PyPy – a progress report

ACCU 2006/Python UK, Oxford

Michael Hudson mwh@python.net
Heinrich-Heine-Univeristät Düsseldorf

What is PyPy?

- PyPy is:
  - An implementation of Python, written in Python
  - An open source project
  - A STREP (“Specific Targeted REsearch Project”), partially funded by the EU
  - A lot of fun!

Demo

- We can currently produce a binary that looks very much like CPython to the user
- It’s fairly slow (around the same speed as Jython)
- Can also produce binaries that are more capable than CPython -- stackless, thunk, ...

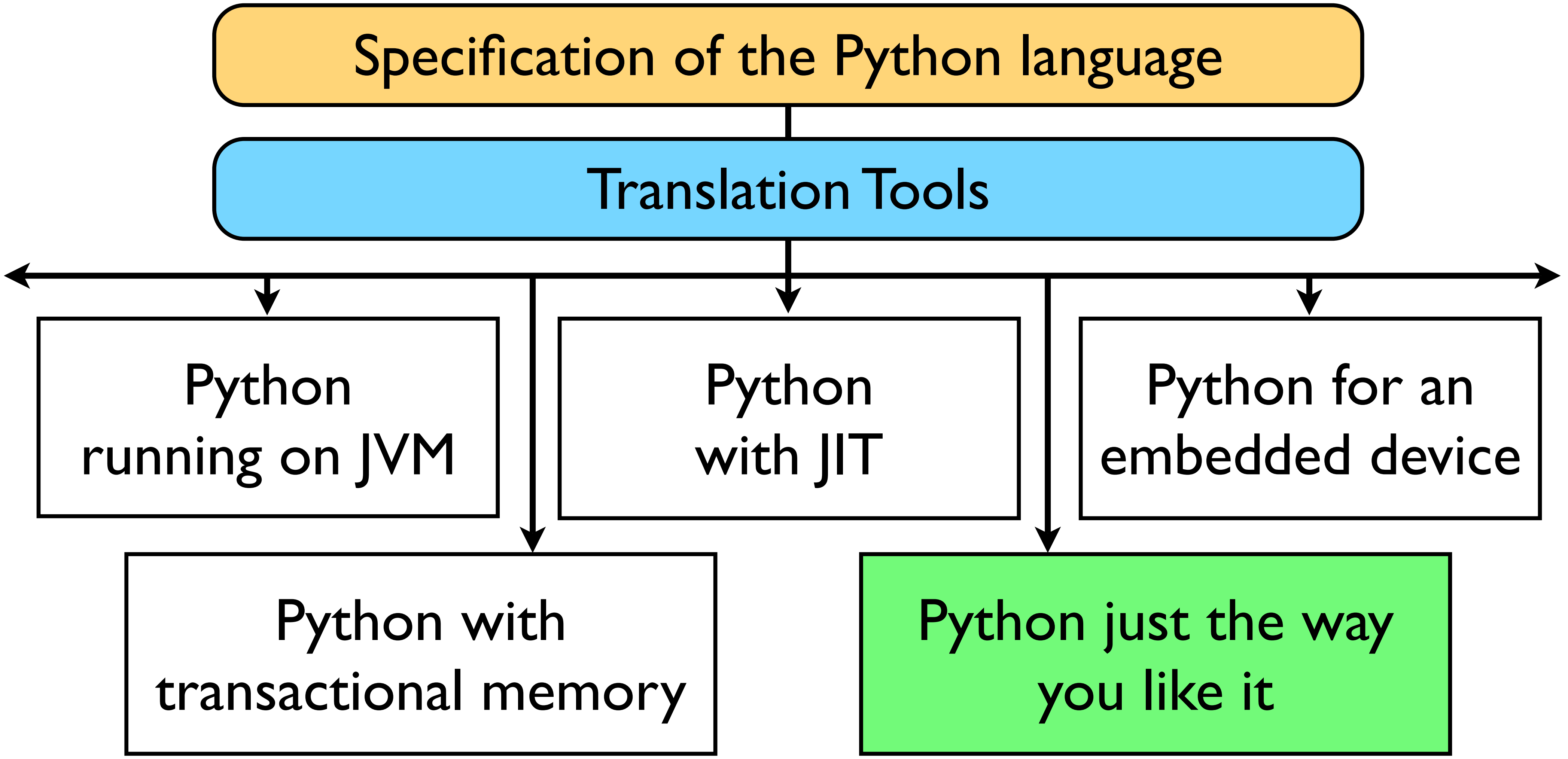
Motivation

- PyPy grew out of a desire to modify/extend the *implementation* of Python, for example to:
  - increase performance (psyco style JIT compilation, better garbage collectors)
  - add expressivity (stackless-style coroutines, logic programming)
  - ease porting (to new platforms like the JVM or CLR or to low memory situations)

Lofty goals, but first...

