Capitolo 1

Python

Funziona doc e help
Pythonic e nonPythonic
Curiosità: zen di python, comando **import** this
Doc string = Commento ⇒ * oppure """
Leggibilità
introspezione: capacità di un linguaggio di distribuire informazioni e metadati rispetto a sé stesso e ai propri costrutti. python è moooolto introspettivo.
Modulo: file python o in una altro linguaggio che posso rendere disponibile a python tramite import

1.1 Generic Utilities

Commento É possibile commentare con i simboli #, una riga sola, oppure con """ più righe di testo e """ alla fine

Comando 'dir' >>> dir (nomelib) : utilizzabile esclusivamente sulle cose già importate. Mi dice quali funzioni sono presenti nella libreria
Inoltre il comando dir, applicato ad una variabile, mi dice quali cose posso effettuare ad una variabile.
Le cose da poter fare si dividono in due quelle con __funct__ (doppio under score nome 'dunder') e quelle senza.

Comando 'help' Mi aiuta per sapere come funziona una particolare funzione singola o di tutte quelle di una libreria. Ciò che leggeremo saràuna versione sintetica di quello segnato nella documentazione sorgente online

locals 'locals' metodo builtin che da accesso a tute le variabili locali disponibili nel prompt interattivo. dizionario il quale collega al nome della variabile una possibile variabile elencate come dunder quelli già presenti, quelli senza sono quelli importati. Se definisco una variabile questa successivamente sarà presente in locals [usare locals().keys()].
funzionando come libreria:
>>> a=3 sto definendo una variabile e appare come valore in locals
>>> locals()['a']
>>> out 3
Globals fa la stessa cosa con le variabili globali

Slice, ':' Slice(start, stop, step) è la funzione : che utilizziamo negli array si usa per dire da dove dobbiamo partire(il meno indica che dobbiamo partire dal fondo).

```
a=(2:) %da 2 in poi
b=(1:100:2)  %1 a 100 a passi di 2
```

Pep 8 ??

__main__

```
if __name__=='__main__'
```

Quando faccio eseguire un file.py da shell o tramite editor, come pyzo, viene inizializzata subito una variabile: '__name__'. il codice tramite questo if quindi capisce se il pile.py è stato eseguito come modulo assestante oppure come modulo importato in un secondo modulo, in tal caso la variabile 'name' assume come valore la stringa corrispondente al nome del file.py

Importantissimo quando usiamo python c'è una differenza fondamentale fra quando eseguiamo un file, o modulo(stessa cosa), e quando invece importiamo un contenuto di un modulo da un altro modulo in particolare quando importiamo un modulo da un altro modulo quello che sta dentro l' if non viene eseguito. = __main__ quando il file viene eseguito da solo o uguale ad altro altrimenti.

1.2 Librerie

Pacchetti aggiuntivi alla standard lib già installata di default create da esterni.
Esistono 3 livelli per le librerie. Noi potremo scrivere un quarto, ovvero programmi solo nostri.
La libreria standard di python è inclusa automaticamente nel programma mentre gli altri pacchetti come matplotlib, numpy, scipy, math, sono tutti indipendenti e da aggiungere
Come faccio a capire cosa trovo nel core nella standard in numpy e così via: sul browser 'python built-in functions' mi dice le funzioni principali già presenti nel codice le altre vanno importate. 'type' per esempio è una funzione built-in.
Non sono state inserite tutte le altre per una questione di memoria, nella standard sono inserite le funzioni essenziali.
Uso gli import per evitare 'name clash' conflitti di nomi quindi importando nuove funzioni ma mettendo il prefisso della libreria si può evitare il clash.

1.3 Installazione librerie aggiuntive

Da linea di comando vado nella cartella Python38[1](o Python che sia), vado nella directory Lib e faccio il comando:

```
pip install NomeNuovaLibreria
```

