CSC253 midterm blog post comments
2014-10-27

Renan Zanelatto -
https://github.com/renansz/URochester/blob/master/CSC453/midterm_post.md

compiler -> interpreter

two variables `a` e `b (a and b)`
`x = string_concatenate(v, w, f, next_instr); (maybe put a comment next to it to say that you'll zoom into it later?)`
minor typos: This is our the **simplified** version

Good: Simplified version of the overall logic flow

very detailed

Jinze An & Christian Djoe
https://github.com/raign/CSC253-Midterm/wiki/Python-While-Loop
explicitly trace through the sample program

have a summary of some sort?

what's "heading pending"? (I need ideas for a heading that makes sense there. All the headings were pretty last-minute)

- "main opcodes involved"?

good discussion of overflow

it would be good to briefly talk about the order that the code executes in the loop. right now there are good static descriptions of several opcodes, but what's missing is the control flow -- maybe add it in the CONTROL FLOW section? trace through your x=1, x+=1 example

- maybe even put that at the end of the introduction to give the "big picture" overview

Might be good to have a "big picture" at the beginning that links all the pieces below together.

Might be able to give yourself more 'space and time' to explain other details by removing details on popping/pushing to the value stack.

Qiyuan Qiu & Jeffrey White

https://github.com/jwhite37/csc253Midterm/blob/master/README.md

personal pet peeve: capitalize Python and spelling errors

instanciated -> instantiated

“Function is like a prototype of Frame” -- what does that mean? template? shell, or placeholder
“Code” -> “Code object” to be more precise

Maybe follow Renan’s example and simplify your code snippets to only the relevant parts for the tutorial, especially the struct definitions

“Python Internals” section header could be renamed something else -- tracing execution

---
Dong Chen

http://dongchen-coder.github.io

A lot of code excerpts, not as much explanation

Grammar…(e.g. “By given a Python code”)

giving instructions on how to create lists in Python and showing how to disassemble is unnecessary

too much space spent diving into background that your audience already has a firm grasp on. giving refreshers on the fly while diving into the core mechanics of the script would work better

“**Assemble Code:** “ -- “disassembled code”?

The resulting assemble code is listed below. -- “disassembled”

- also, this long paragraph can be broken up into a bulleted list with one element for each bytecode to explain it. that will make it easier to read

Single quotation marks look weird in the code

For the “Diving into Python bytecode” section, it would be clearer to trace through at least part of the execution of your code example instead of just showing the code

- maybe just showing ONE iteration of your loop and how that executes through all of the functions that you listed. otherwise it will just be like describing pieces of code without much context

---
Doug Miller
https://github.com/dooglius/CSC253-shouldworkthistime/blob/master/README.md

Three empty headings?
should simplify your code excerpts
the output prediction is cute, but a bit tangential considering the purpose of the tutorial. the code example also ends up injecting extra bytecodes that don't contribute meaningfully to the focus of the tutorial, which is iteration over the list

as a conventions thing, wrap in-line code snippets like "GET_ITER" in <code></code>

Interesting quiz element

The last 2 paragraphs before "Thus, the final value of x is [20, 40]", are very dense and might need clarification with small code snippets.

Three blank bullet points is a little bit confusing, but the quiz is very interesting.

There's A LOT of text without being broken up by headings, so think of heading names

Repeating each line of ~5 bytecodes later in the explanation could help, since there's a lot of bytecodes there

---
Hao Xu & Xi Jin
https://github.com/highpowerxh/CSC453BlogPost/blob/master/CSC453_Midterm_Blog_Post.md

Really not necessary to show the raw disassembly

- if you're going to show it anyways, say how you got the human-readable version out using dis.dis()

"So the Highlight part" - highlight shouldn't be capitalized unless it's a proper noun (e.g., a technical term), can say "highlighted"

lots of small typos like "deined", "wierd", etc. just do some proofreading
"build-in" -> "built-in"

Jonathan McLinn and Yennifer Hernandez

https://github.com/jmclinn/253midterm/blob/master/README.mdff
“The concatenation happens on line 4” -- is this Line 3?

good subsetting to simplify code snippets

“if statement” (the “if” could be in a code block. i think the syntax is `if` but not 100% sure)

good extra Observations section
“Py_MEMCPY vs memcpy” -- maybe mention that it’s a macro so it inlines the code for efficiency to avoid a function call

---
Shaowei Su & Xin Xu
https://github.com/shaowei-su/CSC453/blob/master/TutorialForWhile.md

minor typos, e.g., “uesd”, “throuth”

there are a ton of code blocks, but not much explanation to go along with them

- it would be good to trace through your actual code example rather than just describing what each bytecode does. i like how you use continue and break in your example, but it would be good to show a trace through the loop to see exactly when the continue and break lines executed

“In Python, `continue` and `break` work**s the** same as in other languages like C and Java” ???

- clarify a bit more

CSC253 midterm blog post comments
2014-10-27