- CPython is hardly a bad implementation of Python but:
  - it's written in C, which makes porting to, for example, the the CLR hard
  - while psyco and stackless exist, they are very hard to maintain as Python evolves
  - some implementation decisions would be very hard to change (e.g. refcounting)

Enter the PyPy platform



Specification of the Python language
Translation Tools
Python running on JVM
Python with JIT
Python for an embedded device
Python with transactional memory
Python just the way you like it

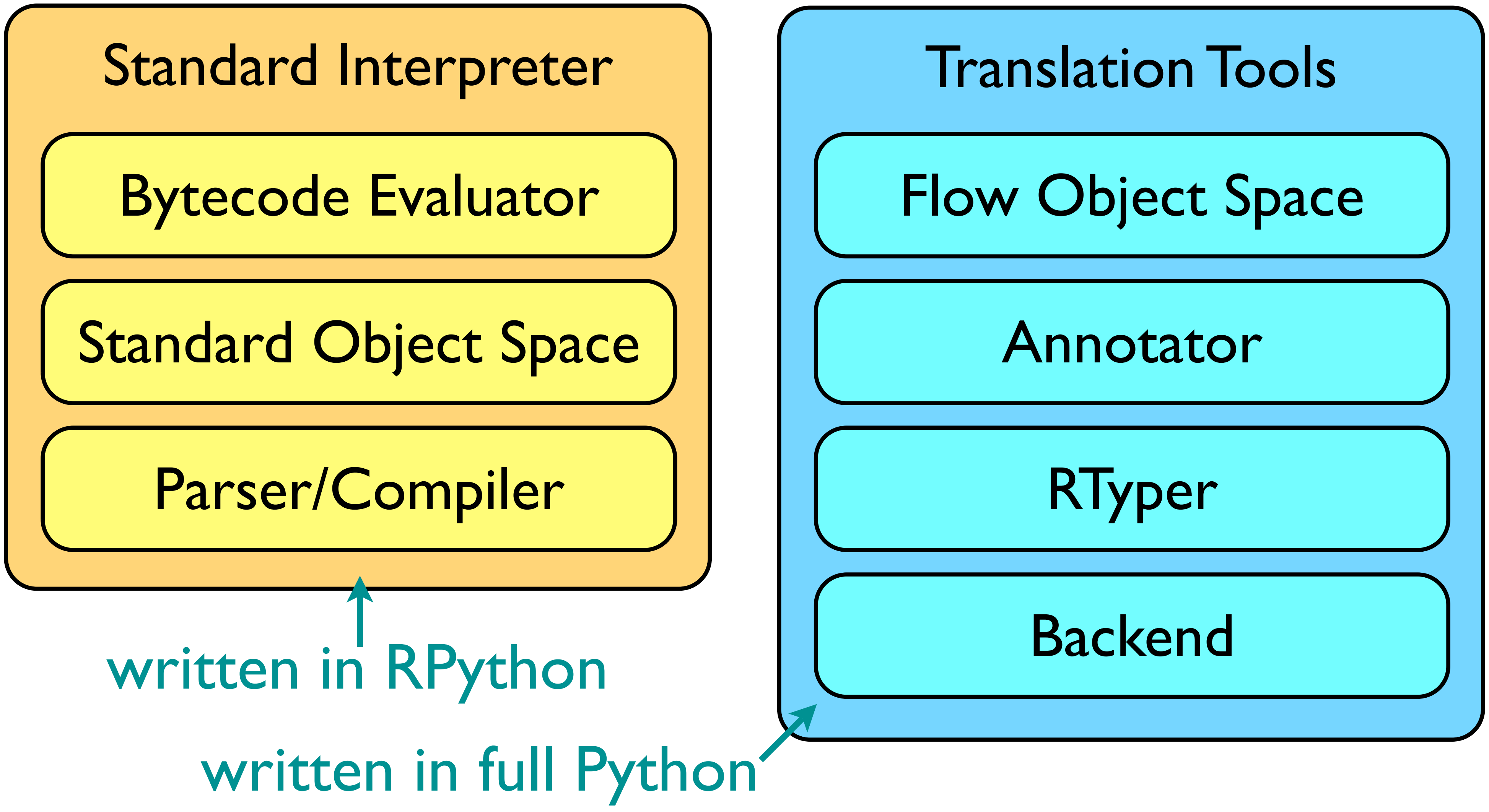
How do you specify the Python language?

