# ExplainCode: A Simple Language to Write and Run Algorithms Like a Human

**By Devansh Purohit · July 2025**

---

## What Is ExplainCode?

Have you ever written pseudocode that looked super clear but couldn't actually run it?

**ExplainCode** fixes that.

It's a lightweight language that lets you write algorithms the way you *think*—like steps in plain English—and then turns that into real Python code that you can run.

---

## Why It Exists

Most of us start learning algorithms with pseudocode. It's great for planning but useless if you want to actually see it work.

Here's what we usually face:

- Pseudocode doesn't run
- Real code has too much syntax
- Beginners get stuck before they even start

**ExplainCode** bridges this gap by letting you:

- Write clean, step-by-step instructions
- Use readable keywords like `Set`, `IF`, `FOR`, etc.
- Actually run the logic without needing to learn complex syntax

---

## 📦 How It Works (At a Glance)

1. You write your logic in a `.epd` file (like pseudocode).
2. The ExplainCode parser reads your steps.
3. It converts your steps into working Python code.
4. You get prompted for input—and boom, it runs!

---

## The Building Blocks of ExplainCode

Here's what writing in ExplainCode looks like:

```
ALGORITHM AddNumbers
INPUT: a, b
OUTPUT: sum

STEP 1: Set sum ← a + b
STEP 2: RETURN sum

END ALGORITHM
```

Simple, right?

---

## 🔍 What's Supported

| Feature | How It Looks in ExplainCode | What It Becomes in Python |
|---|---|---|
| Assignment | `Set x ← 5` | `x = 5` |
| IF condition | `IF x > 0 THEN ... END IF` | `if x > 0:` |
| | | |

| | | |
|---|---|---|
| FOR loops | `FOR i ← 1 to n DO ... END FOR` | `for i in range(1, n+1):` |
| WHILE loops | `WHILE cond DO ... END WHILE` | `while cond:` |
| RETURN | `RETURN x` | `return x` |
| PRINT | `PRINT x` | `print(x)` |

---

## 💻 Let's Look at an Example

Here's how you'd write **Binary Search** in ExplainCode and return the result in variable `c`.

```
ALGORITHM BinarySearch
INPUT: A, target
OUTPUT: c

STEP 1: Set low ← 0
STEP 2: Set high ← len(A) - 1
STEP 3: Set c ← -1
STEP 4: WHILE low <= high DO
STEP 5:     Set mid ← (low + high) // 2
STEP 6:     IF A[mid] == target THEN
STEP 7:         Set c ← mid
STEP 8:         BREAK
STEP 9:     END IF
STEP 10:    IF A[mid] < target THEN
STEP 11:        Set low ← mid + 1
STEP 12:    ELSE
STEP 13:        Set high ← mid - 1
STEP 14:    END IF
STEP 15: END WHILE
STEP 16: RETURN c

END ALGORITHM
```

You don't worry about indentation or syntax—just focus on the logic.

---

## ⚙ How It Actually Runs

1. You run this command:

```
python explaincode_compiler.py binary_search.epd
```

2. It asks for inputs like:

```
→ A = [2, 4, 6, 8, 10]
→ target = 8
```

3. It shows you the Python code **and the result**:

```
✅ Output: 3
```

Boom. Done.

---

## 📈 Why It's Useful

- **For students**: No syntax confusion, just logic.
- **For teachers**: You can write testable algorithms in plain English.
- **For developers**: Great for prototyping or explaining code to non-tech folks.
- **For AI/LLMs**: Perfect for translating natural language to code and back.

---

## What’s Coming Next

We’re already working on:

- Support for custom functions and recursion
- A visual playground (like Scratch for algorithms)
- Type hints and better error messages
- Exporting to JavaScript or C++
- Debug mode with step-by-step tracing

---

## In Summary

ExplainCode isn’t just a language—it’s a new way to think about coding.

It helps you focus on what matters: the logic, not the syntax. Whether you’re learning, teaching, or prototyping, it’s a tool that speaks your language.

---

### Want to try it?

Write your `.epd`, and run:

```
python explaincode_compiler.py my_algorithm.epd
```

Done!