ChocoPy

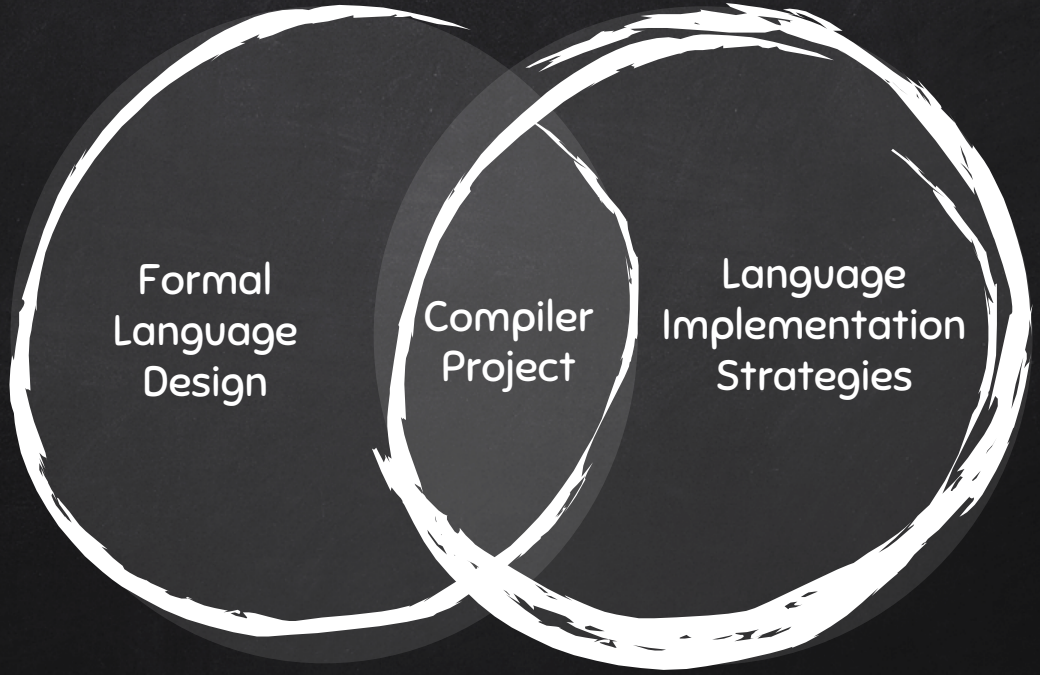
A Programming Language for Compilers Courses

Rohan Padhye, Koushik Sen, Paul N. Hilfinger
(UC Berkeley)

The set-up

CS164 at UC Berkeley

Course content



12 weeks

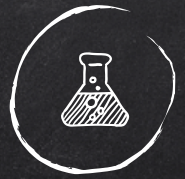
Build a full Compiler

Translates valid <source language> programs into <assembly language>

(working in teams)

ChocoPy

A language that screams “Compile Me!”



Why did we develop ChocoPy?

