



**AP COMPUTER SCIENCE PRINCIPLES: THE
ULTIMATE GUIDE TO GET A 5
-Technicallyneverhere (Reddit) !!-**



Using this guide: by ME!!

- Knowing all the concepts are obviously important but knowing all the highlights that are in red is SUPER DUPER IMPORTANT OR you're cooked.
- I recommend looking through your code and UNDERSTANDING what each part does even if it's ChatGPTed.
- FOCUS ON BIG IDEA 3!!!
- Do tons of practice and locate your weak points!
- And...even if you don't get a 5, you're a 5 in my eyes 🌟
- I just studied all of this before my exam and got a 4!!

BIG IDEA 1: CREATIVE DEVELOPMENT

13%

Programming is a collaborative and creative process that brings ideas to life through the development of software

A **computing innovation** uses a computer program to take in data, transform data and output data

Collaboration can occur in the planning, testing or designing step of the process!!
Benefits of collaboration include:

- + ability to exchange ideas and discuss different solutions to different problems
- + Multiple perspectives, allow for improvements in software
- + Addresses misunderstanding and clarifies misconception
- + Development of thinking skills
- + Increased student responsibility
- + Eliminates bias

Pair programming is when two programmers develop software side-by-side at one computer on the same algorithm. 🙌

User interface: the inputs and outputs that allow a user to interact with a piece of software

It is important for your element to have meaningful names.

Camel case: stopButton

Input: data that is sent to a computer for processing into a device

Output: data sent from a program to a device

Program is a collection of instructions that a computing device executes

A **code segments** are smaller collection of statements that are a part of a program

Program event refers to an action or occurrence that takes place within a computer program, such as a button click, mouse movement, or keyboard input. It is used to trigger specific actions or behaviors in the program.

Event driven program: a type of program that respond to events triggered by user actions, system events or other sources

Sequential program: happens in order

THE DEVELOPMENT PROCESSES

Iterative development process: programmers develop working prototypes of their programs and go back through the stages of their development method

Incremental development process: programmers break the program they are working on into smaller pieces and make sure that each piece works before adding it to the whole

Program Documentation

A description of how something in your program works

Benefits:

- + crucial for understanding how every part of program works
- + fosters collaboration

Most common form of program documentation: **Comments** which are directly written into the program itself

Ex:

```
//This on event allows so that clicking this button will bring it to the next screen
```

Library: a collection of functions that can be used in different programs

- how each function works
- A complete list of parameter
- What (if anything) is returned

This is known as **Application Program Interface (API)**

ERRORS (KNOW ALL OF THEM OR ELSE)

Syntax Errors: A syntax error occurs when the spelling and/or punctuation rules of the programming language aren't followed. For example, forgetting to close a set of parentheses or spelling a variable wrong, could cause your entire program to crash. A syntax error could also be failing to indent properly. It's that dumb semicolon thing.

Example:

```
a ← expression  
display (A)
```

A is not the same as the variable 'a' because variables are case sensitive so it doesn't function properly.

Logic Errors: A mistake in a program's base logic that causes unexpected behavior.

```
a = 95
IF (a > 90)
  DISPLAY("You got an A.")
IF
(a > 80)
  DISPLAY("You got a B.")
IF
(a > 70)
  DISPLAY("You got a C.")
```

This program is supposed to print out 'You got an A' but it prints out 'You got an A' 'You got a B', 'You got a C'.

That's because 95 is greater than 70, 80 and 90!!

Run-Time Errors: An error that occurs when the program is running. You'll be able to start your program if you have a run-time error, but something will go wrong when you're trying to use it.

Ex: Display (5/0)

5/0 is undefined so the program crashes

Overflow Errors: an error that occurs when a computer tries to handle a number that's outside of its defined range of values.

Ex: $x < -2000 * 365$

Display (x)

The result is a large number.

Debugging is the process of finding and fixing errors.
HOORAY YOU PASSED BIG IDEA 1!!!!

BIG IDEA 2: DATA 22%

Data: collection of facts

Number base: the number of digits or digit combos that a system uses to represent values

Decimal system with a base 10: which only uses combinations of 0-9 to represent values
This represents the number 5,729

Power of 10	10^3	10^2	10^1	10^0
Value represented (base 10)	1000	100	10	1
Amount of value	5	7	2	9

Binary system only uses combinations of 0 and 1

This represents the number 5

Power of 2	2^3	2^2	2^1	2^0
Value represented (base 10)	8	4	2	1
Amount of value	0	1	0	1

The 1 signifies that that's what the number is made up of. In this case it is 4 and 1 which add up to 5.

These binary digits are known as **bits**. **Bit** - smallest unit of info stored or manipulated on a computer (0/1) - basic building blocks of storage just like amino acids for protein

- False/true, on/off...

8 bits form a byte. 8 bits has 256 unique combinations

- Black = 1 and white = 0
- [255,255,255] or [1111111,1111111,1111111] (3 bytes of data)

Analog Data

Data that is **measured continuously**. It changes very smoothly like the volume of music or the position of a runner.

As the time changes on a clock, data is being recorded constantly.

If you want to measure ALL of this data, this is known as **digital data**.

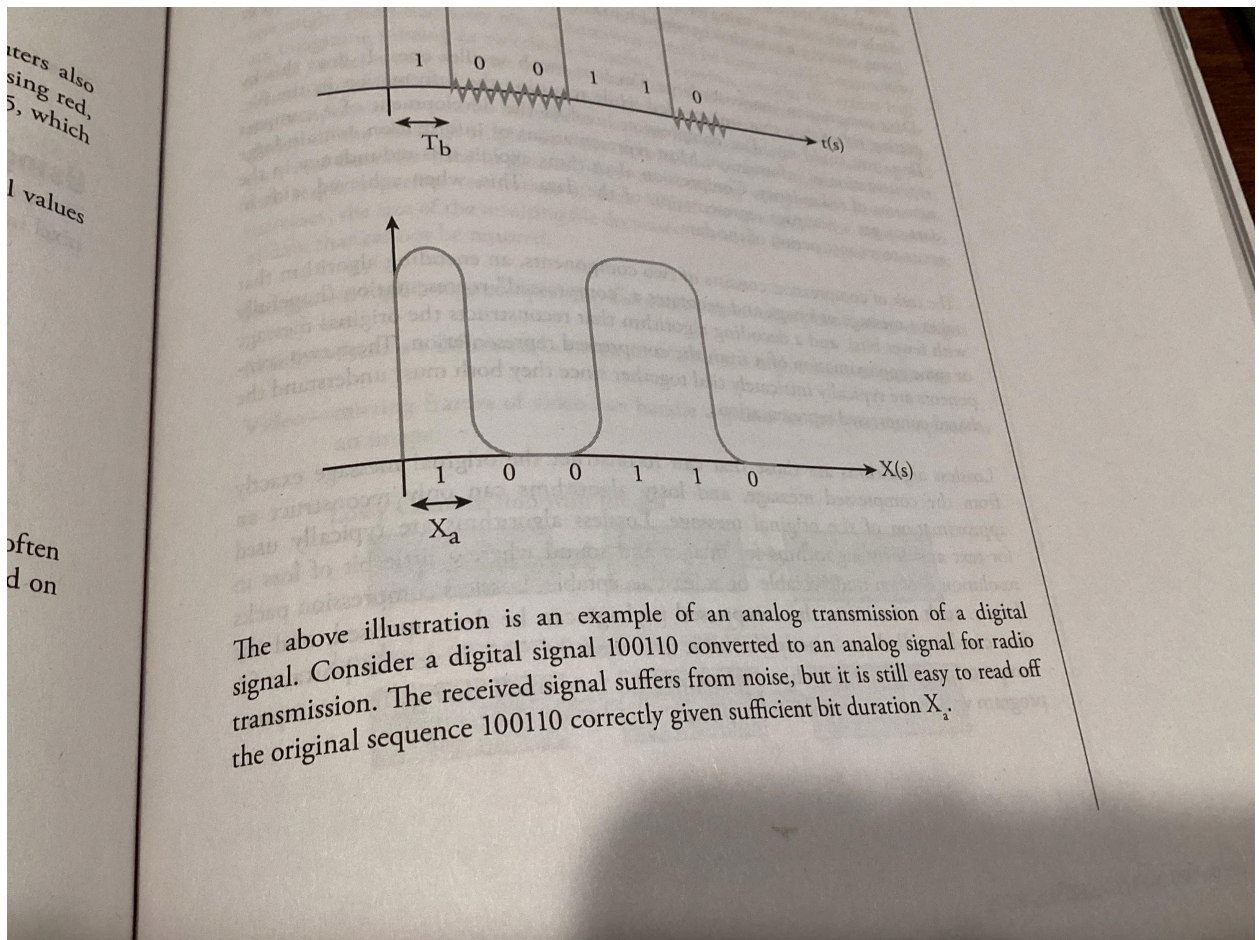
Analog data can be represented digitally by using a **sampling** technique. The values of the analog signal are measured and recorded at regular intervals. These intervals are known as **samples**.

