

Teaching Programming to Novices Using the codeBoot Online Environment

Marc Feeley and Olivier Melançon

Université 
de Montréal

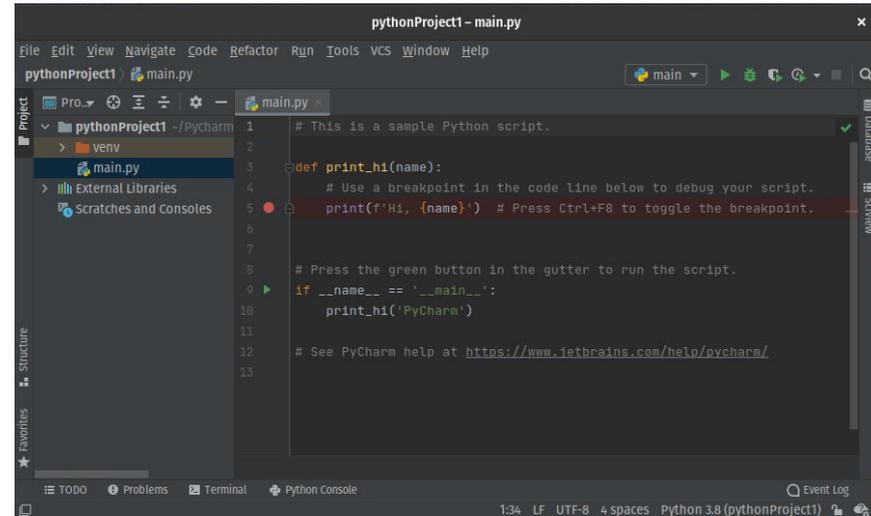
Why codeBoot?

- Fall 2020 semester was done through distance learning
- “Programming 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]

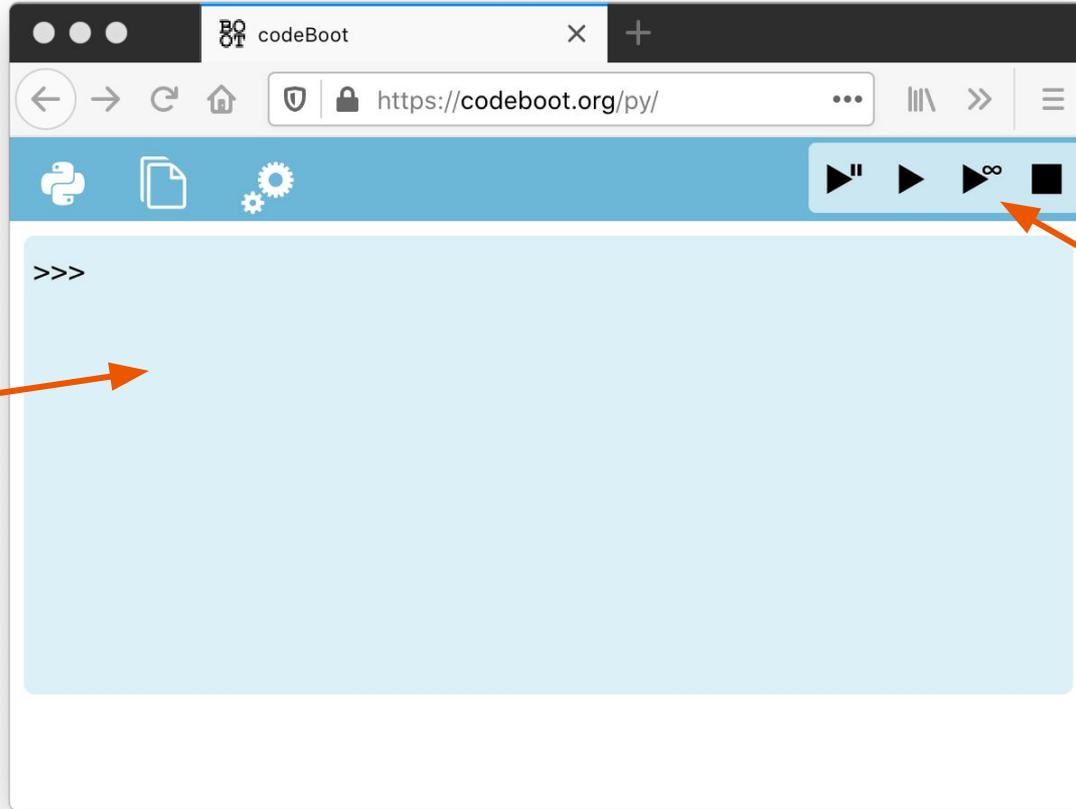
No support for fine-grained single-stepping and hyperlink creation

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?