- The way we did it was to write an interpreter for Python in *RPython* – a subset of Python that is amenable to analysis
- This lets us write unit tests for our specification/implementation that run on top of CPython
- Can also test entire specification/ implementation in same way

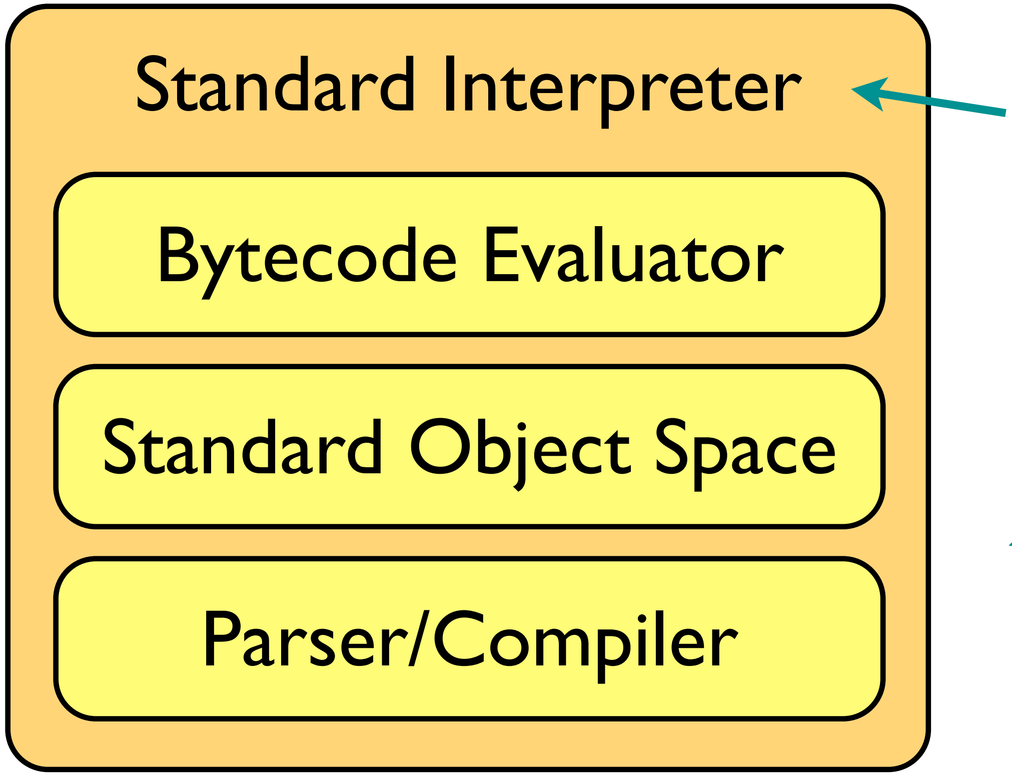
The "What is RPython?" question

- Restricted Python, or RPython for short, is a subset of Python that is static enough for our analysis toolchain to cope with
- First and foremost it *is* Python
- Definition is basically "what our tools accept" – so changes as toolchain does
- Somewhat Java-like – classes, methods, no pointers

In more detail...



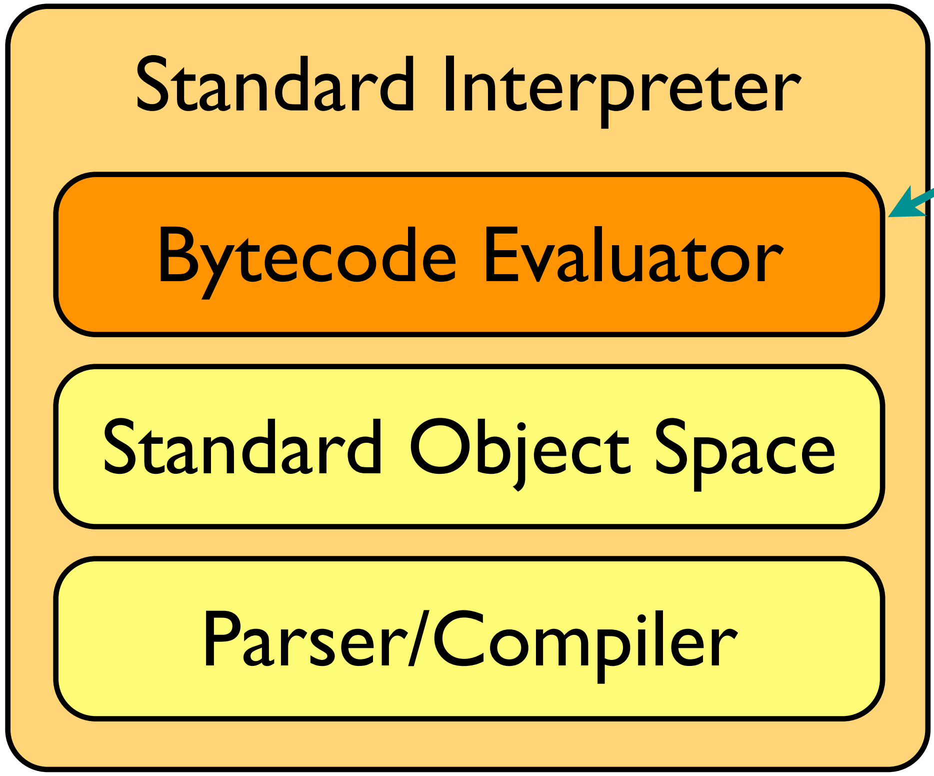
The Standard Interpreter



The standard interpreter does roughly speaking the same job as CPython does

CPython can be split along the same lines with enough imagination – hardly a coincidence!

