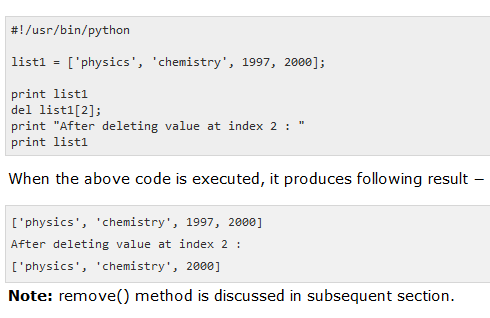
**Python**

**Control Structures**

* For, while loops
* Nested Loops
* Break
* Continue
* Pass
  + The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

**Lists**

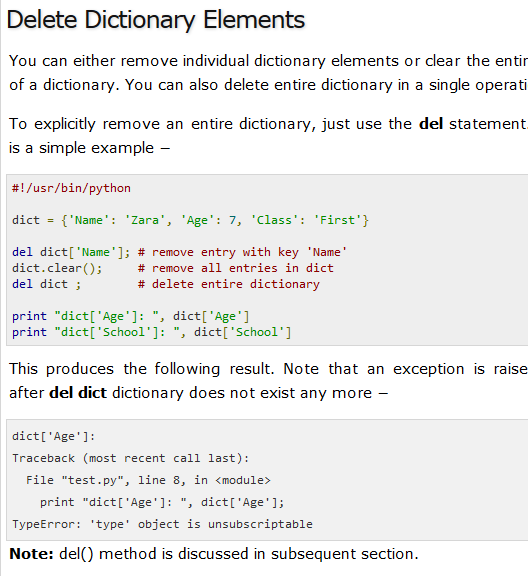
* Like and array



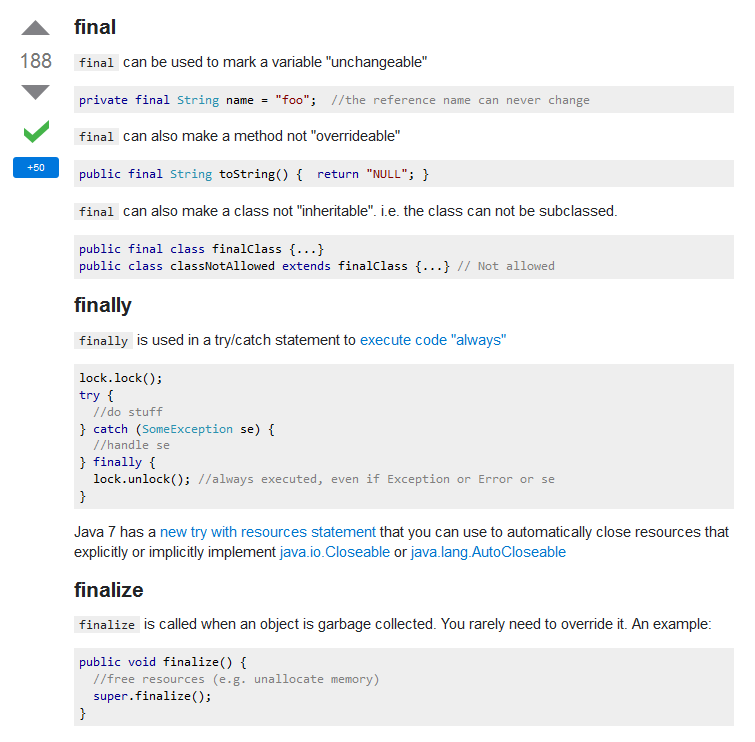
**Tuple**

* A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

**Dictionary**



**Java**



**Biology**

**Genome**

* A genome is an organism’s complete set of genetic instructions. Each genome contains all of the information needed to build that organism and allow it to grow and develop.