The screenshot shows the codeBoot web interface. At the top, there's a browser window with the URL `https://codeboot.org/py/#`. Below the browser, there's a blue header bar with a Python logo, a file icon, a gear icon, and a blue button labeled "112 steps". To the right of the button are navigation icons: a play button with "1", a right arrow, a right arrow with an infinity symbol, and a square stop button. Below the header, there's a light blue box containing the command `>>> load("spiral.py")`. To the right of this box is a grid-based playground showing a spiral drawing with a red triangle cursor. Below the command box is a code editor with a file named "spiral.py" containing the following code:

```
def spiral(n):  
    if n > 0:  
        fd(n); lt(90)  
        spiral(n-5)  
  
clear(250, 100); goto(0, -40); spiral(80)
```

A yellow tooltip is visible over the code, showing the current state of the function: `n: 45` and `spiral: <function spiral #0>`. The tooltip also shows `None` above it.

Environment bubble

When single-stepping, displays the result of the evaluated expression and variables which are in scope

Local file

Files are local to the browser

Step counter

Conveys a sense of execution cost

Playground

Allows to draw with:

- turtle module
- pixels module
- manipulation of the DOM

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

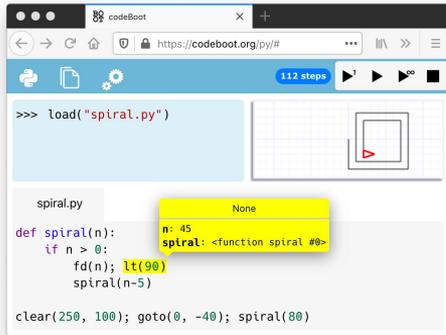
"Once evaluation of a Job starts, it must run to completion before evaluation of any other Job starts." - ECMAScript 2020 Language specification

codeBoot's Python interpreter

Solution:

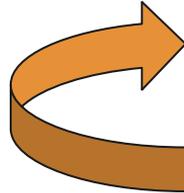
- Continuation Passing Style
 - CPS allows to save the state of a Python program as a continuation
 - Calling the continuation executes one step of the code
- Trampoline
 - A trampoline is used to avoid a stack overflow in CPS (JavaScript doesn't guarantee TCO)
 - It also allows to pause the execution of the Python code when needed
 - Manage interface between interpreter and UI

codeBoot's Python interpreter

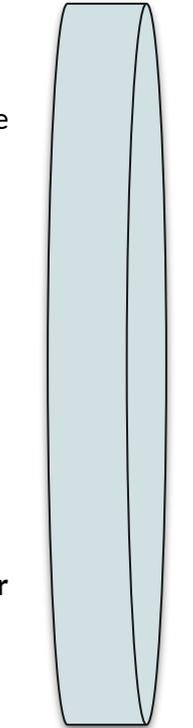


UI

(4) When the user resumes execution, the trampoline **calls the continuation**



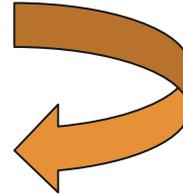
(3) When needed, the trampoline gives back control to the browser



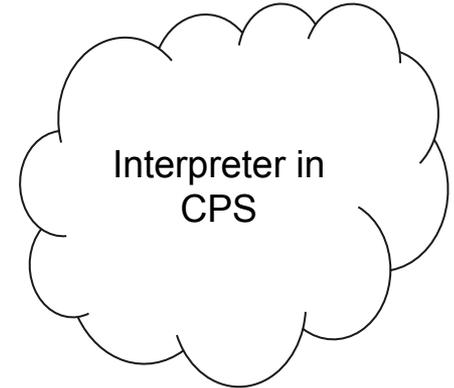
Trampoline

What is the code compiled to?

