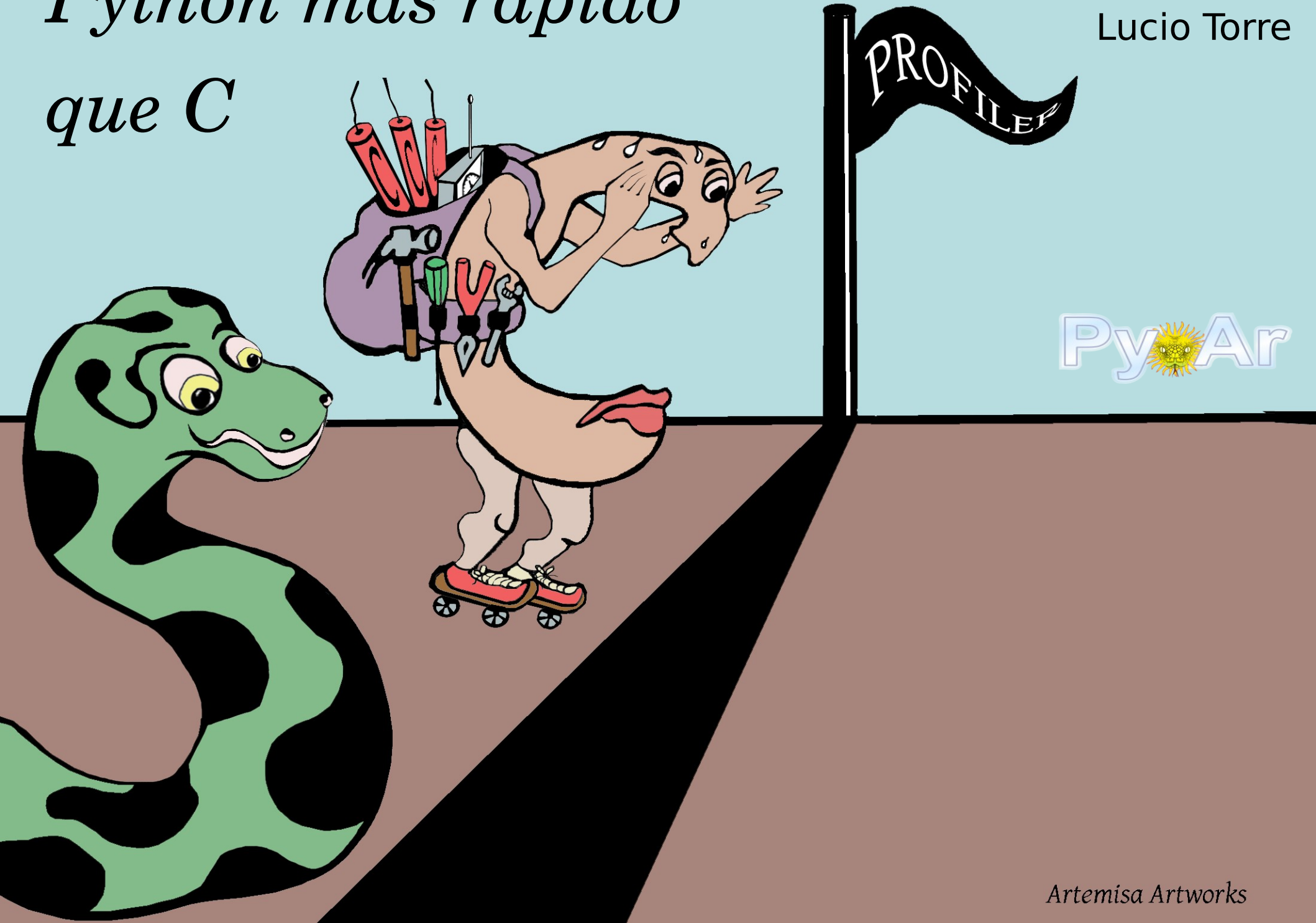
Python más rápido que C

Facundo Batista
Lucio Torre



Artemisa Artworks

Objetivos

- Divertirse
- Comparar tiempos de ejecución en Python y C
- Sin perder de vista otros factores
  - Errores tontos (< 5 seg)
  - Errores complicados (> 10 min, una taza de café y una patada al sofá)
  - Cantidad de lineas de código
  - Complejidad de diseño
- ¡Pero haciendo énfasis en el tiempo!
  - ¿Es C más rápido que Python?
  - ¿Cuándo? ¿Cuánto?

