Corso Programmazione 2014-2015

(docente)
Fabio Aiolli
E-mail: aiolli@math.unipd.it
Web: www.math.unipd.it/~aiolli

(assistenti laboratorio)
M. Donini, M. Virgulin

Dipartimento di Matematica
Torre Archimede, Via Trieste 63

Orario delle lezioni e esercitazioni

~32 ore di lezioni in aula P200

- Mercoledi
  - Ore 11:30 - 12:15
- Giovedi,Venerdi'
  - Ore 11:30 – 13:00

~32 ore di esercitazioni in laboratorio (Paolotti)

- Martedi, Ore 14:00 – 17:00
- ISCRIVERSI ALLA LISTA UNIWEB!

Risorse per il corso

A. Downey, J. Elkner, C. Meyers. "Pensare da Informatico, Imparare con Python" scaricabile liberamente da http://www.python.it/doc/Howtothink/HowToThink_ITA.pdf

B. F.Aiolli. "Programmazione (scientifica) con Python" ESCULAPIO

C. Altri testi saranno consigliati nel corso..

Altro materiale sarà disponibile sul sito web del corso:

http://www.math.unipd.it/~aiolli/corsi/1415/prgxmat/prg.html

Il GOOGLE GROUP del corso:

http://groups.google.it/group/prxmat15pd

Il GOOGLE GROUP del corso:

http://groups.google.it/group/prxmat15pd

ISTRUZIONI x l'iscrizione

- Nickname: <Nome> <Cognome> (separato da spazio)

(per esempio, Fabio Aiolli, Michele Donini, ecc.)

- Inserire indirizzo di posta elettronica

Esercizi a Punti

Prima di tutto, bisogna iscriversi alla lista UNIWEB
"IscrizioneLaboratorio" (istruzioni sul gruppo!!!)

REGOLE (provvisorie)

Ogni settimana:

- Vengono proposti uno o più esercizi
- Lo studente può sottomettere una soluzione entro una certa scadenza
- Un sottoinsieme delle consegne verranno corrette (circa un terzo)

Valutazione:

- I progetti che vengono corretti ricevono una valutazione X >= 0 (X dipende dalla difficoltà degli esercizi)
- I progetti consegnati ma non corretti ricevono Y punti, dove Y è la media dei punti ottenuti dai progetti che sono stati corretti
- Gli altri studenti ricevono 0 punti!

Podio: Le soluzioni di alcuni progetti selezionati verranno pubblicate

Esame Scritto

Parte Propedeutica

- Domande a risposta multipla, tipicamente sulla sintassi del Python e/o nozioni teoriche di base + semplici programmi

Prima Parte (costa molti punti malus)

- Analisi e implementazione di programmi Python

Seconda Parte (vale pochi punti bonus)

- Analisi e implementazione di algoritmi complessi

Colloquio (per esame da 8 crediti, vecchio ord.)

- Contenuti di Introduzione alla Programmazione

Contenuti del corso

(a spanne)

- Fondamenti di Programmazione
- Panoramica sul linguaggio Python
- Strutture dati ed algoritmi
- Programmi x il calcolo scientifico e i giochi

Due parole su Python

- http://it.wikipedia.org/wiki/Python
- http://www.python.org/~guido/

- **Python** è un linguaggio di programmazione ad alto livello, orientato agli oggetti
- Ideato da GUIDO VAN ROSSUM (Matematico/Informatico) all'inizio degli anni novanta. Il nome fu scelto per via della passione di van Rossum per i MONTY PYTHON e per la loro serie televisiva Monty Python's Flying Circus (commedia televisiva britannica di enorme successo dei primi anni settanta)

Obiettivi di Python

- linguaggio semplice, intuitivo e potente quanto i suoi maggiori avversari
- open source, in modo che ognuno avrebbe potuto partecipare al suo sviluppo
- un codice facilmente comprensibile, come l'inglese parlato
- ottimo per i compiti di tutti i giorni, poiché in grado di consentire tempi di sviluppo brevi

Noi e Python

- Adatto alla programmazione scientifica
- Linguaggio semplice (programmi leggibili e brevi)
- Buon primo linguaggio di programmazione: poche tecnicalità
- Paradigma OOP, come i linguaggi più avanzati. Stile di programmazione spendibile nel mondo del lavoro

Programmazione (scientifica)

Iniziamo..

PARTE 1

Definizioni Fondamentali

Algoritmo

DEFINIZIONE

- Insieme completo delle regole che permettono la soluzione di un determinato problema

DEFINIZIONE OPERATIVA

- Procedura effettiva che indica le istruzioni (passi) da eseguire per ottenere i risultati voluti a partire dai dati di cui si dispone

Algoritmo: Esempi

Esempio Culinario:

- Ricetta x cucinare gli spaghetti

Esempio Turistico:

- Indicazioni per raggiungere un albergo

Esempi sui numeri:

- Insieme di passi per verificare se un numero è dispari, pari, primo, ecc.