(1) The trampoline **starts the execution** of compiled code

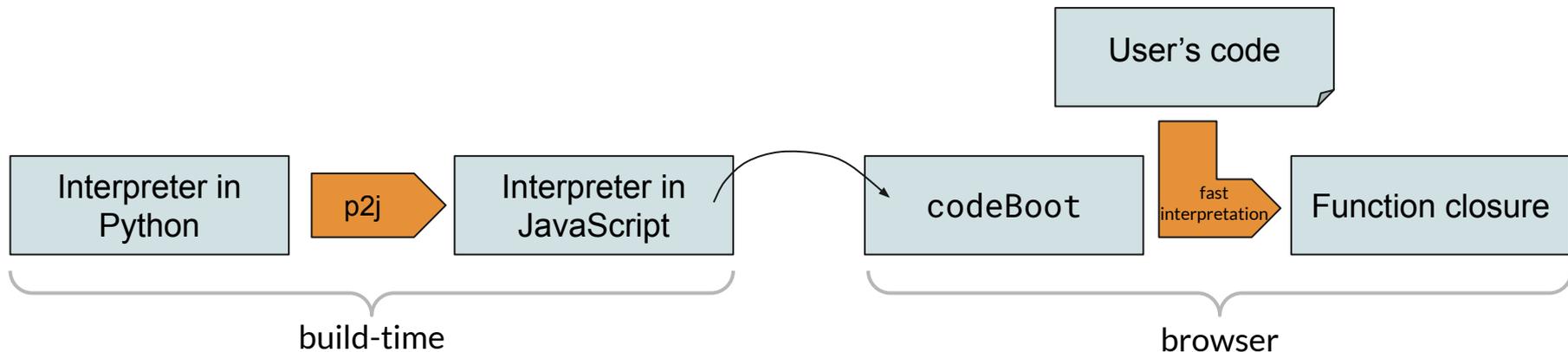


(2) The interpreter **returns a continuation** with the current state of the program



codeBoot's Python interpreter

- 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



codeBoot's Python interpreters

Implementation of the Python construct `obj.attr`

Compiled function

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

Code for evaluating the object

`obj_code` is the code for evaluating `obj` in the 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
```

Compiler for attribute access

The function `gen_Attribute` compiles the `obj.attr` construct to a function

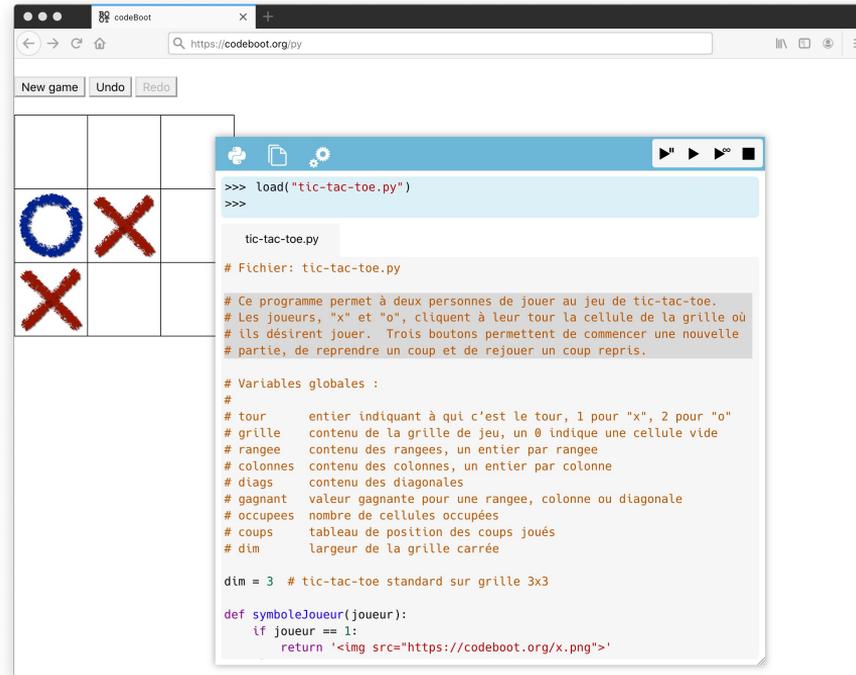
Continuation for expression end

The `expr_end_cont` is a special continuation which indicates the end of an expression to the trampoline

Web applications in codeBoot

Web applications

Python programs can be bundled as web application.



The screenshot shows a web browser window with the URL `https://codeboot.org/py`. The browser interface includes a search bar, navigation buttons (back, forward, refresh, home), and a menu icon. Below the browser, there is a Tic-Tac-Toe game grid. The grid is a 3x3 table with the following content:

Below the grid, there are three buttons: "New game", "Undo", and "Redo". Overlaid on the right side of the browser is a code editor window. The code editor shows the following Python code:

```
>>> load("tic-tac-toe.py")
>>>
```

tic-tac-toe.py

```
# Fichier: tic-tac-toe.py

# Ce programme permet à deux personnes de jouer au jeu de tic-tac-toe.
# Les joueurs, "x" et "o", cliquent à leur tour la cellule de la grille où
# ils désirent jouer. Trois boutons permettent de commencer une nouvelle
# partie, de reprendre un coup et de rejouer un coup repris.

