Strumenti per la programmazione funzionale in Python

Map/Reduce

Paradigma distribuito (esiste anche un framework di Google con lo stesso nome) per la computazione dei dati.

Diviso in due fasi:

- Mappatura dei dati (filtri ed elaborazioni sui singoli elementi)
- Riduzione (aggregazione dei risultati della prima fase)

Strumenti per la programmazione funzionale in Python

Map/Reduce

Diviso in due fasi:

- Mappatura dei dati (filtri ed elaborazioni sui singoli elementi)
- Riduzione (aggregazione dei risultati della prima fase)

Essendo distribuito non si può assumere la sequenza dei dati

La riduzione può avvenire in più step.

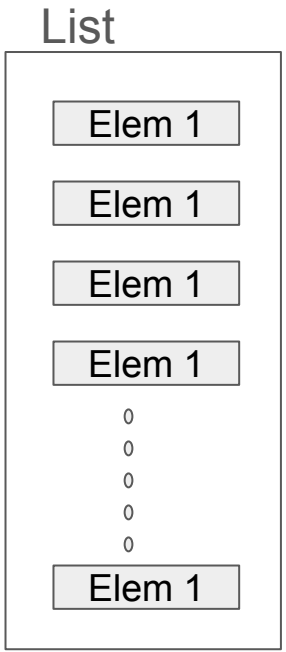
Strumenti per la programmazione funzionale in Python

Map/Reduce

Utile in database distribuiti in cui i dati sono in nodi diversi e la computazione non può avvenire in un unico nodo.

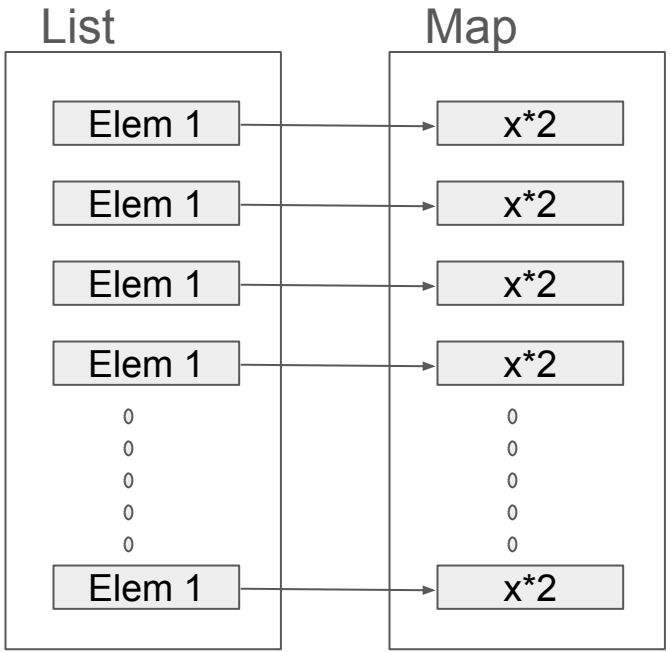
Strumenti per la programmazione funzionale in Python

