Complementi di Programmazione

Programmazione Funzionale

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

Filippo Muzzini

Paradigmi di programmazione

Avete già visto diversi paradigmi

- Imperativo
- Ad oggetti

Paradigmi di programmazione

Avete già visto diversi paradigmi

- Imperativo
  - Il programma è una serie di istruzioni che vengono eseguite una dopo l'altra
  - Il programmatore scrive le funzioni per risolvere il problema
- Ad oggetti
  - Il programma è organizzato in oggetti che esibiscono metodi
  - incapsulano degli attributi che non vengono esibiti

Paradigmi di programmazione

Avete già visto diversi paradigmi

- Imperativo
  - Es. C
- Ad oggetti
  - Es. Java

Paradigmi di programmazione

Vi sono altri paradigmi (per esempio):

- Funzionale
- Logico

Paradigmi di programmazione

Vi sono altri paradigmi (per esempio):

- Funzionale
  - Programmazione in stile matematico
  - si dichiara che funzione utilizzare per ottenere un risultato
- Logico
  - Si esprime il problema sotto forma di vincoli
  - Si deve trovare un soluzione che soddisfi i vincoli

Paradigmi di programmazione

Vi sono altri paradigmi (per esempio):

- Funzionale
  - Es. LISP
- Logico
  - Es. Prolog

Paradigma Funzionale

E’ un paradigma in stile dichiarativo. Ovvero si dichiara che valore deve assumere un dato nome.

Tipicamente ad un nome viene assegnato il risultato di una funzione.

Si dice che utilizza “espressioni” come in matematica.

Le espressioni vengono valutate per portare al risultato.

Focus su cosa risolvere.

Paradigma Funzionale

- Le Funzioni utilizzate in questo paradigma vengono definite Pure
  - Stesso input implica stesso risultato
  - Non modificano gli argomenti in input
- Non si utilizzano i cicli
  - Si utilizza la ricorsione
- Sono linguaggi di ordine superiore
  - le funzioni possono essere passate come parametri o ritornare come valore
- I nomi (che in altri linguaggi chiamiamo variabili) sono immutabili (per questo non si chiamano variabili)

Paradigma Funzionale in Python

In Python abbiamo tutti gli ingredienti per poter programmare in stile funzionale.

- Le Funzioni utilizzate in questo paradigma vengono definite Pure
  - POSSIBILE
- Non si utilizzano i cicli
  - POSSIBILE
- Sono linguaggi di ordine superiore
  - LE FUNZIONI SONO OGGETTI
- I nomi (che in altri linguaggi chiamiamo variabili) sono immutabili (per questo non si chiamano variabili)
  - POSSIBILE

Paradigma Funzionale in Python

In Python abbiamo tutti gli ingredienti per poter programmare in stile funzionale.

- Le Funzioni utilizzate in questo paradigma vengono definite Pure
  - POSSIBILE

Basta non modificare i parametri in ingresso e al limite copiarli e lavorare sulle copie

Paradigma Funzionale in Python

In Python abbiamo tutti gli ingredienti per poter programmare in stile funzionale.

- Non si utilizzano i cicli
  - POSSIBILE

Se necessario utilizzare la ricorsione

Paradigma Funzionale in Python

In Python abbiamo tutti gli ingredienti per poter programmare in stile funzionale.

- Sono linguaggi di ordine superiore
  - LE FUNZIONI SONO OGGETTI

In Python le funzioni sono oggetti come gli altri, pertanto possono essere passate come argomenti o ritornate da altre funzioni.

Paradigma Funzionale in Python

In Python abbiamo tutti gli ingredienti per poter programmare in stile funzionale.

- I nomi (che in altri linguaggi chiamiamo variabili) sono immutabili (per questo non si chiamano variabili)

Il programmatore può consapevolmente non modificare un variabile (Python).