Familiarity

Specification

Artifacts

Modern Target

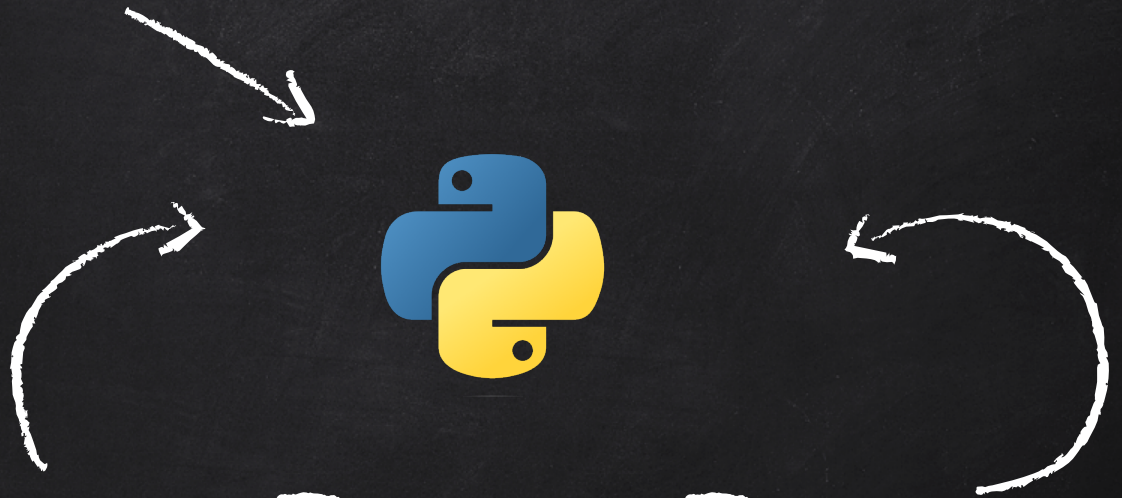
The Language

What does a ChocoPy program look like?

ChocoPy Programs

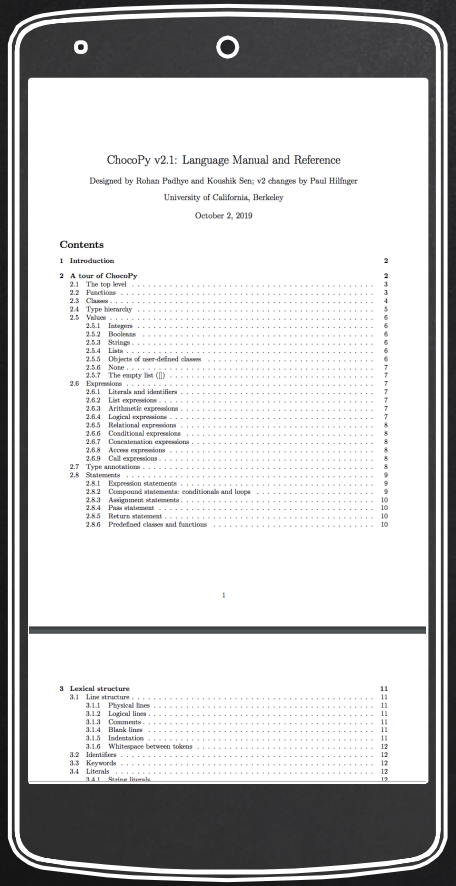
```
def contains(items:[int], x:int) -> bool:
  i:int = 0

  while i < len(items):
    if items[i] == x:
      return True
    i = i + 1

  return False

if contains([4, 8, 15, 16, 23], 15):
 print("Item found!")
```

```
class Animal(object):
  makes_noise:bool = False

  def make_noise(self: "Animal"):
    if self.makes_noise :
      print(self.sound())

  def sound(self: "Animal") -> str:
    return "???"

class Cow(Animal):
  def __init__(self: "Cow"):
    self.makes_noise = True

  def sound(self: "Cow") -> str:
    return "moo"
```



ChocoPy ⊂ Python

Every valid ChocoPy program
can be executed in a Python interpreter
(to get same result)



ChocoPy v2.1: Language Manual and Reference

Designed by Rohan Padhye and Koushik Sen; v2 changes by Paul Hilfinger

University of California, Berkeley

October 2, 2019

Contents

1

Language Reference Manual

Comprehensive (36 page) specification of the ChocoPy language.

- Lexical structure of programs
- Formal grammar of syntax
- Typing rules
- Formal operational semantics

Formal Typing rules & Operational Semantics

VAR-INIT

$$\frac{O(id) = T \quad O, M, C, R \vdash e_1 : T_1 \quad T_1 \leq_a T}{O, M, C, R \vdash id{:}T = e_1}$$

ATTR-INIT

$$\frac{M(C, id) = T \quad O, M, C, R \vdash e_1 : T_1 \quad T_1 \leq_a T}{O, M, C, R \vdash id{:}T = e_1}$$

ATTR-READ

$$\frac{O, M, C, R \vdash e_0 : T_0 \quad M(T_0, id) = T}{O, M, C, R \vdash e_0.id : T}$$

LIST-SELECT

$$\frac{O, M, C, R \vdash e_1 : [T] \quad O, M, C, R \vdash e_2 : int}{O, M, C, R \vdash e_1[e_2] : T}$$

RETURN-E

$$\frac{O, M, C, R \vdash e : T \quad T \leq_a R}{O, M, C, R \vdash \texttt{return}\ e}$$

