Cenni di Programmazione Funzionale

“To Iterate is Human, To Recuse, Divine”

Cos'è la Programmazione Funzionale?

La programmazione funzionale è un paradigma di programmazione in cui l'operazione fondamentale è l'applicazione di funzioni ai loro argomenti.

Un programma non è altro che una funzione, definita in termini di altre funzioni, che elaborano l'input fornita al programma e restituiscono il loro risultato finale.

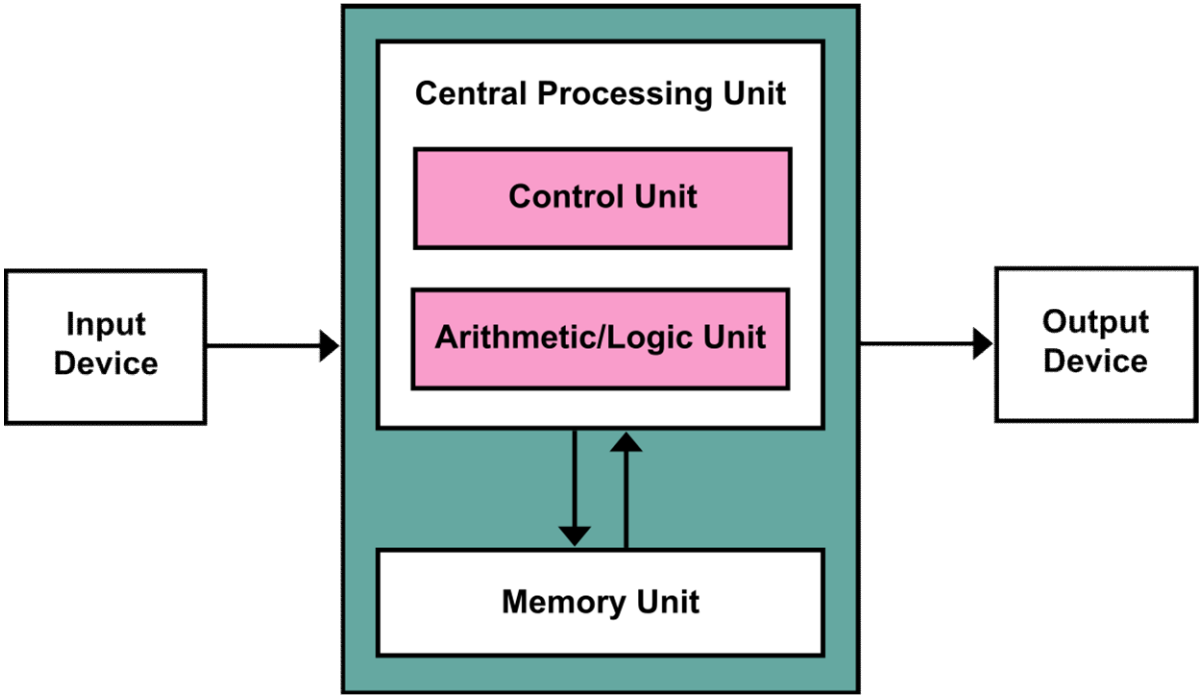
Tutte queste funzioni sono da intendersi come "funzioni matematiche", che vengono chiamate **FUNZIONI PURE**: a parità di input restituiscono sempre lo stesso output (no *side effects*)

Caratteristiche della Programmazione Funzionale

1. Le funzioni sono “first class objects” (sono trattate come qualsiasi altro tipo di dato)
2. La **RICORSIONE** è la principale struttura di controllo. Non esistono altri modi di iterare
3. Il focus principale è sul processare liste
4. La valutazione di funzioni avviene in modalità **LAZY**. Per esempio è possibile definire funzioni su liste infinite
5. Per evitare gli effetti colletarelli il più possibile, ogni variabile non può cambiare valore, ma è di tipo *read-only*
6. Vengono utilizzate ***High Order Functions***: funzioni che operano su funzioni, che operano su funzioni, …