Recuerden

We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil.

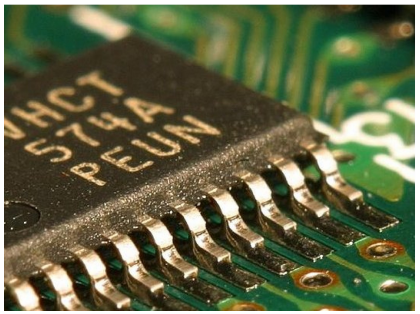
Donald Knuth

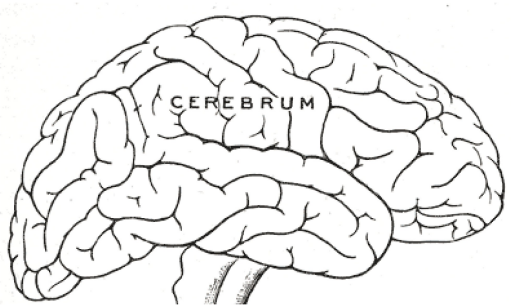
Structured Programming with goto Statements
ACM Journal Computing Surveys
Vol 6, No. 4, Dec. 1974. p.268

Gap

Bajo nivel
(código final)

Alto nivel
(ideas)





ASM C Python

¡este camino lo tiene que recorrer alguien!

Multiplicar un número

ASM

```
  .file    "mult.c"
  .text
.globl main
  .type    main, @function
main:
  leal    4(%esp), %ecx
  andl    $-16, %esp
  pushl    -4(%ecx)
  pushl    %ebp
  movl    %esp, %ebp
  pushl    %ecx
  subl    $16, %esp
  movl    $5, -16(%ebp)
  movl    $10, -12(%ebp)
  movl    -16(%ebp), %eax
  imull    -12(%ebp), %eax
  movl    %eax, -8(%ebp)
  addl    $16, %esp
  popl    %ecx
  popl    %ebp
  leal    -4(%ecx), %esp
  ret
  .size    main, .-main
  .section    .note.GNU-stack,"",@progbits
```

C

```
void main(void)
{
int a=5, b=10, c;
c = a * b;
}
```

Python

```
a = 5
b = 10
c = a * b
```

Traduciendo

- C se pasa a ASM en tiempo de compilación
- Python a instrucciones de su VM:

```
2           0 LOAD_CONST               1 (5)
            3 STORE_FAST               0 (a)

3           6 LOAD_CONST               2 (10)
            9 STORE_FAST               1 (b)

4          12 LOAD_FAST                0 (a)
           15 LOAD_FAST                1 (b)
           18 BINARY_MULTIPLY
           19 STORE_FAST               2 (c)
           22 LOAD_CONST               0 (None)
           25 RETURN_VALUE
```

¿Cómo comparamos?

ptrace

```
int main()
{   pid_t child;
    BIGNUM *count;

    child = fork();

    count = BN_new();

    if(child == 0) {
        ptrace(PTRACE_TRACEME, 0, NULL, NULL);
        execl("/usr/bin/python", "python", "nada.py", NULL);

    } else {
        int status;
        while(1) {
            wait(&status);
            if(WIFEXITED(status))
                break;
            ptrace(PTRACE_SINGLESTEP, child, NULL, NULL);
            BN_add(count, count, BN_value_one());
        }
        printf("Total %s\n", BN_bn2dec(count));
    }
    return 0;
}
```

Cuánto ejecutamos

Vemos cuantas instrucciones se ejecutan...

- Haciendo Nada, para ver el costo por estructura
- Multiplicando un sólo número
- Realizando 100 mil multiplicaciones

Ej 0	**Nada**	**Una**	**100k**	**por op**
C	101047	101054	701055	6
Python	15767781	15834089	93349599	775

¡Python es más de 100 veces más lento!

¿Posta?

Multiplicamos un número

¿Cuanto se tarda en todo el proceso?

	edición	compilación	ejecución
C	x	y	z
python	x/2	0	z * 100
calculadora	x/5	0	q (q <<x)

Para definir una metrica, lo importante es el contexto

Un caso más real

Tomamos muchos valores de algún lado:

- Generamos previamente un millón de pares de enteros
- Los leemos y multiplicamos

Ej 1	Code [m]	Run [s]
C	10	3,5
Python	4	19,8
	Factor:	5,66

Vemos que C es mucho más rápido que Python en este caso, donde el procesador hace todo

En Python hicimos mucho más que multiplicar dos *ints.*

Soportando big nums

Casi lo mismo que antes...