Sampling - recording an analog signal at regular discrete moments and converting them or digital signals (can be stored on digital media)

Data Abstraction - filtering out specific details to focus on the info needed to process the data.

Digital data must be formatted in a finite set of possible values while analog data can be finite. For example watching a video on YouTube: you can see how long of a video you are watching, like 10 minutes. That's digital data, meanwhile seeing a live game at a venue it's continuous so it's analog.

Digital data is a simplified representation that leaves out extra details as you wouldn't know the time down to the millisecond. **Using digital data (leaves out details) to approximate real world analog data is considered an abstraction.**



Calculations :):

- To Calculate the largest value: do the power of 2 -1

For example what's the largest value you can represent with 7 bits:

2 to the 8th -1 which is 255

- To calculate the largest amount of numerical values you just do the power like 2 to the 8th which is 256. That's because we count 0-255

DATA COMPRESSION

File types like mp3, mp4, jpg all use data compression, without it a 3 minute song would be over 100MB.

-
- **Data compression** - set of steps of packing data into a smaller space while allowing for the original data to be seen

Two way process - make a data package smaller or decompress packet to original form

- Useful in computing to save disk space or to reduce bandwidth
- Deals with a string of bytes and compressing it down to a smaller set of bytes, taking less bandwidth.
- Condense large files by getting rid of data that isn't needed while retaining the information in the file
- Contain large amounts of redundancy to get a compact representation of the data
- 1) an encoding algorithm that takes a message or image
- 2) and generates a compressed representation and a decoding algorithm that reconstructs the original message or an approximation of it.

DEPENDS ON TWO THINGS

1. The amount of **redundancy** or repeated information
2. The method you use to **compress** your file

Run length encoding works by replacing repeating data with a run that represents the number and value of the repeated data.

For example FFFFFIIIIIVVVVVVEEEE could be written as 5F6I7V4E



LOSSLESS DATA COMPRESSION

- allows you to reduce your file size without sacrificing any of the original data in the process. Run length encoding is an example of this.
- **Inverting pixel colors and brightness values**
- can reconstruct the original message exactly from the compressed message
- Used mainly for text

STUDY FOR THE AP STUDY FOR THE AP STUDY FOR THE AP STUDY AP OR PERISH

-
- Packs data in such a way that the compressed package can be decompressed and the data can be pulled out the same way it was given.
 - **Very important** ^^ for programs that if a small change happens it can make it unusable.

LOSSY DATA COMPRESSION

- sacrifices some data in order to achieve greater compression
- **Converting to grayscale, lowering resolution**
- can only reconstruct an **approximation** of the original message
- Used mainly for images and sound
- High degrees of compression and result in smaller compressed files but some number of original pixels, sound waves, or frames are removed FOREVER. Loss of a noise/quantity
- Amount of compression  Size of resulting file 

Fewer bits does not necessarily mean less information

-The amount of size reduction from compression depends on both the amount of redundancy in the original data representation and the compression algorithm applied

EXTRACTING INFORMATION FROM DATA

Large data sets: **big data**

Correlation DOES NOT CAUSE: Correlations refer to the statistical relationship between two or more variables. It measures how closely these variables are related to each other, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation)

There could also be **outliers:** data points that significantly deviate from the overall pattern or trend

Metadata is data about data. **IT DOES NOT AFFECT THE DATA ITSELF!!** Changes and deletions made to metadata do not change the primary data

IT IS USED TO help find and organize data.

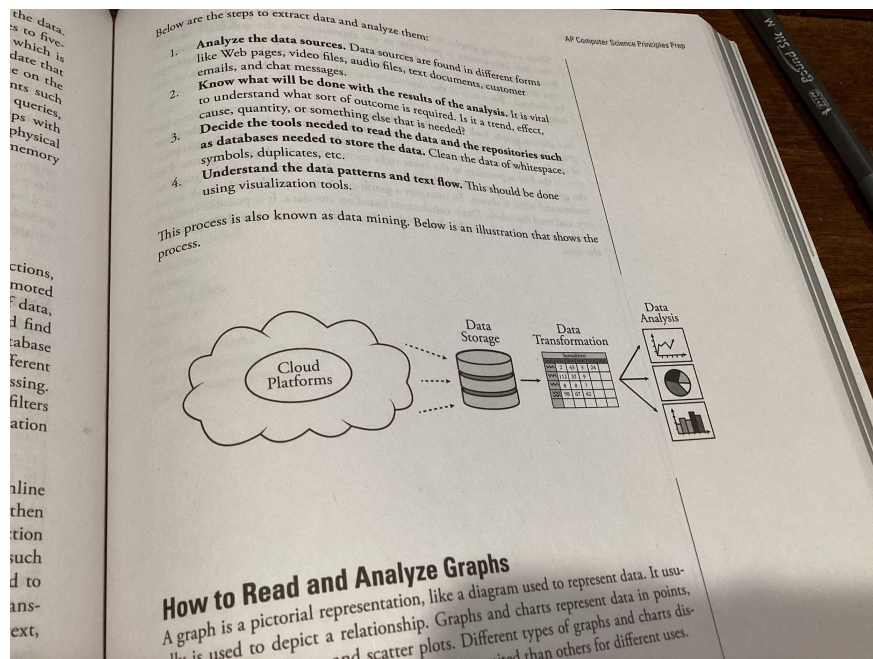
Data may not be uniform so you have to clean data. **Cleaning data** creates a uniform of data.

Data sets pose **challenges** regardless of size, such as:

1. The need to clean data
2. Incomplete data
3. Invalid data
4. The need to combine data sources

Large data sets are difficult to process using a single computer and may require **parallel** systems. Problems of **bias** are often created by the type or source of data being collected. Bias is not eliminated by simply collecting more data

The process of examining very large data sets to find useful information such as patterns is known as **data mining**.



