python recursion

It keeps getting better.

John Hanley

30th September 2025

Connect with me at https://www.linkedin.com/in/jhanley714

agenda

- improvements
- recursion
- recursion in Scheme
- TCO
- stability
- python
- recent interpreters

transparent improvements

- diverse speedups, in libs and interpreter
- quadratic string append
- bare `except` vs. CTRL/C
- timsort speed from Tim Peters
- `dict` storage & iteration order from Raymond Hettinger
- "zero cost" `try:` blocks, also by Raymond

quadratic $O(n^2)$ string append

```
n = 1_000_000
s = ""
for _ in range(n):
    s += " hello"
return s
```

versus

```
text = []
for _ in range(n):
    text.append(" hello")

return "".join(text)
```

zero cost `try:` blocks

old way:

- Execute `try:` –> push some state onto the stack.

new way:

- Occasionally `raise` an exception –> go find some state.

coming soon

Interpreter 3.14 hits the streets October 7^{th}.

It will feature template t-strings, similar to formatted f-strings.

motivation

StackOverflow question:

> Why does python limit recursion depth?

cheaper, lazy Python frames

Buried in the 3.11 rel notes we find:

- Streamlined the internal frame struct to contain only essential information

For most user code, no frame objects are created at all. As a result, nearly all Python functions calls have sped up significantly.

Most Python function calls now consume no C stack space, speeding them up.

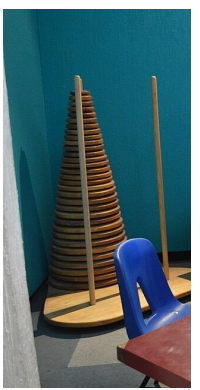
recursion

```
"""Recursively solve the famous Towers of Hanoi problem."""

Peg = str

def hanoi(n: int, source: Peg, target: Peg, aux: Peg) -> None:
    if n == 0:
        return  # base case
    hanoi(n - 1, source, aux, target)
    print(f"Move disk {n} from {source} to {target}")
    hanoi(n - 1, aux, target, source)

hanoi(3, "A", "C", "B")
```

towers in motion

```
    hanoi(n - 1, source, aux, target)
    print(f"Move disk {n} from {source} to {target}")
    hanoi(n - 1, aux, target, source)

hanoi(3, "A", "C", "B")
```

- Move disk 1 from A to C
- Move disk 2 from A to B
- Move disk 1 from C to B
- Move disk 3 from A to C
- Move disk 1 from B to A
- Move disk 2 from B to C
- Move disk 1 from A to C



the lambda calculus


```
#lang racket

(+ 1 2 3)
6
```

factorial – tail call optimization

```
(define (fact-slow n)  ; TCO is disabled, so the call stack grows
  (if (zero? n)
      1
      (* n (fact-slow (sub1 n)))))


(fact-slow 5)
120
```

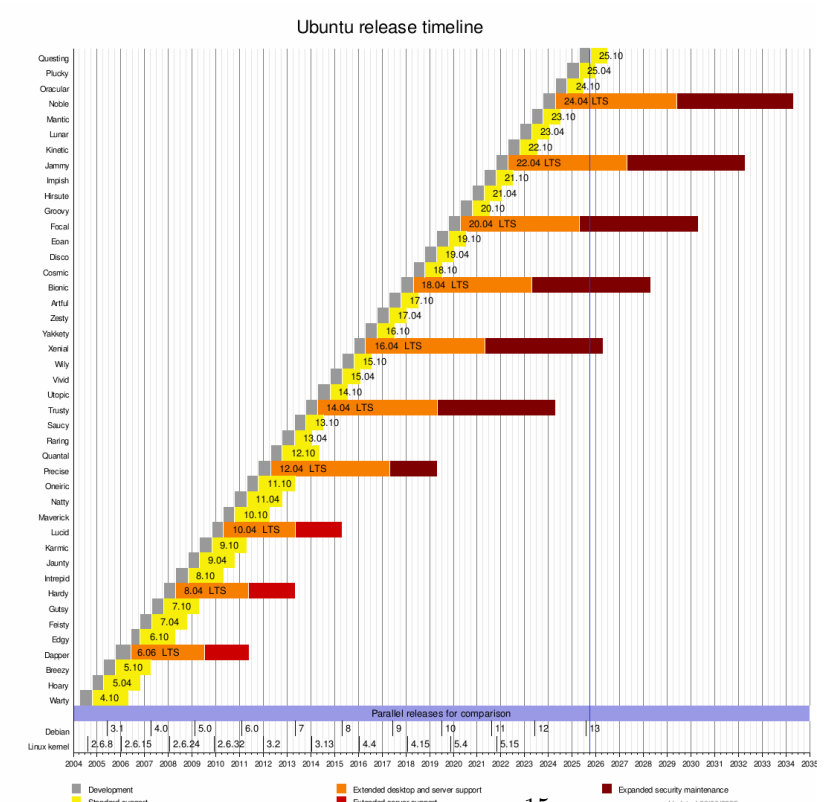
factorial – tail call optimization

```
(define (fact-slow n)  ; TCO is disabled, so the call stack grows
  (if (zero? n)
      1
      (* n (fact-slow (sub1 n)))))

(define (fact n)
  (_fact n 1))

(define (_fact n acc)
  (if (zero? n)
      acc
      (_fact (sub1 n) (* acc n))))
```