The Standard Interpreter

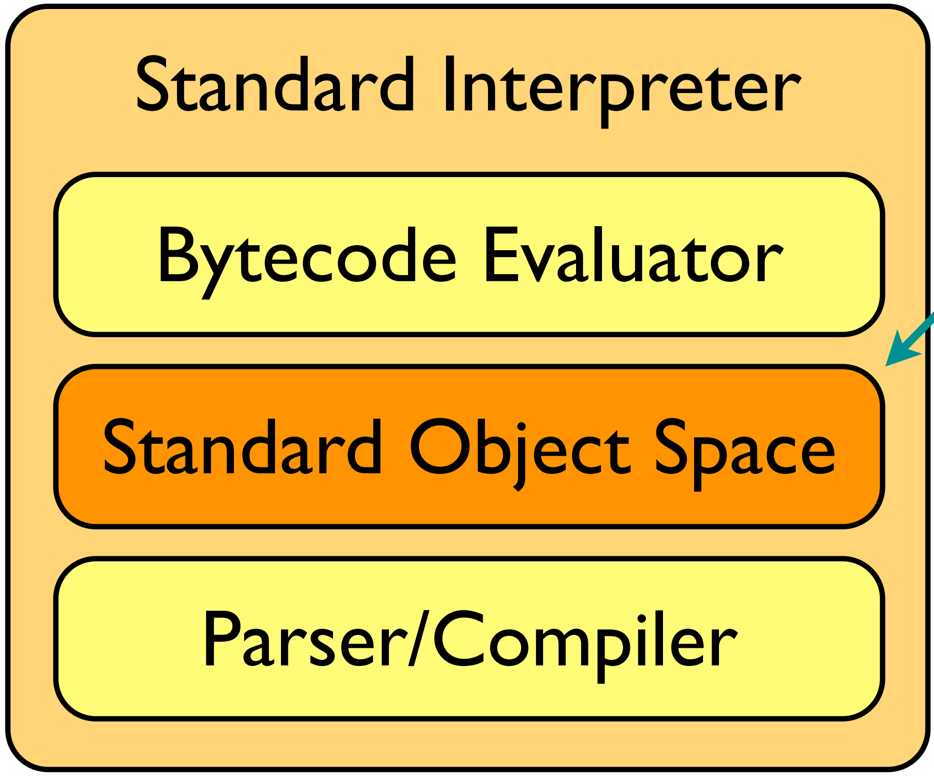


The bytecode evaluator evaluates the same bytecodes as CPython but treats objects as black boxes – it doesn't care if they are Python-like values, abstract Variables or even fruit

$\boxed{2} + \boxed{3} = \boxed{5}$

$\boxed{\texttt{Variable}} + \boxed{\texttt{Constant}} = \boxed{\texttt{Variable}}$

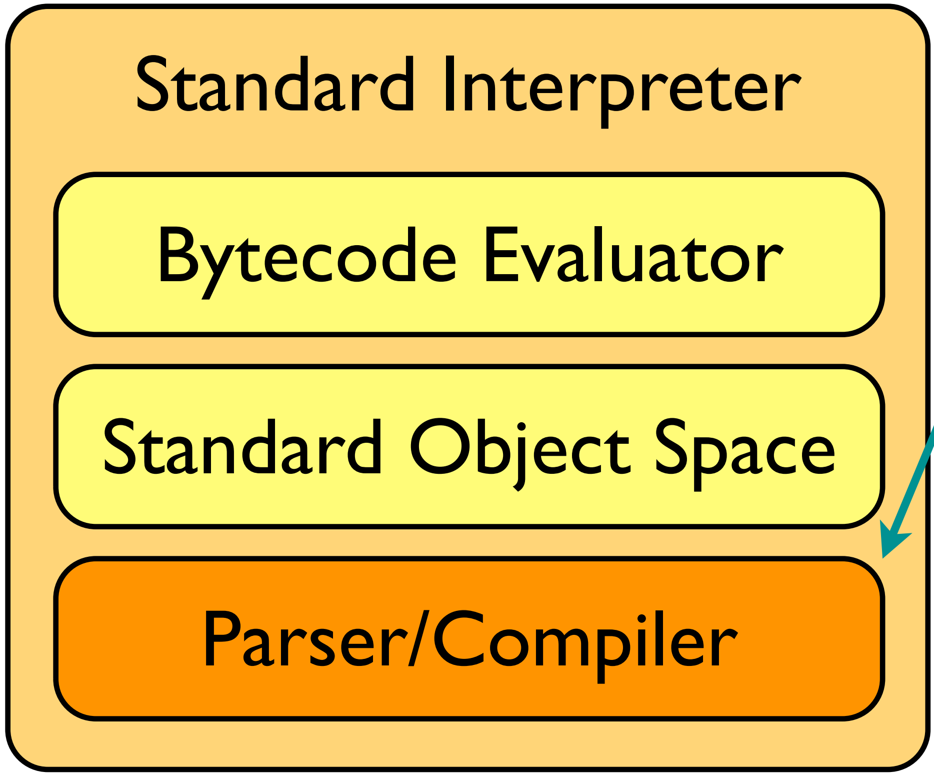
The Standard Interpreter



The Standard Object Space implements objects that look very much like CPython's – integers, lists, dictionaries, classes, etc