Programs such as spreadsheets help efficiently organize and find trends in information

-combining or comparing data in some way, such as adding up a list of numbers, or finding the student who has the highest GPA

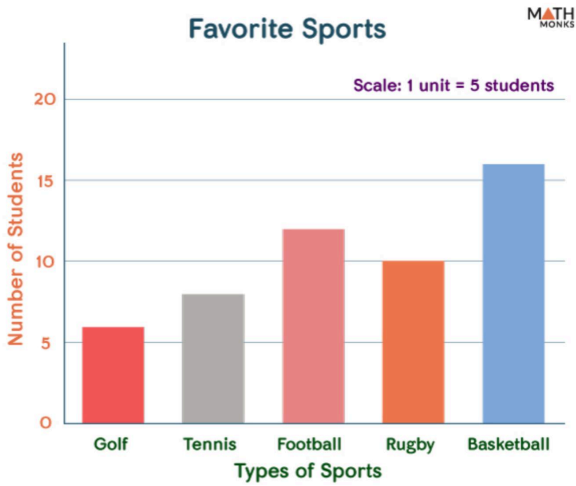
Data Transformation Examples:

- Modifying every element of a data set. This can be an arithmetic modification, although it isn't necessarily one.
 - Ex. Multiplying each number by some constant value (Like if you wanted to convert a list of measurements from liters to millilitres.)
 - Another non-arithmetic example is adding a grade level or class rank to a list of student records.
- Filtering a data set by category, as mentioned above.
 - Besides time or value, data sets can also be filtered by quality, such as which extracurricular activities a group of students are in.
- Combining or comparing data in some way.
 - Ex. Comparing the average SAT score of students going to all the colleges in one state and combining that data with average scores from other states.
- Creating data visualization tools.
 - Ex. graphs, charts, and word-bubbles.

Charts (have a basis on this)

Bar-Chart

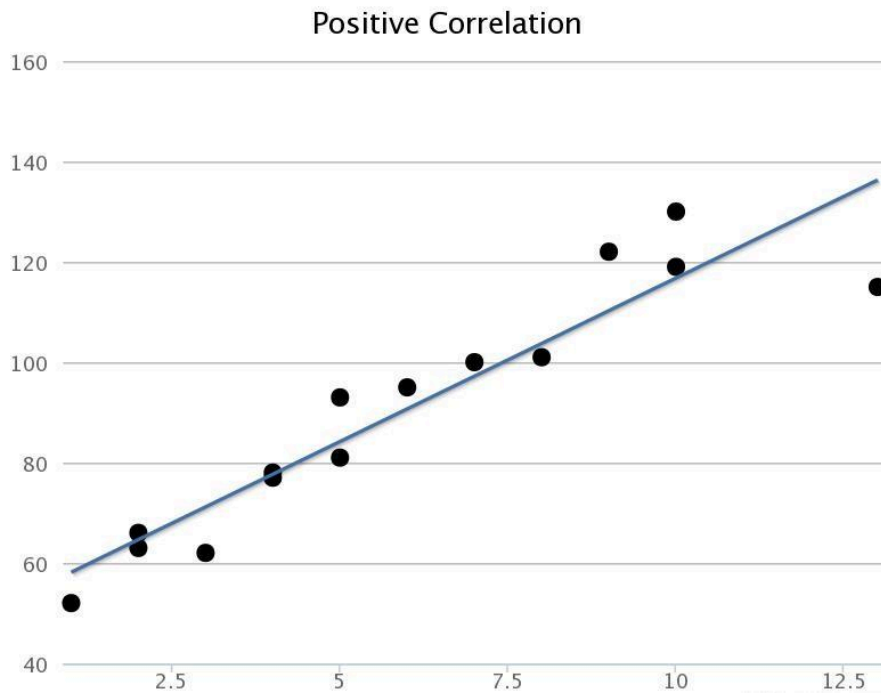
- a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represents
- Use vertical or horizontal bars to represent the values.



More students prefer basketball over golf.

Scatter plots

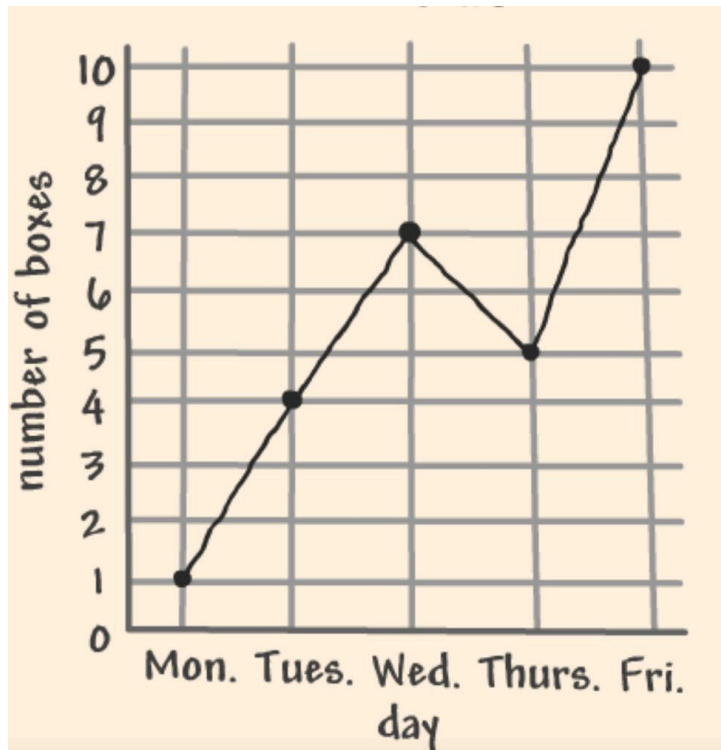
- uses dots to represent values for two different numeric variables
- Use correlation to compare the relationship between two variables. (Pearson r)
- Causation \neq Correlation
- Strong = 1 Weak = 0 Negative/Positive .



Strong positive correlation

Line graphs

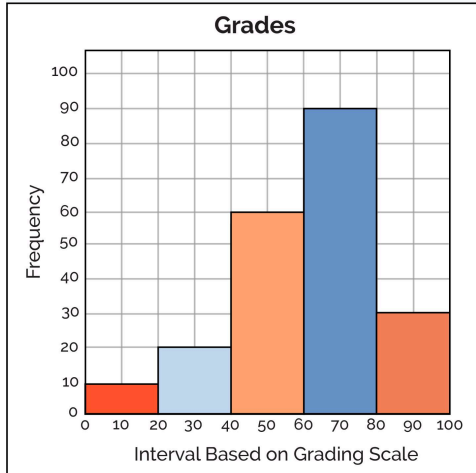
- Use lines to represent the values



There was an increase in boxes on Friday.

Histogram/Bar graph

- Uses bars to represent values
- Usually use frequency and ranges



Grades were about 60-80.

BIG IDEA 3: ALGORITHMS AND PROGRAMMING



35%

EVERYTHING IN THIS UNIT IS IMPORTANT!!

A **variable** is a placeholder in your program for a value. It's usually represented by letters or words.

You can assign values to variables through the assignment operator.

Text:

```
a ← expression
```

Block:

```
a ← expression
```

Evaluates `expression` and then assigns a copy of the result to the variable `a`.

Global vs Local Variable:

Note: I'm telling you RIGHT NOW that this is going to show up in those questions where it asks you to modify the code.

Global Variable:

STUDY FOR THE AP STUDY FOR THE AP STUDY FOR THE AP STUDY AP OR PERISH

-
- can be used anywhere
 - Variable is used outside of an event

Example:

```
varclicks = 0  
Onevent ('button', 'click')  
clicks = clicks +1
```

Local variable:

- used only in part of the code it was created, deleted once onevent is done.
- Variable is inside an event

Example:

```
onEvent('button', 'click')  
varclicks=1
```

AVOID LOCAL VARIABLES!!