**Prokaryotes**

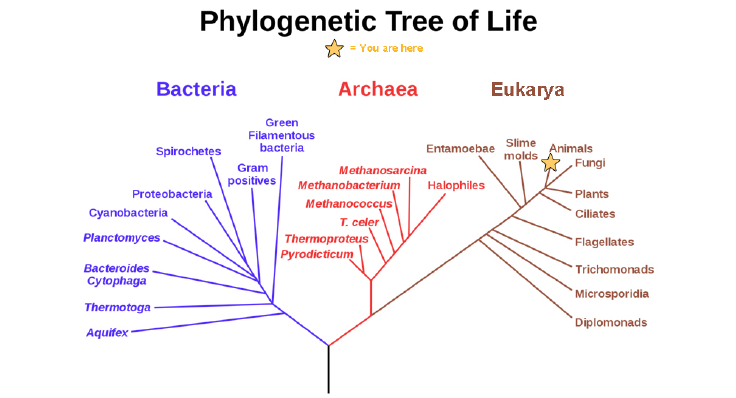
* are single-celled or colonial organisms that do not have membrane bound nuclei

**eukaryotes**

* do have membrane-bound organelles and a membrane-bound nucleus.

**A phenotype**

* (from Greek phainein, meaning "to show", and typos, meaning "type") is the composite of an organism's observable characteristics or traits, such as its morphology, development, biochemical or physiological properties, phenology, behavior, and products of behavior (such as a bird's nest).



**A phylogenetic tree is**

* a diagram showing the evolutionary relationships among biological species based on similarities and differences in genetic
* or physical traits or both. A phylogenetic tree is composed of nodes and branches.

**Atomic Number-** Protons

**Mass number-** Together, the number of protons and the number of neutrons determine an element’s

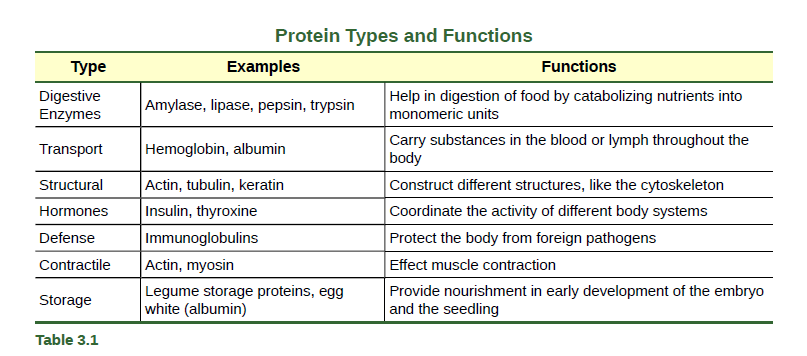
**Ionic bonds** are formed between ions with opposite charges

**Covalent bonds**

* Another way the octet rule can be satisfied is by the sharing of electrons between atoms to form covalent bonds. These bonds are stronger and much more common than ionic bonds in the molecules of living organisms.
* Covalent bonds are commonly found in carbon-based organic molecules, such as our DNA and proteins. Covalent bonds are also found in inorganic molecules like H2O, CO2, and O2. One, two, or three pairs of electrons may be shared, making single, double, and triple bonds, respectively.
* The more covalent bonds between two atoms, the stronger their connection. Thus, triple bonds are the strongest.

**Proteins**

are one of the most abundant organic molecules in living systems and have the most diverse range of functions of all macromolecules. Proteins may be structural, regulatory, contractile, or protective; they may serve in transport, storage, or membranes; or they may be toxins or enzymes. Each cell in a living system may contain thousands of proteins, each with a unique function. Their structures, like their functions, vary greatly. They are all, however, polymers of amino acids, arranged in a linear sequence.



**Amino acids** are the monomers that make up proteins. Each amino acid has the same fundamental structure, which consists of a central carbon atom, also known as the alpha (α) carbon, bonded to an amino group (NH2), a carboxyl group (COOH), and to a hydrogen atom. Every amino acid also has another atom or group of atoms bonded to the central atom known as the R group.

**DNA and RNA**

The two main types of nucleic acids are deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). DNA is the genetic material found in all living organisms, ranging from single-celled bacteria to multicellular mammals. It is found in the nucleus of eukaryotes and in the organelles, chloroplasts, and mitochondria. In prokaryotes, the DNA is not enclosed in a membranous envelope.

