Teaching Programming to Novices Using the codeBoot Online Environment

Marc Feeley and Olivier Melançon
Why codeBoot?

● Fall 2020 semester was done through distance learning

● “Programmation 1” course
  ○ Mandatory programming course of undergraduate CS degree
  ○ Large class of students with little experience in programming
  ○ Fall 2020 was the first time Python was used to teach
Why codeBoot?

We needed an environment with:

- Simple UI with no installation required (to avoid overwhelming novices)
- Fine-grained single-stepping at subexpression level
- Shareable state using hyperlinks that can be embedded in documents (PDF, HTML, ...)

An IDE aimed at professional developers can be overwhelming for novices
Existing tools and environments

Python programming environment

- PyCharm [JetBrains, 2014]
  - Python IDE for professional developers
- Jupyter [Project Jupyter, 2014]
  - Web environment for Julia, Python and R
  - Aimed at data transformation, numerical simulation and statistical modelling
Existing tools and environments

Online teaching environment for Python

- Pythy [Edwards, Tilden, Allevato, 2014]
- Online Python Tutor [Guo, 2013]
  - Web-based
  - Step forwards and backwards, data-structure visualisation
  - Generate shareable hyperlink to current execution point
  - Server-side execution (no event-driven programming)
Existing tools and environments

Python interpreter for the browser

- Brython [Quentel, 2012]
- Pyodide [Iodide, 2018]
- Skulpt [Graham, 2013]

No support for fine-grained single-stepping and hyperlink creation
Overview

1. What is codeBoot?

2. How we implemented an interpreter which allows single-stepping in the browser?

3. Web applications with codeBoot
What is codeBoot?
What is codeBoot?

REPL console

Code execution controls
From left to right:
- Execute one step
- Execute with animation
- Execute to the end
- Stop execution

The UI has been kept intentionally to the bare minimum
What is codeBoot?

- **Environment bubble**: When single-stepping, displays the result of the evaluated expression and variables which are in scope.
- **Step counter**: Conveys a sense of execution cost.
- **Playground**: Allows to draw with:
  - turtle module
  - pixels module
  - manipulation of the DOM
- **Local file**: Files are local to the browser.
codeBoot’s Python interpreter
codeBoot's Python interpreter

Challenges:

● Single-stepping needs UI updates to be handled during Python code execution
  ○ Showing the environment bubble
  ○ Incrementing the step counter
  ○ Drawing in turtle

● Browsers require JavaScript code to execute until completion before handling any other event including UI updates

"Once evaluation of a Job starts, it must run to completion before evaluation of any other Job starts." - ECMAScript 2020 Language specification
Solution:

- **Continuation Passing Style**
  - CPS allows to save the state of a Python program as a continuation
  - Calling the continuation executes one step of the code

- **Trampoline**
  - A trampoline is used to avoid a stack overflow in CPS (JavaScript doesn’t guarantee TCO)
  - It also allows to pause the execution of the Python code when needed
  - Manage interface between interpreter and UI
codeBoot's Python interpreter

Interpreter in CPS

Trampoline

UI
codeBoot's Python interpreter

(1) The trampoline starts the execution of compiled code

Trampoline

Interpreter in CPS
codeBoot's Python interpreter

(1) The trampoline starts the execution of compiled code

(2) The interpreter returns a continuation with the current state of the program
codeBoot's Python interpreter

(1) The trampoline starts the execution of compiled code

(2) The interpreter returns a continuation with the current state of the program

(3) When needed, the trampoline gives back control to the browser
codeBoot's Python interpreter

1. The trampoline starts the execution of compiled code.
2. The interpreter returns a continuation with the current state of the program.
3. When needed, the trampoline gives back control to the browser.
4. When the user resumes execution, the trampoline calls the continuation.
codeBoot's Python interpreter

Interpreter in CPS

(1) The trampoline starts the execution of compiled code
(2) The interpreter returns a continuation with the current state of the program
(3) When needed, the trampoline gives back control to the browser
(4) When the user resumes execution, the trampoline calls the continuation

What is the code compiled to?