(bit different on the inside though)

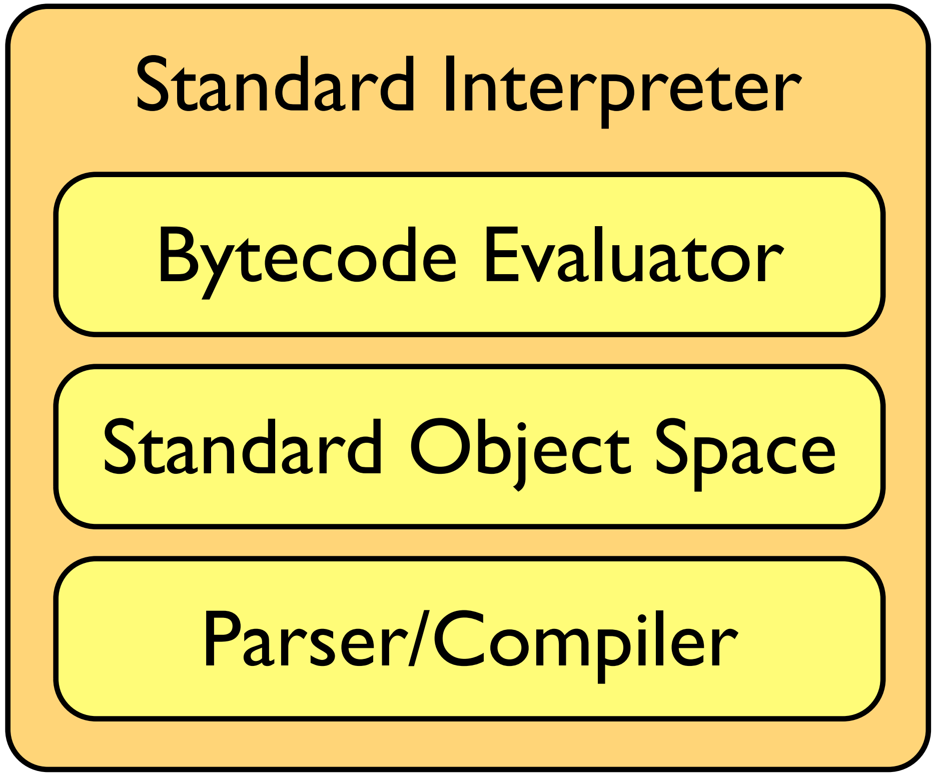
The Standard Interpreter



The parser and compiler, well, parse Python code and compile to the same bytecode as CPython uses

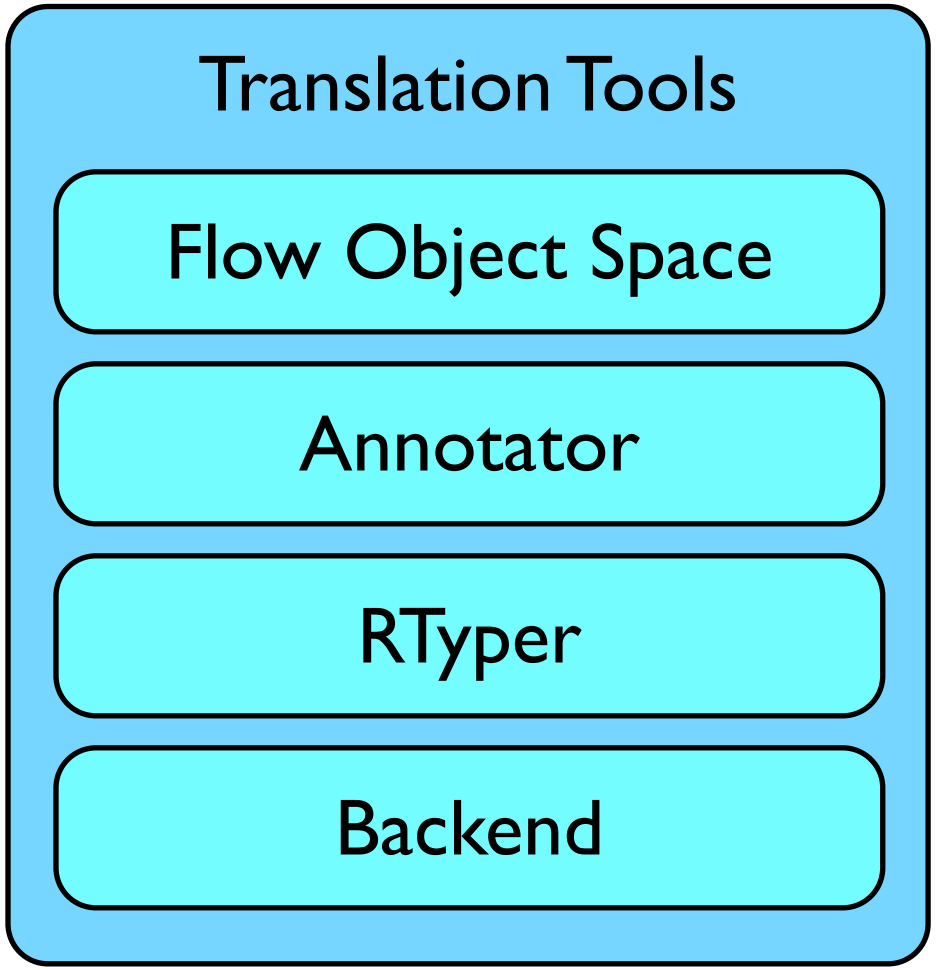
Will sometime soon allow runtime modification of the grammar of the language