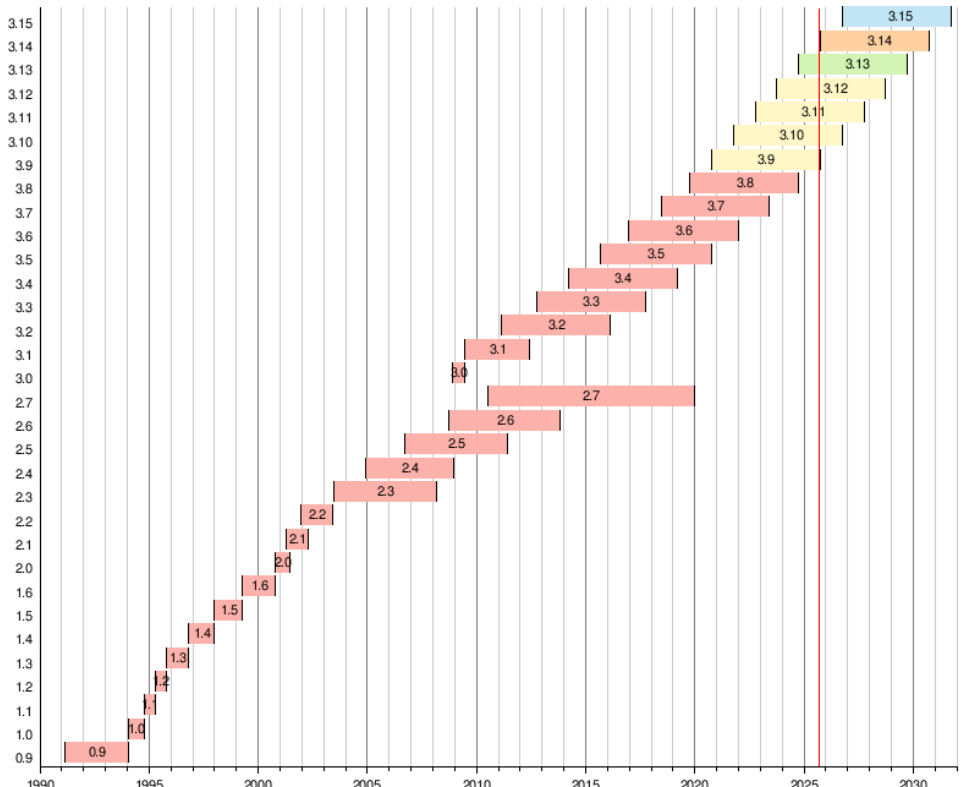
timing

```
Factorial 250000 found by #<procedure:fact-slow> in 29848 msec

Factorial 250000 found by #<procedure:fact>      in 10349 msec
```

stability – ubuntu



stability – python interpreter



stability – PEP-719 interpreter releases

- 3.13.1: Tuesday, 2024-12-03
- 3.13.2: Tuesday, 2025-02-04
- 3.13.3: Tuesday, 2025-04-08
- 3.13.4: Tuesday, 2025-06-03
- 3.13.5: Wednesday, 2025-06-11
- 3.13.6: Wednesday, 2025-08-06
- 3.13.7: Thursday, 2025-08-14

future:

- 3.13.8: Tuesday, 2025-10-07
- 3.13.9: Tuesday, 2025-12-02
- 3.13.10: Tuesday, 2026-02-03 ...

Makefile

```
measure: \
  $(REC)/py3.13 \
  $(REC)/py3.12 \
  $(REC)/py3.11 \
  $(REC)/py3.10 \
  $(REC)/py3.9 \
  $(REC)/py3.8 \

py%:
        mkdir $@
        cd $@ && uv venv --python $(shell basename $*)
        cd $@ && python $(DIR)/count.py
```

counting

```
def iterative_count(i: int, ceil: int) -> int:
    assert 0 == i, i  # Count up from zero, please.
    while i < ceil:
        i += 1
    return i


def recursive_count(i: int, ceil: int) -> int:
    if i < ceil:
        return int(recursive_count(i + 1, ceil))
    return i
```

counting

```
def main(n: int = 30_000_000) -> int:
    """Chooses large N, appropriate for current interpreter."""
    sys.setrecursionlimit(n)
    if sys.version_info < (3, 11):
        n = 42_780  # max feasible value for interpreter 3.10.16
    if sys.version_info < (3, 10):  # noqa
        n = 72_287  # interpreters 3.8.20 & 3.9.23

    assert recursive_count(0, n) == n

    return n
```

results

```
3.14.0rc2  30,000,000  1.274513 sec;  23,538,402.4 count/sec
3.13.7     30,000,000  1.282322 sec;  23,395,061.2 count/sec
3.12.8     30,000,000  1.268134 sec;  23,656,808.2 count/sec
3.11.11    30,000,000  1.246084 sec;  24,075,419.4 count/sec
3.10.16        42,780  0.008580 sec;   4,986,031.8 count/sec
3.9.23         72,287  0.013965 sec;   5,176,249.8 count/sec
3.8.20         72,287  0.016485 sec;   4,384,959.7 count/sec
```

questions?

image credits

- https://en.wikipedia.org/wiki/Tower_of_Hanoi
  - AlejandroLinaresGarcia:
  - https://en.wikipedia.org/wiki/File:UniversumUNAM34.JPG
- https://xkcd.com/297 – Randall Munroe
- https://asset.conrad.com/media10/isa/160267/c1/-/en/860048_BB_00_FB/image.jpg
- https://en.wikipedia.org/wiki/Ubuntu_version_history
  - https://upload.wikimedia.org/wikipedia/en/timeline/swfdceo7tl9n28j9pamsv083nwmv0l7.png
- https://en.wikipedia.org/wiki/History_of_Python
  - https://upload.wikimedia.org/wikipedia/en/timeline/3aeqr87p1hb7nohu4c5lnlkdiiypums.png