

Laboratorio Algoritmi e Strutture Dati

2024/25

Laboratorio di Algoritmi

Scopo del laboratorio

- Nel corso di Algoritmi e Strutture Dati imparo soprattutto a:
 - ▶ Capire come funziona un algoritmo
 - ▶ Verificarne la correttezza
 - ▶ Calcolarne la complessità (soprattutto computazionale)
- Nel Laboratorio di Algoritmi imparo:
 - ▶ Ad implementare algoritmi in un linguaggio orientato alla prototipazione: Python
 - ▶ A modificare i programmi/algoritmi
 - ▶ A testare sperimentalmente il funzionamento dei programmi provando con vari tipi di input di dimensione diversa
 - ▶ A descrivere, con una relazione, gli esiti della sperimentazione

3 CFU ...

Tipo di lavoro

- *"Il Laboratorio di Algoritmi consiste nello svolgimento da parte dello studente di un compito didattico aggiuntivo nell'ambito dell'Insegnamento di Algoritmi e Strutture Dati"*
- Soprattutto lavoro a casa
 - ▶ Il lavoro è **individuale** e gli esercizi sono diversi tra studenti
- Si dovrà (probabilmente) installare del software a casa

Valutazione

- E' un'idoneità (senza voto)
 - ▶ Gli esercizi a casa vanno consegnati tutti entro la scadenza indicata
 - ★ Consegna di codice e relazione su moodle
 - ★ Caricamento del codice su apposite piattaforme
 - ▶ Non saranno accettate consegne in ritardo
 - ▶ Il giorno dell'orale vengono poste domande sugli esercizi svolti
 - ▶ Deve essere lavoro **individuale** quindi ci si aspetta che il codice sia diverso, ma **sicuramente** lo sarà la relazione e ancora di più i risultati sperimentali!!
 - ▶ In caso di "duplicazioni" (anche se individuate all'orale) sarà necessario svolgere un altro progetto
 - ▶ Analogamente se non si è in grado di raccontare/motivare cosa fatto

T_EX e L^AT_EX

L^AT_EX

- L^AT_EX *non* è un programma WYSIWYG (*what you see is what you get*)
- Non possiede un'interfaccia grafica per visualizzare in *tempo reale* il documento
- L^AT_EX è un linguaggio di markup utilizzato per generare testi
- La formattazione di equazioni matematiche è considerata migliore di quella ottenuta da altri editor di testo
- Si **compila** un file di testo con il sorgente (.tex) e si genera l'output (per esempio PDF)

T_EX

- T_EX è il “motore” di L^AT_EX
- Il nome deriva dalle prime tre lettere della parola "Tecnologia" in greco ($\tau\epsilon\chi$)
- Si pronuncia "tec"
- La storia di T_EX è lunga e complicata...

T_EX e L^AT_EX

- T_EX si occupa della formattazione dei documenti e interessa soprattutto chi progetta i template dei documenti
- L^AT_EX si occupa del contenuto, quindi è per chi scrive i documenti
- Come informatici ci potrebbero interessare entrambi gli aspetti ...
- ... ma ci focalizziamo sul secondo
- L^AT_EX è un insieme di macro costruite sopra T_EX che consentono di indicare i capitoli, i paragrafi le tabelle o le figure
- In L^AT_EX scrivo `\section{...}` in T_EX indicherei il tipo di carattere, l'altezza ecc.