The Standard Interpreter

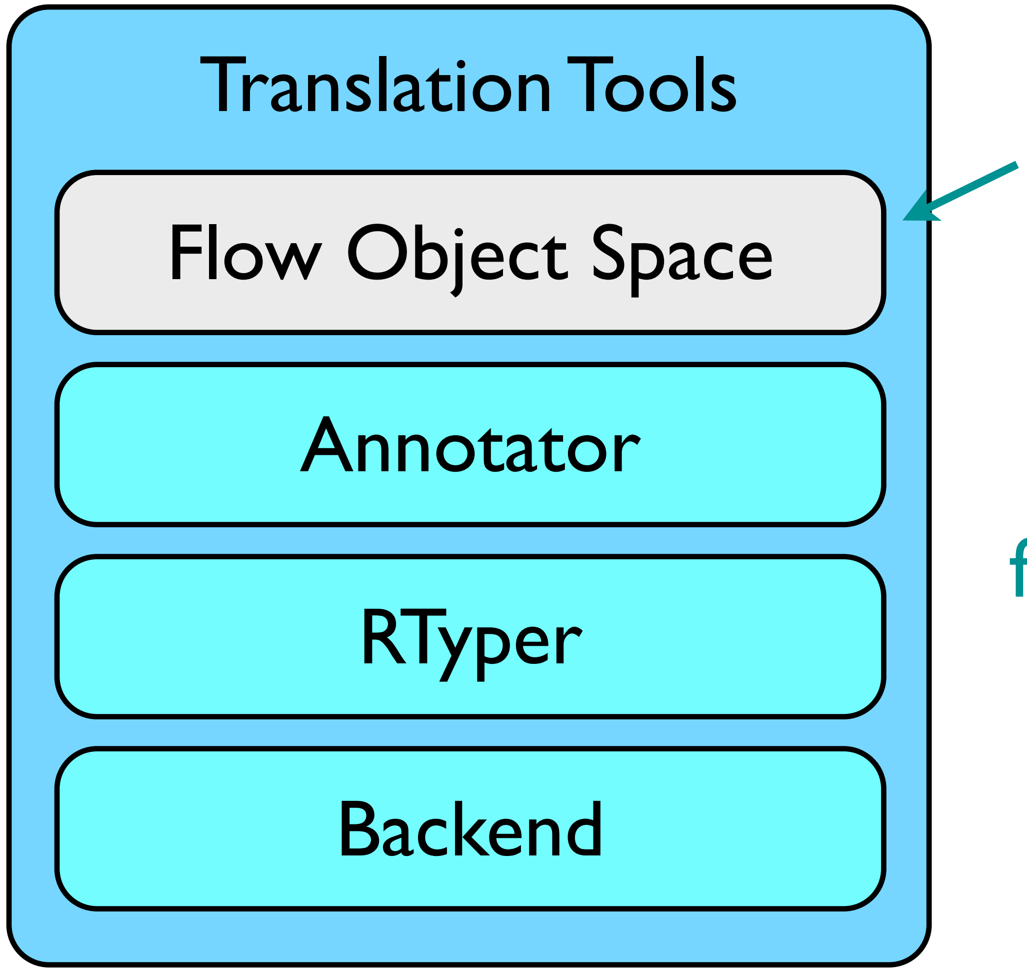


The standard interpreter is pretty stable now, implementing Python 2.4.2, apart from some work to come on the parser and compiler

