Type Errors

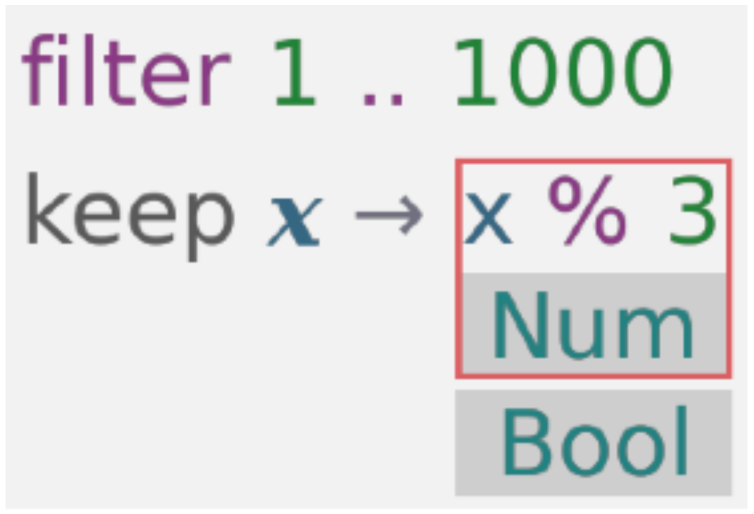
Static typing has many benefits, but usually it adds verbosity and requires deciphering complicated type errors.

```
In file included from /usr/include/c++/4.6/algorithm:63:0,
                 from error_code.cpp:2:
/usr/include/c++/4.6/bits/stl_algo.h: In function '_RandomAccessIterator std::__find(_RandomAccessIterator, _RandomAccessIterator, const _Tp&,
std::random_access_iterator_tag) [with _RandomAccessIterator = __gnu_cxx::__normal_iterator*, std::vector > >, _Tp = int]':
/usr/include/c++/4.6/bits/stl_algo.h:4403:45:   instantiated from '_IIter std::find(_IIter, _IIter, const _Tp&) [with _IIter = __gnu_cxx::__normal_iterator*,
std::vector > >, _Tp = int]'
error_code.cpp:8:89:   instantiated from here
/usr/include/c++/4.6/bits/stl_algo.h:162:4: error: no match for 'operator==' in '__first.__gnu_cxx::__normal_iterator::operator* [with _Iterator = std::vecto *,
_Container = std::vector >, __gnu_cxx::__normal_iterator::reference = std::vector&]() == __val'
/usr/include/c++/4.6/bits/stl_algo.h:162:4: note: candidates are:
/usr/include/c++/4.6/bits/stl_pair.h:201:5: note: template bool std::operator==(const std::pair&, const std::pair&)
/usr/include/c++/4.6/bits/stl_iterator.h:285:5: note: template bool std::operator==(const std::reverse_iterator&, const std::reverse_iterator&)
```

A snippet from a long C++ type error (from https://codegolf.stackexchange.com/a/10470)

Type *Mismatches*

When creating an expression in a place where its type doesn't match, a fragment is created. The red frame indicates a type mismatch, with the fragment type and expected type both displayed.



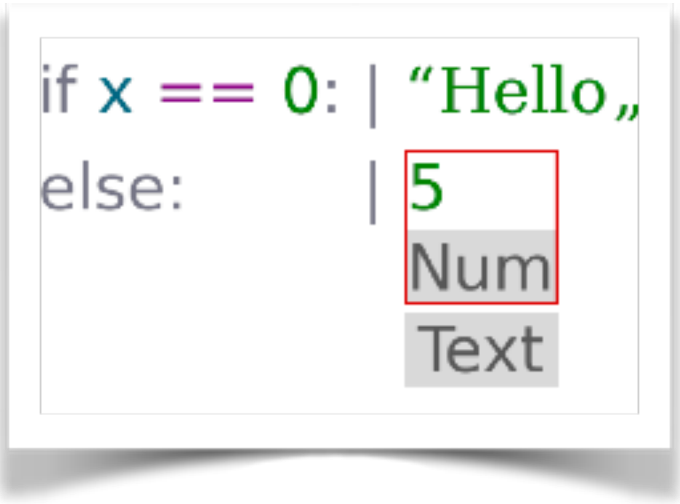
```
filter 1 .. 1000
keep x → x % 3
         Num
         Bool
```

Blame Assignment

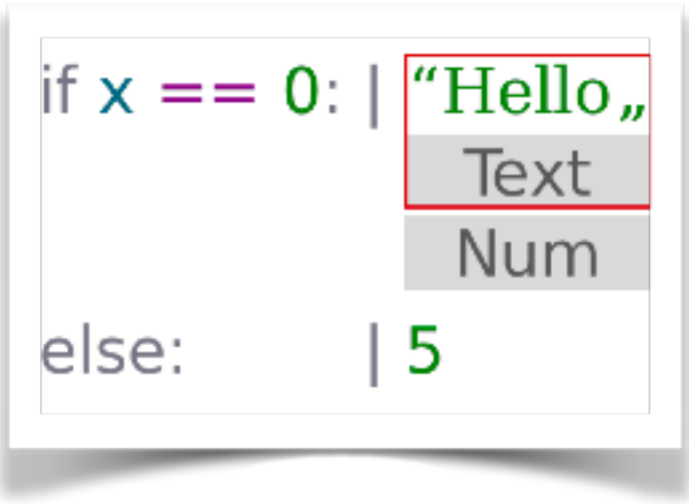
Traditionally compilers arbitrarily assign blame - often editing one piece of code results in an error somewhere seemingly unrelated.

Lamdu uses the order in which code is written for to assign type mismatches to newly written code.

"Hello" typed first



```
if x == 0: | "Hello„
else:      | 5
             Num
             Text
```



```
if x == 0: | "Hello„
             Text
             Num
else:      | 5
```

5 typed first