- Usamos enteros de entre 0 y 32 dígitos
- En Python no hay que hacer nada
- En C tuvimos que usar una biblioteca externa

Ej 2	Code [m]	Run [s]
C	58	37
Python	0	39
	Factor:	1,05

¡Rayas y centollas!

¿Multiplicando enteros
Python es tan rápido como C?

¡A la hoguera!

¡Herejía, herejía!

(acá abandonamos el primer día de trabajo)

Enfocándonos

Tratamos de apuntar a performance

- Dejamos de usar tanto disco

Explotamos ventajas de C

- Acceso a memoria
- Multiplicación pura

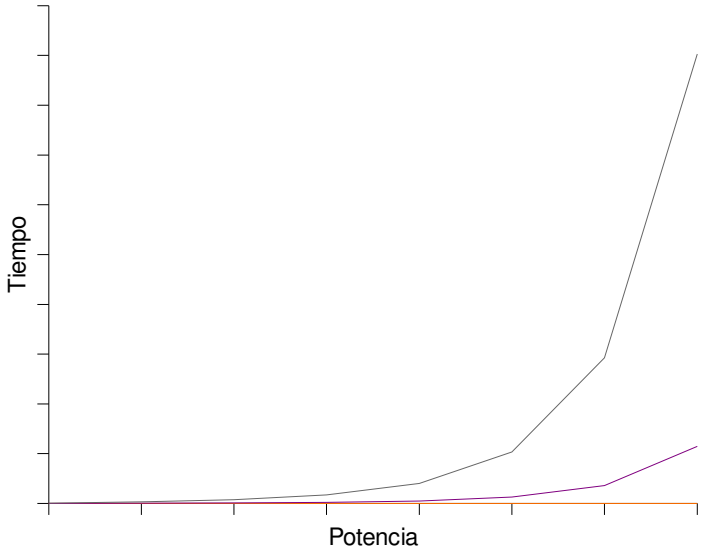
¡Multiplicamos matrices!

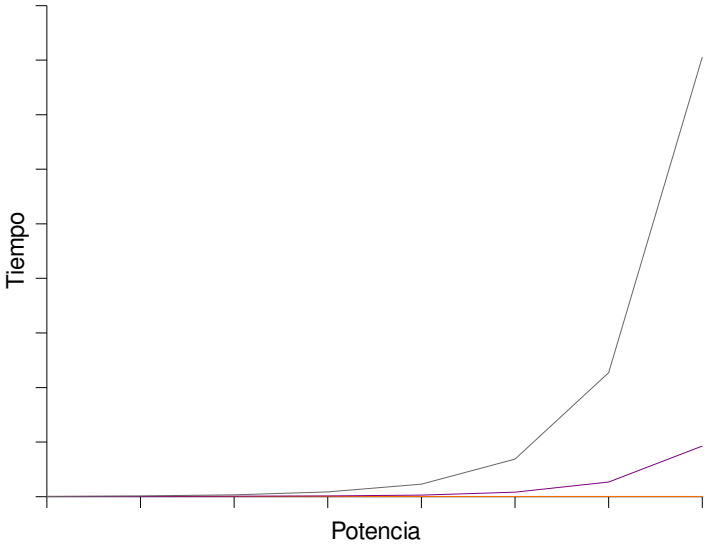
- Tiempo de codeo: C: 50 min Py: 18 min
- De 10x10, 100x100 y 200x200
- Multiplicamos por si misma muchas veces
- len(str(m[0][0])) de la última: 3518!