Altri esempi (piu' difficili) : MCD(a,b), mcm(a,b)

- Per esempio l'algoritmo di Euclide per il calcolo del MCD (che può essere usato anche per il calcolo del mcm!) -> mcm(a,b)=(ab)/MCD(a,b)

Algoritmo MCD

1) $a=|a|$, $b=|b|$
2) ordina a e b in modo tale che $a > b$
3) *Finche' b* rimane diverso da 0
   1) $t=b$
   2) $b=a \bmod b$
   3) $a=t$
4) $MCD(a,b)=a$

Algoritmo: Caratteristiche

- Esprimibile con un numero finito di istruzioni
- Istruzioni eseguibili da un elaboratore
- Insieme di istruzioni di cardinalità finita
- Tempo di esecuzione di ogni istruzione finito
- Elaboratore ha una memoria
- Calcolo per passi discreti
- Non esiste limite alla lunghezza dei dati di ingresso
- Non c'e' un limite alla memoria disponibile
- Numero di passi esecuzione eventualmente illimitato

(vedere Macchina di Turing it.wikipedia.org/wiki/Macchina_di_Turing)

Macchina di Turing

- La macchina può agire sopra un nastro che si presenta come una sequenza di caselle nelle quali possono essere registrati simboli di un ben determinato alfabeto finito; essa è dotata di una testina di lettura e scrittura (I/O) con cui è in grado di effettuare operazioni di lettura e scrittura su una casella del nastro

- Dati iniziali sul nastro
- La macchina ha uno stato interno
- Ad ogni passo, dipendentemente dallo stato in cui si trova e dal carattere letto, la macchina
  - Modifica il contenuto della casella
  - Si sposta a destra o sinistra di una posizione
  - Cambia il suo stato interno
- Si dimostra che essa è equivalente, ossia in grado di effettuare le stesse elaborazioni, a tutti gli altri modelli di calcolo piu' complessi!!!

Linguaggi

LINGUAGGI NON FORMALI (ambigui)

- Linguaggio Naturale
- Linguaggio della musica e della pittura
- Linguaggio del corpo
- Ecc.

LINGUAGGI FORMALI (di programmazione)

- Linguaggi NON ambigui, regolati da regole grammaticali precise
- Possono essere classificati in base al loro livello di astrazione

Linguaggi di Alto Livello (LAL)

ESEMPI FAMOSI

- Imperativi: PASCAL, FORTRAN,COBOL, C
- Ad oggetti: CPP, JAVA,Python

Appositi software (compilatori) si occupano di tradurre le istruzioni scritte in questi linguaggi (cosiddetto codice sorgente, un file di testo), nell'equivalente codice direttamente eseguibile dalla macchina (cosiddetto codice eseguibile, binario)

Python e Java sono in realtà interpretati (JVM,PVM)! Compilano in ByteCode (indipendente dalla macchina)

Caratteristiche LAL strutturati

SEQUENZA

- Le istruzioni vengono eseguite in sequenza nell'ordine in cui compaiono in un *blocco* di istruzioni

SELEZIONE

- Strutture di controllo decisionali:
  - P.e. Se <espressione> esegui <BloccoIstruzioniV>
    altrimenti esegui <BloccoIstruzioniF>

ITERAZIONE

- Strutture di controllo iterative:
  - P.e. Fintanto che <espressione> esegui <BloccoIstruzioni>
  - Oppure, Esegui <BloccoIstruzioni> fintanto che <espressione>

Complessita' degli Algoritmi

- Complessita' Polinomial (P) : Il numero di passi e' proporzionale in modo polinomiale alla cardinalita' dell'input

- Complessita' Non-Deterministic Polinomial (NP) : Sono noti algoritmi che terminano in un numero di passi polinomiale usando un numero indeterminato di macchine in parallelo, oppure utilizzando l'algoritmo di Gastone

Complessita' degli Algoritmi (2)

Casi:

- Ottimo: I dati presentati sono i migliori possibili per l'algoritmo
- Pessimo: I dati presentati sono i piu' sfavorevoli per l'algoritmo
- Medio: Comportamento in media al variare dei dati possibili in ingresso

Notazione:

- $O(f(n))$: valutazione caso pessimo
  - La quantita' di risorse richiesta cresce NON PIU' di f(n)
- $\Omega(g(n))$: valutazione caso ottimo
  - La quantita' di risorse richiesta cresce NON MENO di g(n)
- $\Theta(h(n))$: casi ottimo e pessimo hanno simili prestazioni
  - La quantita' di risorse richiesta cresce COME h(n)

Algoritmi di Ricerca

- Ricerca MIN e MAX
- Ricerca di un valore in una collezione
- Ricerca di un valore in una collezione ordinata
- Ricerca degli zeri di una funzione

Algoritmi di Ordinamento

- Ordinamento degli elementi in una generica collezione
- Fondere due collezioni ognuna di esse gia' ordinata
- Ordinare senza usare confronti

Algoritmi di Ottimizzazione

- Problema del Commesso Viaggiatore (TSP)
- Problema dei Cammini Minimi (SP)
- Problemi IA