Hello world!

```
\documentclass[] {article}

\begin{document}

Hello World!

\end{document}
```

Vari Hello world

<http://helloworldcollection.de/>

Risorse

Editor Online

- **Overleaf** <https://www.overleaf.com>

Editor Offline

- **TeXstudio**
- **TexMaker** <http://www.xmlmath.net/texmaker/>
- **Kile**
- **Gummi (linux)**

Documentazione

- Documenti su moodle

Python 1/3

Linguaggio Python

- Interpretato
- Interattivo
- Ad oggetti
- Incorpora
 - ▶ moduli
 - ▶ eccezioni
 - ▶ tipizzazione dinamica
 - ▶ tipi di dati dinamici di alto livello
 - ▶ classi
- Molto potente, sintassi chiara
- Programmi in C di 100 linee possono essere ridotti a 20 linee in Python
- Portabile

Per cosa è utile

- Python è un linguaggio di programmazione general-purpose con un'ampia *standard library* per:
 - ▶ Elaborazione stringhe (espressioni regolari, Unicode, differenze tra file)
 - ▶ Protocolli Internet (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, programmazione CGI)
 - ▶ Ingegneria del software (unit testing, logging, profiling)
 - ▶ Interfacce per sistemi operativi (system calls, filesystems, TCP/IP sockets)
- E soprattutto molte altre estensioni

<https://docs.python.org/3/faq/general.html>

Libri?!?



- <https://docs.python.org/3.7/tutorial/>
- 2.* vs 3.*

Alternative per sviluppo SW

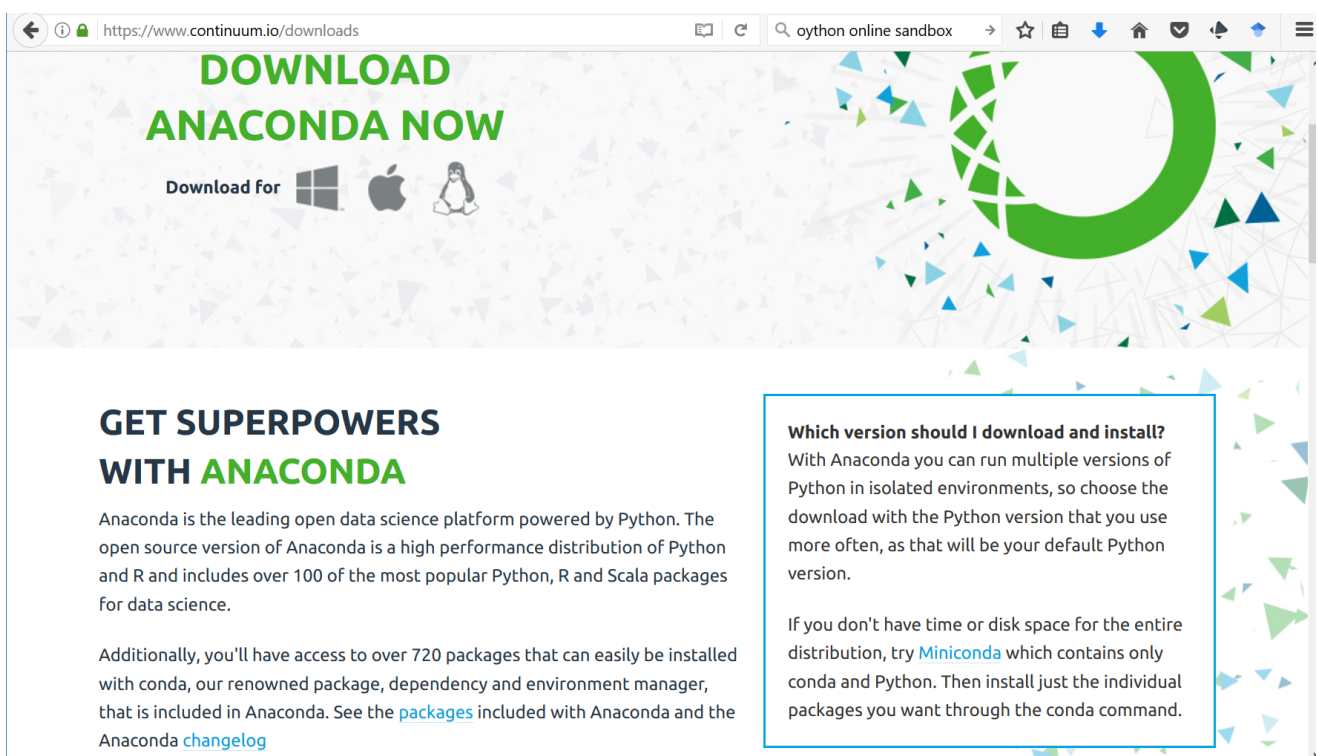
- Installare Python (<https://www.python.org/>)
 - ▶ Usare la shell / eseguire un programma .py
- Usare Python in una Console online (<https://www.python.org/>)
- Installare IPython (Jupyter - <http://jupyter.org/>)
 - ▶ Shell interattiva con completamento con tab, history...
- Jupyter Notebook: Web-based interactive computational environment (<https://jupyter.org/try>)
- **Colab Notebook** (Google: <https://colab.research.google.com/notebook>)
- Installare una distribuzione
Esempio **Anaconda** (distribuzione di Python con più di 100 package, NumPy, Pandas, SciPy, Matplotlib, Jupyter ...)

