

Introduction to Programming

Introduction to Programming – Quick Reference Notes

1. What Is Programming?

- **Definition:** The process of designing, writing, testing, and maintaining code that tells a computer how to perform tasks.
 - **Goal:** Transform a problem or requirement into a set of precise, executable instructions.
-

2. Core Concepts

Concept	Description	Typical Example
Algorithm	Step-by-step procedure to solve a problem.	Sorting a list of numbers.
Syntax	Formal rules that define how code must be written.	<code>if (x > 0) { ... }</code> in C-style languages.
Semantics	Meaning of syntactically correct statements.	<code>x = x + 1</code> increments <code>x</code> .
Variable	Named storage location for data that can change.	<code>int count = 0;</code>
Data Type	Classification of values (e.g., integer, string, boolean).	<code>float</code> , <code>char</code> , <code>bool</code> .
Control Flow	Structures that dictate the order of execution.	<code>if/else</code> , <code>for</code> , <code>while</code> .
Function / Procedure	Reusable block of code that performs a specific task.	<code>def add(a, b): return a + b</code>
Library / API	Pre-written code that provides additional functionality.	<code>math.sqrt()</code> , <code>java.util.ArrayList</code> .
Debugging	Finding and fixing errors (bugs) in code.	Using breakpoints, print statements.
Compilation / Interpretation	Translating source code to executable form.	<code>gcc</code> (compile), Python REPL (interpret).

Introduction to Programming

3. Programming Paradigms

Paradigm	Key Idea	Typical Languages
Procedural	Code organized as a sequence of instructions and procedures.	C, Pascal
Object?Oriented (OO)	Data and behavior are bundled into objects; emphasizes inheritance & polymorphism.	Java, Python, C++
Functional	Computation expressed as evaluation of pure functions; avoids mutable state.	Haskell, Lisp, Scala
Declarative	Describe <i>what</i> should be done, not <i>how</i> ; includes SQL, HTML, Prolog.	
Event?Driven	Flow controlled by events (user actions, messages).	JavaScript (browser), C# (GUI)

4. Basic Program Structure (Example in Python)

```
# 1. Import modules (libraries)
import math

# 2. Define constants
PI = math.pi

# 3. Define functions
def area_of_circle(radius):
    """Return the area of a circle given its radius."""
    return PI * radius ** 2

# 4. Main execution block
if __name__ == "__main__":
    r = float(input("Enter radius: "))
    print(f"Area = {area_of_circle(r):.2f}")
```

Typical sections

1. **Imports** – bring in external code.
2. **Constants** – values that never change.
3. **Functions / Classes** – reusable logic.

Introduction to Programming

4. **Entry point** – code that runs when the script starts.
-

5. Writing Your First Program (Pseudo?Steps)

1. **Understand the problem** – what input, what output, any constraints?
2. **Design an algorithm** – outline steps in plain language or flowchart.
3. **Choose a language** – consider environment, libraries, learning curve.
4. **Translate algorithm to code** –