

Complementi di Programmazione

# Performance C and Python

CdL Informatica - Università degli studi di Modena e Reggio Emilia  
AA 2023/2024

Filippo Muzzini

# Profiling

In python è possibile profilare i tempi di esecuzione con

```
python3 -m cProfile foo.py
```

# Profiling

E' possibile anche profilare specifiche parti di codice:

```
import cProfile, pstats, io

from pstats import SortKey

pr = cProfile.Profile()

pr.enable()

# ... do something ...

pr.disable()

s = io.StringIO()

sortby = SortKey.CUMULATIVE

ps = pstats.Stats(pr, stream=s).sort_stats(sortby)

ps.print_stats()

print(s.getvalue())
```

# C

Anche in C è possibile misurare i tempi di esecuzione:

```
#include <sys/time.h>
```

```
int main() {
```

```
    long start, end;
```

```
    struct timeval timecheck;
```

```
    gettimeofday(&timecheck, NULL);
```

```
    start = (long)timecheck.tv_sec * 1000 + (long)timecheck.tv_usec / 1000;
```

```
    usleep(200000); // 200ms
```

```
    gettimeofday(&timecheck, NULL);
```

```
    end = (long)timecheck.tv_sec * 1000 + (long)timecheck.tv_usec / 1000;
```

```
    return 0;
```

```
}
```

# C vs Python

Spesso quindi si scrive un programma in Python e poi si utilizzano funzioni C per le parti più onerose:

- Codice semplice per la maggior parte del programma
- Parte più lenta diventa molto più veloce

# C vs Python

Come si fa?

Si scrive la funzione C

```
int square(int i) {  
    return i * i;  
}
```

# C vs Python

```
int square(int i) {  
    return i * i;  
}
```

la si compila

```
cc -fPIC -shared -o my_functions.so my_functions.c
```

# C vs Python

Si scrive il programma Python che carica la libreria C e la utilizza

```
from ctypes import *  
  
so_file = "/Users/pankaj/my_functions.so"  
  
my_functions = CDLL(so_file)  
  
print(my_functions.square(10))  
  
print(my_functions.square(8))
```



# C vs Python

Si scrive il programma Python che carica la libreria C e la utilizza

```
from ctypes import *  
so_file = "/Users/pankaj/my_functions.so"  
my_functions = CDLL(so_file)  
print(my_functions.square(10))  
print(my_functions.square(8))
```

Modulo utilizzato  
per caricare la  
libreria

# C vs Python

Si scrive il programma Python che carica la libreria C e la utilizza

```
from ctypes import *
```


```
so_file = "/Users/pankaj/my_functions.so"
```

```
my_functions = CDLL(so_file)
```

```
print(my_functions.square(10))
```

```
print(my_functions.square(8))
```

Caricamento  
libreria da file



# C vs Python

Si scrive il programma Python che carica la libreria C e la utilizza

```
from ctypes import *
```

```
so_file = "/Users/pankaj/my_functions.so"
```

```
my_functions = CDLL(so_file)
```

```
print(my_functions.square(10))
```

```
print(my_functions.square(8))
```

Utilizzo della  
funzione



# C vs Python

Sappiamo già che le performance non sono comparabili.

C è molto più veloce

Python è molto più comodo