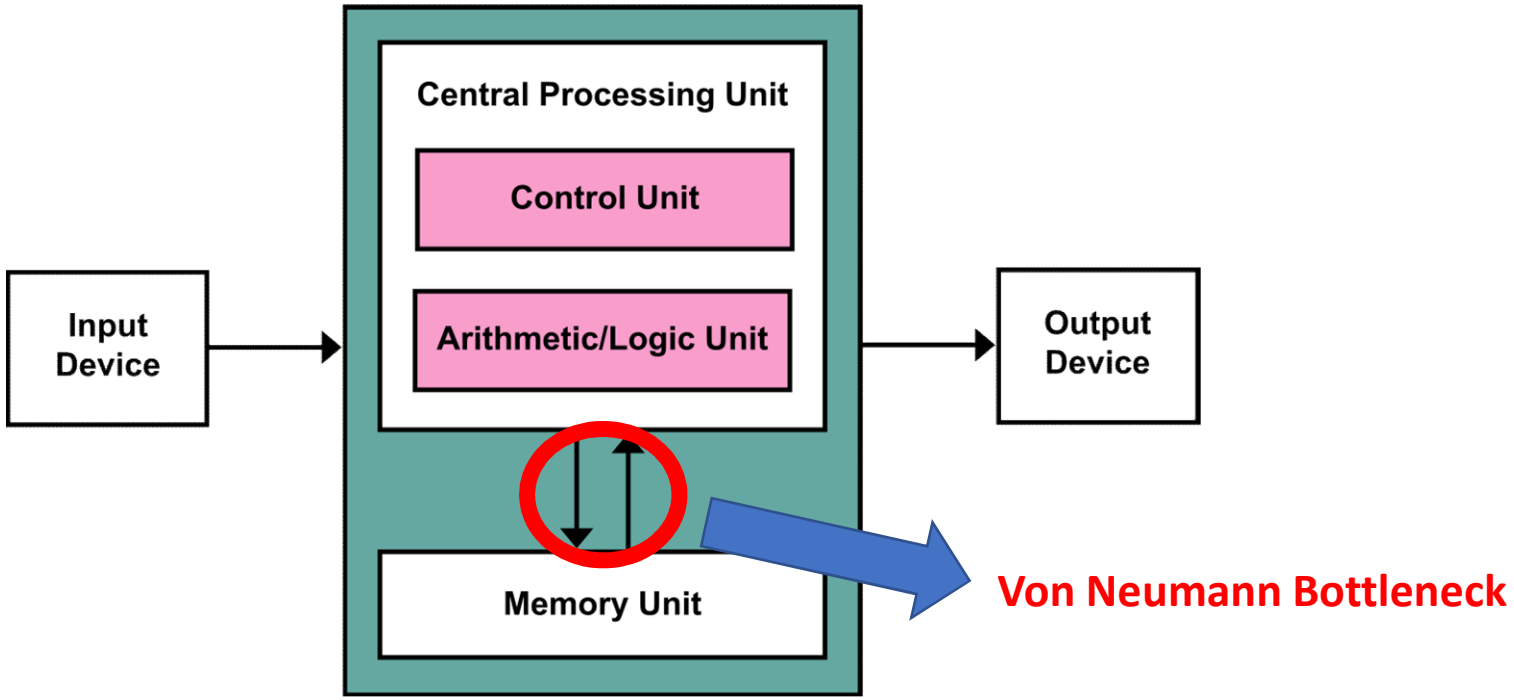
Caratteristiche della Programmazione Imperativa

La maggior parte dei linguaggi di programmazione sono basati su un paradigma di programmazione imperativo.
Questi linguaggi assumono, spesso implicitamente, che saranno eseguiti su un "**Computer di von Neumann**"