- create variables once
- They should be on top of the code
- They should be outside an on event and a function

Each variable can only hold one data value at a time. It can be reassigned.

Example:

```
Animal = "cat"  
Animal = "dog"
```

The variable Animal is now "dog"

Data types are different categories of data that your computer can represent. For example: integers, strings, lists and booleans.

Integers can be positive or negative

Example: 5

Strings are represented by quotation marks

Example: 'We are passing AP CSP'

Substring is part of an existing string.

- **The + operator can be used to join or concatenate two strings or a number.**

Example: 'sun' + 'rise' = sunrise

- **/n: new line**
- **Slice (str, start, length) returns the substring split up of character from the string at the starting and what length.**

Example: **Slice(cat,1,2) : ca t**

A **list** is an ordered sequence of elements. They are also known as arrays.

Example: foodList = ["fish", "chicken", "beef"]

Boolean can only represent two values: true or false : computers use this to make decisions inside conditionals (if then)

If it's true, we execute the code segments if not then the else is executed!!



Relational and Boolean Operators	
Text and Block: a = b a ≠ b a > b a < b a ≥ b a ≤ b	The relational operators =, ≠, >, <, ≥, and ≤ are used to test the relationship between two variables, expressions, or values. A comparison using relational operators evaluates to a Boolean value. For example, a = b evaluates to true if a and b are equal; otherwise it evaluates to false.

The **NOT operator** is used to reverse what the condition evaluates to. If a condition is true, the operator will evaluate to false, and vice versa.

NOT OPERATOR: !

NOT true: false
NOT false: true

Text: NOT condition Block: NOT (condition)	Evaluates to true if condition is false; otherwise evaluates to false.
---	--

The **AND operator** is used to combine two conditions. The operator will only evaluate to true if both conditions are met

AND OPERATOR: &&

True AND False: False
True AND True: True
False AND False: False

Text: condition1 AND condition2 Block: (condition1) AND (condition2)	Evaluates to true if both condition1 and condition2 are true; otherwise evaluates to false.
---	---

The **OR operator** also involves two conditions. In this case, the operator will evaluate to true if one condition or the other is met.

OR OPERATOR: ||

True or False: True

True or True: True
False or False: False

Text:
condition1 OR condition2

Block:

condition1 OR condition2

Evaluates to true if condition1 is true or if condition2 is true or if both condition1 and condition2 are true; otherwise evaluates to false.

There can be nested conditionals. **Nested conditional** statements are conditional statements inside conditional statements.

```
IF (age=40)
{
ticket = 20
}
else
{
  If (age = 20)
  ticket = 30
  Else
  ticket = 25
}
```

So basically if the age was 30, the first if would be completely ignored then we would go to the else, and we would again ignore the if (age=20) so the ticket is 25.

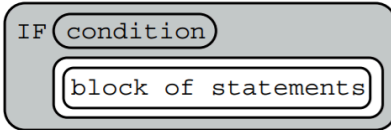
If statement (conditionals)



Text:

```
IF(condition)
{
  <block of statements>
}
```

Block:



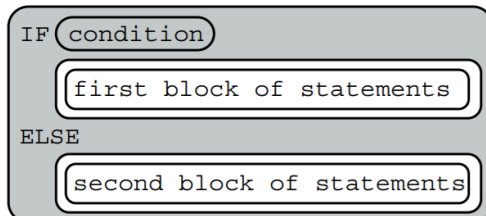
The code in **block of statements** is executed if the Boolean expression **condition** evaluates to **true**; no action is taken if **condition** evaluates to **false**.

Else statement

Text:

```
IF(condition)
{
  <first block of statements>
}
ELSE
{
  <second block of statements>
}
```

Block:



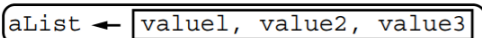
The code in **first block of statements** is executed if the Boolean expression **condition** evaluates to **true**; otherwise the code in **second block of statements** is executed.

Lists:

Text:

```
aList ← [value1, value2, value3, ...]
```

Block:



Creates a new list that contains the values **value1**, **value2**, **value3**, and **...** at indices **1**, **2**, **3**, and **...** respectively and assigns it to **aList**.

An **element** is an individual value in a list. In the picture above, value 1, value 2 and value 3 are all elements. Each element is assigned an index value.

Index numbers start at 1.

Lists can be used to create **data abstraction**.

Data abstraction simplifies a set of data by representing it in some general way. Basically focusing on the main thing.

Using a list can allow you to work easily with individual elements.

Accessing an element by index

Text: <code>aList[i]</code> Block: <code>aList [i]</code>	Accesses the element of <code>aList</code> at index <code>i</code> . The first element of <code>aList</code> is at index <code>1</code> and is accessed using the notation <code>aList[1]</code> .
--	--

This operation allows you to single out an element in a list based on its index number!

Assigning the value of an element of a list to a variable

Text: <code>x ← aList[i]</code> Block: <code>x ← aList [i]</code>	Assigns the value of <code>aList[i]</code> to the variable <code>x</code> .
--	---

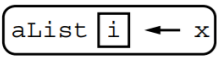
This allows you to assign a variable to a certain element within a list.

```
grocery_list = ["milk", "eggs", "cheese"]  
change = "soap"  
grocery_list[2] = change  
print (grocery_list)
```

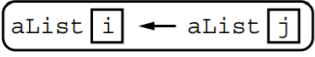
The code's output: ["milk", "eggs", "soap"]

Assigning a value to an element outright



Text: <code>aList[i] ← x</code> Block: 	Assigns the value of <code>x</code> to <code>aList[i]</code> .
---	--

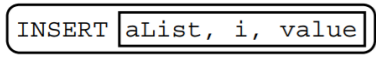
Assigning the value of one element in the list to another

Text: <code>aList[i] ← aList[j]</code> Block: 	Assigns the value of <code>aList[j]</code> to <code>aList[i]</code> .
--	---

```
grocery_list = ["milk", "eggs", "cheese"]  
  
grocery_list[0] = grocery_list[2]  
  
print (grocery_list)
```

The code's output: ["cheese", "eggs", "cheese"]

Insert elements at a given index

Text: <code>INSERT(aList, i, value)</code> Block: 	Any values in <code>aList</code> at indices greater than or equal to <code>i</code> are shifted one position to the right. The length of the list is increased by 1, and <code>value</code> is placed at index <code>i</code> in <code>aList</code> .
--	---

This allows you to insert a value into the index position you want. It will increase the length of the list and shift everything greater than or equal to that index down by one place.

For example if you insert a new value to the index value 4, what was originally there will move to the index value 5.

Appending or adding elements to the end of the list



Text: APPEND(aList, value) Block: APPEND aList, value	The length of aList is increased by 1, and value is placed at the end of aList.
--	---

This allows you to add values to the end of your list

New index is added

Removing: eliminates an element from the list

Text: REMOVE(aList, i) Block: REMOVE aList, i	Removes the item at index i in aList and shifts to the left any values at indices greater than i. The length of aList is decreased by 1.
--	--

Removes element in given list an given index

All indexes shift!!

Determine the length of a list through this

Text: LENGTH(aList) Block: LENGTH aList	Evaluates to the number of elements in aList.
--	---

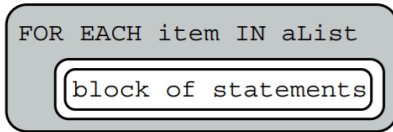
Looping through Lists

You can also use loops to transverse or go through a list. This can be a complete or partial transversal.

