_bounds) on the [computational complexity](https://en.wikipedia.org/wiki/Analysis_of_algorithms) of [randomized algorithms](https://en.wikipedia.org/wiki/Randomized_algorithm), especially online algorithms.

The emergence of the internet has motivated the development of algorithms for finding equilibria in games, markets, computational auctions, peer-to-peer systems, and security and information markets. [Algorithmic game theory](https://en.wikipedia.org/wiki/Algorithmic_game_theory)[[57]](https://en.wikipedia.org/wiki/Game_theory#cite_note-algorithmic-game-57) and within it [algorithmic mechanism design](https://en.wikipedia.org/wiki/Algorithmic_mechanism_design)[[58]](https://en.wikipedia.org/wiki/Game_theory#cite_note-r11-58) combine computational [algorithm design](https://en.wikipedia.org/wiki/Algorithm_design) and analysis of [complex systems](https://en.wikipedia.org/wiki/Complex_system) with economic theory.

# 10. Networks

1. The application layer is the only part of a communications process that a user sees, and even then, the user doesn't see most of the work that the application does to prepare a message for sending over a network. The layer converts a message's data from human-readable form into bits and attaches a header identifying the sending and receiving computers.

2. The presentation layer ensures that the message is transmitted in a language that the receiving computer can interpret (often ASCII). This layer translates the language, if necessary, and then compresses and perhaps encrypts the data. It adds another header specifying the is language as well as the compression and encryption schemes.

3. The session layer opens communications and has the job of keeping straight the communications among all nodes on the network.

It sets boundaries (called bracketing) for the beginning and end of the message, and establishes whether the messages will be sent half-duplex, with each computer taking turns sending and receiving, or full-duplex, with both computers sending and receiving at the same time. The details of these decisions are placed into a session header.

4. The transport layer protects the data being sent. It subdivides the data into segments, creates checksum tests - mathematical sums based on the contents of data - that can be used later to determine if the data was scrambled. It can also make backup copies of the data. The transport header identifies each segment's checksum and its position in the message.

5. The network layer selects a route for the message. It forms data into packets, counts them, and adds a header containing the sequence of packets and the address of the receiving computer.

6. The data-link layer supervises the transmission. It confirms the checksum, then addresses and duplicates the packets. This layer keeps a copy of each packet until it receives confirmation from the next point along the route that the packet has arrived undamaged.

7. The physical layer encodes the packets into the medium that will carry them - such as an analogue signal, if the message is going across a telephone line - and sends the packets along that medium.

8. An intermediate node calculates and verifies the checksum for each packet. It may also reroute the message to avoid congestion on the network.

9. At the receiving node, the layered process that sent the message on its way is reversed. The physical layer reconverts the message into bits. The data-link layer recalculates the checksum, confirms arrival, and logs in the packets. The network layer recounts incoming packets for security and billing purposes. The transport layer recalculates the checksum and reassembles the message segments. The session layer holds the parts of the message until the message is complete and sends it to the next layer. The presentation layer expands and decrypts the message. The application layer converts the bits into readable characters, and directs the data to the correct application.

Computer networking is the engineering discipline concerned with communication between computer systems or devices. Networking, routers, routing protocols, and networking over the public Internet have their specifications defined in documents called RFCs. Computer networking is sometimes considered to be a sub-discipline of telecommunications, computer science and information technology. Computer networks rely heavily upon the theoretical and practical application of these scientific and engineering disciplines.

A computer network is any set of computers or devices connected to each other with the ability to exchange data.

Networks are classified according to different criteria (e.g., geographical area, architecture, topology, network protocol they use). According to the geographical area networks are classified into PAN LAN MAN and WAN. (PAN) Personal Area Network typically includes a laptop, a mobile phone or a PDA. Local area network (LAN) is a network that spans a relatively small space (e.g., a building) and provides services to a small number of people. The LAN's in question would usually be connected via "backbone" lines. The first LAN was invented by a Law Doctor. Metropolitan Area Network (MAN) is a network that is too large for even the largest of LAN's but is not on the scale of a WAN. It also integrates two or more LAN networks over a specific geographical area (usually a city). Wide Area Network (WAN) is a network where a wide variety of resources are deployed across a large domestic area or internationally, therefore, WANs cover a country or a continent. An example of this is a PSTN (Public Switched Telephone Network) or a multinational business that uses a WAN to interconnect their offices in different countries. The largest and best example of a WAN is the Internet, which is a network that consists of many smaller networks.

Depending on the number of people that use a Local Area Network, a peer-to-peer or client-server method of networking may be used. A peer-to-peer network is where each client shares their resources with other workstations in the network. Examples of peer-to-peer networks are small office networks where resource use is minimal. A client-server network is where every client is connected to the server and each other. Client-server networks use servers in different capacities. These can be classified into two types: single-service servers, where the server performs one task such as file server, print server, etc.; and multiservice servers which can not only perform as a file servers and print servers, but they also conduct calculations and use these to provide information to clients.

Network topology is the schematic description of a network arrangement, connecting various nodes (sender and receiver) through lines of connection. There are bus, ring, star, mesh, thee and hybrid topologies. Bus topology is a network type in where every computer and network device is connected to single cable. Ring topology is a network type in where computers forms a ring as each computer is connected to another computer, with the last one connected to the first. Exactly two neighbours for each device. In star type of topology all the computers are connected to a single hub through a cable. Mesh topology is a point-to-point connection to other nodes or devices. There are partial and full mesh topologies. Tree topology has a root node and all other nodes are connected to it forming a hierarchy. It is also called hierarchical topology. Hybrid topology is type of topology which is a mixture of two or more topologies.