Translation Tools



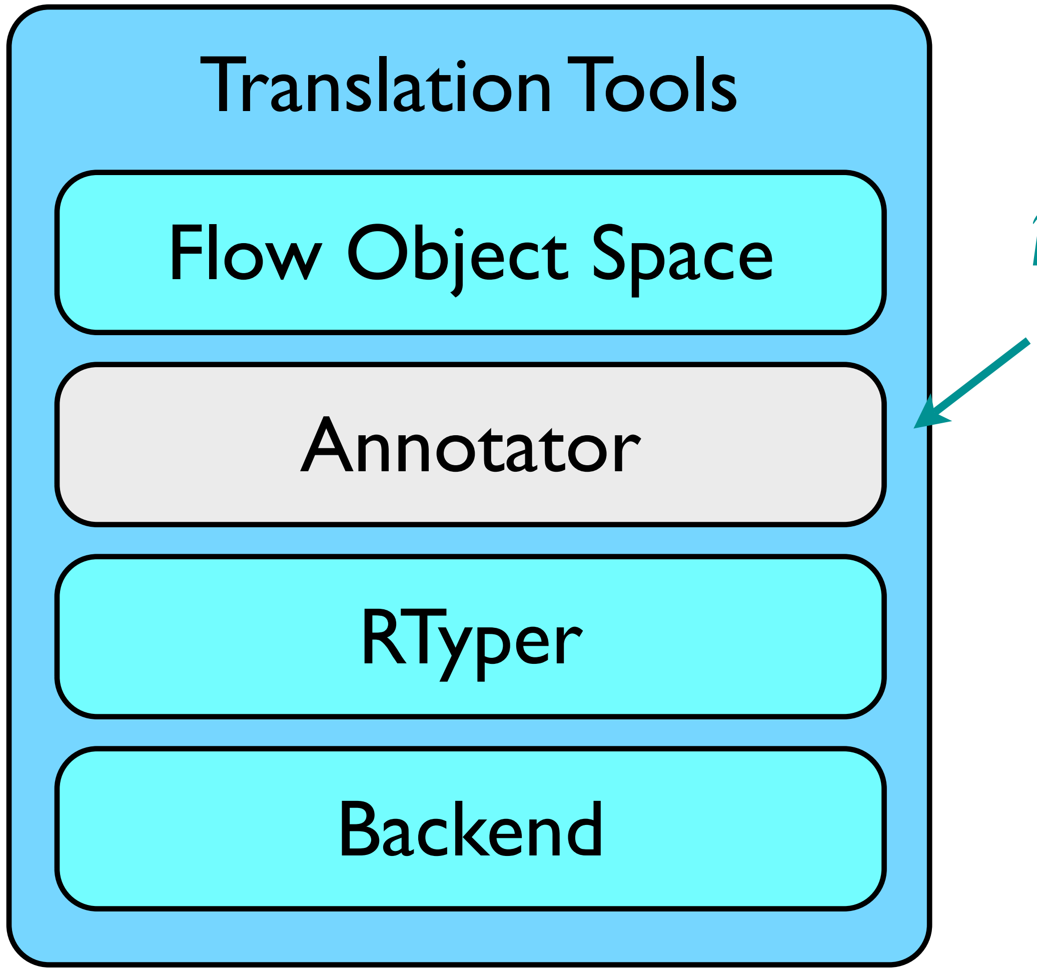
Translation Tools



Analyzes a single code object to deduce control flow

We have a funky pygame flow graph viewer that we use to view these flow graphs
(demo)