Text:

```
FOR EACH item IN aList
{
  <block of statements>
}
```

Block:



The variable `item` is assigned the value of each element of `aList` sequentially, in order, from the first element to the last element. The code in `block of statements` is executed once for each assignment of `item`.

Filter a List

Creating a subset of elements from the original list !!

An example is the linear search which checks each element of a list in order until the desired value is found or all elements in the list have been checked.

Binary Search

Most basic way to search through a list: called a **linear or sequential search algorithm** and it checks each value of a list in order until the result is found

Binary search algorithm starts in the **middle** of a sorted data set and **eliminates half** of the data based on what it's looking for. It repeats the process until the desired value is found. **THE LIST HAS TO BE IN ORDER**

Example:

1, 1, 2, 3, 3, 4, 5, 7, 9, 11, 12

Finding where 12 was.

The middle is 4 BUT 12 is larger than 4 so disregard everything under that.

5, 7, 9, 11, 12

Value 9 now but it's less than 12 so eliminates everything before and including that value

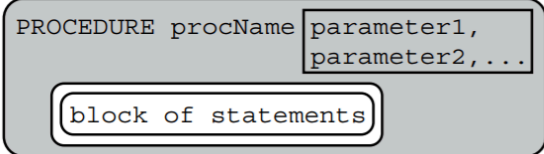
11, 12...

The process would go on until the program either found 12 or went through all values in the list

Binary search is more efficient than linear search

PROCEDURES

A **procedure** is a group of programming instructions. They are also called methods or functions. You can use a procedure to use the same set of instructions again and again without having to rewrite the code.

Procedures and Procedure Calls	
<p>Text:</p> <pre>PROCEDURE procName(parameter1, parameter2, ...) { <block of statements> }</pre> <p>Block:</p>  <p>The diagram shows a large rounded rectangle representing a procedure block. Inside, the text 'PROCEDURE procName' is followed by a smaller rounded rectangle containing 'parameter1, parameter2, ...'. Below this is another rounded rectangle containing 'block of statements'.</p>	<p>Defines <code>procName</code> as a procedure that takes zero or more arguments. The procedure contains <code>block of statements</code>.</p> <p>The procedure <code>procName</code> can be called using the following notation, where <code>arg1</code> is assigned to <code>parameter1</code>, <code>arg2</code> is assigned to <code>parameter2</code>, etc.:</p> <pre>procName(arg1, arg2, ...)</pre>

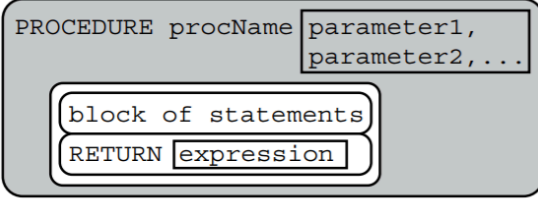
Procedures often require some sort of parameter in order to work. **Parameters** are the input variables of a procedure.

YOU DON'T ALWAYS NEED TO USE PARAMETERS

When you call a procedure, your program acts like those lines of codes are written out.

Arguments are values passed into the procedure.



<p>Text:</p> <pre>PROCEDURE procName(parameter1, parameter2, ...) { <block of statements> RETURN(expression) }</pre> <p>Block:</p> 	<p>Defines <code>procName</code> as a procedure that takes zero or more arguments. The procedure contains <code>block of statements</code> and returns the value of <code>expression</code>. The <code>RETURN</code> statement may appear at any point inside the procedure and causes an immediate return from the procedure back to the calling statement.</p> <p>The value returned by the procedure <code>procName</code> can be assigned to the variable <code>result</code> using the following notation:</p> <pre>result ← procName(arg1, arg2, ...)</pre>
--	---

You can use the **return** statement which you can then use without printing it. It is used in a function to specify the value that should be returned when the function is called.

RETURNS terminate the program.

Procedural Abstraction

- Procedures allow you to solve a large problem based on the solution to smaller subproblems.
- Procedures can help you simplify your code and improve its readability.

Instead of this:

```
first_number = 7  
second_number = 5  
sum_value = first_number + second_number  
print (sum_value)
```

```
first_number = 8  
second_number = 2  
sum_value = first_number + second_number
```

```
print (sum_value)
```

```
first_number = 9
```

```
second_number = 3
```

```
sum_value = first_number + second_number
```

```
print (sum_value)
```

you get this:

```
def summing_machine(first_number, second_number):
```

```
    sum_value = first_number + second_number
```

```
    print (sum_value)
```

```
summing_machine(5, 7)
```

```
summing_machine(8, 2)
```

```
summing_machine(9, 3)
```

THE ALGORITHM

An **algorithm** is a set of instructions used to accomplish a specific task or solve a problem.

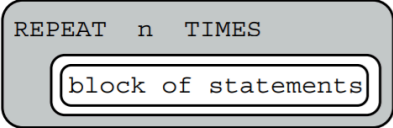
Sequencing, selection and iteration

Sequencing : consists of steps that go in order

Selection:(If else) Selection refers to the process of making a decision based on a condition or criteria. It allows the program to choose between different paths of execution.

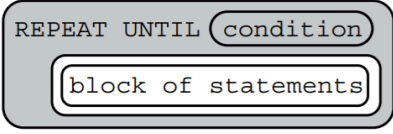
Iteration: (Loop) Iteration refers to the repetition of a set of instructions until a specific condition is met. It allows programs to perform tasks repeatedly without having to write repetitive code.

They are also called **loops**.

<p>Text: REPEAT n TIMES { <block of statements> }</p> <p>Block:</p> 	<p>The code in <code>block of statements</code> is executed <code>n</code> times.</p>
---	---

The `n` times is how many times the loop will run.

The second type of loop is a **Repeat Until condition loop (for loop)** where the loop will continue to run until a condition is met. We know how many times the loop will run.

<p>Text: REPEAT UNTIL(condition) { <block of statements> }</p> <p>Block:</p> 	<p>The code in <code>block of statements</code> is repeated until the Boolean expression <code>condition</code> evaluates to <code>true</code>.</p>
--	---

While loops run while a condition is met and end when that condition is no longer true. It checks the condition before executing the code block. We don't know how many times the code will run.

3 parts of a while loop

1. `var i = 0`: initializes the counter variable `i` to 0
2. Boolean expression: checks condition of variable
3. Statement which increase/decreases the variable `i++`

A for loop

for (var i = 0 ; i < 4, i ++)

Infinite loops are loops that continue to repeat indefinitely because the condition controlling the loop is always true or there is no condition at all.

EXPRESSIONS

Text and Block: <code>a + b</code> <code>a - b</code> <code>a * b</code> <code>a / b</code>	<p>The arithmetic operators <code>+</code>, <code>-</code>, <code>*</code>, and <code>/</code> are used to perform arithmetic on <code>a</code> and <code>b</code>.</p> <p>For example, <code>17 / 5</code> evaluates to <code>3.4</code>.</p> <p>The order of operations used in mathematics applies when evaluating expressions.</p>
Text and Block: <code>a MOD b</code>	<p>Evaluates to the remainder when <code>a</code> is divided by <code>b</code>. Assume that <code>a</code> is an integer greater than or equal to 0 and <code>b</code> is an integer greater than 0.</p> <p>For example, <code>17 MOD 5</code> evaluates to <code>2</code>.</p> <p>The <code>MOD</code> operator has the same precedence as the <code>*</code> and <code>/</code> operators.</p>

MOD operator: modulo: `a Mod b`: `a` is divided by `b` and MOD gives you what the remainder would be.