Map/Reduce



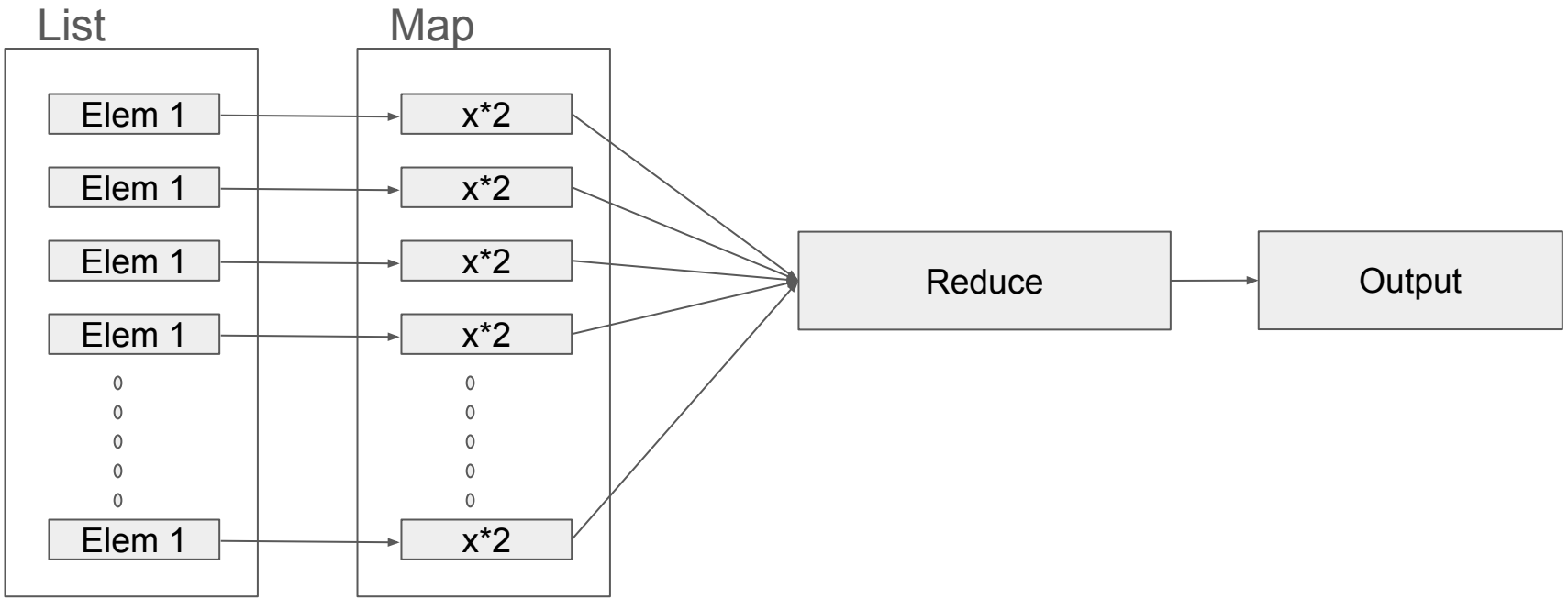
Strumenti per la programmazione funzionale in Python

Map/Reduce



Strumenti per la programmazione funzionale in Python

Map/Reduce



Strumenti per la programmazione funzionale in Python

Map/Reduce in Python

In python esiste la funzione **reduce()** che è parte della libreria **functools**.

Questa funzione esegue in locale e non in un ambiente distribuito ma è comunque utile.

Applica ripetutamente una funzione di due argomenti sugli elementi di una sequenza in modo da ridurre la sequenza ad un unico valore.

Strumenti per la programmazione funzionale in Python

Map/Reduce in Python

Applica ripetutamente una funzione di due argomenti sugli elementi di una sequenza in modo da ridurre la sequenza ad un unico valore.

Più nel dettaglio:

ad ogni iterazione il primo argomento è il risultato delle passate precedenti e il secondo è un elemento (non ancora processato) della lista.

La prima passata non ha il risultato delle passate precedenti quindi si usa il primo elemento della lista

Strumenti per la programmazione funzionale in Python

Map/Reduce in Python

La prima passata non ha il risultato delle passate precedenti quindi si usa il primo elemento della lista

Oppure

Si può specificare che valore utilizzare come primo risultato (fake)

Strumenti per la programmazione funzionale in Python

Map/Reduce in Python

Es. somma

```
reduce(lambda x,y: x+y, [1,2,3,4])
```

Es. creazione di un set da una lista

```
reduce(lambda x,y: x.union(set([y])), set())
```

Strumenti per la programmazione funzionale in Python

Map/Reduce in Python

Es. somma

```
reduce(lambda x,y: x+y, [1,2,3,4])
```

Es. creazione di un set da una lista

```
reduce(lambda x,y: x.union(set([y])), [1,2,3,4], set())
```

Strumenti per la programmazione funzionale in Python

Map/Reduce in Python

Questo paradigma è efficace sia quando la computazione può essere distribuita, sia quando può essere trattata come flusso.

Questo approccio riduce le risorse necessarie alla computazione

Strumenti per la programmazione funzionale in Python

Eval

Eval è una funzione che valuta un’espressione **passata sotto forma di stringa**

In pratica la stringa viene eseguita dall’interprete come se fosse codice del programma.

eval(“print(‘ciao’)”)

Strumenti per la programmazione funzionale in Python

Eval

Eval è una funzione che valuta un’espressione **passata sotto forma di stringa**

Si possono passare anche variabili dal programma “principale”, sia come globali sia come locali al codice della stringa

eval(“print(‘ciao’)”, globals, locals)

Strumenti per la programmazione funzionale in Python

Eval

```
a = 10

b = 20

eval(“print(f’c: {c} d: {d}’)”, {‘c’: a},{‘d’:b})
```

Strumenti per la programmazione funzionale in Python

Eval

Il valore di ritorno di eval è il risultato dell’espressione valutata

eval(“1+1”) -> 2

Strumenti per la programmazione funzionale in Python

Eval

Eval può valutare qualsiasi espressioni con tutti i problemi relativi alla sicurezza.