1.3.1 Math e Import con libreria

Abbiamo usato la libreria math per fare qualche esempio di funzioni in python e dell'import.
>>> import nomelib
Contiene un alto numero di funzioni e costanti matematiche
Per usare una funzione di una libreria, per esempio math, è necessario anteporre il nome della libreria dal quale si prende l funzione:
>>>a=4
>>>math.sqrt(a)
>>>out 2
oppure posso utilizzare direttamente la funzione facendo
>>> from math import sqrt
e usare direttamente sqrt
>>> sqrt(a)
>>> out 2
É possibile importare tutte le funzioni con **import ***, '>>>from math import *'. Questa cosa è pericolosa perché si può non controllare più quello che si sta facendo potremmo non sapere più da quale libreria prendiamo le funzioni.
Importando possiamo ridefinire quello che stiamo importando **import numpy as np** a volte può non essere necessario o sconveniente. 'as' sta per alias
Cosa succede esattamente quando faccio **import**? Meccanismo interno di import: fisicamente quando importo numpy creo un collegamento un path al luogo dove è situata la libreria quando viene installata

Python path

Dove vivono sul disco numpy? Come faccio a saperlo?: nomelib.__file__ e produce in out il path.
Ma se volessi importare librerie non dalla directory di sistema devo usare il **pythonpath**. Servirà **__init.py__** Quando dovremo creare i nostri pacchettini dovremo disporre di un pythonpath per poterlo cercare durante un import.
Generalmente se l'installazione è giusta il path potrebbe già essere quello corretto

Per cercare il path di un file o libreria: numpy.__file__ (due underscore per lato)
__*init*__.py

1.3.2 Argparse, Opzioni Aggiuntive

The argparse module makes it easy to write user-friendly command-line interfaces. The program defines what arguments it requires, and argparse will figure out how to parse those out of sys.argv. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid arguments.
To parse, parser = analizzare, analizzatore
Permette di parlare con l'utilizzatore del programma di impostare l'help, di comunicargli determinate cose o errori o come compilare il prompt per far partire il codice:
Per utilizzabilità e comprensibilità del codice.
Si utilizza per dare alcune opzioni possibili al momento del lancio del programma tramite prompt dei comandi, **Opzioni di comando**.
Sviluppando software riutilizzabili bisogna fare in modo che se si dovesse modificare qualcosa non si debba accedere necessariamente al codice sorgente ma farlo tramite opzioni aggiuntive.

[1]Al momento era questo nome per la versione che avevo installato, il sistema non penso cambierà

Dovremo documentare il codice e documentare anche le opzioni senza dover guardare necessariamente il codice.

Argparse si utilizza principalmente in 3 step:

1. creare un "parser", " ArgumentParser object e contiene tutte le informazioni necessarie per leggere la riga di comando
2. Aggiungere Argomenti, riempiono il parser con le giuste informazioni da prendere dal terminale a momento del lancio. Si usa add_argument()
3. Analizzare gli argomenti si effettua semplicemente chiamando la funzione parse_args()

Le versione di python inferiori a 3.9 non supportano l'argomento exit_on_error

1.3.3 Logging

come debuggare?
debugger GDB.
Cioè come scovare gli errori. come quando si inserivano i print nel mezzo.
il print non è la via per andare in paradiso non è molto flessibile. sarebbe meglio usare il modulo logging. Questo fa delle stampe sul terminale ma è più strutturato potendo assegnare un livello di pericolosità(warning, Error, info, debug), per esempio:
>>>logging.error('aiuto!')
>>>aiuto!
>>>logging.warning('aiuto...')
>>>aiuto...
>>>logging.info('aiuto')
>>>
>>>logging.debug('aiuto')
>>>
gli utlimi due non printano niente.
Esistono livelli di criticità e per delle informazioni e dei debug non è necessario printare nulla perché non ci sono problemi grossi, e questo aiuta perché non rovina il codice come i tanti print. Questi messaggi appaiono runnando il codice
Logging mi permette di debuggare il codice senza dover modificare il codice sorgente, cosa che non va mai fatta
esiste una funzione che seleziona il livello di criticità che vogliamo far printare. Il codice pieno di logging commentando cosa sta avvenendo è una cosa ben vista.
c'è la possibilità di runnare in modalità debug **in argparse c'è una modalità di loglevel che setta logging**

Altre librerie

- Numpy Useremo principalmente numpy perché ci mette a disposizione la vettorizzazione, si può lavorare con array.
- Gestione del tempo Modulo time in stdlib, oppure timeit, di python misura il tempo di esecuzione di un pezzetto di codice usiamolo per confrontare due codici che fanno la stessa cosa. Ricordiamoci che il tempo è relativo alla propria macchina
- Random Esempio: random.gauss mi genera numeri casuali secondo una distribuzione casuale dobbiamo dare μ media e σ devstd, generalmente ci è sufficiente numpy.
  python uses Mersenne twister as the core generator(nella documentazione python)
- os.path sono utili le funzioni os:

  ```
  >>> import os\\
  >>> os.path.exist(C:)\\
  >>> True\\
  ```

  utili per cercare, collegare percorsi.
- Glob lista il contenuto di una cartella crea sposta file e cartelle
  verificare global
- Seaborn, plot. Libreria per visualizzare i plot.