27 mod 4: 3

PEMDAS EXISTS!! DO MATHEMATICAL EXPRESSIONS WITH PEMDAS

Developing Algorithms

Algorithms are step-by-step instructions or procedures used to solve a problem or complete a task. They provide a clear set of instructions that can be followed to achieve a specific goal.

Examples of Existing Algorithms:

- Determining the max. or min. value in a group of two or more numbers
- Solving math problems: calculating sums, averages, etc.
- Determining a robot's path through a maze (route-finding algorithm)
- Compressing data
- Sorting a list

Generating Random Values

Text:

RANDOM(a , b)

Block:

RANDOM

Generates and returns a random integer from a to b, including a and b. Each result is equally likely to occur.

For example, RANDOM(1, 3) could return 1, 2, or 3.

Simulations: the process of creating a model or representation of a real world system or phenomenon on a computer.

Simulations are an example of abstraction!!

- used to represent real world events and conditions so you can investigate and draw conclusions about them without dealing with some of the complications of the real world
- Used to predict and plan
-

Disadvantages: may include bias based on what the simulation creator chose to include or exclude. It can also be out of scale

Algorithmic Efficiency:

A **problem** is a task that an algorithm is trying to solve while an **instance** of the problem is a problem with a specific input.

Decision problems: yes or no answer

Optimization problem: wants the best answer! example: finding the shortest path between two cities

An algorithm's **efficiency** is an estimate of how many computational resources (power, memory, time) it uses.

Or in a nutshell how many times a statement or statement group
Or much easier: how many steps it takes

When finding the max number of list elements find the closest 2 to the power of

Like 200: 2 to the 8th is closer

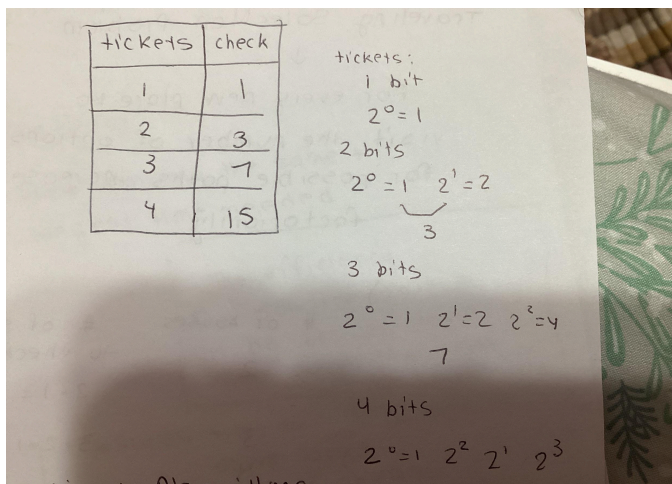
- **Polynomial efficiency of lower are said to run in a reasonable amount of time**

Ex: n^2 , log base 2 of n

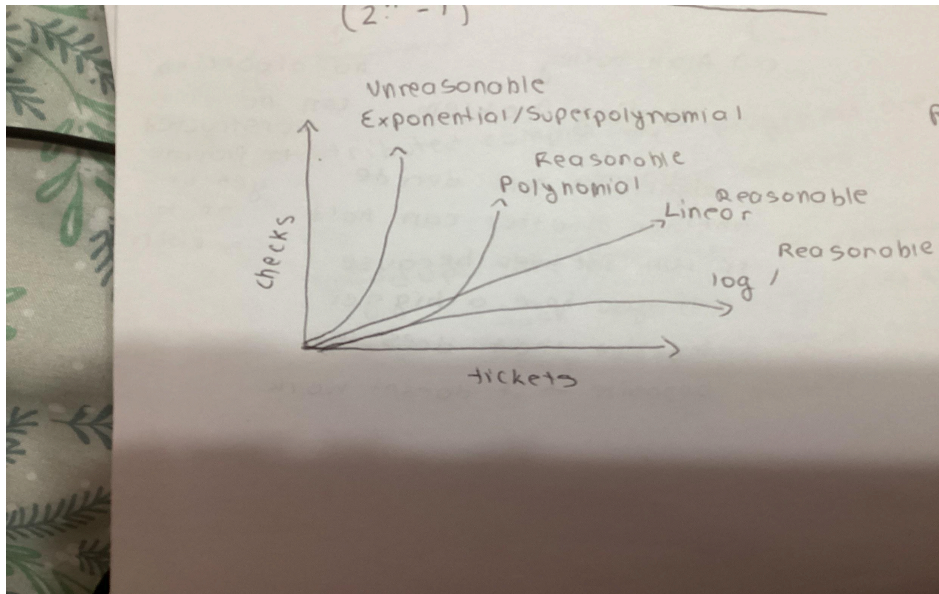
- **Exponential or factorial efficiency run in an unreasonable amount of time**

Ex: 2 to the n or $n!$

The number of checks a program needs to make is the largest number you can make with those bits.



Here are some graphs 



Some problems can't be solved in a reasonable amount of time so they turn to an approximate solution which is known as **heuristic**.

A **decidable** problem is a decision problem.

If an algorithm can't be written that's always capable of providing a correct yes or no answer: this is an **undecidable problem**. Solvable in some cases but there is no algorithm that will solve the problem in all cases.

Example: Halting problem created by Alan Turing

The halting problem asks that if a computer is given a random program, can an algorithm ever be written that will answer the question, will this program ever stop running?, for all programs? By proving that there wasn't, Turing demonstrated that some problems can't be completely solved with an algorithm

HOORAY YOU PASSED BIG IDEA 3!!!

BIG IDEA 4: COMPUTER SYSTEMS AND NETWORKS 15%

Internet: a computer network consisting of interconnected networks that use standardized, open (**NONPROPRIETARY**) communication protocols

- + Open network meaning any computing device can join as long as they follow the rules (protocol).
- + They connect computer networks which are systems of computing devices.

A **computing device** is a physical machine that can run a program. They connect with other computing devices to form a computing system.

A **computing network** is a group of computing devices that can share data with each other.

A **computing system:** a group of computing devices and programs working together for a common purpose

When you send or receive data from the internet, you have to get data from one place or another but sometimes it's too much data so...the data is broken up into **packets**.

Packets contain a section of the data you want to send and comes with a header that contains **metadata** (data about data) to tell the routers where the packet is from, where it's going and how it should be reassembled.

Computing devices create these packets and send them out through paths. **Paths are sequences of connected computing devices known as ROUTERS.**

The process of finding a path to take is known as **routing**

Packets can arrive at their destination in order or out of order! :(

Bandwidth: the rate of data transfer it allows from one device to another

Or

In simple words: the maximum amount of data a network connection can move in a certain amount of time

Latency: how late the bits arrive

It's measured in **bits per second of megabits per second!**

Internet Protocol:

In order for computing devices to communicate with each other over the internet, they all have to use the same protocol!

A **protocol** is a standard set of rules that everyone agrees on. They are OPEN or NONPROPRIETARY.

Two major protocols:

TCP/IP:

Transmission Control Protocol/Internet Protocol:

The TCP governs how packets are created and reassembled while the **IP** moves packets to their destinations. It also dictates how devices are given addresses to communication with each other (**IP ADDRESSES:** unique numerical label assigned to each device)

- reliable, ordered and error checked delivery of data packets
- It MAY be received out of order but the computer double checks and makes sure to add instructions to reassemble it on the other side

-
- Some MAY not be delivered but the computer checks to be redelivered again

Two versions of IP:

IPv4:

Ex: 74.125.20.113: split into 4 numbers all ranging 0-255
2 to the 32 possible values

★ IPv6:

Ex: 2001:0db8:0000:0042:8a2e....

