Hiding Iteration with Recursion

INTRODUCTION



Gerald BrittonIT SOLUTIONS DESIGNER

@GeraldBritton www.linkedin.com/in/geraldbritton

Foundations



Mathematical foundation

E.W. Dijkstra - Recursive Programming

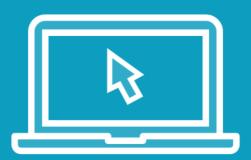
- http://oai.cwi.nl/oai/asset/9253/9253A.pdf
- "Go To Statement Considered Harmful."

$$F_{n} = \begin{cases} 0 & if \ n = 0 \\ 1 & if \ n = 1 \\ F_{n-1} + F_{n-2} & if \ n > 1 \end{cases}$$

Recurrence

$$F_{10} = F_9 + F_8 = F_8 + F_7 + F_7 + F_6 = \dots$$

$$S_n = \begin{cases} 0 & \text{if } n = 0 \\ n + S_{n-1} & \text{if } n > 0 \end{cases}$$



Tail Recursion



Avoid stack exhaustion

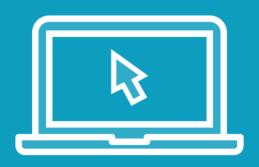
Function f calls function g at the end

If function g leads to $f \Rightarrow f$ is tail recursive

Function f and g may be the same

Turn recursion into iteration

Only one stack frame per function



Demo 2

Trampolining



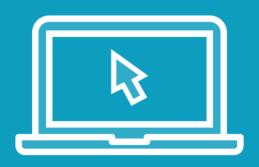
Python does not optimize tail calls

Need workaround

Trampolining!

GO TO

Generators!



Demo 2

Recap



Recursion and mathematical roots

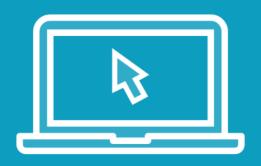
Python support and limitations

Tail recursion

Trampolining and the tramp function

Garbage collection

Pure iteration is usually faster



Count of expedited orders

...with backordered items

Use tail recursion and the tramp function

List[0] = head

List[1:] = tail

Summary



Recursion in Python

Limitations and mitigations

- Tail recursion
- Trampolining

Costs

- Extra object creation and destruction
- Loss of traceback information

Should you use recursion?

- It depends!