Anaconda

- Distribuzione Python
- Permette di installare semplicemente pacchetti (es NumPy and SciPy)
- Include:
 - ▶ **conda**, open source package and environment management system
 - ▶ **Spyder** Scientific Python Development Environment un semplice IDE
 - ▶ **IPython**
 - ▶ **Jupyter** (eseguire `jupyter notebook` da terminale)




Install Anaconda

<https://www.anaconda.com/distribution/>



The screenshot shows the Anaconda download page. At the top, it says "DOWNLOAD ANACONDA NOW" in green. Below that, it says "Download for" followed by icons for Windows, macOS, and Linux. The main heading is "GET SUPERPOWERS WITH ANACONDA". The text describes Anaconda as a leading open data science platform powered by Python, which includes over 100 of the most popular Python, R, and Scala packages for data science. It also mentions that users will have access to over 720 packages that can be installed with conda. A callout box on the right asks "Which version should I download and install?" and provides guidance on choosing the Python version to download, suggesting to choose the version used most often as the default. It also mentions the option to try Miniconda, which contains only conda and Python, and then install individual packages through the conda command.

DOWNLOAD ANACONDA NOW

Download for   

GET SUPERPOWERS WITH ANACONDA

Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R and Scala packages for data science.

Additionally, you'll have access to over 720 packages that can easily be installed with conda, our renowned package, dependency and environment manager, that is included in Anaconda. See the [packages](#) included with Anaconda and the Anaconda [changelog](#)

Which version should I download and install?
With Anaconda you can run multiple versions of Python in isolated environments, so choose the download with the Python version that you use more often, as that will be your default Python version.

If you don't have time or disk space for the entire distribution, try [Miniconda](#) which contains only conda and Python. Then install just the individual packages you want through the conda command.

The Shell

- Si invoca python dalla linea di comando
- Utile per matematica di base, per provare idee
- Non si scrivono programmi completi nell'interprete
- Non si può salvare ciò che si scrive

Interprete

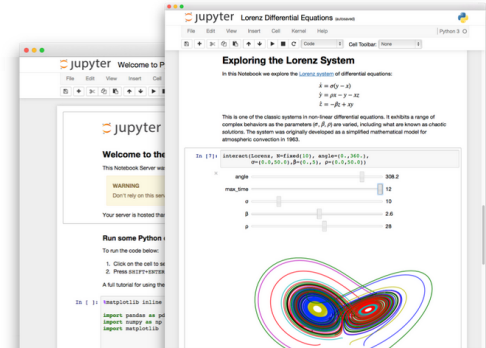
```
>>>print("Hello, World!")
Hello, World!
>>>var = 9+2
>>>var*11
121
```