Matrices

Ej 3

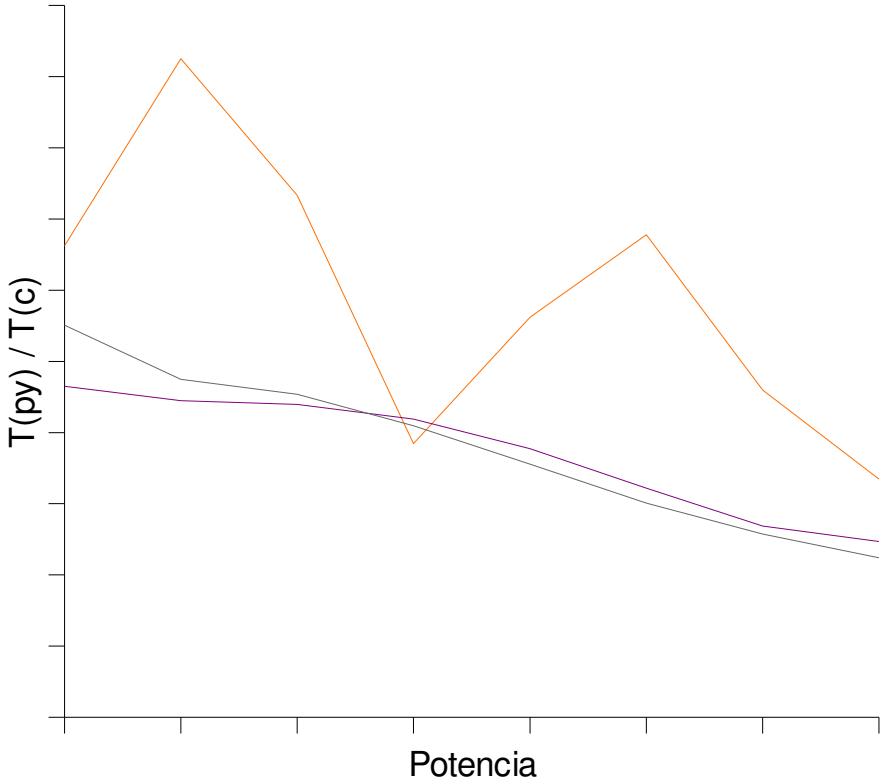
	Python			C		
**	**10x10**	**100x100**	**200x200**	**10x10**	**100x100**	**200x200**
2	0,05	1,94	15,81	0,00	0,21	1,44
5	0,07	7,84	62,27	0,00	0,88	6,56
10	0,09	18,42	148,74	0,01	2,10	16,39
20	0,12	43,03	342,14	0,02	5,14	41,78
40	0,90	100,59	811,04	0,08	13,33	114,01
80	1,07	256,28	2072,23	0,08	39,82	344,09
160	1,52	718,41	5850,62	0,17	133,66	1136,15
320	2,88	2293,46	18052,94	0,43	463,98	4028,33





Relación entre ambos

	T(py) / T(c)		
**	**10x10**	**100x100**	**200x200**
2	13,25	9,30	11,01
5	18,50	8,89	9,49
10	14,67	8,79	9,08
20	7,69	8,38	8,19
40	11,24	7,55	7,11
80	13,56	6,44	6,02
160	9,20	5,37	5,15
320	6,69	4,94	4,48



Python es más parecido a C cuanto más grande es el cálculo

(no tenemos ni idea por qué)

Saliendo de los números

Realizamos un ejercicio más clásico

- Tomamos un texto grande (todo Shakespeare)
- Generamos al azar una lista de tokens de ese texto
- El ejercicio es escribir en otro archivo esas palabras y en que posición del texto original aparecen...

Ej 4	Code [m]	Run [s]
C	40	47
Python	22	144
...		
Python	+5	60
	Factor:	1,28

Python es tan rápido cómo C
si lo optimizamos un poco

Optimizando

Optimizamos después
de revisar tiempos

Es más fácil optimizar código correcto que corregir código optimizado

Enfocándonos (de nuevo)

Tratamos de eliminar overheads comunes

- No escribimos más a disco
- Mostramos la palabra que aparece más cerca del final (y en que posición aparece)

Ej 5	Code [m]	Run [s]
C	1	0,25
Python	1	11,00
	Factor	44,00

C es mucho más rápido que Python en este caso

Porque modelamos la solución minimizando el uso de memoria dinámica.

Conclu...

C es más rápido que Python, especialmente si...

- Dejamos que el procesador haga todo el trabajo
- Nos escapamos lo más posible de la memoria dinámica

Pero esto lo encontramos ajustando las tareas para que tengan esas propiedades.

En los experimentos pensados sin forzar esto, Python es tan rápido como C

...siones

Muchas veces C es más
rápido que Python,
pero no siempre se justifica
el esfuerzo extra

"Premature optimization
is the root of all evil"

Donald Knuth