L’informatica e Python al Liceo: conoscere, pensare, progettare, realizzare

Maurizio Boscaini
ITI Marconi di Rovereto
Via Monti 1, 38068 Rovereto (TN)
maurizio.boscaini@gmail.com

L'informatica, intesa come scienza dell'informazione, è una disciplina di sicuro interesse nel percorso formativo liceale. Il linguaggio di programmazione Python semplifica la possibilità di toccare con mano, a vari livelli, concetti e problematiche interessanti, utili ed importanti. La programmazione può così esprimere effettivamente le potenzialità del proprio unicum formativo: l’essere al contempo contenuto specifico e collante interdisciplinare. Il beneficio atteso ed in parte già riscontrato è quello di favorire il processo formativo, invitando gli alunni alla realizzazione di progetti, semplici o complessi, frutto di conoscenza e ragionamento. È questo un tentativo di riequilibrare l’attuale insegnamento dell’informatica, spesso ridotto al mero uso delle nuove tecnologie, riportando al centro dell’azione didattica lo studio teorico come imprescindibile fase della crescita intellettuale e l’attività laboratoriale come essenziale momento di verifica e di rielaborazione personale.

1. Introduzione

La comunicazione e l'informazione sono intorno a noi. Il mondo virtuale sempre più frequentemente condiziona il mondo reale. Internet e l'*Information and Communication Technology* (ICT) hanno raggiunto in quasi tutto il pianeta livelli di diffusione e pervasività impensabili solo quindici anni fa.

Sono soprattutto i giovani i voraci consumatori di questi nuovi media. L'alfabetizzazione informatica delle nuove generazioni sembra ormai raggiunta all'inizio del triennio della scuola superiore.

In questo quadro, quale senso può avere l'insegnamento autonomo dell'informatica? Quali contributi specifici può oggi dare questa disciplina ad uno studente, che frequenta il triennio di un Liceo? Tali domande sottendono il quesito pedagogico, culturale e soprattutto epistemologico di fondo: la formazione informatica quale contributo può dare alla società della conoscenza?

Inoltre, l’attuale quadro dell’offerta didattica nella scuola superiore italiana mette a frutto le potenzialità di questa disciplina?

A. Andronico, L. Colazzo (Eds.): DIDAMATICA 2009 – ISBN 978-88-8443-277-3

Cercheremo di dare alcune risposte, partendo dalla presentazione di alcuni quadri di riferimento per contestualizzare meglio la tematica.

2. L'informatica nel Liceo

I licei tradizionali, classico, scientifico ed artistico, non prevedono l'insegnamento dell'informatica nei loro piani di studio.

Il Liceo scientifico con PNI (Piano Nazionale Informatica) contempla l'insegnamento della matematica integrato con quello dell'informatica. Tale tipologia liceale in questo scritto non viene presa in considerazione, dal momento che, a dispetto del significato dell'acronimo PNI, curiosamente l'informatica non risulta avere in questo ambito dignità e valenza di disciplina autonoma e non prevede docenti specializzati.

L'insegnamento delle scienze dell'informazione hanno segnato un punto di arrivo valido ed interessante con l'istituzione della disciplina autonoma "Informatica e sistemi automatici" nella sperimentazione del Liceo scientifico e tecnologico (LST) progetto "Brocca" [Commissione Brocca, 1992] (anche se ci troviamo di fronte alla deludente prospettiva che tale ordine di Liceo venga cancellato dall'attuale Ministero della Pubblica Istruzione).

I programmi ministeriali di questa materia sono oltremodo estesi a fronte di uno spazio orario di sole tre ore settimanali. È previsto lo studio dei sistemi e l'invito a scegliere tecnologie informatiche, che siano anche tecnologie-strumento volte a favorire un approccio didattico multidisciplinare e progettuale. Nel documento ministeriale troviamo che:

> *"la disciplina informatica e sistemi automatici ha lo scopo di introdurre lo studente all'analisi ed alla soluzione dei problemi con i metodi tipici della tecnologia e nello stesso tempo di offrire supporti tecnici all'indagine scientifica".*

I programmi svolti dalle varie scuole risultano anche molto diversi tra loro ed evidenziano in più di un caso la difficile sintesi tra i sistemi automatici e l'informatica. Vi sono docenti, che privilegiano le tematiche dell'elettronica e delle telecomunicazioni, mentre altri si concentrano quasi esclusivamente sui linguaggi di programmazione.

