Considerazioni sul corso Basi di Informatica.

• **Cosa dev'essere (in breve)** Per **una parte degli studenti** del primo anno il corso Basi di Informatica è il primo ed ultimo contatto con un esperto della materia. Per loro il corso deve servire come **alfabetizzazione e orientamento** su strumenti di produttività tecnica e scentifica. Lo spettro dev'essere molto ampio privilegiando la capacità di orientarsi, se necessario a scapito dell'approfondimento.

• **Cosa NON dev'essere (in breve)** Progettare algoritmi solidi ed efficienti è relativamente facile per un buon matematico. Ma non è con un corso di programmazione che si diventa buoni matematici. Quindi il corso di informatica **non dev'essere una introduzione alla programmazione**. Eventualmente, per questo c'è un corso al secondo anno. Inoltre **non deve aveve come obbiettivo il calcolo scentifico** perché interessa solo la parte degli studenti che seguirà il corso del secondo anno.

• **Cosa intendo per alfabetizzazione** Purtroppo, il matematico medio è informaticamente meno competente non solo del fisico medio ma spesso persino di biologi, geologi, ecc.

Il matematico medio usa strumenti di produttività più adatti ad ambienti di "amministrativi" che ad ambienti "scentifici/tecnici". Per esempio, l'unico linguaggio a marcatori che conosce è `LaTeX`. Spesso non distingue bene i confini tra IDE e linguaggio. Non conosce gli strumenti per lo sviluppo e collaborativo e distribuito di progetti (`Dropbox` è il massimo che riesce a concepire). Non conosce i paradigmi e gli strumenti *reproducible research (RR)* che, anche se non lo riguardano nell'immediato, è comunque formativo.

Il corso del primo anno deve aprire gli orizzonti. E rompere il circolo vizioso che porta a riprodurre l'ignoranza delle generazioni precedenti. Il corso dovrebbe assicurare che lo studente

1. conosca la struttura del file system di `Unix`
2. sappia interagire con la linea di comando della shell (`Bash` è la scelta ovvia)
3. sappia cosa sono i linguaggi a marcatori (avere una vaga idea della struttura di e filosofia di `LaTeX`, `Html5`, `MarkDown`)
4. conosca le problematiche che i *control version systems* provano a risolvere (per esempio conosca i concetti di base di `git`)
5. conosca le problematiche relative alla RR e quali sono gli strumenti a disposizione per la pubblicazione di RR (io suggerirei `Jupyter`, più cenni ad altri)
6. Il linguaggio di programmazione da introdurre deve avere qualche utilità anche a chi non avrà né occasione né incentivi ad imaparne un altro. Sono quindi da **evitare linguaggi di nicchia** per quanto interessanti. Sarebbe una perdita di tempo per gli studenti meno motivati.

I linguaggi interpretati hanno grossi vantaggi di fruibilità. Suggerirei per esempio `Python`. È uno dei due linguaggi più usati nelle *Data Sciences* e forse il più usato nell'*Artificial Intelligence/Machine Learning*. Ma il vero argomento a favore è che facilmente spendibile anche per applicazioni di basso profilo (applicazioni alla didattica, amministrazione, gestione di un piccolo sito web). Questo è un aspetto essenziale per gli studenti per cui questo è l'unico corso di informatica.

Altri linguaggi più adatti al calcolo scientifico (`C++`, `MatLab/Octave`, ecc.) possono essere presi in considerazione per il corso del secondo anno. (`Julia` potrebbe essere un ottimo compromesso, ma è ancora troppo di nicchia.)

• **Fattibilità** Mi rendo conto della difficoltà di progettare un corso (e soprattutto l'esame) con un programma di così ampio respiro. Ma è importante non riprodurre una generazione di matematici restii all'uso di tecnologie evolute.