Jupyter(Lab) Notebook

- Ambiente di calcolo interattivo Web-based usato per creare notebook IPython
- Ipython: Command shell per elaborazione interattiva con introspezione (?variabile), completamento con tab, history, ...
- Un notebook IPython è un documento JSON contenente una lista ordinata di celle di input/output che possono contenere codice, testo, matematica, plot e rich media
- Notebook IPython possono essere convertiti in vari formati open standard (HTML, LaTeX, PDF, Markdown, Python)
 - ▶ "Download As" nell'interfaccia web
 - ▶ nbconvert in a shell

```
jupyter nbconvert Test.ipynb -to latex
```
- <https://jupyter.org>

Jupyter

[Install](#)[About](#)[Resources](#)[Documentation](#)[NBViewer](#)[Widgets](#)[Blog](#)[Donate](#)

The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.



Language of choice



Share notebooks



Interactive widgets



Big data integration

Jupyter



Hosted by Rackspace

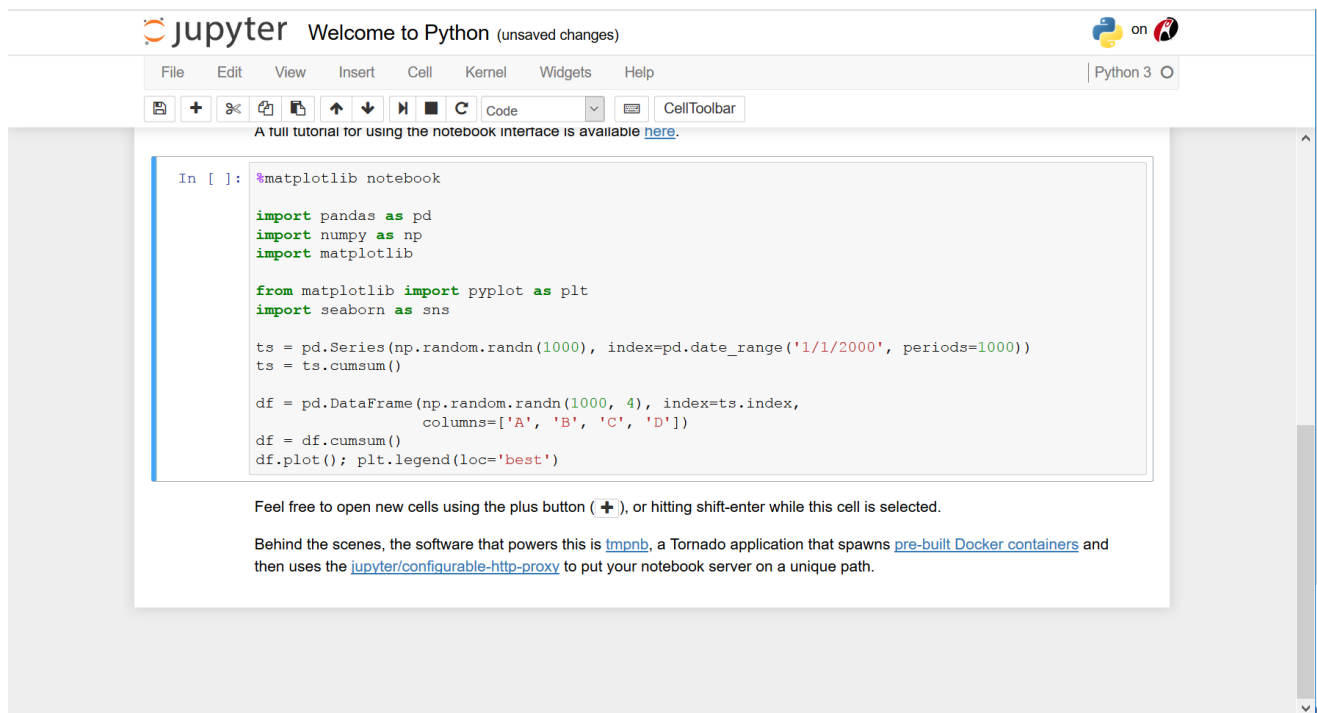
[Files](#) [Running](#) [Clusters](#)

Select items to perform actions on them.