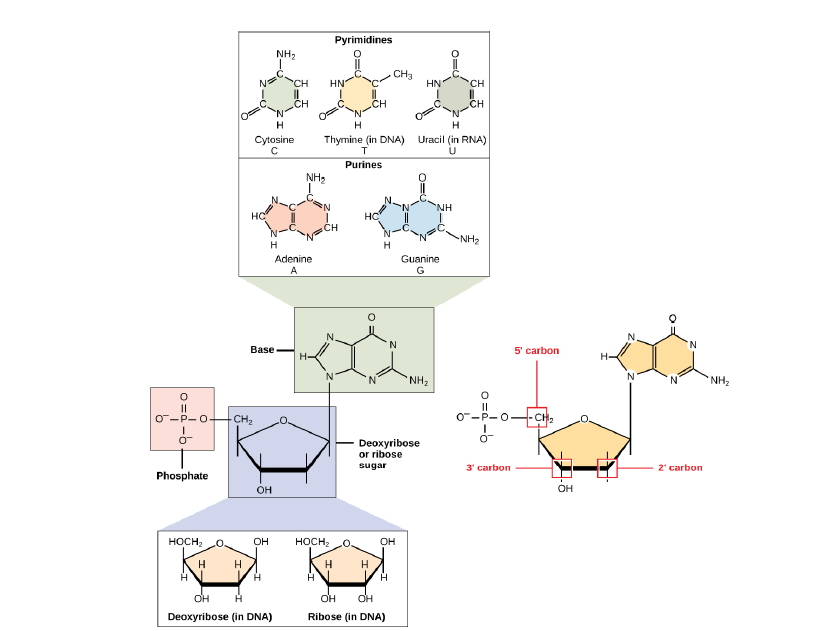
The entire genetic content of a cell is known as its genome, and the study of genomes is genomics. In eukaryotic cells but not in prokaryotes, DNA forms a complex with histone proteins to form chromatin, the substance of eukaryotic chromosomes.

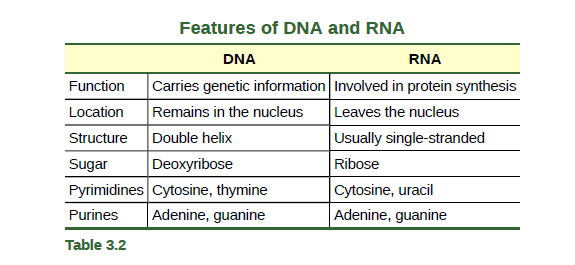
A chromosome may contain tens of thousands of genes. Many genes contain the information to make protein products; other genes code for RNA products. DNA controls all of the cellular activities by turning the genes “on” or “off.”

The other type of nucleic acid, RNA, is mostly involved in protein synthesis. The DNA molecules never leave the nucleus but instead use an intermediary to communicate with the rest of the cell. This intermediary is the messenger RNA (mRNA).

Other types of RNA—like rRNA, tRNA, and microRNA—are involved in protein synthesis and its regulation.

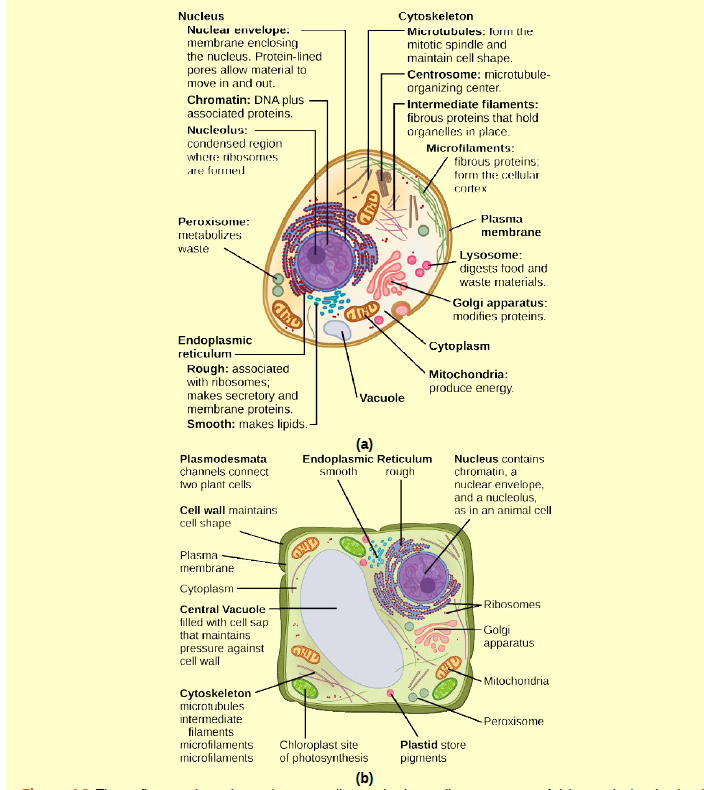
DNA and RNA are made up of monomers known as nucleotides. The nucleotides combine with each other to form a polynucleotide, DNA or RNA. Each nucleotide is made up of three components: a nitrogenous base, a pentose (fivecarbon) sugar, and a phosphate group (Figure 3.31). Each nitrogenous base in a nucleotide is attached to a sugar molecule, which is attached to one or more phosphate groups.