UI

codeBoot's Python interpreter

Interpreter in CPS
codeBoot's Python interpreter

- The interpreter is based on the *fast interpretation* technique
  - Transforms the program's Abstract Syntax Tree into a function closure
- Implemented in Python
  - Compiled to JavaScript by p2j

![Diagram showing the process of codeBoot's Python interpreter]

**Interpretation Process**

- **User's code**
- **p2j** (free online compiler)
- **codeBoot** (Python interpreter)
- **Function closure**
Implementation of the Python construct `obj.attr`

```python
def gen_Attribute(cte, ast, obj_code, name):
    def call_getattribute(rte, cont, obj):
        ctx = Context(rte, cont, ast)
        return sem_getattribute(ctx, obj, om_str(name))
    def code(rte, cont):
        expr_end_cont = do_expr_end(cont, ast)
        return obj_code(rte, lambda rte, val:
            call_getattribute(rte, expr_end_cont, val))
        return cte, code
```
Implementation of the Python construct `obj.attr`

def gen_Attribute(cte, ast, obj_code, name):
    def call_getattribute(rte, cont, obj):
        ctx = Context(rte, cont, ast)
        return sem_getattribute(ctx, obj, om_str(name))

    def code(rte, cont):
        expr_end_cont = do_expr_end(cont, ast)
        return obj_code(rte,
                        lambda rte, val:
                            call_getattribute(rte, expr_end_cont, val))

    return cte, code
```python
def gen_Attribute(cte, ast, obj_code, name):
    def call_getattribute(rte, cont, obj):
        ctx = Context(rte, cont, ast)
        return sem_getattribute(ctx, obj, om_str(name))
    def code(rte, cont):
        expr_end_cont = do_expr_end(cont, ast)
        return obj_code(rte,
                        lambda rte, val:
                        call_getattribute(rte, expr_end_cont, val))
    return cte, code
```

Implementation of the Python construct `obj.attr`

Compiled function
The code function encapsulates the meaning of the `obj.attr` operation

Compiler for attribute access
The function `gen_Attribute` compiles the `obj.attr` construct to a function

codeBoot's Python interpreters

The code function encapsulates the meaning of the `obj.attr` operation.
def gen_Attribute(cte, ast, obj_code, name):
    def call_getattribute(rte, cont, obj):
        ctx = Context(rte, cont, ast)
        return sem_getattribute(ctx, obj, om_str(name))
    def code(rte, cont):
        expr_end_cont = do_expr_end(cont, ast)
        return obj_code(rte,
                        lambda rte, val:
                            call_getattribute(rte, expr_end_cont, val))
    return cte, code
def gen_Attribute(cte, ast, obj_code, name):

def call_getattribute(rte, cont, obj):
    ctx = Context(rte, cont, ast)
    return sem_getattribute(ctx, obj, om_str(name))

def code(rte, cont):
    expr_end_cont = do_expr_end(cont, ast)
    return obj_code(rte, lambda rte, val:
        call_getattribute(rte, expr_end_cont, val))

return cte, code
Web applications in codeBoot
Python programs can be bundled as web application. Programs and execution snapshots can be shared through hyperlinks.
Web applications

- User interaction beyond textual console input/output:
  - browser alert(), prompt() and confirm()
  - getMouse() is a built-in function to get the location and state of the mouse
  - onclick and onkeypress event handlers that execute Python code

- Three kinds of graphical interface:
  - Drawing with the turtle module
  - Drawing on a rectangular grid of pixels
    - Pixels can be set with setPixel(x, y, color)
    - getMouse() can report coordinates in the pixel rectangle
  - Manipulating the browser’s Document Object Model
codeBoot was designed to teach programming to novices:

- Fully in-browser
- Fine-grained single-stepping
- Shareable state using hyperlinks
- Interface to DOM and event-handling in Python

Future work:

- Implements a subset of Python
- More advanced programming courses
codeBoot is available at codeboot.org/py, you are welcome to try it!
In quiz-mode a StyleError exception is raised when the student uses a blocked feature.
The `pixels` module allows drawing on a grid of any size.