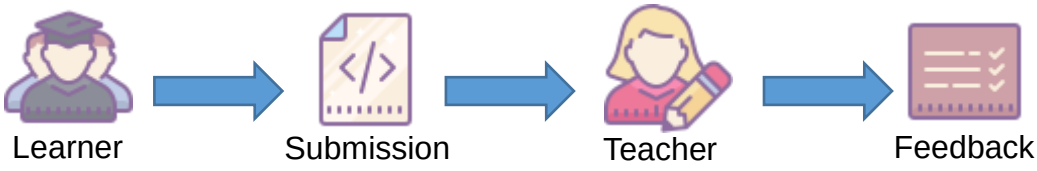
Pedal

Pedagogical Feedback Library for Python

https://pedal-edu.github.io



Giving Feedback Is Hard!

In a 2019 study by Denny et al, the #1 question among non-researching computing education teachers was as follows:

"How and when is it best to give students feedback on their code to improve learning?"

What do you do when you need to…?

- **Check** more precise conditions and patterns than just input/output
- **Revise** a hundred lines of complicated, dependent unit tests?
- **Track** what feedback was triggered for your students?
- **Regrade** your students' submissions with a changed grading script?

Pedal Makes It Easier

Pedal has a wide set of features:

- Structurally identify student mistakes using Python Code
- Reusable collection of student misconceptions
- Unit testing functions in classic assertion style
- Sophisticated sandboxing and mocking
- Type checking and flow analysis

And is meant for teachers:

- Compatible with any other autograding platform that supports arbitrary pure Python code execution
- Successfully deployed in BlockPy, Jupyter, VPL, and more!
- Fully Open-source, completely free

Hear our talk! Session 7E: Python Debugging Saturday 11:35AM-12PM

Read our paper! Pedal: An Infrastructure for Automated Feedback Systems (SIGCSE '20)

Get in touch! Luke Gusukuma (lukesg08@vt.edu) and Cory Bart (acbart@udel.edu)

Join us on GitHub! https://github.com/pedal-edu/pedal

Pedal

Pedagogical Feedback Library for Python

https://pedal-edu.github.io

Minimal boilerplate

```
from pedal.quick import *
setup_pedal()
```

No fuss installation

```
pip install pedal
```

Rich feedback primitives with metadata

```
gently("You have the wrong output", label="wrong_output")
compliment("Great progress!")
set_success()
give_partial(.5)
```

Enhanced error messages through our flow and type analyzer

TypeError
unsupported operand type(s) for +: 'int' and 'str'

Incompatible types
You used an addition operation with a number and a string on line 2. But you can't do that with that operator. Make sure both sides of the operator are the right type.

Detect structural mistakes with declarative searches

```
if find_matches("answer = 42"):
    explain("You may not simply embed the answer.", label="cheated")

if not find_matches("for _item_ in ___: _item_"):
    gently("You are not using the iteration variable", label="no_iter")
```

All your classic assertions, plus a bunch of new ones

```
assert_equal(student.call('sum', [1,2,3]), 6)
assert_prints(student.call("say_hello"), "hello")
match_signature('compare', int, str, returns=bool)
```

Sophisticated sandboxing to better contextualize errors

I ran:

```
    sum([1, 2, 3])
```

But your code had the following error on line 17 in <sum>

…