Possiamo guardare all'informatica sia come palestra della logica e del metodo scientifico sia come contributo all'unitarietà dei tre anni di corso.

3. Tra tecnologia e scienza

3.1 Anche le parole contano

Il fatto che più lingue abbiamo termini semanticamente differenti per indicare la disciplina, che stiamo considerando, contribuisce probabilmente ad alimentare incertezza e confusione.

In inglese, l’aspetto scientifico dell’informatica è ricordato espressamente dall’uso del termine *computer science*.

Il termine italiano informatica, ripreso dalla crasi effettuata in lingua francese di *inform(ation electronique ou autom)atique*, nasconde probabilmente un po’ troppo l’aspetto teorico e concettuale della disciplina.

Infatti, nonostante più fonti, tra le quali citiamo Wikipedia, riportino che “L'informatica è una scienza interdisciplinare che riguarda tutti gli aspetti del trattamento dell'informazione mediante procedure automatizzabili” [Wikipedia, Informatica, 2009], l’accezione tecnologica sembra spesso essere decisamente preponderante rispetto a quella scientifica.

3.2 Le certificazioni

L’interpretazione più tecnicistica e legata all’elaboratore prevale probabilmente per più ragioni. Negli ultimi anni, a mio avviso, anche le certificazioni informatiche hanno giocato un ruolo più o meno importante in tal senso. Le certificazioni si possono collocare in due macrocategorie: quelle relative all’alfabetizzazione informatica e quelle specialistiche.

Mentre le prime si rivolgono potenzialmente a tutti, le seconde sono ristrette per lo più alla nicchia di coloro che svolgono una professione nell’ambito dell’ICT.

Questo presumibilmente ha fatto sì che la percezione di cosa debba dare la formazione scolastica informatica si riduce all’uso dell’elaboratore e dei principali programmi di *office automation*.

I programmi di certificazione, tra i quali sicuramente il più diffuso e conosciuto è l’ECDL (*European Computer Driving Licence*) di AICA (Associazione Italiana per l’Informatica ed il Calcolo Automatico), hanno il merito di aver contribuito all’alfabetizzazione digitale. Quello che viene certificato sono conoscenze e competenze per lo più di tipo operativo. Una conseguenza di questo è che, malgrado tutto, si tende a ridurre e forse a snaturare il significato di informatica (mi riferisco al significato percepito in generale almeno nella scuola italiana), che diviene molto *computer* e poco *science*.

Partendo da analoghe constatazioni, Giorgio Casadei motiva l'utilità di istituire "Le Olimpiadi del Problem Solving" per il primo ciclo scolastico con la finalità di [Casadei, 2008]:

- far conoscere l'informatica come disciplina scientifica,
- rendere più ricco (per gli studenti) l'insieme di conoscenze e competenze per il problem solving.
-
- Tali considerazioni sono ancor più valide nel secondo ciclo di istruzione.

3.3 Nel mezzo del cammin...

Consideriamo la seguente domanda di ricerca: per uno studente di un liceo (non solo di uno scientifico o di uno scientifico e tecnologico) quale può essere e quale valenza può avere un percorso informatico che vada oltre l'alfabetizzazione ma non si perda neppure nei meandri dei tecnicismi?

L'ipotesi è che tale percorso non solo sia possibile, ma che apporti numerosi benefici didattici e formativi. I paragrafi che seguono offrono alcuni brevi spunti e riflessioni basate su esperienze svolte sul campo negli ultimi anni. Tali esperienze, pur non avendo la pretesa di offrire una dimostrazione dell'assunto dato, possono almeno far intuire la bontà di una tale proposta didattica.

4. L'importanza ed il ruolo dei linguaggi di programmazione

Nel processo intellettivo, che porta lo studente ad affrontare un problema ed ad individuare la sua soluzione, la traduzione dell'algoritmo risolutivo in pseudocodifica ed il flow-chart è un passaggio molto utile. Che lo studente arrivi a scrivere un programma in linguaggio di programmazione è, a mio parere, indispensabile.

