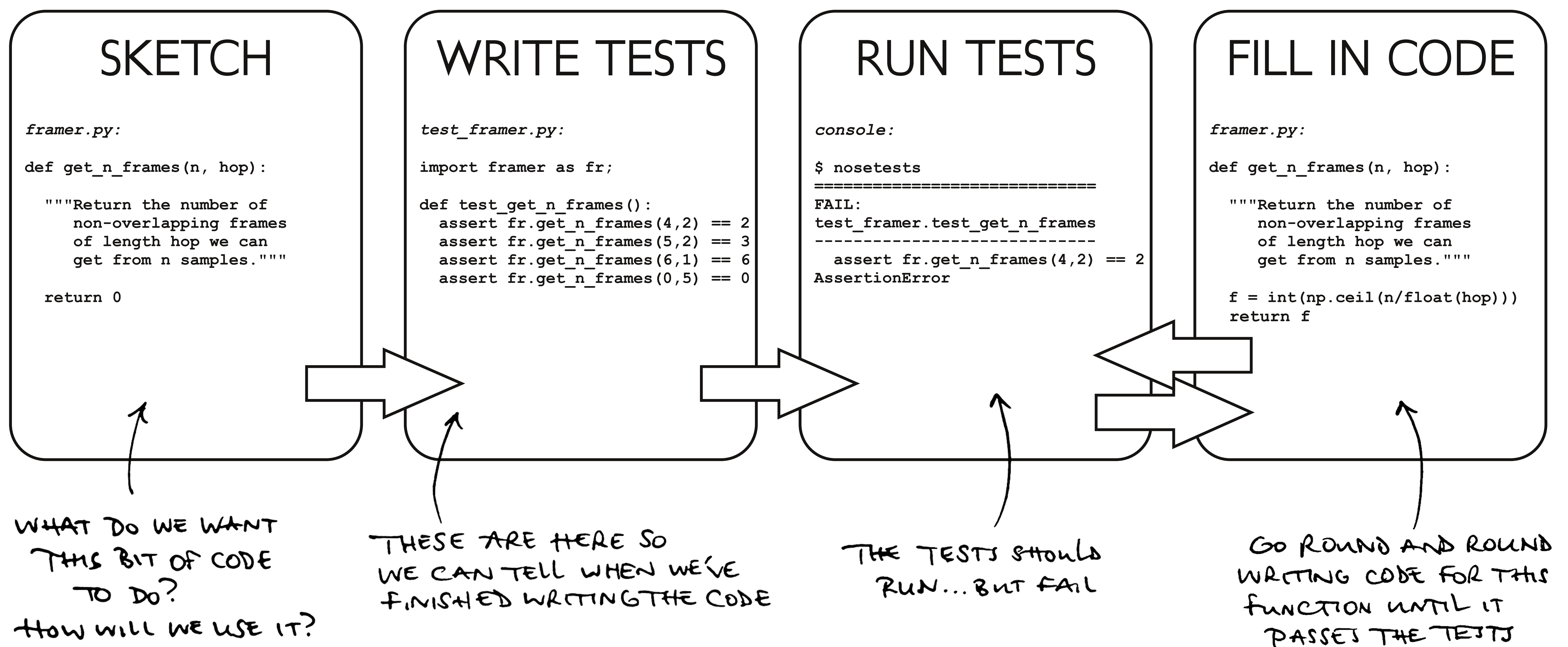


HOW CAN TEST-DRIVEN DEVELOPMENT FIT WITH RESEARCH SOFTWARE?

WHAT IS TEST-DRIVEN DEVELOPMENT?



HOW DO YOU DO THAT WITH RESEARCH CODE?

LOW-LEVEL STUFF

When the task is clear enough that you can work out the answers “by hand” for simple inputs:

- Use simple synthetic test data, not real-world data
- Include trivial or null inputs
- Test any utility code like framing, file I/O
- What's the simplest input that makes it fail?

Try to arrange work into units small enough to be reasoned about this way

EXPERIMENTAL STUFF

Not easy to work out what results are expected?

- You're testing the implementation—not the quality of the method it uses (that's for the paper)
- Seek “minimum valid behaviour” for method
- Existing methods, no matter how bad, should pass your tests if implemented correctly

Tests provide the basic code safety net, enabling wild experimental changes without screwing up

WATCH



Video from ISMIR 2012 tutorial
<http://bit.ly/UrPFGF>



Code written during that tutorial
<http://bit.ly/SLfMuD>

READ



Real-world C++ audio analysis example
<http://bit.ly/Z4XpWE>



More handouts and guides
<http://bit.ly/SLfUuc>