**Components of Prokaryotic Cells**

All cells share four common components: 1) a plasma membrane, an outer covering that separates the cell’s interior from its surrounding environment; 2) cytoplasm, consisting of a jelly-like cytosol within the cell in which other cellular components are found; 3) DNA, the genetic material of the cell; and 4) ribosomes, which synthesize proteins. However, prokaryotes differ from eukaryotic cells in several ways.



**UNIX**

**Basic UNIX commands**

* Note: not all of these are actually part of UNIX itself, and you may not find them on all UNIX machines. But they can all be used on turing in essentially the same way, by typing the command and hitting return. Note that some of these commands are different on non-Solaris machines - see SunOS differences.
* If you've made a typo, the easiest thing to do is hit CTRL-u to cancel the whole line. But you can also edit the command line (see the guide to More UNIX).
* UNIX is case-sensitive.

**Files**

* ls --- lists your files
* ls -l --- lists your files in 'long format', which contains lots of useful information, e.g. the exact size of the file, who owns the file and who has the right to look at it, and when it was last modified.
* ls -a --- lists all files, including the ones whose filenames begin in a dot, which you do not always want to see.
* There are many more options, for example to list files by size, by date, recursively etc.
* more filename --- shows the first part of a file, just as much as will fit on one screen. Just hit the space bar to see more or q to quit. You can use /pattern to search for a pattern.
* emacs filename --- is an editor that lets you create and edit a file. See the emacs page.
* mv filename1 filename2 --- moves a file (i.e. gives it a different name, or moves it into a different directory (see below)
* cp filename1 filename2 --- copies a file
* rm filename --- removes a file. It is wise to use the option rm -i, which will ask you for confirmation before actually deleting anything. You can make this your default by making an alias in your .cshrc file.
* diff filename1 filename2 --- compares files, and shows where they differ
* wc filename --- tells you how many lines, words, and characters there are in a file
* chmod options filename --- lets you change the read, write, and execute permissions on your files. The default is that only you can look at them and change them, but you may sometimes want to change these permissions. For example, chmod o+r filename will make the file readable for everyone, and chmod o-r filename will make it unreadable for others again. Note that for someone to be able to actually look at the file the directories it is in need to be at least executable. See help protection for more details.

**File Compression**

* gzip filename --- compresses files, so that they take up much less space. Usually text files compress to about half their original size, but it depends very much on the size of the file and the nature of the contents. There are other tools for this purpose, too (e.g. compress), but gzip usually gives the highest compression rate. Gzip produces files with the ending '.gz' appended to the original filename.
* gunzip filename --- uncompresses files compressed by gzip.
* gzcat filename --- lets you look at a gzipped file without actually having to gunzip it (same as gunzip -c). You can even print it directly, using gzcat filename | lpr

**printing**

* lpr filename --- print. Use the -P option to specify the printer name if you want to use a printer other than your default printer. For example, if you want to print double-sided, use 'lpr -Pvalkyr-d', or if you're at CSLI, you may want to use 'lpr -Pcord115-d'. See 'help printers' for more information about printers and their locations.
* lpq --- check out the printer queue, e.g. to get the number needed for removal, or to see how many other files will be printed before yours will come out
* lprm jobnumber --- remove something from the printer queue. You can find the job number by using lpq. Theoretically you also have to specify a printer name, but this isn't necessary as long as you use your default printer in the department.
* genscript --- converts plain text files into postscript for printing, and gives you some options for formatting. Consider making an alias like alias ecop 'genscript -2 -r \!\* | lpr -h -Pvalkyr' to print two pages on one piece of paper.
* dvips filename --- print .dvi files (i.e. files produced by LaTeX). You can use dviselect to print only selected pages. See the LaTeX page for more information about how to save paper when printing drafts.