L'importanza di "sporcarsi le mani" col codice si evidenzia in più occasioni in classe in maniera lampante. La teoria viene meglio digerita quando l'alunno è portato a:

- mettere per iscritto i propri ragionamenti,
- utilizzare un formalismo stretto,
- verificare i risultati,
- indagare, correggere, migliorare.
-

In tal modo, si attua efficacemente il motto didattico "dal fare al sapere", in andata e ritorno. L'elaborato codificato lascia sempre spazio alla personalità ed alla creatività. A riprova di questo il fatto che il codice realizzato dagli studenti

nelle verifiche e nelle esercitazioni, anche per problemi molto semplici, risulta quasi sempre vario ed originale (se non hanno copiato :-)).

4.1 La scelta del linguaggio

Per dare un contributo all’insolubile dibattito su quale sia il migliore linguaggio di programmazione per la didattica, riporto alcune osservazioni svolte come docente di “Informatica e sistemi automatici” in un LST.

4.2 Delphi: troppo visuale

Il Pascal rimane un linguaggio didatticamente valido e Delphi un ottimo ambiente di sviluppo. Calare, però, questi linguaggi ed ambienti in un percorso liceale con un quadro orario limitato per la materia risulta difficile. Tre sono le difficoltà più evidenti (che in altri contesti si rivelano, al contrario, punti di forza):

- la dichiarazione obbligatoria dei tipi di dato,
- la complessità del paradigma di programmazione ad oggetti (nell’Object Pascal di Delphi),
- l’ambiente di sviluppo visuale di Delphi, che facilita la realizzazione di applicazioni belle e funzionali, ma sacrifica, purtroppo, il processo di apprendimento, nascondendo i meccanismi logici alla base della programmazione.

4.3 C/C++: troppo criptico

Il C ha ancora il suo fascino. Passare a questo ambiente dopo gli effetti speciali di Delphi può avere per il docente quasi un buon sapore di antico (quando le cose erano lì e si capivano).

Gli studenti si trovano però di fronte anche ad altri problemi:

- la sintassi del linguaggio è difficile e criptica,
- è oltre modo difficile se non addirittura impossibile costruire programmi di una certa complessità, che possano essere soddisfacenti, fattibili e comprensibili per tutti gli alunni della classe (anche arrivando a farlo ci si arriva comunque al termine del corso, quando ormai è troppo tardi!),
- è difficile riuscire ad arrivare con una classe di Liceo alla programmazione ad oggetti del C++.

4.4 Python: eureka!

Python permette un approccio alla scienza dell’informazione ed alla codifica abbastanza diverso dai precedenti linguaggi.

La sua introduzione non è comunque semplice. Mancano ancora una base di esperienze didattiche ampie e consolidate e testi idonei, cui poter fare riferimento per supportare le attività di teoria e laboratorio.

Nello specifico dei libri si fa notare sicuramente il testo di Allen B. Downey, Jeffrey Elkner e Chris Meyers: “How to Think Like a Computer Scientist” [Downey et al., 2002] disponibile in rete anche nella traduzione italiana di Alessandro Pocaterra: “Pensare da informatico: Imparare con Python”. Il testo, però, pur nascendo dall’incontro di insegnanti e programmatori, risente di una forte impostazione didattica americana. È rivolto quasi esclusivamente alla pratica e finisce così per assomigliare più ad un manuale d’uso e ad un *tutorial* piuttosto che ad un testo adatto alla scuola italiana, cui facciamo riferimento.

Elenchiamo ora alcuni vantaggi dell’uso di Python nella didattica.

4.4.1 “No magic”...o quasi

Python può essere insegnato, spiegando subito quello che si sta facendo. Non si nasconde nulla, o quasi. Non si rimanda ad argomenti che si affronteranno più avanti. Come col vecchio BASIC lo studente può capire fin da subito quello che sta scrivendo e soprattutto perché.

Molti autori per dimostrare questa affermazione riportano le righe ed i concetti necessari per scrivere e capire il classico “Hello World!”: una semplicissima riga in Python contro alcune righe, ma soprattutto concetti quali: funzione, classi, importazione di moduli, etc. nella maggior parte degli altri linguaggi di programmazione.