E’ buona norma non far valutare stringhe generiche oppure settare accuratamente globals e locals in modo da non dare accesso a funzioni critiche

Strumenti per la programmazione funzionale in Python

Eval

Eval può valutare qualsiasi espressioni con tutti i problemi relativi alla sicurezza.

E’ buona norma non far valutare stringhe generiche oppure settare accuratamente globals e locals in modo da non dare accesso a funzioni critiche

Strumenti per la programmazione funzionale in Python

Exec

exec() è molto simile ad eval ma più completo

- Può eseguire anche istruzioni (non solo espressioni)
- Può eseguire codice compilato

Strumenti per la programmazione funzionale in Python

Exec

exec() è molto simile ad eval ma più completo

- Può eseguire anche istruzioni (non solo espressioni)

Es.

exec(“import subprocess”)

Strumenti per la programmazione funzionale in Python

Exec

exec() è molto simile ad eval ma più completo

- Può eseguire codice compilato

Anche in python vi è un formato intermedio (bytecode). Ci si riferisce a questo con compilato

Strumenti per la programmazione funzionale in Python

Exec

exec() è molto simile ad eval ma più completo

- Può eseguire codice compilato

Anche in python vi è un formato intermedio (bytecode). Ci si riferisce a questo con compilato

exec(compiled)

Strumenti per la programmazione funzionale in Python

Compile

compile() compila una stringa (o un file) in bytecode

compile(source, file, mode)

Strumenti per la programmazione funzionale in Python

Compile

compile() compila una stringa (o un file) in bytecode

compile(source, file, mode)

stringa da compilare

compile(“print()”, “<string>”, mode)

Strumenti per la programmazione funzionale in Python

Compile

compile() compila una stringa (o un file) in bytecode

compile(source, file, mode)

file da compilare

compile(“”, “source.py”, mode)

Strumenti per la programmazione funzionale in Python

Compile

compile() compila una stringa (o un file) in bytecode

compile(source, file, mode)

Modalità:
‘exec’ se contiene delle istruzioni
‘eval’ se contiene una singola espressione
‘single’ se contiene una singola istruzione interattiva

Strumenti per la programmazione funzionale in Python

Functools

Modulo utile per la programmazione funzionale in Python

Fornisce funzioni di ordine superiore per eseguire operazioni comuni nella programmazione funzionale

Fornisce anche due classi per rappresentare funzioni: **partial** e **partialmethod**

Strumenti per la programmazione funzionale in Python

Functools

Partial rappresenta una funzione in cui alcuni argomenti sono definiti prima di chiamarla

```
def somma(x, y):
    return x+y


sommapartial = partial(somma, y=1)
```

Strumenti per la programmazione funzionale in Python

Functools

```
def somma(x, y):
    return x+y


sommapartial = partial(somma, y=1)
```

Quando si chiama **sommapartial** basterà passare solo un argomento. La y è stata definita prima.

Strumenti per la programmazione funzionale in Python

Functools

partialmethod() è simile a partial() ma lavora sui metodi di una classe.

E’ possibile definire un metodo partiale che richiama un altro metodo ma con dei parametri preimpostati

Strumenti per la programmazione funzionale in Python

Functools

```
class A:
    def print_str(self, s):
        print(s)
    print_ciao = partialmethod(print_str, s='ciao')
```

Strumenti per la programmazione funzionale in Python

Functools

cmp_to_key trasforma una funzione di comparazione in una funzione chiave

Es.

sorted() ordinava i valori di una lista

sorted([3,4,6,2,8,1])

Vi era la possibilità di specificare una funzione chiave

sorted([(1,2), (2,1)], key=lambda x:x[1])

Strumenti per la programmazione funzionale in Python

Functools

cmp_to_key trasforma una funzione di comparazione in una funzione chiave

Es.