RETURN

$$\frac{\texttt{<None>} \leq_a R}{O, M, C, R \vdash \texttt{return}}$$

VAR-READ

$$\frac{E(id) = l_{id} \quad S(l_{id}) = v}{G, E, S \vdash id : v, S, \_}$$

VAR-ASSIGN-STMT

$$\frac{G, E, S \vdash e : v, S_1, \_ \quad E(id) = l_{id} \quad S_2 = S_1[v/l_{id}]}{G, E, S \vdash id = e : \_, S_2, \_}$$

LIST-SELECT

$$\frac{G, E, S_0 \vdash e_1 : v_1, S_1, \_ \quad G, E, S_1 \vdash e_2 : int(i), S_2, \_ \quad v_1 = [l_1, l_2, \ldots, l_n] \quad 0 \leq i < n \quad v_2 = S_2(l_{i+1})}{G, E, S_0 \vdash e_1[e_2] : v_2, S_2, \_}$$

RETURN-E

$$\frac{G, E, S \vdash e : v, S_1, \_}{G, E, S \vdash \texttt{return}\ e : \_, S_1, v}$$

ChocoPy: Language features

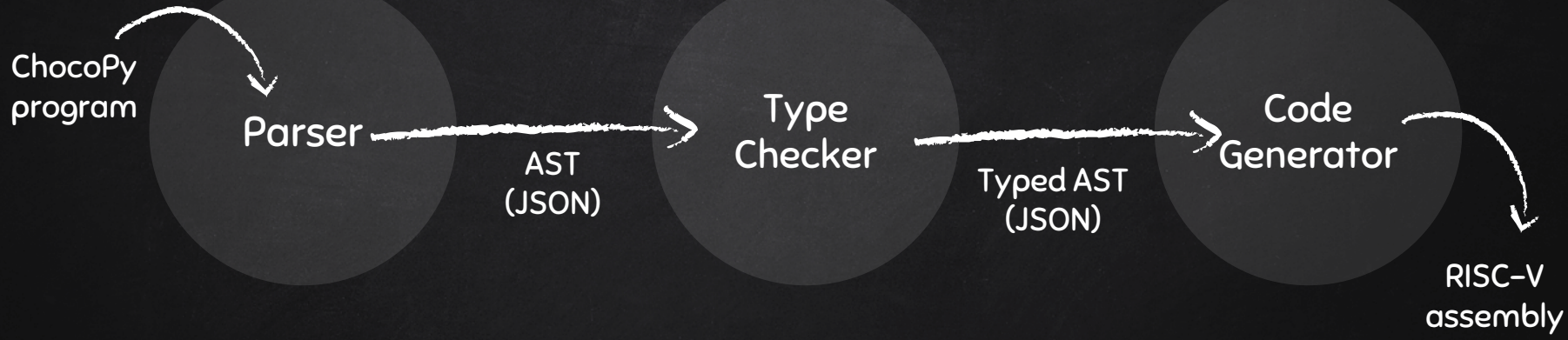
- Static typing with nominal subtyping
- Primitive types, objects, lists, None
- Top-level and nested functions
- Global, local, nonlocal variables
- Classes, attributes, methods

- Native dictionaries
- List comprehension
- Exceptions
- Default arguments
- Lambdas, closures