**Directories**

* Directories, like folders on a Macintosh, are used to group files together in a hierarchical structure.
* mkdir dirname --- make a new directory
* cd dirname --- change directory. You basically 'go' to another directory, and you will see the files in that directory when you do 'ls'. You always start out in your 'home directory', and you can get back there by typing 'cd' without arguments. 'cd ..' will get you one level up from your current position. You don't have to walk along step by step - you can make big leaps or avoid walking around by specifying pathnames.
* pwd --- tells you where you currently are.

**Finding things**

* ff --- find files anywhere on the system. This can be extremely useful if you've forgotten in which directory you put a file, but do remember the name. In fact, if you use ff -p you don't even need the full name, just the beginning. This can also be useful for finding other things on the system, e.g. documentation.
* grep string filename(s) --- looks for the string in the files. This can be useful a lot of purposes, e.g. finding the right file among many, figuring out which is the right version of something, and even doing serious corpus work. grep comes in several varieties (grep, egrep, and fgrep) and has a lot of very flexible options. Check out the man pages if this sounds good to you.

**About other people**

* w --- tells you who's logged in, and what they're doing. Especially useful: the 'idle' part. This allows you to see whether they're actually sitting there typing away at their keyboards right at the moment.
* who --- tells you who's logged on, and where they're coming from. Useful if you're looking for someone who's actually physically in the same building as you, or in some other particular location.
* finger username --- gives you lots of information about that user, e.g. when they last read their mail and whether they're logged in. Often people put other practical information, such as phone numbers and addresses, in a file called .plan. This information is also displayed by 'finger'.
* last -1 username --- tells you when the user last logged on and off and from where. Without any options, last will give you a list of everyone's logins.
* talk username --- lets you have a (typed) conversation with another user
* write username --- lets you exchange one-line messages with another user
* elm --- lets you send e-mail messages to people around the world (and, of course, read them). It's not the only mailer you can use, but the one we recommend. See the elm page, and find out about the departmental mailing lists (which you can also find in /user/linguistics/helpfile).

**About your (electronic) self**

* whoami --- returns your username. Sounds useless, but isn't. You may need to find out who it is who forgot to log out somewhere, and make sure \*you\* have logged out.
* **finger & .plan files**
* of course you can finger yourself, too. That can be useful e.g. as a quick check whether you got new mail. Try to create a useful .plan file soon. Look at other people's .plan files for ideas. The file needs to be readable for everyone in order to be visible through 'finger'. Do 'chmod a+r .plan' if necessary. You should realize that this information is accessible from anywhere in the world, not just to other people on turing.
* passwd --- lets you change your password, which you should do regularly (at least once a year). See the LRB guide and/or look at help password.
* ps -u yourusername --- lists your processes. Contains lots of information about them, including the process ID, which you need if you have to kill a process. Normally, when you have been kicked out of a dialin session or have otherwise managed to get yourself disconnected abruptly, this list will contain the processes you need to kill. Those may include the shell (tcsh or whatever you're using), and anything you were running, for example emacs or elm. Be careful not to kill your current shell - the one with the number closer to the one of the ps command you're currently running. But if it happens, don't panic. Just try again :) If you're using an X-display you may have to kill some X processes before you can start them again. These will show only when you use ps -efl, because they're root processes.
* kill PID --- kills (ends) the processes with the ID you gave. This works only for your own processes, of course. Get the ID by using ps. If the process doesn't 'die' properly, use the option -9. But attempt without that option first, because it doesn't give the process a chance to finish possibly important business before dying. You may need to kill processes for example if your modem connection was interrupted and you didn't get logged out properly, which sometimes happens.
* quota -v --- show what your disk quota is (i.e. how much space you have to store files), how much you're actually using, and in case you've exceeded your quota (which you'll be given an automatic warning about by the system) how much time you have left to sort them out (by deleting or gzipping some, or moving them to your own computer).
* du filename --- shows the disk usage of the files and directories in filename (without argument the current directory is used). du -s gives only a total.
* last yourusername --- lists your last logins. Can be a useful memory aid for when you were where, how long you've been working for, and keeping track of your phonebill if you're making a non-local phonecall for dialling in.