Paradigma Funzionale in Python

Esempio classico: Fattoriale

```
def fattoriale(x):
    return 1 if x<=1 else return x*fattoriale(x-1)


fact10 = fattoriale(10)
```

… **da qui in poi non devo cambiare fact10 (immutabile)**

Paradigma Funzionale in Python

Esempio classico: Fattoriale

```
def fattoriale(x):
    return 1 if x<=1 else return x*fattoriale(x-1)

fact10 = fattoriale(10)
```

… da qui in poi non devo cambiare fact10 (immutabile)

Versione ricorsiva
Non si usano loop

Paradigma Funzionale in Python

Esempio classico: Fattoriale

```
def fattoriale(x):
    return 1 if x<=1 else return x*fattoriale(x-1)

fact10 = fattoriale(10)
```

… da qui in poi non devo cambiare fact10 (immutabile)

Il risultato della funzione è sempre quello (dato lo stesso input)

Paradigma Funzionale in Python

Esempio classico: Fattoriale

```
def fattoriale(x):
    return 1 if x<=1 else return x*fattoriale(x-1)

fact10 = fattoriale(10)
```

… da qui in poi non devo cambiare fact10 (immutabile)

Non modifico gli argomenti (non modifico proprio nulla)

Paradigma Funzionale in Python

Esempio classico: Fattoriale

```
def fattoriale(x):
    return 1 if x<=1 else return x*fattoriale(x-1)

fact10 = fattoriale(10)
```

… da qui in poi non devo cambiare fact10 (immutabile)

Non modifico gli argomenti (non modifico proprio nulla)

Paradigma Funzionale in Python

La ricorsione può saturare lo stack. Per questo Python ha un limite di chiamate ricorsive permesse.

Lo si può modificare con

import sys

sys.setrecursionlimit(<massimo_chiamate>)

Vantaggi della programmazione funzionale

- Maggiore facilità di debug.
  - Il risultato di una funzione è dato solamente dagli argomenti
- Funzioni tendenzialmente più semplici
  - assenza di cicli da seguire
  - assenza di flussi complessi
- Struttura più interpretabile da un compilatore
  - maggiori ottimizzazioni
  - possibili parallelizzazioni automatiche
  - lazy evaluation -> esecuzione solo quando si ha effettivamente bisogno del risultato

Svantaggi della programmazione funzionale

- Cambio di paradigma implica cambio di abitudini
  - mentalità diversa
  - struttura del codice diversa
  - Dopo un po’ di allenamento comunque diventa automatico
- Poco utilizzata
  - l’imperativo è più semplice

Strumenti per la programmazione funzionale in Python

Python mette a disposizione diversi strumenti (e sintassi) utili al fine della programmazione funzionale

Alcuni di essi li abbiamo già visti senza sapere che fossero funzionali

Strumenti per la programmazione funzionale in Python

List Comprehension

Costrutto sintattico per la creazione di liste (o dizionari) senza un ciclo for classico (ma innestato dentro la lista)

[expression **for** value **in** iterable **if** condition]

Sintassi con stesse parole di un ciclo for ma la semantica è differente

Il risultato è una lista risultante da tutti i risultati di expression calcolati con ogni value nell'iterabile (se la condizione è vera)

Strumenti per la programmazione funzionale in Python

List Comprehension

```
[k*k for k in range(1, n+1)]
```

```
[k for k in range(1,n+1) if n % k == 0]
```

Strumenti per la programmazione funzionale in Python

List Comprehension

Set

```
{ k*k for k in range(1, n+1) }
```

Dictonary

```
{ k : k*k for k in range(1, n+1) }
```

Strumenti per la programmazione funzionale in Python

List Comprehension

Vi possono essere anche più for

[x for iter2 in iter1 for x in iter2]

Notare che i nomi vengono letti da sinistra verso destra

[x for x in iter2 for iter2 in iter1] -> Errore perchè iter2 non si sà cosa sia quando viene visto

Strumenti per la programmazione funzionale in Python

Lambda

Fondamento della programmazione funzionale.

Ogni funzione ha argomenti e risultato. Funzioni Pure

Potenzialmente un intero programma può essere scritto con sole funzioni lambda (Turing Completo)

Strumenti per la programmazione funzionale in Python

Lambda

lambda argomenti**:** espressione

Questa funzione può avere un numero qualsiasi di argomenti ma solo un'espressione, che viene valutata e restituita

Strumenti per la programmazione funzionale in Python

Lambda

Esempio filter() -> filtra in base al risultato di una funzione (True tiene, False scarta)

filter(lambda x: (x % 2 != 0), iter1)

Esempio map() -> mappa il risultato di una funzione

map(lambda x: x*2, iter1)