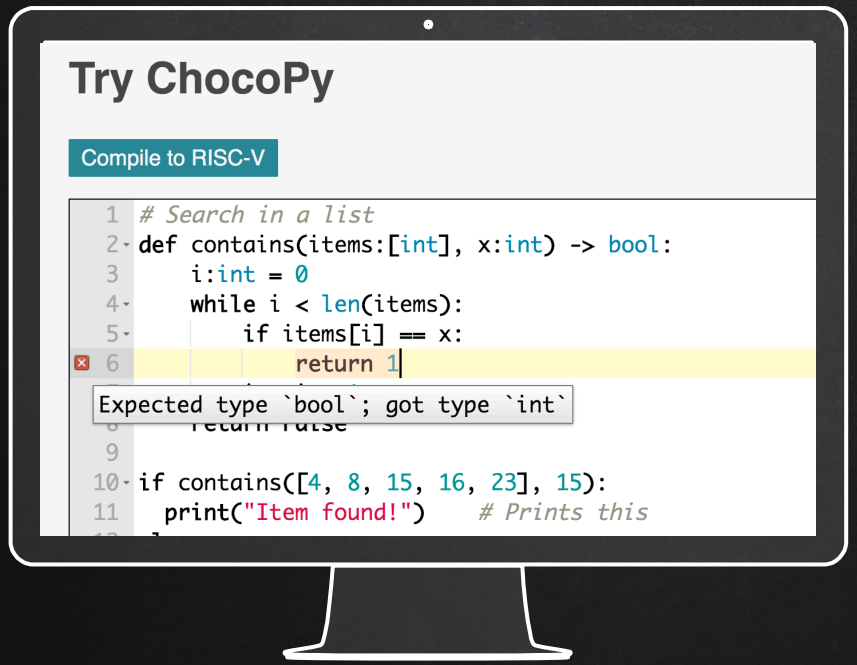
The Project

What do students work with?

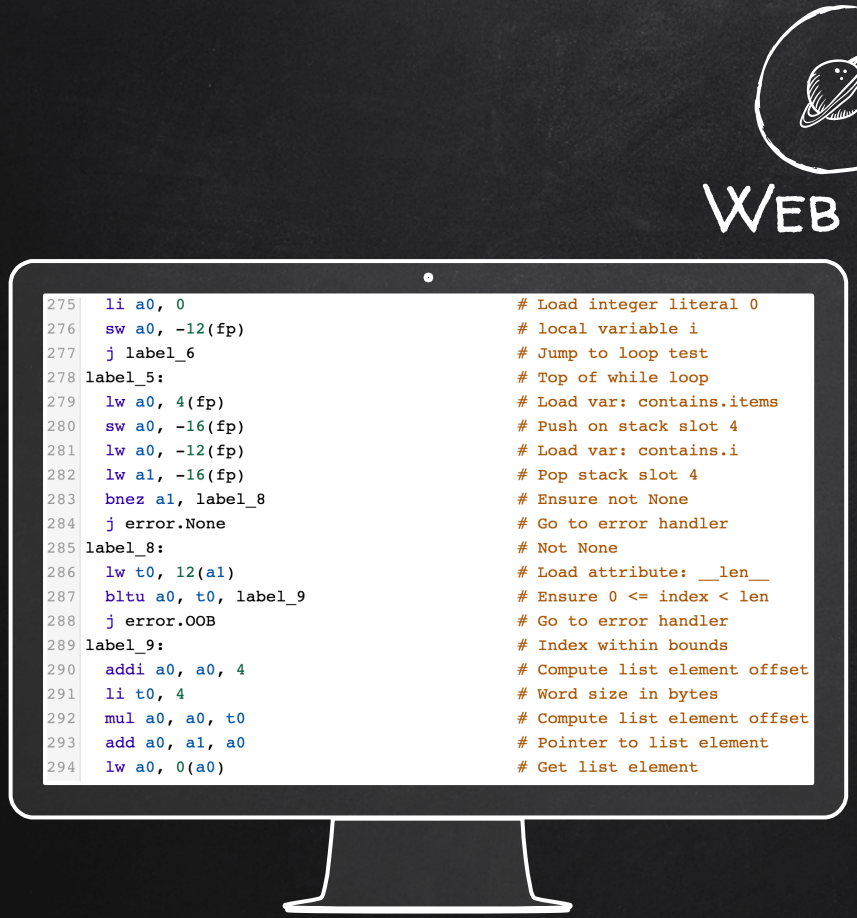
A compiler in 3 parts (=assignments)



Web IDE



- Powered by student or reference compiler
- Self-documenting assembly
- Step-through debugging in browser