[Upload](#) [New](#)

<input type="checkbox"/>	
<input type="checkbox"/>	communities
<input type="checkbox"/>	datasets
<input type="checkbox"/>	featured
<input type="checkbox"/>	Welcome Julia - Intro to Gadfly.ipynb
<input type="checkbox"/>	Welcome R - demo.ipynb
<input type="checkbox"/>	Welcome to Haskell.ipynb
<input type="checkbox"/>	Welcome to Python.ipynb
<input type="checkbox"/>	Welcome to Spark with Python.ipynb
<input type="checkbox"/>	Welcome to Spark with Scala.ipynb

Jupyter



The screenshot shows the Jupyter Notebook interface with the title "Welcome to Python (unsaved changes)". The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The right side shows "Python 3" and a refresh icon. Below the menu is a toolbar with icons for saving, adding cells, undo, redo, and other actions. A code cell is selected, containing the following Python code:

```
In [ ]: %matplotlib notebook

import pandas as pd
import numpy as np
import matplotlib

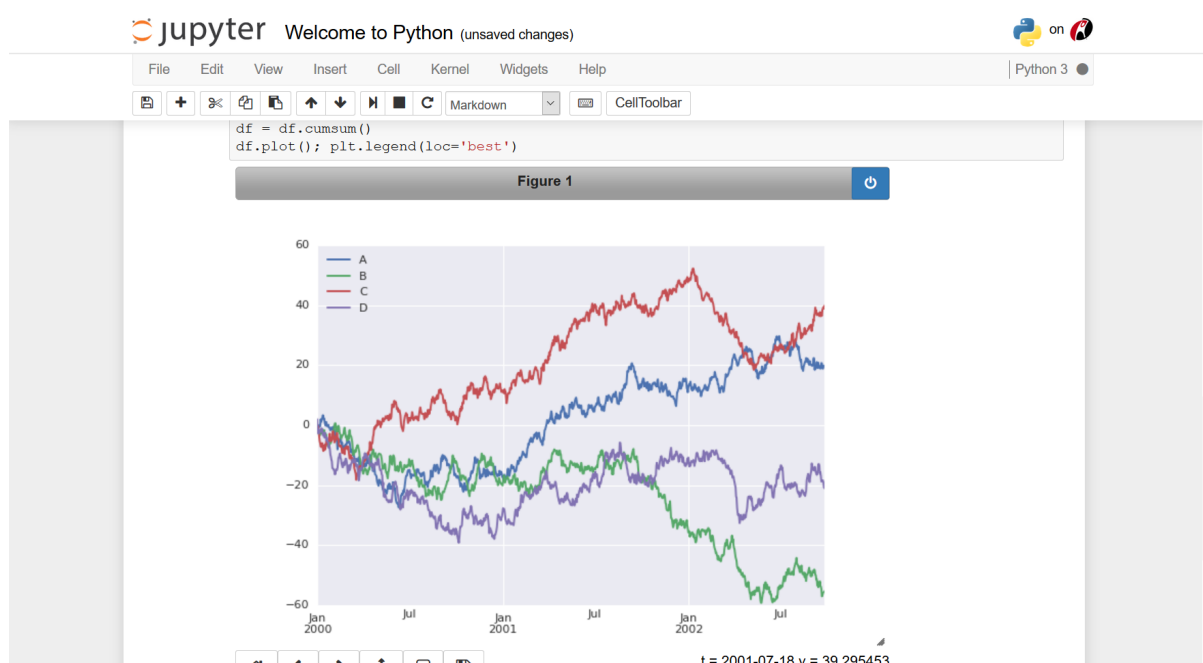
from matplotlib import pyplot as plt
import seaborn as sns

ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000', periods=1000))
ts = ts.cumsum()

df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,
                  columns=['A', 'B', 'C', 'D'])
df = df.cumsum()
df.plot(); plt.legend(loc='best')
```

Below the code cell, there is a message: "Feel free to open new cells using the plus button (+), or hitting shift-enter while this cell is selected." followed by a paragraph: "Behind the scenes, the software that powers this is [tornado](#), a Tornado application that spawns [pre-built Docker containers](#) and then uses the [jupyter/configurable-http-proxy](#) to put your notebook server on a unique path."