4.4.2 Sintassi e paradigmi

Python ha una sintassi semplice, che aiuta il processo di astrazione e la focalizzazione del problema. L’indentazione obbligatoria per rappresentare i blocchi di codice costringe ad una scrittura ordinata e pulita.

L’apprendimento può seguire un percorso graduale. Si parte utilizzando la semplicità del paradigma imperativo per poi passare eventualmente alla programmazione ad oggetti. Si arriva, infine, a toccare alcuni aspetti della programmazione funzionale.

4.4.3 Open source, multipiattaforma ed interattivo

La filosofia dell’open source si sta diffondendo sempre più nella didattica e nella ricerca. Python è in questo solco. Da qui discende abbastanza naturalmente l’indipendenza dal sistema operativo, che svincola il docente dal

dover imporre agli studenti una determinata piattaforma per lo studio e gli esercizi.

Python dispone di una shell interattiva (anche questa in comune col BASIC degli home computer), nella quale lo studente può provare tecnica e ragionamento, verificando immediatamente gli effetti delle proprie azioni.

5. Esperienze e proposte

Presento di seguito alcune esperienze svolte in questi anni al Liceo con l'obiettivo di dare un piccolo assaggio delle potenzialità di Python nella didattica.

5.1 Pygraph e VPython

Per quanto riguarda l'analisi matematica e la geometria nella loro veste grafica, è molto utile ed originale la libreria interamente in Python **pygraph** [Zambelli, 2009].

Sempre per stimolare l'interesse degli alunni e con un occhio rivolto alla fisica possiamo utilizzare VPython per la simulazione di semplici esperimenti. In questo caso, l'immediatezza della visualità si unisce la possibilità di introdurre concetti informatici come classi ed oggetti. Riporto un esempio col relativo output grafico (vedi Fig. 1), preso da http://vpython.erikthompson.com/, che simula il lancio di una palla da un edificio.

```
from visual import *

scene.width = 800
scene.height = 600

scene.autoscale = 0
scene.range = (100,100,100)
scene.center = (0,40,0)
scene.background = color.white

ball = sphere(pos=(0,102,0),radius=2,
       color=color.red)
ground = box(pos=(0,1,0),size=(100,2,10),
       color=color.green)
building = box(size=(6,100,6),
pos=(0,50,0), color=color.blue)

gravity = 9.8   # m/s**2
velocityX = 6   # m/s
seconds = 0
dt = .01

finished = False
while not finished:
```

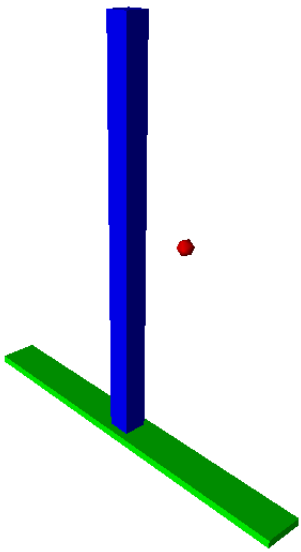

Fig. 1 – Esempio di VPython

```
    rate(100)
    seconds += dt
    # y(t) = y0 + v0*t + .5 * a * t**2
    ballY = 100 - .5 * gravity * seconds**2
    ballX = velocityX * seconds

    ball.pos = vector(ballX,ballY,0)

    if ballY - 2 <= 0:
        finished = True
        print "Tempo di caduta:", seconds, "sec."
     Print "Distanza percorsa (asse X):", ballX, "m"
```

Nel programma, inizializzo l'ambiente di visualizzazione, creo i tre oggetti utili alla simulazione (ball, ground, building), imposto le costanti fisiche d'interesse, quindi faccio cadere la palla fino a che non tocca il suolo. Infine, stampo il tempo impiegato e la distanza orizzontale percorsa dalla palla nella caduta.

L'esercizio può essere proposto dopo alcune settimane di introduzione al linguaggio ed alla programmazione. Gli allievi possono abbastanza agevolmente comprendere tutto ciò che viene scritto, costruendo passo passo, accompagnati dall'insegnante, l'esperimento di simulazione.