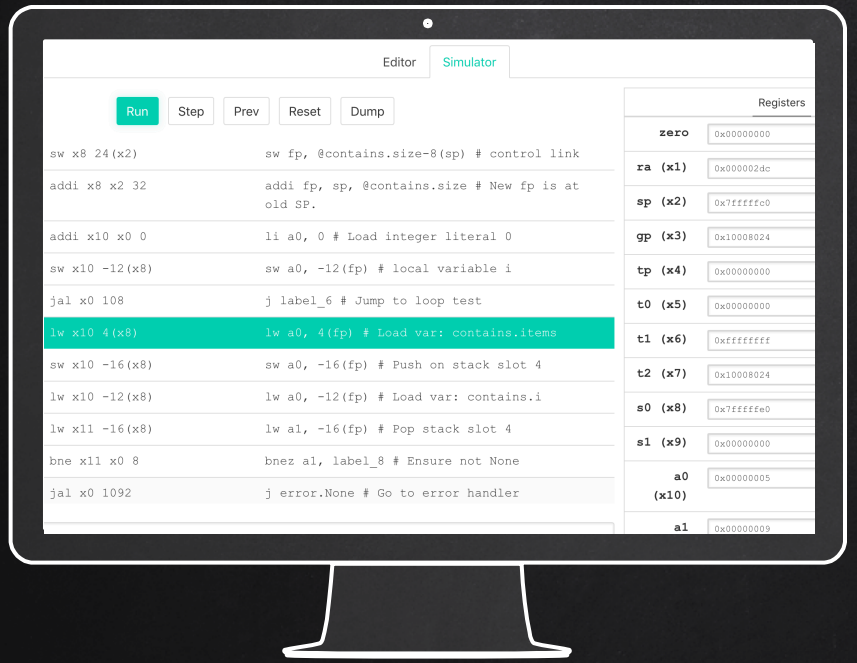


Web IDE

```
275   li a0, 0                          # Load integer literal 0
276   sw a0, -12(fp)                    # local variable i
277   j label_6                         # Jump to loop test
278 label_5:                            # Top of while loop
279   lw a0, 4(fp)                      # Load var: contains.items
280   sw a0, -16(fp)                    # Push on stack slot 4
281   lw a0, -12(fp)                    # Load var: contains.i
282   lw a1, -16(fp)                    # Pop stack slot 4
283   bnez a1, label_8                  # Ensure not None
284   j error.None                      # Go to error handler
285 label_8:                            # Not None
286   lw t0, 12(a1)                     # Load attribute: __len__
287   bltu a0, t0, label_9              # Ensure 0 <= index < len
288   j error.OOB                       # Go to error handler
289 label_9:                            # Index within bounds
290   addi a0, a0, 4                    # Compute list element offset
291   li t0, 4                          # Word size in bytes
292   mul a0, a0, t0                    # Compute list element offset
293   add a0, a1, a0                    # Pointer to list element
294   lw a0, 0(a0)                      # Get list element
```

- Powered by student or reference compiler
- Self-documenting assembly
- Step-through debugging in browser

Web IDE



- Powered by student or reference compiler
- Self-documenting assembly
- Step-through debugging in browser

Assignment resources

Language reference manual

Reference compiler

Web IDE

Java-based starter code

RISC-V implementation guide

Auto-grader

Experience

How do it go?

Takeaways from 2 ½ semesters

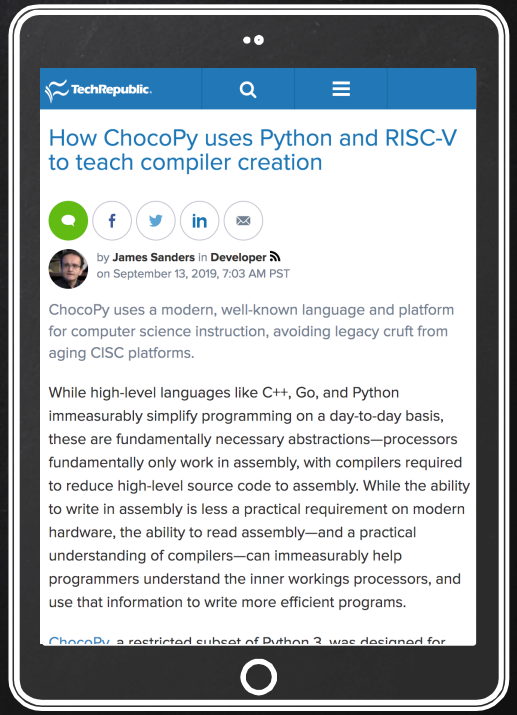
- Near-zero learning curve
- Language extensions intuitive
- Web-based IDE works well
- Student compilers beat CPython

- Lots of text to read
- Project is quite large
- Auto-grading error cases is tricky

Media

Hacker News (Front Page #4)

TechRepublic.com



CHOCOPY.ORG

Running your own course?
instructors@chocopy.org