Jupyter



The screenshot shows the Jupyter Notebook interface with the title "Welcome to Python (unsaved changes)". The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The right side shows "Python 3" and a refresh icon. Below the menu is a toolbar with icons for saving, adding cells, undo, redo, and other actions. A code cell is selected, containing the following Python code:

```
df = df.cumsum()
df.plot(); plt.legend(loc='best')
```

Below the code cell, a line plot is displayed. The plot is titled "Figure 1" and shows four time series (A, B, C, D) plotted against time. The x-axis is labeled with "Jan 2000", "Jul", "Jan 2001", "Jul", "Jan 2002", and "Jul". The y-axis ranges from -60 to 60. The legend indicates that A is blue, B is green, C is red, and D is purple. The plot shows that series C (red) has the highest values, followed by A (blue), B (green), and D (purple). The plot is titled "Figure 1" and has a power button icon.

Markdown

- Un modo per scrivere contenuto nel Web
- Scritto in "plaintext", caratteri normali con alcuni caratteri speciali
- Usato per commenti in GitHub
- Learning curve poco ripida (si impara in 10 minuti)
- Poche cose, in modo semplice (italic, bold, headers, lists...)
 - ▶ Corsivo: (_). Esempio `_corsivo_`
 - ▶ Neretto (**). Esempio `**neretto**`
 - ▶ Header (#): Esempi (# Header One)... (### Header Three).
 - ▶ Inline link: link text tra [] link tra parentesi ()
Esempio: [Visit GitHub] (www.github.com)
 - ▶ Immagine come link: ![TestoAlternativo](http://xxx.jpg)
 - ▶ Liste: * prima di ogni item
 - ▶ Oppure numeri

Colab

Un benvenuto a Colaboratory

File Modifica Visualizza Inserisci Runtime Strumenti Guida

+ Codice + Testo Copia su Drive

Cos'è Colaboratory?

Colaboratory o, in breve, "Colab" ti permette di scrivere ed eseguire codice Python nel tuo browser con i seguenti vantaggi:

- Nessuna configurazione necessaria
- Accesso gratuito alle GPU
- Condivisione semplificata

Che tu sia **studente**, **data scientist** o **ricercatore AI**, Colab può semplificarti il lavoro. Guarda [questo video introduttivo su Colab](#) per ulteriori informazioni oppure inizia qui sotto.

Introduzione

Il documento che stai leggendo non è una pagina web statica, ma un ambiente interattivo chiamato **blocco note Colab** che ti permette di scrivere ed eseguire codice.

Ad esempio, qui vediamo una **cella di codice** con un breve script Python che calcola un valore, lo archivia in una variabile e stampa i risultati:

```
[1] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day

86400
```

Per eseguire il codice nella cella sopra, selezionala con un clic e poi premi il pulsante Riproduci a sinistra del codice o usa la scorciatoia da tastiera "Comando/Ctrl+Invio". Per modificare il codice, fai clic sulla cella e inizia a modificare.