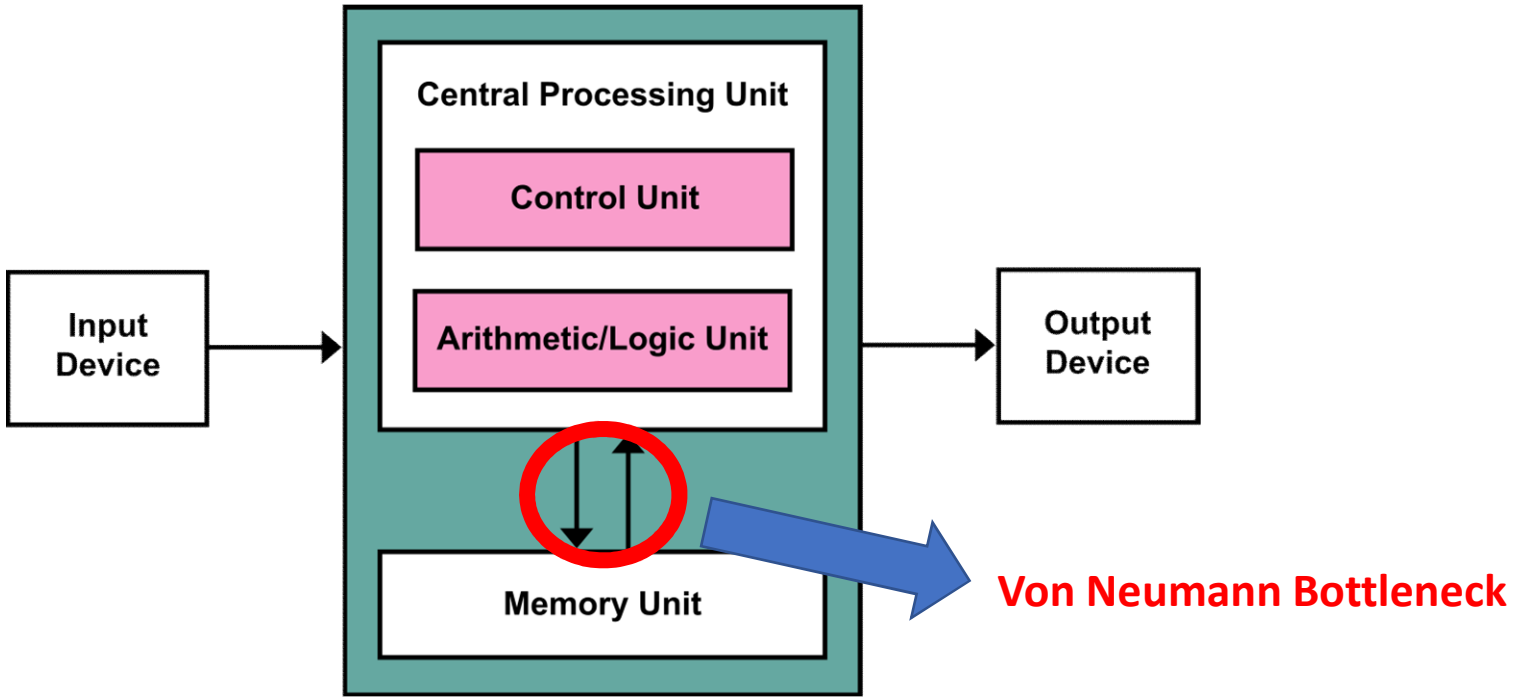
Caratteristiche della Programmazione Imperativa

La maggior parte dei linguaggi di programmazione sono basati su un paradigma di programmazione imperativo.
Questi linguaggi assumono, spesso implicitamente, che saranno eseguiti su un "**Computer di von Neumann**"



Von Neumann Bottleneck

La maggior parte del traffico dati tra la CPU e la memoria sono riguarda i dati, ma dove andare a trovarli e/o memorizzare.
I linguaggui di von Neumann usano le variabili per imitare le celle di memoria di un computer; le variabili sono trattate come fossero delle celle di memoria.



Assegnamento

Un comando di assegnamento divide un programma in due mondi diversi:

1. Il primo mondo comprende il ***right hand side*** dell'assegnamento, ed è dove avviene la parte importante dei calcoli. Esempio: c := **c + a[i]*b[i]**

2. Il secondo è il mondo dei "comandi", di cui l'assegnamento è l'elemento principale. Questo secondo mondo è quello disordinato con NESSUNA PROPRIETÀ MATEMATICA UTILE. Per assemblare un risultato come sequenza di "comandi", i linguaggi di von Neumann offrono delle primitive comuni **for**, **while**, **if-then-else**.