These are hexadecimal numbers.

2 to the 128 possible values

UDP:

Used Datagram Protocol

Offers a way to deliver a faster stream of information by eliminating error checking which TCP/IP does.

- does not guarantee delivery or order of packets

THE WORLD WIDE WEB or www.

World Wide Web: a system of web pages, programs and files

This is governed by the **HTTP (hypertext transfer protocol) protocol which controls how web page data is transmitted (enables communication between web browsers)**

★ TCP, IP and UDP are used to transmit data over a variety of NETWORKS while HTTP is used to transmit data over THE WORLD WIDE WEB (pls don't get it mixed up :))

Scalability: the capacity for the system to change in size and scale to meet new demands

- + **Fault tolerance:** it can function properly even in the event of one part failing 🔥

One major aspect of a fault tolerant system is the presence of **redundancy**: the inclusion of extra components that can be used to mitigate failure of a system of other components fail

Benefits of Fault Tolerance:

- helps reduce hardware malfunctions (issues or failures with physical computer components) and cyber attacks (deliberate attempts by individuals to gain unauthorized access)
- Increases reliability of a system
- Keeps the system from shutting down
- Reduces damage by some cyber attacks (example: a Distributed Denial of Service Attack (DDoS) takes place when a server or network is overwhelmed with a flood of traffic, causing it to slow or even crash. In this situation, having a redundant server or network connection could allow you to go around the attack and continue to operate.)
- Makes it easier for a system to expand

Disadvantages of Fault Tolerance:

- + requires more resources
- + Expensive in materials and maintenance to build new resources

COMPUTING

Parallel Computing: a program is broken into smaller sequential computing operations using multiple PROCESSORS

Advantages:

- + helps save a lot of time
- + Scales more effectively

Sequential computing: traditional method of executing instructions in a sequential order

Distributed computing: multiple DEVICES are used to run a program : allows users to share information

Advantages:

- + allows people to solve problems that they wouldn't be able to due to lack of storage or too much processing time

Calculating EXECUTION TIME

The word problem we are using: For example if a program has three steps that take 40, 50 ,and 80 seconds then...

- I. **Sequential:** a sequential solution takes as long as the **sum of all steps** in the program. The sequential solution would take 170 seconds to complete.
- II. **Parallel Computing:** depends on the **number of cores (individual processing unit)** . The more cores the faster the solution. You have to find the fastest two processors

40 + 50: 90

40 + 80: 120

50 + 80: 130

Or if there are a known 2 processors like 40, 50, 60 and 80

You would find the longest out of the first two and second two which is 50 and 80 then add them so it is 130.

- III. **The speed up:** this is calculated by dividing the time it took to complete the task sequentially by the time it took to complete the task in parallel

Speedup is the measure of how much faster one solution or algorithm performs compared to another solution or algorithm when solving the same problem.

Speed limits:

Before you start a speed calculation problem, make sure you know whether or not all steps are independent. Don't just assume they all are. (The question should tell you if they are or not.)

It's those khan academy questions that goes like ok the startup takes 5 seconds so you would add 5 seconds to the total.

HOORAY YOU PASSED BIG IDEA 4!!!!

BIG IDEA 5: IMPACT OF COMPUTING

26%

Beneficial effects of computing innovations:

- + Machines have vastly improved the medical field
- + Engineers can collect data and design products
- + Communications have flourished
- + We can create, share and sell creative works

Harmful effects of computing innovations:

- + The Digital divide: people can be harmed due to unequal access to technology combined with the increasing importance of technology in the world
- + Computing bias: technology exacerbate currently existing human biases perpetuating inequality
- + Legal and ethical concerns: example being copyright law

-
- + Safe computing: risks in day to day computing use such as being infected with a virus
 - + Loss of privacy
 - + Replacement of humans by computing innovations leading to unemployment
 - + Dependence on technology
 - + Negative health outcomes
-
- **Targeted advertising** is intended to help businesses turn a profit, but it incentivizes the collection of private information and has the potential to be abused.
 - **Machine learning and data mining** have greatly benefitted many fields, but their findings are also susceptible to biases and may unintentionally contribute to discrimination.

THE DIGITAL DIVIDE

Digital divide refers to the gaps between those who have access to technology and the internet and those who don't.

FACTORS THAT INFLUENCE THE DIGITAL DIVIDE:

- Demographics:
 - a. Younger people are more likely to be comfortable with the technology.
 - b. People with higher levels of education tend to use the internet more.

- Socioeconomic
 - a. people with higher incomes are more likely to be able to purchase and maintain technology

- Geographic
 - a. Some areas allow more internet access than another

Harmful impacts of the digital divide

Examples:

Educational opportunities

-
- During the 2020 COVID pandemic many schools across the US shifted to virtual learning systems. Some students without stable internet connections suffered educationally.

Employment Opportunities

- Those without internet access may be at a disadvantage in terms of finding and applying for jobs. They also may be hindered from being able to do their jobs or access resources.

Reducing the Digital Divide

- Organizations can release educational resources to teach people how to navigate the internet. They can release digital literacy programs: programs that teach people how to use the internet.
- Local and national governments can fund businesses that provide internet access to areas that don't have access.

COMPUTING BIAS

Biases are tendencies or inclinations, especially those that are unfair or prejudicial.

Examples of bias:

- Criminal risk assessment tools are used to determine that a defendant will commit another crime . The algorithms are trained to pick out patterns and make decisions based on historical data.
- Facial recognition systems are trained on data sets that contain fewer images of women and minorities.
- Recruiting algorithms can be biased against certain races of gender.

What can we do to prevent bias in computing?

- Use diverse data sets
- Review algorithms for potential biases
- Increase diversity in the tech industry

Machine learning models are computer programs that can learn from data and make predictions or decisions without being programmed.

CROWDSOURCING

Citizen science: scientific research that the general population helps to conduct. Ordinary citizens help contribute data to research projects using computing devices.

Example: counting birds they see at local feeders or smth 🦆

Citizen science gives a wide range of people the ability to contribute to scientific studies and provide more diverse data for scientists.

Crowdsourcing is the practice of getting a large amount of input or information from people on the internet. Citizen science IS A FORM OF CROWDSOURCING!!

Organizations do not need to pay for the information they get.

LEGAL AND ETHICAL CONCERNS !!

Intellectual Property is the creations of the mind such as inventions, literary and artistic works, designs and images used in commerce

It's very easy to access and distribute intellectual property found on the internet but one of the ways to protect it is through copyright

Public Domain: these are creative works that are free to use without permission

Copyright is the legal right that the creator of a work has to it.

Two types of copyright:

Economic rights: rights to financial benefits from the use of work

Moral rights: the right to claim authorship or the right to prevent harmful changes

If you claimed the said content was your own you are guilty of PLAGIARISM!!

Plagiarism is when you take the content of someone else and claim it as your own.

Ex: AP SEMINAR TURNITIN.COM coming in clutch with my 26 percent AI 😞

Legal ways to use the IP of others

Creative Commons is a public copyright license that creators use when they want to GIVE others the right to use their work.

Fair Use allows the use of copyrighted material without permission for limited purposes such as new reporting.

INTRODUCING THE ACRONYM **PANE!** (Just how ap CSP gives me pain) 😞

Purpose: new purpose to make it original

Amount: small amount of work

Nature: use works based in effects

Effect: don't make money off of someone's work

Open sourcing: allows for work to be freely distributed, and modified.

