- Algorithms are Step-by-step instructions that solve a problem if followed exactly.
- High-level language: Easier to read, write, and understand (Python, C++, Java, PHP).
- Low-level language: Directly executed by a computer (machine/assembly language).

Errors

Types of error	When it occurs	How to identify	eg
Syntax error	The program structure is incorrect	Syntax error in the program	Forgetting: in an if statement
Runtime error	While the program is running	Program crashes	10/0
Semantic error	The program runs, but gives the wrong result	Incorrect output	forgetting to divide by 100 when calculating %

ParseError – Syntax issues (e.g., missing parentheses or quotes).
TypeError – Mismatched types (e.g., adding string + int).
NameError – Using a variable before it exists or due to typos.i.eror in the name
Value error- occurs when a function receives an argument of the right type but an inappropriate value.egint("hello") # ValueError because "hello" cannot be converted to an integer

Type Conversion

int(x) → converts x to an integer (truncates floats, parses numeric strings)
float(x) → converts x to floating-point
str(x) → converts x to string

Operation	Purpose
len()	Returns the **length** of a string, list, tuple, or other iterable
%	**Modulus**: returns the remainder of a division (whole number operation)
//	**Floor division**: returns the quotient as an integer, discarding the remainder
/	**Decimal (floating-point) division**
[i]	**Indexing**: returns the element at position i (e.g., s[3] → 4th character)
[start: end]	**Slicing**: returns a subset from start to end-1

.append(x)	Adds element x to the end of a list
.insert(i, x)	Inserts element x at position i in a list
.pop(i)	Removes and returns the element at index i (default last)
.remove(x)	Removes the first occurrence of x in a list
.index(x)	Returns the index of the first occurrence of x
.count(x)	Counts how many times x appears
.split(sep)	Splits a string into a list at each occurrence of sep
.join(iterable)	Joins elements of an iterable into a string, separated by the string
.upper()	Converts string to uppercase
.lower()	Converts string to lowercase
.replace(old, new)	Replaces all occurrences of old with new in a string
.strip()	Removes leading and trailing whitespace from a string
range(start, stop, step)	Generates a sequence of numbers from start to stop-1
type()	Returns the **data type** of a variable or value
int(), float(), str()	Converts values to integer, float, or string

Modules are data objects: Just like variables, modules contain Python elements such as functions, classes, or variables, eg, import math.,import turtle

Method	**Usage**	**Description**
forward(distance)	t.forward(100)	Moves the turtle forward by the specified distance.
backward(distance)	t.backward(50)	Moves the turtle backward by the specified distance.
left(angle)	t.left(90)	Turns the turtle **left** by the specified angle (degrees).

right(angle)	t.right(45)	Turns the turtle **right** by the specified angle (degrees).
penup()	t.penup()	Lifts the pen, so moving the turtle does **not** draw a line.
pendown()	t.pendown()	Puts the pen down to resume drawing.
color(color_name)	t.color("blue")	Sets the pen color.
fillcolor(color_name)	t.fillcolor("red")	Sets the **fill color** for shapes.
begin_fill()	t.begin_fill()	Starts filling a shape. Must be followed by end_fill().
end_fill()	t.end_fill()	Completes the shape filling.
write(string)	t.write("Hello")	Writes text at the current turtle position.
pensize(size)	t.pensize(3)	Sets the thickness of the pen.
speed(number)	t.speed(10)	Sets the turtle's drawing speed (1–10 or "fastest").
goto(x, y)	t.goto(100, 50)	Moves the turtle to specific coordinates.
setheading(angle)	t.setheading(90)	Points the turtle in a specific direction (absolute angle).
home()	t.home()	Moves the turtle to the origin (0,0) and points east.

Function -is a named sequence of statements that belong together. Organize programs into chunks matching how we think about solutions.eg, **def drawSquare(t, sz):**

Docstrings-A string right after the function header used for documentation, eg,"""Make turtle t draw a square of side sz.""".

The **return** statement ends the function immediately. Without return, Python returns None.

Local Variables are created inside a function, while global variables are outside a function.

Functional decomposition-Break a large problem into smaller functions.

Main function: Organizes the program's main step

Flow of execution: Functions are defined, but execute only when called

Incremental development: Build the program in small, testable steps

Unit tests -confirm the correctness of the function at each step