Le variabili che definisci in una cella possono essere usate in seguito in altre celle:

<https://colab.research.google.com/>

La relazione

Relazione per esercizi

Deve contenere:

- **breve** introduzione che descrive il problema
- una **breve** descrizione delle caratteristiche teoriche degli algoritmi e delle strutture dati utilizzate
- una valutazione a priori delle **prestazioni attese** degli algoritmi analizzati sperimentalmente
- una descrizione degli **esperimenti** che verranno fatti (non un semplice elenco)
- la **documentazione** del codice implementato
- i risultati sperimentali, sia in **tabelle** che con **grafici**
- l'**analisi** completa di tali risultati, effettuata in modo critico

La teoria

- Fa riferimento a quanto studiato nel corso di Algoritmi e Strutture Dati
- Deve essere solo la parte finalizzata all'esperimento
- Non va bene un semplice copia/incolla dagli appunti (libro)
 - ▶ Anzi, forse sarebbe anche troppo...
- Bisogna descrivere gli aspetti più importanti e come questi indichino indirettamente quali test eseguire
- Se serve un teorema, basta mostrarne l'applicazione non serve la dimostrazione

Documentazione del codice

La documentazione deve includere:

- uno schema del contenuto e delle interazioni fra i moduli
- uno schema delle classi
- un'analisi delle scelte implementative effettuate
 - ▶ se erano possibili alternative, indicare perché è stata fatta una certa scelta
- una descrizione dei metodi implementati, indicando in particolare l'input/output e la funzione svolta

Descrizione degli esperimenti condotti

Bisogna descrivere:

- i dati utilizzati
 - ▶ Se sono stati generati automaticamente, come questo avviene
 - ▶ Altrimenti da dove provengono e quali sono le loro caratteristiche
- Specifiche della piattaforma di test (hardware, sistema operativo);
- Quali misurazioni vengono effettuate
 - ▶ Che tipo di misure
 - ▶ Quante volte si eseguono i vari test
- Come si effettuano le misurazioni (porzioni di codice osservate, numero di run effettuati)

Presentazione risultati sperimentali

Presentati sia in tabelle che con grafici

- Le tabelle devono contenere tutti i dati (al limite in un file allegato)
 - I valori nelle tabelle devono avere un numero di cifre significative appropriato (python può fornire numeri con precisione arbitraria)
 - Un grafico serve per evidenziare l'andamento di un valore, ma non sostituisce la tabella
 - A volte possono essere presentati vari grafici per una tabella per mostrare aspetti diversi
 - Un grafico non chiaro o che non mostri qualcosa di interessante è inutile
 - Non importa la bellezza di un grafico
 - Tutti grafici, le tabelle e le figure devono essere
 - ▶ Descritti da una didascalia (lunga qb...)
 - ▶ Citati nel testo
- `\label{} ... \ref{}`

Analisi dei risultati sperimentali

- Un esperimento **non** è una semplice collezione di dati
- I risultati di ogni esperimento vanno commentati ed analizzati in modo critico, citando i grafici e le tabelle corrispondenti
- Nell'analisi si verifica se le ipotesi teoriche vengono verificate con i dati sperimentali
- Al termine dell'analisi degli esperimenti un paragrafo di conclusioni è spesso utile per sintetizzare i risultati ottenuti

Aspetti da controllare sulle relazioni

- Le relazioni devono seguire il formato indicato nelle slide del corso
- La lingua in cui si scrive la relazione (Italiano o Inglese) deve essere uniforme. Evitare didascalie, grafici o tabelle in Inglese quando il testo è in Italiano (ovviamente i nomi in Inglese, tipo quicksort, non si traducono)
- I valori numerici sperimentali riportati nelle relazioni devono essere espressi con un numero appropriato di cifre significative (non ha probabilmente senso scrivere "tempo di esecuzione, 0.034769464 s)
- Come norma generale la dimensione del testo in figure e grafici dovrebbe essere analoga a quella del testo nel corpo della relazione. Per figure e tabelle è sempre necessaria una breve didascalia che descriva il contenuto
- Ricordarsi di includere un paragrafo con le Conclusioni in cui si riassumono le analisi effettuate dai programmi e descritte nelle relazioni

Esercizi Laboratorio di Algoritmi

Dettagli nella pagina Moodle di Laboratorio di Algoritmi