Open access refers to research available to the general public free of restrictions like academic journals. They are often free of copyright.

SAFE COMPUTING:

- Search engines can track your search history and use it to suggest websites and ads which is known as **TARGETED MARKETING.**

-
- Devices, websites and networks can collect information about a user's location like their IP address.

Personally Identifiable Information: This is the information that can be used to identify you

It includes: age, race, phone number, medical info, biometric data, social security number

Benefits and Harms of Information Collection

Benefits:



- helps enhance your experience online
- The Tik tok for you page rely on collection of personal information

Harms:

- your identity could be stolen
- You could be stalked!
- Companies that collect personal information could put their users at a risk if they're hit by a data breach

Think before you post guys

Other dangers of computing

- your computer may be infected with a virus or a worm. A **virus** is a malicious program that can gain unauthorized access to something and copy itself. It must be activated by the user. By contrast worms can operate independently.  
- Computer viruses are a type of **malware or malicious software** that is intended to damage. It includes ransomware (makes computer inaccessible until a ransom is paid) and adware (displays unwanted ads to slow down computer)

Protection against malware

Security patch: update to app and fixes bugs

Firewall: monitors incoming traffic but cannot identify and block all malware

Antivirus software: scans files and identifies malware

- Scammers online can take advantage of others. **Phishing** for example works by tricking users into providing their personal information by posing as a trustworthy group.
- **Keylogging** which records your keystrokes to gain access to passwords and other information.
- The information you send over public networks has the potential to be intercepted, analyzed and modified through a **rogue access point : a wireless access point that gives unauthorized access to a secure network. They can modify, analyze and intercept data.**

Passive interception: can read data but not manipulate it

Active interception: manipulates data

- **Distributed denial of service (DDos)** multiple computers overwhelm website with too many requests

Principles of Safe Computing

Authentication measures prevent people from gaining unauthorized access to your account.

- I. A strong password: string passwords often use a variety of characters such as uppercase letters, number and symbols
- II. Multi factor authentication: way to control who gets access to your accounts by requiring multiple methods
 1. Knowledge: password or pin or verification questions
 2. Possession: something you own like a one time password sent to your phone or an access badge
 3. Inheritance: biometrics: fingerprints or your voice

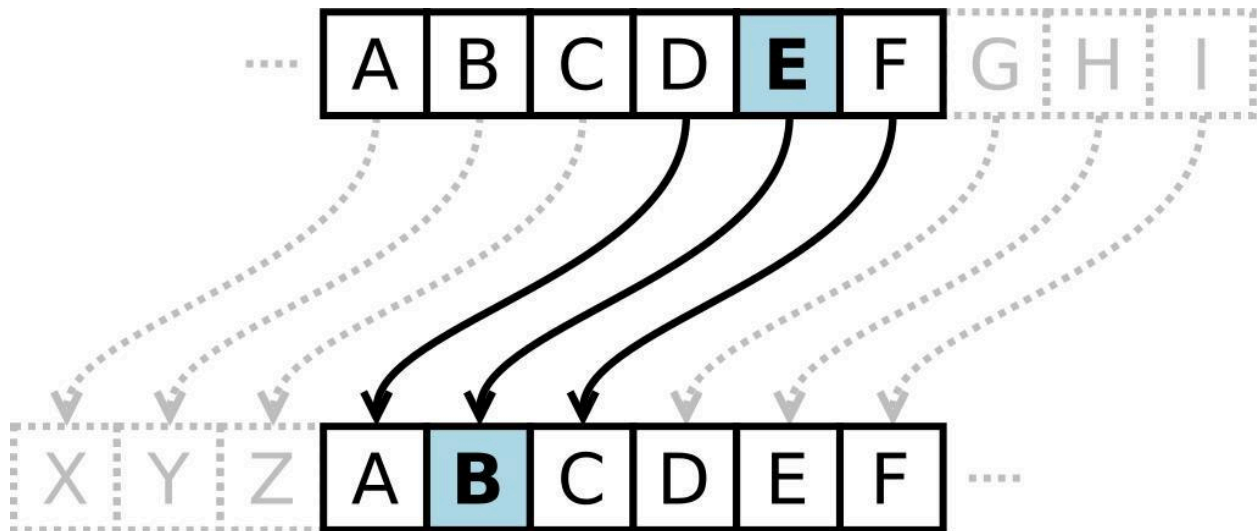
Encryption is the process of encoding data to prevent unwanted access.

Encrypted data is **ciphertext** while **unencrypted** data is **plaintext**.

Decryption is the process of decoding data.

Both of these encryption methods use a key to keep their messages secret

CAESAR CIPHER YAAAAAAA



Symmetric key encryption uses one key for both encrypting and decrypting code.

Asymmetric or Public key encryption uses a public key to encrypt but a private key to decrypt this message.

The public key encryption system relies on digital certificates.

These are issued by Certificate Authorities (CAs) to trusted sites. They allow other computers to verify that a website is what it says it is. These certificates are essential to the public key encryption system because they foster trust between websites. Think of the certificates to be a little like the signature on a check—once we see that signature, we know that the check is trustworthy.

STUDY FOR THE AP STUDY FOR THE AP STUDY FOR THE AP STUDY AP OR PERISH

The sending computer uses the public key of the receiving computer. The sender must verify the identity behind the public key. The certificate authority verifies their ownership of the domain, signs it with their own name and public key and returns it.

The user trusts the client trusts the certificate authority trusts the server

Digital certificates contain a copy of the public key from the certificate holder and is matched to the private key to verify if it's real.

Let me provide an example!

For example, a creator might have a specific watermark whether it's a pen name or a signature. So the digital certificate authorities have a copy of the real one cuz there are a lot of copycats out there. So the digital certificate authorities fact check to see if they are real and then lets the user know. :)

Lowkey the only section in unit 5 that's so confusing 😞

HOORAY YOU PASSED BIG IDEA 5!!!



GENERAL ADVICE TO HELP U FEEL BETTER 😊

- Read ANY code presented in the exam **LINE BY LINE, CHARACTER BY CHARACTER.** So many people have gotten coding questions wrong because they didn't look through the code carefully.
- If you're stuck on a question for some time, **skip over it for now.** You have at most 2 minutes to answer each question on a test.
- DON'T OVERTHINK
- KNOW THE VOCAB
- Remember the code is all is pseudo code INDEXES FOR LISTS START AT 1 NOT 0
- For programming questions, take your time following each step. Keep track of variable values on scratch paper. Trace loops carefully, and make sure you understand what EACH function or line does IN ADVANCE.
- Spend more time on the coding questions or the robot ones (THOSE ARE SO FRIGGING ANNOYING OMG)
- You can get like a minimum of 10 wrong and the FRQ exceptionally brilliant if you want to get that 5 so you DON'T punch your computer 😊
- When doing your practice tests, experiment with one piece of print paper

-
- FRQs should be 3-4 sentences and code SPECIFIC to your code. It's like that college essay thing where it should be as if you cover the name then you know whose it is. DON'T MAKE IT GENERIC

Credits:

This compilation of notes is from our class notes, Princeton, Quizlet and Fiveable. Hats off to them. By ME and (REDACTED).

^,,,^

(. . .)

/ づ You want my heart?

^,,,^

(. . .)

/ づ♡ Ok here's my heart

^,,,^

(. . .)

U♡C~ Nevermind, you don't deserve it

^,,,^

(. . .)

/ づ Just kidding! Here! <3



GOOD LUCK!! :)

STUDY FOR THE AP STUDY FOR THE AP STUDY FOR THE AP STUDY AP OR PERISH