Vi era la possibilità di specificare una funzione chiave

sorted([(1,2), (2,1)], key=lambda x:x[1])

In questo caso ad ogni elemento è applicata la funzione key. E poi gli elementi vengono ordinati usando questo risultato

Strumenti per la programmazione funzionale in Python

Functools

cmp_to_key trasforma una funzione di comparazione in una funzione chiave

Es.

Vi era la possibilità di specificare una funzione chiave

sorted([(1,2), (2,1)], key=lambda x:x[1])

Notare che non avviene nella funzione lambda il confronto tra due elementi. (necessario all'ordinamento)

Strumenti per la programmazione funzionale in Python

Functools

cmp_to_key trasforma una funzione di comparazione in una funzione chiave

Es.

supponiamo di avere oggetti più complessi da ordinare

```
class Data:
    def __init__(self, anno, mese, giorno):
        self.anno = anno
        …
```

Strumenti per la programmazione funzionale in Python

Functools

cmp_to_key trasforma una funzione di comparazione in una funzione chiave

```
class Data:
    def __init__(self, anno, mese, giorno):
        self.anno = anno
        …
```

Non è banale restituire una chiave che sorted possa utilizzare per l'ordinamento

Strumenti per la programmazione funzionale in Python

Functools

cmp_to_key trasforma una funzione di comparazione in una funzione chiave

Possiamo però scrivere una funzione di confronto tra due date

Strumenti per la programmazione funzionale in Python

Functools

```
def cmp_date(x,y):
    if x.anno < y.anno
        return -1
    elif x.anno > y.anno:
        return 1
    else: #caso stesso anno
        if x.mese < y.mese:
            retrun -1
        elif x.mese > y.mese:
            return 1 …
```

E poi utilizzare cmp_to_key per avere una funzione chiave

cmp_to_key(cmp_date)

Strumenti per la programmazione funzionale in Python

Functools

total_orderign è un decoratore per classi che fornisce i metodi di confronto per quella classe.

Basandosi sul metodi __eq__ e uno tra gli altri metodi di confronto (__lt__(), __le__(), __gt__(), __ge__()) inferisce gli altri

Strumenti per la programmazione funzionale in Python

Functools

Quindi basterà implementare due metodi per avere gli altri

```
@total_ordering
class A:
    def __eq__(self, other):
        …
    def __lt__(self, other):
        …
```

Strumenti per la programmazione funzionale in Python

Functools

LRU_cache è un decoratore che fa in modo di evitare la chiamata ad una funzione se è già stata invocata con gli stessi argomenti.

Memorizza il risultato e ritorna direttamente quello senza rieseguire

La funzione deve essere deterministica!

Strumenti per la programmazione funzionale in Python

Functools

LRU_cache è un decoratore che fa in modo di evitare la chiamata ad una funzione se è già stata invocata con gli stessi argomenti.

```
@lru_cache()

def somma(x,y):

    return x+y
```

Se invoco somma(1,2) più volte la funzione verrà eseguita una sola volta

Strumenti per la programmazione funzionale in Python

Functools

LRU_cache è un decoratore che fa in modo di evitare la chiamata ad una funzione se è già stata invocata con gli stessi argomenti.

```
@lru_cache(maxsize=10)
def somma(x,y):
    return x+y
```

Numero massimo di memorizzazioni

Se invoco somma(1,2) più volte la funzione verrà eseguita una sola volta

Strumenti per la programmazione funzionale in Python

Itertools

Itertools è un modulo che fornisce funzioni per generare sequenze di dati iterabili.

Molto utile in programmazione funzionale

Strumenti per la programmazione funzionale in Python

Itertools

iteratori infiniti

count(10) -> 10,11,12 …

cycle([1,2,3]) -> 1, 2, 3, 1, 2, 3, …

repeat(‘a’) -> a,a,a,a…

Strumenti per la programmazione funzionale in Python

Itertools

iteratori che modificano la sequenza

accumulate([1,2,3,4]) -> 1 3 6 10

chain(‘abc’, ‘def’) -> a b c d e f

zip_longest(‘ABC’, ‘xyz’) -> Ax By Cz

starmap(pow, [(1,2), (3,4)] -> applica pow(*x) -> 1 81

Strumenti per la programmazione funzionale in Python

Itertools

iteratori combinatori

permutation([1,2,3]) -> tutte le permutazioni possibilie

combination([1,2,3]) -> tutte le combinazioni possibili