5.2 Calcolo della distanza di Hamming

Le funzioni Python sottostanti si sono rivelate un utile strumento per apprendere principi della teoria dell'informazione e delle telecomunicazioni digitali.

```
def distanza_hamming(p1, p2):
    '''Ritorna la distanza di Hamming tra due parole binarie
    di uguale lunghezza, distanza data dal numero di posizioni
    in cui i bit corrispondenti nelle parole sono diversi
    '''
    distanza = 0
    for i in range(len(p1)):
        if p1[i] != p2[i]:
            distanza += 1
    return distanza

def distanza_minima_hamming(codice):
    '''Ritorna la distanza minima di Hamming
    tra qualsiasi coppia di parole di un codice
    '''
    distanza_minima = len(codice[0])
    for p1 in codice:
        for p2 in codice:
            if p1 != p2:
```

```
                distanza = distanza_hamming(p1, p2)
                if distanza < distanza_minima:
                    distanza_minima = distanza
    return distanza_minima

codice = ['11000', '10100', '10010', '10001', '01100']
print distanza_minima_hamming(codice)
```

Nelle due ultime righe, definisco un codice di cinque parole (o stringhe) di cinque caratteri e ne stampo la distanza minima di Hamming (in questo caso uguale a 2). Questa distanza misura il numero di sostituzioni necessarie a trasformare una parola del codice in un’altra. In questo modo, ho una stima, dipendente dal codice scelto, della capacità di rilevare o correggere gli errori di trasmissione dovuti a segnali indesiderati (rumore), che trasformano una parola in un’altra.

L’esercizio chiarifica l’importanza della scelta di un codice rispetto ad un altro.

5.3 La ricorsione

Lo studio della ricorsione fornisce un contributo interessante alla logica ed al ragionamento. Istruire un elaboratore perchè risolva problemi di tipo fisico-matematico attraverso la ricorsività è un buon modo per farla apprezzare.

Ecco un conciso esempio di calcolo ricorsivo della sequenza di Fibonacci:

```
def fibonacci(n):
    if n < 2:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

for i in range(15):
    print fibonacci(i),
```

che produce il seguente output:

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
```

Nella funzione, l’esperienza didattica mostra come il passaggio dalla definizione matematica della successione di Fibonacci alla sua espressione in Python risulti non solo semplice ma anche “abbastanza uniforme”. Gli studenti, infatti, scrivono codici molto simili, dal momento che devono sottostare all’uso coatto dell’indentazione e non si possono dimenticare terminatori di istruzione e/o delimitatori di blocco di codice (visto che non sono previsti).

6. Conclusioni e sviluppi futuri

Spero di aver mostrato come vi siano valide argomentazioni per capire la necessità di una chiara distinzione anche terminologica tra i cosiddetti “elementi di informatica”, intesi come uso dell’elaboratore, dei programmi di *office automation* e dei principali servizi Internet, e la scienza dell’informazione.

Quest’ultima può dare contributi utili ed importanti nel panorama didattico dei licei italiani, soprattutto nell’attuale società della conoscenza.

Si aprono ampi spazi per approfondire le esperienze illustrate e per ricercare nuove esperienze didattiche seguendo più direzioni. Python può essere provato come collante interdisciplinare non solo con la fisica e la matematica, ma anche con la letteratura italiana e straniera, attraverso la scrittura di programmi per l’analisi stilometrica, o con la chimica, grazie ad interessanti progetti come Biopython (http://biopython.org/), o ancora con le scienze della terra e la biologia, grazie a strumenti come QGIS (http://www.qgis.org/) per la programmazione di GIS (*Geographical Information System*).

Bibliografia

[Casadei, 2008] Casadei G., Le Olimpiadi di Problem Solving, http://www.viagonzagadue.it/problemsolving/Olimpiadi%20di%20problem%20solving_Casadei.ppt, 2008

[Commissione Brocca, 1992] “Commissione ministeriale Brocca” http://www.edscuola.it/archivio/norme/programmi/trienniobrocca.pdf, 598-606

[Downey et al, 2002] Downey A. B., Elkner J., Meyers C., How to Think Like a Computer Scientist: Learning with Python, Green Tea Press, Needham, MA, 2002

[Wikipedia, Informatica, 2009] AA.VV. http://it.wikipedia.org/wiki/Informatica

[Zambelli, 2009] Zambelli D., http://www.fugamatematica.blogspot.com/, 2009