Avoid frustration by teaching your students these **6** recurring patterns in **error messages**.

Applying Software Engineering Anti-patterns to Programming Error Messages

Context

- Programming error messages have **frustrated** and **misled** novice programmers for over 60 years
- What **concretely** makes an error message **misleading** or **unhelpful**?

What's an anti-pattern?

Anti-patterns are **common**, **recurring** features of programming error messages that either **do not help** the programmer or **actively prevent programmers from resolving their issue**

Catalog of anti-patterns

ONE-SIDED CONFLICT

highlights the location where the inconsistency was **discovered**, which might not be the **source of the problem**

IMPLICIT SUGGESTION

wording can be **misinterpreted as a suggestion** of how to fix the error when **no such suggestion is intended**

CASCADING ERROR

induced by a previously reported error; **not** as a **direct result of a true error**

TOKEN SOUP

shows **high density** of seemingly incoherent **tokens**

COMPILER-SPEAK

jargon only relevant to **programming language implementation**

CHARGED TERMINOLOGY

word choices are unjustifiably **harsh**, **graphic**, or **extreme**

Pilot Study

- N = **297** participants
- Quantitative survey: participants labelled error messages from Java, C, Python with anti-patterns

Results

- Programmers **clearly perceive anti-patterns**
  - Evidence that **anti-patterns** do **exist**!
- **Python has fewer anti-patterns** than C, Java

Eddie Antonio Santos, Ioannis Karvelas, Brett A. Becker

Becker et al.: 73 CS1 students: 21% of students believe error messages after the first always correspond to true error. 11% of students believe errors after the first never correspond to true programming errors.

CPCT+: on Blackbox 425,812±474 errors reported vs. Panic Mode 981,628

Denny et al. 2021: jargon has a strong negative correlation (r = .65) with programming error message comprehension.

Dy and Rodrigo: 15/28 (54%) `x expected` fixed with something other than `x`. 12/17 (71%) of `; expected` fixed with something other than a semicolon.

Kohn: 15% of novices insert a comma if presented with `missing comma`. 25% of novices insert `x` if given message `missing x`.

Barik et al.: Developers prefer extended arguments over simple claims, but prefer a resolution even more than that.

Denny et al. 2021: GCC token soup was 59th/60 (second least readable) message.

Denny et al. 2023: Codex: 57% (48%) provides correct error message explanations. 47% (33%) provides correct fixes.

- CHARGED TERMINOLOGY: `illegal start of expression` (90%), `illegal start of type` (88%)
- COMPILER SPEAK: `SyntaxError: EOL while scanning string literal` (61%)
- IMPLICIT SUGGESTION: `')' expected` (90%), `'.class' expected` (90%), `'(' expected` (87%), `';' expected` (86%), `<identifier> expected` (84%), `expected ')' before numeric constant` (84%), `expected declaration specifies or '…' before string constant` (81%), `expected expression before 'int'` (81%), `expected declaration specifiers or '…' before numeric constant` (78%), `expected declaration specifiers before 'if'` (78%), `class, interface, or enum expected` (76%), `expected ';' before int` (74%)
- TOKEN SOUP: `expected '=', ',', ';', 'asm' or '__attribute__' before '<' token` (96%)

```
fn multiply_pairs(pairs: &Vec<(f64, f64)>) -> Vec<f64> {
    pairs.iter().map(|(a, b)| a * b).collect()
}
```

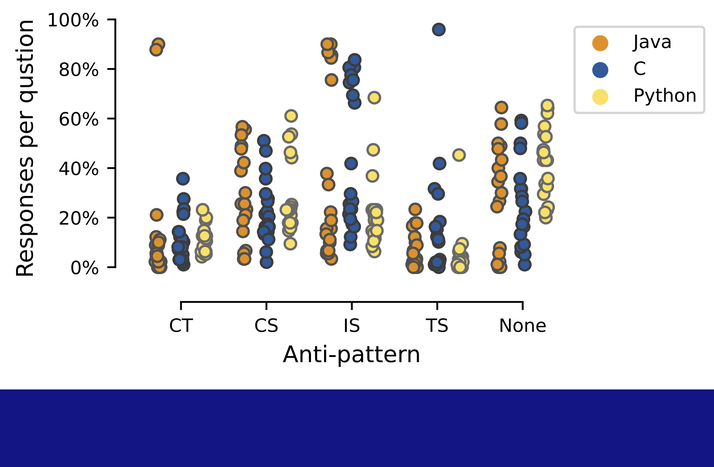
Execution — Close

Standard Error

```
   Compiling playground v0.0.1 (/playground)
error: expected one of `!`, `(`, `)`, `+`, `,`, `::`, or `<`, found `>`
 --> src/lib.rs:1:40
  |
1 | fn multiply_pairs(pairs: &Vec<(f64, f64)>) -> Vec<f64> {
  |                                        ^ expected one of 7 possible tokens

error: expected one of `->`, `where`, or `{`, found `>`
 --> src/lib.rs:1:42
  |
1 | fn multiply_pairs(pairs: &Vec<(f64, f64)>) -> Vec<f64> {
  |                                          ^ expected one of `->`, `where`, or `{`

error: could not compile `playground` due to 2 previous errors
```



Scan for more information →