**Connecting to the outside world**

* nn --- allows you to read news. It will first let you read the news local to turing, and then the remote news. If you want to read only the local or remote news, you can use nnl or nnr, respectively. To learn more about nn type nn, then \tty{:man}, then \tty{=.\*}, then \tty{Z}, then hit the space bar to step through the manual. Or look at the man page. Or check out the hypertext nn FAQ - probably the easiest and most fun way to go.
* rlogin hostname --- lets you connect to a remote host
* telnet hostname --- also lets you connect to a remote host. Use rlogin whenever possible.
* ftp hostname --- lets you download files from a remote host which is set up as an ftp-server. This is a common method for exchanging academic papers and drafts. If you need to make a paper of yours available in this way, you can (temporarily) put a copy in /user/ftp/pub/TMP. For more permanent solutions, ask Emma. The most important commands within ftp are get for getting files from the remote machine, and put for putting them there (mget and mput let you specify more than one file at once). Sounds straightforward, but be sure not to confuse the two, especially when your physical location doesn't correspond to the direction of the ftp connection you're making. ftp just overwrites files with the same filename. If you're transferring anything other than ASCII text, use binary mode.
* lynx --- lets you browse the web from an ordinary terminal. Of course you can see only the text, not the pictures. You can type any URL as an argument to the G command. When you're doing this from any Stanford host you can leave out the .stanford.edu part of the URL when connecting to Stanford URLs. Type H at any time to learn more about lynx, and Q to exit.

**Data Structures**

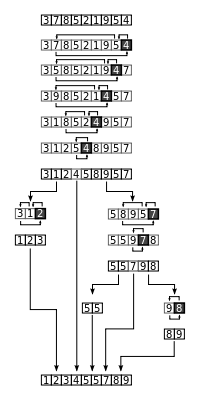
**Bubble Sort**

* Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list. Although the algorithm is simple, it is too slow and impractical for most problems even when compared to insertion sort.[1] It can be practical if the input is usually in sorted order but may occasionally have some out-of-order elements nearly in position.
* [*О*](https://en.wikipedia.org/wiki/Big_o_notation)(*n*2),
* When analyzing algorithms two questions are important: Is the algorithm correct, and how fast does it run.

**Binary Sort**

* On sorted list keep cutting in half

**Quick Sort**



**Abstract Data Types**

* The study of data structures is concerned largely with the need to maintain collections of values. These are sometimes termed containers. Even without discussing how these collections can be implemented, a number of different types of containers can be identified purely by their purpose or behavior. This type of description is termed an abstract data type.
* Types
  + Stack- Like Stack of plates
  + Bag- A bag is the simplest type of abstraction. A good metaphor is a bag of marbles. Operations on a bag include adding a value to the collection, asking if a specific value is or is not part of the collection, and removing a value from the collection.
  + A set
    - is an extension of a bag. In addition to bag operations the set makes the restriction that no element may appear more than once, and also definesseveral functions that work with entire sets. An example would be set intersection, which constructs a new set consisting of values that appear in two argument sets. A venn diagram is a good metaphor for this type of collection.
  + A queue
    - Removes values in exactly the same order that they were inserted. This is termed FIFO order (first-in, first-out). A queue of people waiting in line to enter a theater is a useful metaphor.
  + A map, or dictionary
    - maintains pairs of elements. Each key is matched to a corresponding value. They keys must be unique. A good metaphor is a dictionary of word/definition pairs.

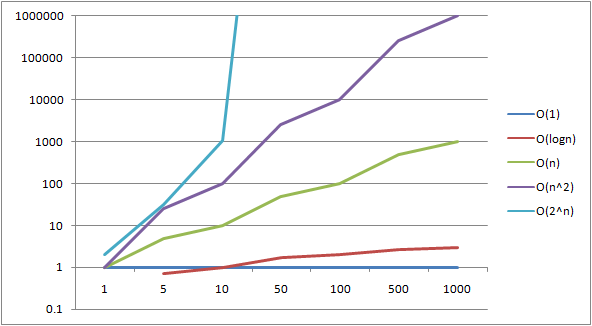
**Angular**

**React JS**

**Big O**

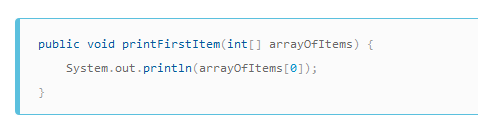
**Big O notation**

* Is used in Computer Science to describe the performance or complexity of an algorithm. Big O specifically describes the worst-case scenario, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