- Sorting, ordinare qualcosa. pagina 14 sono decritti alcuni tipi di sort, bisogna verificare l'ordine medio. python usa il timsort, deriva dal nome tim peterson, linkata. Si usa semplicemente sort

```
>>> l=[2,5,8,3,6,1,4,9,7]
>>> l[2]=10
>>> l
[2, 5, 10, 3, 6, 1, 4, 9, 7]
>>> l.sort()
>>> l
[1, 2, 3, 4, 5, 6, 7, 9, 10]
>>>
```

  Guardare il quicksort con i balli ungheresi su you tube

- re , Matching e corrispondenze.Si agisce in vari step il primo trmaite re.compile("quello che voglio matchare") queto mi definisce una "stringa" che devo trovare nel testo. Per i caratteri devo anteporre il
  . Se volessi trovare i singoli caratteri e non dei comandi, come per esempio n(a capo) ma singolarmente il backslash e la n devo anteporre la lettera rprima delle virgolette (r" "). l'asterisco invece se volessi cercare più volte

pylint !!!, voto al proprio script

Una volta installato è sufficiente da linea di comando eseguire

```
pylint nomefile.py
```

Analizzatore statico di sintassi.
docstring = commento a qualcosa

1.4 tipi di dato

Non è necessario indicare il tipo di dato, inoltre python provvede a modificare il tipo automaticamente e necessario

1.4.1 Lista e tupla

la lista è mutabile è possibile appendere dati e modificarli.

$$l = [1, 2, 3]$$

la tupla invece non è modificabile

>>>t=(1, 2, 3)
>>>t[1] -> 1
>>>t[1] = 100 -> ERROR

Se qualcosa non deve cambiare posso usare la tupla così mo evita possibili cambiamenti da non effettuare. Quello che penso è dunque: "A quindi sto cercando di cambiare qualcosa che non deve essere modificato"

1.4.2 Dizionario

Struttura dati che immagazzina oggetti secondo una chiave come una tabella hash
Utilizza le funzioni hash per raggruppare dati.
Tipo la rubrica dei numeri con nomi in ordine alfabetico e numero.
Per recuperare cose in funzione di una chiave, tipo le tabelle hash, io utilizzo una funzione di hash applico la chiave alla funzione e mi riporta il numero associato ed è molto più veloce ritrovarlo.
Per alcuni chiarimenti sulla funzione hash vedi **??**

1.4.3 Stringa

la stringa è immutabile.
sommare due stringhe è un lavoro di ordine n^2, in questa maniera io sto creando tutte le volte una stringa nuova contenente quelle vecchie
Da python 3.6 esiste la "fstring" all'interno delle parentesi graffe *name* è inserito il nome della stringa o variabile in generale che si sostituisce dentro, con stringhe molto lunghe è più intuitivo rispetto a leggere le variabili in fondo alla funzione

1.4.4 Rappresentazione numeri

Challenge of the day
Come rappresentare i numeri virgola mobile con numeri interi(combinazioni di bit)
IEEE 754 standard, slides 14 lect 2

1.5 Funzioni

Dry=Don't Repeat Yourself
Wet= Write Every Time
Cercando funzioni python c'è la spiegazione di tutte le maniere in cui possiamo scrivere le funzioni, come si utilizzano i 4 fondamenti:

1. argomenti banali
2. Argomenti default
3. Star '*'
4. Doppia star '**'

Permettono di isolare porzioni di codice che eseguono una particolare cosa da poter riutilizzare, se la cosa da ripetere è molto lunga posso evitare di riscriverla per intero, eventuali errori, bug, posso correggerli una volta sola.
Prende in ingresso uno o più argomenti, variabili,(anche zero) e mi restituisce uno o più valori
Nel caso di restituzione di più argomenti vengono restituiti come tupla.
É possibile dare dei valori di default assegnando con l' '=' un valore agli argomenti
Per spiegazioni di alcune funzioni presenti nelle librerie e non, si usa la documentazione python online o il comando help