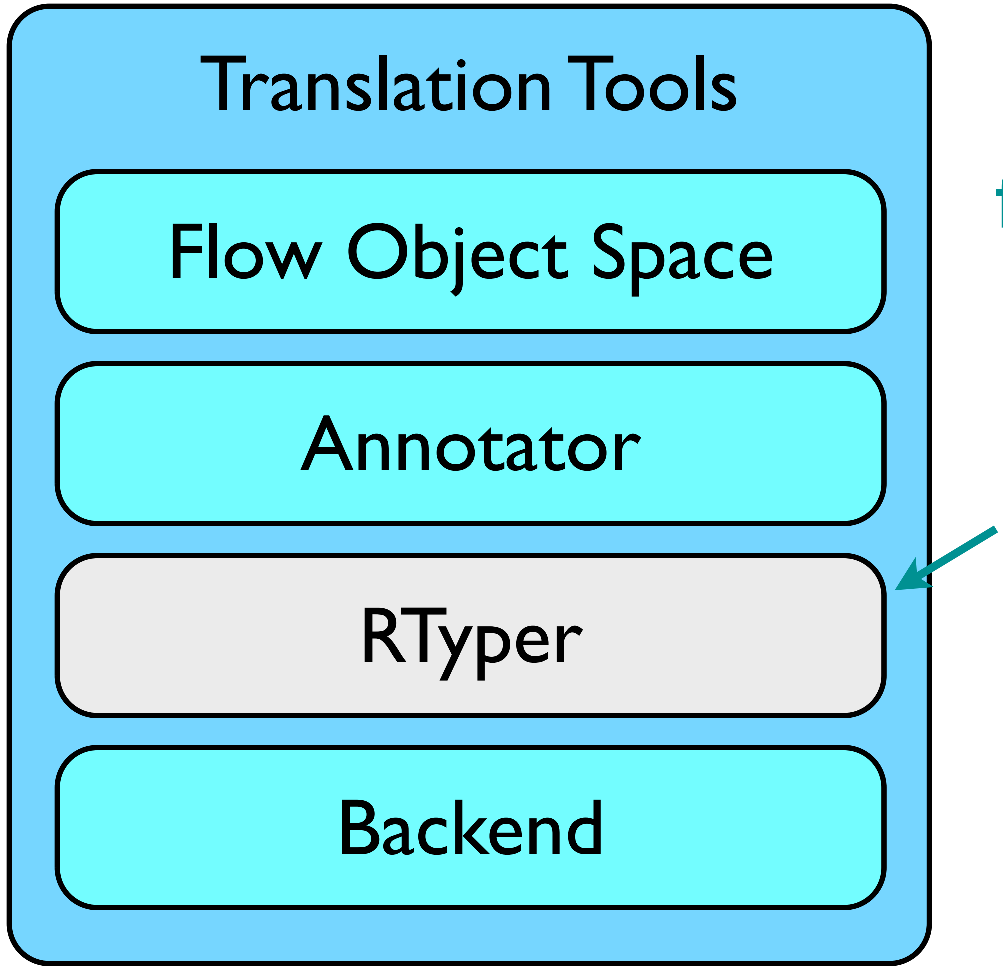
Translation Tools



Analyzes an *entire program* to deduce type and other information

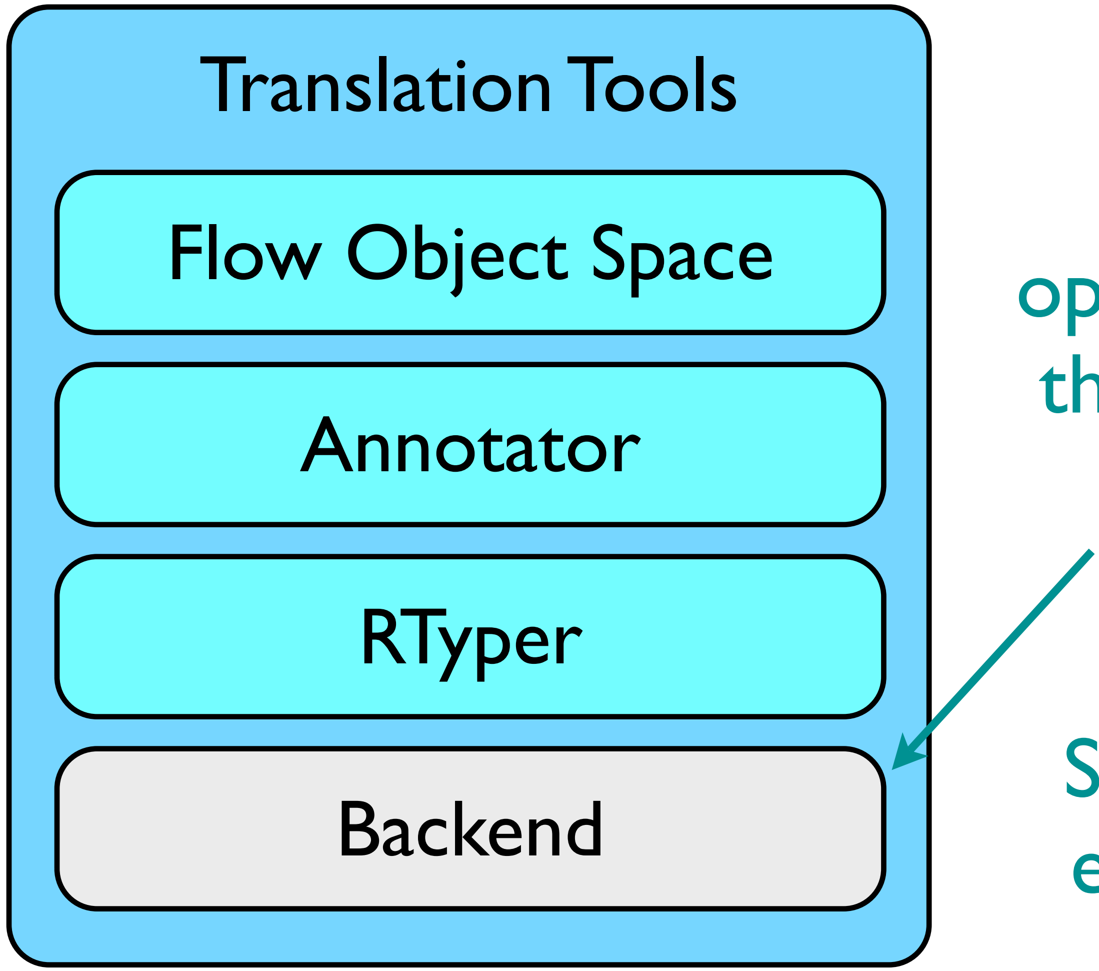
Uses abstract interpretation, rescheduling and other funky stuff

Translation Tools



Uses the information found by the annotator to decide how to lay out the types used by the input program in memory, and translates high level operations to lower level more pointer-ish operations

Translation Tools



Translates low level operations and types from the RTyper to (currently) C, JavaScript or LLVM code

Sounds like it should be easy, in fact a bit painful