Purplex

पर्प्लेक्स Marathi

EiPL & Prompt Problems

Overview

What we're covering today

Two "Natural Language Programming" Activities

EiPL & Prompt Problems — students describe code instead of writing it

A Platform

Purplex — AI-powered grading for these activities

What is EiPL?

You see code. You write what it does — not *how*, but *what*.

EiPL (Explain in Plain Language) flips traditional coding exercises. Instead of writing code, you **describe** existing code in your own words. An AI then tries to recreate the code from your description alone.

Reverse engineering: prove you understand by describing.

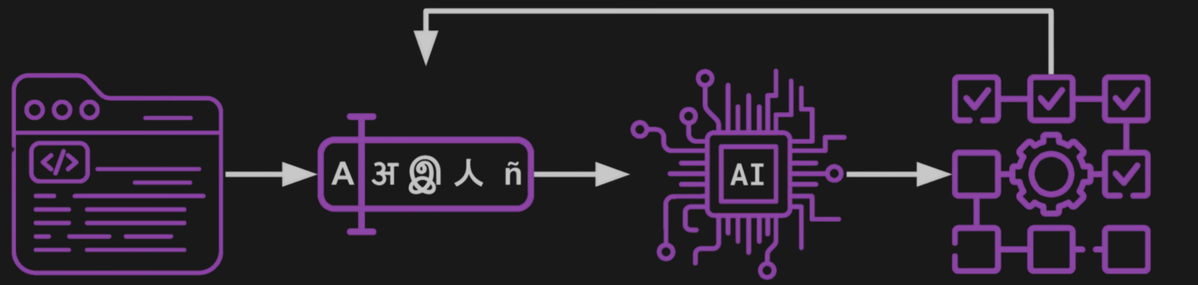
EXAMPLE QUESTION

```
def func_a(x):
    return f"Hello, {x}!"
```

"Describe what this function does in plain language."

Student: "This function takes a name and returns a greeting with that name."

The EiPL Pipeline: Grading Correctness



Learner reads code A shown to them

Learner explains the purpose of code in a language of their choosing

LLM generates code B from the student's description

Test suite checks equivalence of code A and code B

The EiPL Pipeline: Grading Abstraction

The system counts the number of distinct operations present to see if students are giving a line-by-line or high-level explanation.

YOUR EXPLANATION

"The function (1) *initializes a counter variable to zero. It* (2) *converts the input to lowercase and loops through each character and* (3) *checks if it is a vowel. If so, it* (4) *increments the counter. After the loop finishes, it* (5) *returns the counter."*

REFERENCE CODE

```
1  def func_b(x):
2    n = 0
3    for c in x.lower():
4      if c in 'aeiou':
5        n += 1
6    return n
```

5 segments detected

> threshold of **2**

High-level alternative: "Counts how many vowels appear in the text" → **1 segment**

Grading: Two Dimensions

Correctness × Abstraction Level

	Tests Pass	Tests Fail
High-level segments ≤ threshold	COMPLETE	INCOMPLETE
Too detailed segments > threshold	PARTIAL	INCOMPLETE

COMPLETE PARTIAL INCOMPLETE

Example 1: Good Submission

CODE

```
def func_b(x):
    n = 0
    for c in x.lower():
        if c in 'aeiou':
            n += 1
    return n
```

YOUR DESCRIPTION

"This function counts and returns the number of vowels in the given text, ignoring case."

RESULT

100% COMPLETE

3/3 variations passed

1 segment (threshold: 3)

*Captures **what** the function does in one sentence. No mention of loops, counters, or lowercase conversion.*

Example 2: Too Detailed

CODE (SAME)

```
def func_b(x):
    n = 0
    for c in x.lower():
        if c in 'aeiou':
            n += 1
    return n
```

YOUR DESCRIPTION

"The function initializes a counter variable to zero. It converts the input string to lowercase. It then loops through each character and checks if it is a, e, i, o, or u. If so, it increments the counter by one. After the loop finishes, it returns the counter."

RESULT

100% PARTIAL

3/3 variations passed

5 segments > threshold of 3

*Correct — but describes **how**, not **what**. Line-by-line narration lowers your grade even when tests pass.*

Example 3: Incorrect Submission

CODE

```
def func_c(x):
    y = []
    for z in x:
        if isinstance(z, list):
            y.extend(func_c(z))
        else:
            y.append(z)
    return y
```

YOUR DESCRIPTION

“This function removes duplicate elements from a nested list and returns the unique values.”

RESULT

33% INCOMPLETE

1/3 variations passed

Segmentation skipped

Wrong description → wrong code generated → tests fail. Abstraction analysis is never reached.

Prompt Problems: The I/O Variant

Same pipeline, different stimulus

EiPL

See CODE

- You read reference code
- Describe what it does
- Abstraction analysis: **ON**
- Graded on correctness + abstraction

```
def func_a(x):
  return f"Hello, {x}!"
```

Prompt Problem

See I/O

- You see input/output examples
- Infer the operation & describe it
- Abstraction analysis: **OFF**
- Graded on correctness only

```
$ python mystery.py
Enter number: 7
1 1 2 3 5 8 13
```

Same pipeline underneath — only the stimulus changes. Instead of code, you study input output pairs and infer what the program does.

Purple + Perplex = **Purplex**

AI-Powered Grading

Powered by Meta's Llama models via the Llama Impact Grant — fully automated correctness + abstraction scoring

Multilingual Support

Exploring ways to support natural-language programming activities in non-English environments

Progress & Analytics

Real-time dashboards for instructors — per-student, per-problem, per-section insights

Thank You

Questions? (dhsmith4@vt.edu)

purplex.org · Demo course: `ENGGEN131`

Supported by Meta Llama Impact Grant