1.5.1 Funzioni variadiche

Funzione che accettano un numero variabile di argomenti.
funzione os.path.join() concatena cose insieme.
Un esempio è proprio la funzione somma

sum([1,2,..,5])

Si utilizza la star * = numero arbitrario di argomenti
Con lo star possiamo passare ad una funzione come curve_tutti gli argomenti calcolati con fit_curve (∗popt) implementazione su registrazione lunedì mattina 29 settmbre
Doppia star **
mi indica che voglio 'Spacchettare un dizionario', cioè mi deve leggere i singoli valori e non prendere tutto il dizionario insieme.

1.6 Costrutti di controllo

Più che altro per rivedere come si scrivono e alcune curiosità.
Advance iteration, ∼h1.m35
>>>l=[1, 2, 3]

>>>n=len(l) >>>**for** i **in** range n
>>> print(l[i])
>>>""" non pythonico
>>>
>>>**for** element **in** l
>>>... print(element)

1.7 data structure

le strutture dati principali in python sono le tuple, le liste e dizionari.

Operation	Average case	Worst case
Copy	O(n)	O(n)
Append	O(1)	O(1)
Insert	O(n)	O(n)
Get Item	O(1)	O(1)
Set Item	O(1)	O(1)
Delete Item	O(n)	O(n)
Iteration	O(n)	O(n)
min(s), max(s)	O(n)	
Get Length	O(1)	O(1)

Figura 1.1: ordine di operazioni per alcune delle più semplici operazioni che di solito si eseguono

Devo sempre capire cosa ci sto facendo io con i dati, quale operazione faccio più spesso questo come si ripercuote nella complessità
Il dizionario è una struttura dati che associa ad ogni chiave un valore, come un elenco telefonico che associo ad un nome e cognome un numero di telefono.
Possono essere implementati tramite alberi binari(per il momento lo lasciamo stare (10-10-2020)) o tramite hash table.

Funzione hash, python

Mappa un oggetto qualsiasi in un numero compreso tra max e min intero In python esiste già la funzione hash. vedere con help(hash).

```
hash(2) %oppure hash(2.0) two object that compare equal must also have same hash value (2 ==
2
hash20
20
hash(2.2)
46.......202
```

i numeri interi vanno nel proprio numero intero, quelli decimali in numeroni
quando si usa un oggetto come chiave di un dizionario l'oggetto viene trasformato nel suo hash e e due oggetti hanno lo stesso hash sono la stessa chiave
Esempio

```
a=dict()     %a è un dizionario
a[2] = 'two'
a
{2: 'two'}
a[2.] = 'twoooo'
a
{2: 'twoooo'}
a[(1, 2)]=3 % si posso usare una tupla come chiave ma non una lista perché essendo mutabile
```

Quando uso un oggetto come chiave di un dizionario l'oggetto è trasformato come chiave.
Quando uso 2 oggetti con la stessa chiave hash come chiave di un dizionario sono la stessa chiave di un

dizionario 'a' non ha 2 chiavi per due valori perché 2 e 2. hanno la stessa chiave hashing. É sconsigliato usare i numeri in virgola mobile come chiavi di un dizionario la lista è unhashable perché se modifico qualcosa della lista poi diventa un casino, solo gli oggetti immutabili si possono usare come chiave, quindi alla lista devo preferire la tupla.
!!!Solo gli oggetti immmutabili possono essere usati come chiave di un dizionario e due oggetti uguali sono la tessa cosa dal punto di vista della chiave di un dizionario.
Tutti i numeri in virgola mobile senza parte decimale sono comunque ben rappresentabili sul pc e quindi assumono la stessa rappresentazione dei numeri interi
In un dizionario devo calcolare la funzione di hash di una chiave e poi prendere direttamente il valore segnato in quel valore del dizionario. Quindi ordine 1. A meno che non succedano le collisioni, cioè due chiavi hanno lo stesso hash value. Con la funzione hash gli ordini diventano:

Operation	Average case	Worst case
Copy	O(n)	O(n)
Get Item	O(1)	O(n)
Set Item	O(1)	O(n)
Delete Item	O(1)	O(n)
Iteration	O(n)	O(n)

1.8 Algoritmi

Cos'è un algoritmo? Sequenza di istruzioni non ambigue che fanno qualcosa di ben definito come fattorizzare un numero primo.