s

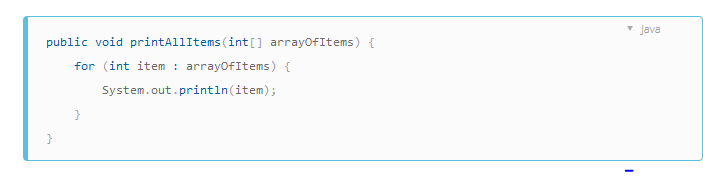
**O(1)**

* This function runs in O(1) time (or "constant time") relative to its input. The input array could be 1 item or 1,000 items, but this function would still just require one "step."



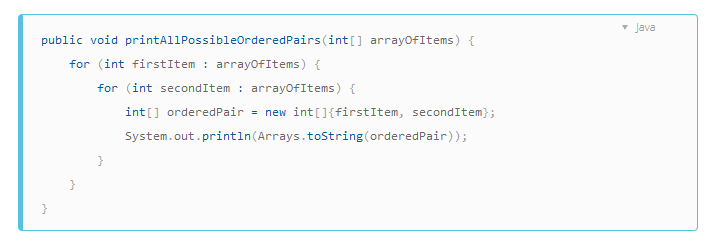
**O(n)**

* This function runs in O(n) time (or "linear time"), where nnn is the number of items in the array. If the array has 10 items, we have to print 10 times. If it has 1,000 items, we have to print 1,000 times.



**O(n^2)**

* Here we're nesting two loops. If our array has nnn items, our outer loop runs nnn times and our inner loop runs nnn times for each iteration of the outer loop, giving us n2n^2n​2​​ total prints. Thus this function runs in O(n2)O(n^2)O(n​2​​) time (or "quadratic time"). If the array has 10 items, we have to print 100 times. If it has 1,000 items, we have to print 1,000,000 times.
* O(N^2) represents an algorithm whose performance is directly proportional to the square of the size of the input data set. This is common with algorithms that involve nested iterations over the data set. Deeper nested iterations will result in O(N^3), O(N^4) etc.



**O(2N)**

* O(2N) denotes an algorithm whose growth doubles with each additon to the input data set. The growth curve of an O(2N) function is exponential - starting off very shallow, then rising meteorically. An example of an O(2N) function is the recursive calculation of Fibonacci numbers:

**O(log N).**

* Logarithms are slightly trickier to explain so I'll use a common example:
* Binary search is a technique used to search sorted data sets. It works by selecting the middle element of the data set, essentially the median, and compares it against a target value. If the values match it will return success. If the target value is higher than the value of the probe element it will take the upper half of the data set and perform the same operation against it. Likewise, if the target value is lower than the value of the probe element it will perform the operation against the lower half. It will continue to halve the data set with each iteration until the value has been found or until it can no longer split the data set.
* This type of algorithm is described as O(log N). The iterative halving of data sets described in the binary search example produces a growth curve that peaks at the beginning and slowly flattens out as the size of the data sets increase e.g. an input data set containing 10 items takes one second to complete, a data set containing 100 items takes two seconds, and a data set containing 1000 items will take three seconds. Doubling the size of the input data set has little effect on its growth as after a single iteration of the algorithm the data set will be halved and therefore on a par with an input data set half the size. This makes algorithms like binary search extremely efficient when dealing with large data sets.

**Example 2**

We can expand the phone book example to compare other kinds of operations and their running time. We will assume our phone book has businesses (the "Yellow Pages") which have unique names and people (the "White Pages") which may not have unique names. A phone number is assigned to at most one person or business. We will also assume that it takes constant time to flip to a specific page.

Here are the running times of some operations we might perform on the phone book, from best to worst:

* O(1) (worst case): Given the page that a business's name is on and the business name, find the phone number.
* O(1) (average case): Given the page that a person's name is on and their name, find the phone number.
* O(log n): Given a person's name, find the phone number by picking a random point about halfway through the part of the book you haven't searched yet, then checking to see whether the person's name is at that point. Then repeat the process about halfway through the part of the book where the person's name lies. (This is a binary search for a person's name.)
* O(n): Find all people whose phone numbers contain the digit "5".
* O(n): Given a phone number, find the person or business with that number.
* O(n log n): There was a mix-up at the printer's office, and our phone book had all its pages inserted in a random order. Fix the ordering so that it's correct by looking at the first name on each page and then putting that page in the appropriate spot in a new, empty phone book.