# Variables globales :
#
# tour      entier indiquant à qui c'est le tour, 1 pour "x", 2 pour "o"
# grille    contenu de la grille de jeu, un 0 indique une cellule vide
# rangee    contenu des rangees, un entier par rangee
# colonnes  contenu des colonnes, un entier par colonne
# diags     contenu des diagonales
# gagnant   valeur gagnante pour une rangee, colonne ou diagonale
# occupees  nombre de cellules occupées
# coups     tableau de position des coups joués
# dim       largeur de la grille carrée

dim = 3 # tic-tac-toe standard sur grille 3x3

def symboleJoueur(joueur):
    if joueur == 1:
        return ''
```

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](#)

Conclusion

codeBoot was designed to teach programming to novices:

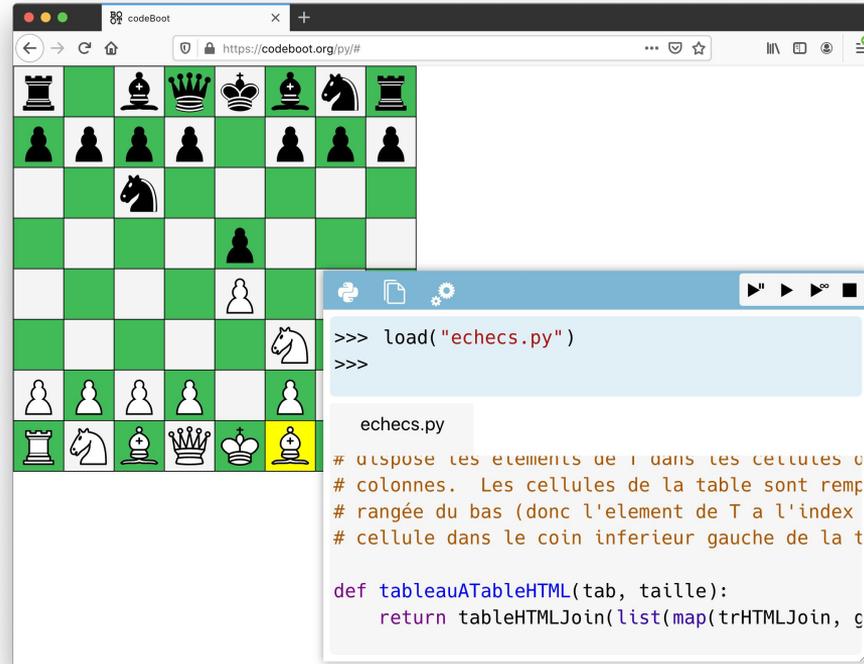
- Fully in-browser
- Fine-grained single-stepping
- Shareable state using hyperlinks
- Interface to DOM and event-handling in Python

Future work:

- Implements a subset of Python
- More advanced programming courses

Conclusion

codeBoot is available at codeboot.org/py, you are welcome to try it!



The screenshot shows a web browser window with the URL `https://codeboot.org/py/#`. On the left, there is a chess board with a green and white checkered pattern. The board contains several chess pieces: a black king, queen, rook, knight, and pawns on the top row; a black knight and a black pawn on the second row; a white pawn on the third row; a white pawn on the fourth row; a white knight on the fifth row; and a white king, queen, rook, and pawns on the bottom row. The bottom-right cell of the board (row 8, column 6) is highlighted in yellow and contains a white king piece.

On the right side of the browser window, there is a Python code editor. The editor has a toolbar with icons for running, saving, and settings. The code in the editor is as follows:

```
>>> load("echecs.py")
>>>

echecs.py
# utilise les elements de T dans les cellules de
# colonnes. Les cellules de la table sont remplis
# rangée du bas (donc l'element de T a l'index
# cellule dans le coin inferieur gauche de la table
def tableauATableHTML(tab, taille):
    return tableHTMLJoin(list(map(trHTMLJoin, g
```