1.8.1 Ricerca binaria

Esempio algoritmo Siamo però in una sequenza già ordinata
Parto dalla metà guardo se è maggiore o minore e seleziono una delle due metà.
i passi da eseguire valgono $\log_2(n)$:

$$2^m = n \Rightarrow m = log_2(n)$$

m = Numero massimo di passi. La differenza tra n e log(n) è grossa per 10^6 sample farò al massimo 19 passi
Il succo è che bisogna sempre cercare la soluzione migliore, *se siamo persone virtuose lo dobbiamo essere sempre, non a tratti si e a tratti no.*

1.8.2 Analisi di un algoritmo, complessità

```
def find_maximum(list_):
    """Find the biggest element in a list.
    """
    maximum = list_[0]
    for value in list_[1:]:
        if value > maximum:
            maximum = value
    return maximum

l = [1, 2, 5, 98, 3, 1672, 6, 34, 651]
print(find_maximum(l))

[Output]
1672
```

Figura 1.2: i ':' si intendono come 'slicing'. leggi fino a.. . vedi python slices.

quello che dobbiamo chiederci per la semplicità e velocità è:

Quante istruzioni fondamentali, n, esegue l'algoritmo?

un assegnamento e cercare un elemento sono rispettivamente una istruzione fondamentale. Nel caso **??** si hanno 3(n-1) massimi. Ma nell' if ci entriamo un numero stocastico di volte non sempre allora il numero di passi può andare da 3n a 4n-1.
Di solito è difficile contare il numero di istruzioni.
Diciamo che l'algoritmo ha complessità n. Quali domande dobbiamo porci?

- numero minimo
- numero massimo
- numero medio

di istruzioni. Il numero esatto è irrisorio può dipendere sia da hardware che software, preoccupiamoci dell'ordine di n, $log(n) < nnlog(n) < n^2$ passi.

big-O notation Abbiamo una lista o una stringa con n elementi, caratteri, quante istruzioni servono per arrivare al fondo per l'algoritmo? diciamo in media. Facciamo attenzione anche ai casi peggiori controlliamo il comportamento asintotico tramite limiti. Quindi si buttano tutti i termini che riescono più lentamente $n^2 > n$ senza usare i coefficienti numerici $\Rightarrow$ l'ordine sarà ordine n^2. Spesso n^2 può essere ridotto a nlog(n).

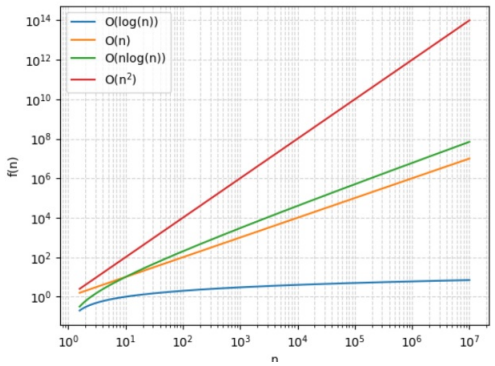

Figura 1.3: Confronto tra difficoltà di algoritmi con ordini di complessità differente

diminuire da n^2a nlog(n) vuol dire aumentare di 6 ordini la velocità, esempio 2 settimane in un giorno. Come misuriamo questo comportamento asintotico:

- brute force, si misura il tempo di esecuzione in diverse condizioni runnandolo e usando timeit
- Per analisi modo difficile contando le esecuzioni valutando il caso peggiore migliore e medio
- Ad occhio, by eye, cioè un loop vale ordine n, due loop sempre ordine n e così via. Ma due loop annidati si ha

In python le funzioni sono programmate sia in c che in python stesso e spesso è difficile specificare la complessità. Nella moltiplicazione di 2 vettori deve esserci un ciclo for dentro non visibile ad occhio.

```
a=numpy.array([1, 2, 3])
a=numpy.array([1, 2, 3])
a*b
array([1, 4, 9])
```

questa moltiplicazione con numpy è un for in c ha a che fare con la vettorizzazione in numpy che vedremo più avanti.

Può essere già una cosa buona considerare i loop del proprio programma poiché non è possibile valutare funzioni prese da altre librerie.