1. Funzioni come First Class

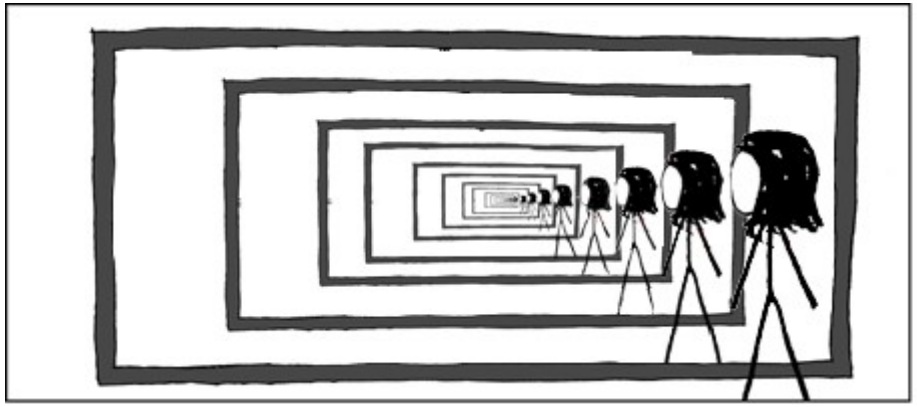
- Esempio: Calcolo di un polinomio e di una sua derivate

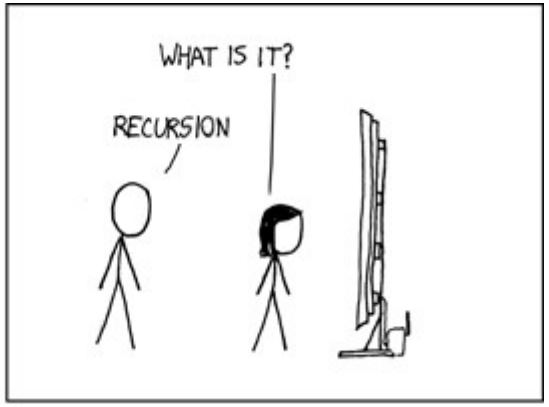
VEDERE NOTEBOOK PYTHON SU GITHUB

2. Funzioni ricorsive

In generale, una definizione **ricorsiva** (o induttiva) è fatta di due parti:

1. Il **caso base** che specifica il risultato per quel caso
2. Il **caso ricorsivo** che specifica lo stesso problema per un suo sottoproblema più semplice da risolvere





2. Funzioni ricorsive: Fattoriale

Definizione di fattoriale con “postfix” operatore **!**:

$$1! = 1$$

$$n! = n*(n-1)!$$

Oppure usando una notazione “prefix” per fattoriale:

Fattoriale(1) = 1

Fattoriale(n) = n***Fattoriale**(n-1)

2. Funzioni ricorsive: Fibonacci

Funzione di Fibonacci:

Female(0) = 1

Female(1) = 1

Female(n) = **Female**(n-1) + **Female**(n-2)

NOTA:

1. In questo caso ci sono due **casi basi**
2. Nel caso ricorsivo ci sono due chiamate ricorsive

2. Funzioni ricorsive: Palindrome

Esercizio: scrivere una definizione ricorsiva per controllare se una data stringa è una palindrome, ovvero se li legge allo stesso modo sia da sinistra verso destra che da destra verso sinistra.

VEDERE NOTEBOOK PYTHON SU GITHUB

3. Processare liste

- Funzioni elementari sulle liste:

```
Head = lambda Ls: Ls[0]
Tail = lambda Ls: Ls[1:]
```

- Esempio definizione della funzione **`sum(Ls)`**
- Definizione di iteratori: un iteratore definisce un flusso di dati, e ne elabora uno alla volta

```
Ls = [1,2,3,4,3,2,1]
iteratore = iter(Ls)
print(type(iteratore))
next(iteratore), next(iteratore)
A = next(iteratore)
```

4. Lazy Evaluation e Liste infinite

- Esempio: funzione counter, lista di numeri primi, funzione enumerate

VEDERE NOTEBOOK PYTHON SU GITHUB