For the below examples, we're now at the printer's office. Phone books are waiting to be mailed to each resident or business, and there's a sticker on each phone book identifying where it should be mailed to. Every person or business gets one phone book.

* O(n log n): We want to personalize the phone book, so we're going to find each person or business's name in their designated copy, then circle their name in the book and write a short thank-you note for their patronage.
* O(n2): A mistake occurred at the office, and every entry in each of the phone books has an extra "0" at the end of the phone number. Take some white-out and remove each zero.
* O(n · n!): We're ready to load the phonebooks onto the shipping dock. Unfortunately, the robot that was supposed to load the books has gone haywire: it's putting the books onto the truck in a random order! Even worse, it loads all the books onto the truck, then checks to see if they're in the right order, and if not, it unloads them and starts over. (This is the dreaded bogo sort.)
* O(nn): You fix the robot so that it's loading things correctly. The next day, one of your co-workers plays a prank on you and wires the loading dock robot to the automated printing systems. Every time the robot goes to load an original book, the factory printer makes a duplicate run of all the phonebooks! Fortunately, the robot's bug-detection systems are sophisticated enough that the robot doesn't try printing even more copies when it encounters a duplicate book for loading, but it still has to load every original and duplicate book that's been printed.

**Research**

Translational research

* Applies findings from basic science to enhance human health and well-being. In a medical research context, it aims to "translate" findings in fundamental research into medical practice and meaningful health outcomes.

**Questions**

Translational research

**Tips**

1. Calm down.

First of all, the most important thing to do is stay calm. If you start freaking out, your body will begin reacting physiologically. For example, your blood pressure will start rising, and your heart may race. Once you start a stress response, you won't be thinking clearly, and you may throw out answers without thinking. Take deep breaths, and tell yourself that it's OK to not know the answer to the question. You'll just have to work through it; there's nothing you can do to change things, but you need to stay calm to find the right answer.

2. Don't say, "I don't know," off the bat. And don't make stuff up.

You should not tell the interviewer you don't know the answer without mulling it over. Then again, be careful not to make stuff up, because your interviewer can see right through that.

3. Ask questions.

Maybe it's the question you don't understand. Ask your interviewer to clarify what she said. Go deeper into the question to see if you can get more details that will help you figure it out.

4. Tell your interviewer what you do know.

If you do have some knowledge of the question, then take the time to tell your interviewer what you do know of the situation. Saying everything out loud can start you on the process of figuring out the problem.

5. Tell them how you would find the answer.

Even if you don't know what the answer is, you can tell the interviewer the steps you would take to figure out the problem. Interviewers ask you hard questions because they want to see what your thought process is. Sometimes, the thought process may be more important than the actual answer. They want to see that you can take initiative and have the resources to come up with a solution on your own, instead of needing someone to hold your hand through problems. While you're trying to find the solution, you can admit to not knowing certain parts; this way, you come off as being honest, and the hiring manager will know you are not trying to fake it. For example, if you need to calculate something and you're not good at math, you can respond with "I can't do the calculations off the top of my head, but I think these calculations will give me the answer. And what I can do is use a calculator to find that answer." Showing a little honesty shows vulnerability and transparency. It also makes you more likable.

6. Know the right time to come clean.

Although we mentioned not admitting to the interviewer that you don't know the answer, there is an exception to this rule. If the answer is something that you will only know through memorization, such as a definition of the word, then it's probably best to admit that you don't know the answer, as it may be impossible to figure it out independently. Here's what you can tell the interviewer: "It's a good question, but I'm sorry, I don't have the answer off the top of my head. I will be sure to follow up with the answer after the interview."

7. Send a follow-up email.

The follow-up email for an interview could become your second chance. Try to talk about the answer you were stumped on, but be smooth when you're talking about it. And make sure you're only naming the mistakes your interviewer caught and not drawing attention to the ones she did not catch. Don't say something like "I'm sorry I did not know the answer to that question." Instead, tell her that after more time and thought, you managed to come up with a couple