COMPUTER PROGRAMMING LANGUAGES

A Programming Language is software that helps you write instructions for your computer. There are several different programming languages, each with their own pros and cons; it depends on what you want to create and the **Level of Abstraction**.

HIERARCHY OF PROGRAMMING LANGUAGES

LANGUAGE LEVEL		DESCRIPTION	EXAMPLE	ABSTRACTION LEVEL
High-Level Language	☺	Easiest for Humans; hardest for Computers	Videos (FPS)	↑ Greater Level of Abstraction
			Images (Pixels)	
Assembly Language	⇄	Requires work for both the Human and Computer to undestand	Colors	
			Decimals	↓Lower Level of Abstraction
Machine Language	☹	Easiest for Computers; hardest for Humans	Binary	

COMPUTERS AND COMPUTATIONAL THINKING

COMPUTATIONAL THINKING	DEFINING COMPUTERS
Decomposition: breaking down a large problem into manageable bits for the computer to execute	Data: we glean meaning from data
Recognizing Patterns: insight into solutions and giving context for solving new problems; understanding that the symbols may change, but not the pattern	Process: what we have the computer do when we engage with it
Abstraction: engaging with information at a lower, more general level where not everything must be understood in order for it to work (see *Abstraction of Code*)	Output: what the computer figures out and, sometimes, shows us
Designing Algorithms: creating a plan of action or list of instructions that a computer can follow and execute	Storage: a place dedicated to the output either immediately upon completion, or later after conditions are met

ABSTRACTION OF CODE IMPLEMENTATION

LESS ABSTRACT ← → MORE ABSTRACT

Programming Language	Pseudocode Language	Natural Language
actual programming language	practice computing language	human language; discussing how to program

VARIABLES

DEFINITION

can be thought of as a name that refers to a value inside of a program

NAMING CONVENTIONS

CAN:	CANNOT:
● start with or contain A-Z	● start with 0-9
● start with or contain a-z	● contain any symbol other than "_"
● contain 0-9	● contain a reserved word
● contain "_"	● contain spaces

DECLARING VARIABLES

declaring a variable lets the program know what process it can perform on the stored input or value

INITIALIZING VARIABLES

the process of assigning a value to the variable once it is declared

THREE LANGUAGE CASES

camelCase	PascalCase

VARIABLE STRUCTURES

"65 is assigned to the variable score!"

int	<--	65	
↑	↑	↑	
variable type	variable name	assignment operator	value

TWO TYPES OF CODE

COMPILATION CODE	
Compiler	*Interpreter*
Compiled	*Interpreted*
changes code to machine readable code all at once	changes code one command at a time
PRO: Faster	PRO: Easier to change and correct
CON: Harder to find errors and fix	CON: Slower
EXAMPLE: C++	EXAMPLE: Python

PROGRAMMING VOCABULARY

What is **SYNTAX**?

> How you organize your program and what language you use to create it

PROGRAMMING VOCABULARY (cont)

What are **KEY** or **RESERVED WORDS**?

> Special jargon unique to each language that have specific, unalterable purposes

What is **DOCUMENTATION**?

> Text and information that comes with a program but does not affect the running of the program

What is **SCOPE**?

> How a program is organized and "controlled;" each language manages and controls scope differently; this may include brackets, white space, or indentation; scope can also refer to specific block of code like a loop

What is **TESTING** or **UNIT TESTING**?

> When you make sure your code (program, software) is behaving as intended; using test cases helps determine if each iteration will work and won't work

What is **EDGE CASING**?

> Using the "edges" of the test, like going one above, one below, or a combination

What is **DEBUGGING**?

> A BUG is an undesireable behavior in a program, so debugging is of identifying and correcting the errors *See Error Types*

What are **COMMENTS**?

> Notes within a code or program that do not affect the execution but may be helpful to the programmer

DATA TYPES

TEXT	NUMERIC	
Character	*Integer*	*True/False*
a single letter, number, or symbol	integers are always whole numbers	*Boolean Variable Examples*
set off by '_'	positive, negative, or zero	boolean check<--false
Examples:'a', '5', '!'	*Floating Point Data*	boolean check <--true
String	a floating decimal value which contains NO fractions	
a combination of characters (number or letter) strung together	*Reserved Words for Numeric Data*	
set of by "_"	short (16 bits)	
Examples: "181240", "Hello!"		

data and data format could change depending on the language being used

ERROR TYPES