Baldini

Me

23 novembre 2020

Indice

0.1 Keep in mind . . . 4

1 Version-Control Git **7**
1.1 installazione . . . 7
1.1.1 comandi . . . 8

2 Ex Cap Python **9**
2.1 Funzioni . . . 9
2.2 Funzione hash, python . . . 9
2.3 Algoritmi . . . 10
2.3.1 Ricerca binaria . . . 10
2.3.2 Analisi di un algoritmo, complessità . . . 10

3 Lezioni da inserire **13**
3.1 Lezione giovedì 8 ottobre . . . 14
3.1.1 Definizioni . . . 14
3.1.2 Come fare le nostre classi . . . 15
3.2 Ottobre 12 2020,mattina . . . 16
3.2.1 Ihnerithance-Ereditarietà . . . 16
3.2.2 Esempi . . . 17

A Cose da fare a casa, no assignement **19**

B Casualità **21**

C Editor di testo **23**
C.1 komodo ide . . . 23
C.2 Atom . . . 23

0.1 Keep in mind

Parola d'ordine: Riproducibilità, Semplicità o difficoltà lettura algoritmo

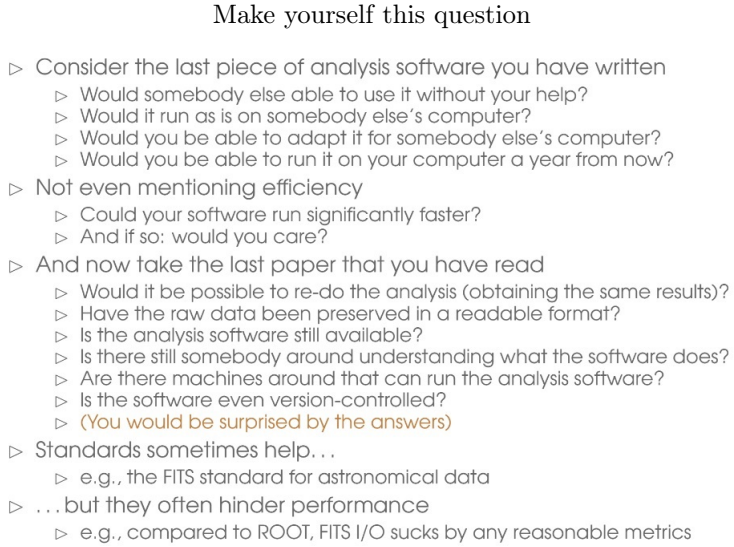

Figura 1: Semplice autocritica al proprio software appena scritto

Workflow

Cosa bisogna fare quando si utilizzano i repositori distribuiti come quelli di git-hub.
Queste sono le regole da seguire per un buon Team's Work:

1. Clone your repository, someone else could change something toghether with you in China
2. Create new branch and check it out, it is more convenient to not work on master branch, if we use another branch of repository we can work safetely with other commit
3. Do all you have to do
4. Add ew file, commit your modified files
5. Create a pull request
6. Have your pull request reviewed
7. Merge the branch on master
8. Tag the master

Questi sono i passaggi che abbiamo seguito durante la lezione del 28 settembre 2020, dove il prof ci ha insegnato a creare il nostro repositorio e esercitato nel workflow

Esercitazione 28 settembre, Pomeriggio, Workflow

Creazione repositorio entrare sul proprio account di github e cliccare sul $'+'$ e "New repository"
Dovremo selezionare le voci che permettono la creazione di 3 file:

- File readme, File di testo che servirà per presentare il proprio progetto e le proprie utilità
- gitignore, serve per non far considerare al repositorio gli aggiornamenti che il programmadi testo(python latex che sia) creano per far leggere al pc i nostri programmi. quindi bisogna selezionare il template, ambiente. Credo se ne possa selezionare uno solo per repositorio
- License, criteri con il quale viene distribuito il nostro programma. Noi usiamo GPLv.3(General public license)

git è un VC distribuito, cioè ho delle copie esatte del repositorio sul cloud sul mio pc e posso averlo su altri pc

Esercitazione guidata Una volta dentro il nostro repositorio online, cliccando su clona ci sono diverse modalità, protocolli col quale comunichiamo col server remoto, noi usiamo HTTPS. Copiamo il link e apriamo la shell di git: 'git bash'
Sulla shell entro nella directory nel quale voglio copiare il repositorio e uso il comando[1]:

```
>>> git clone 'link'
```

Creiamo con un editor di testo, al momento ho usato komodo ide[2] C.1, un file python 'hello world.py'. A questo punto per vedere se è sincronizzato il nostro repositorio si entra nella cartella Lezione_Prova_28/9 e si usa il comando
>>> git status
mi indica lo stato del repositorio locale e se ho file modificati o nuovi, mi indica il master, Questo anche quando sono entrato nel clone del repositorio c'è segnato il branch a lato in blu. adesso devo 'tracciare' il file
Per aggiungere il file si usa
>>>git add nomefile.ext
se la cosa funziona, selezionando nuovamente git status, viene il nome del file in verde ma non è committato. Per questa operazione
>>> git commit -m "initial import." nomefile.ext
!!!**'-m' è importantissimo, va segnato perché avendo selezionato VIM rischio di fare casino**!!!
-m = messaggio che se non si seleziona viene utilizzato l'editor di testo di default e io non lo voglio al momento. il messaggio che voglio è proprio quello che ho modificato nel file committato
É stato necessario inserire la mail di default con
>>> git config –global user.email "andreaforesi1997@gmail.com"
>>> git config –global user.name "Fore11-1997"
il secondo apparentemente non ha funzionato ma quello era comunque il mio nome di github. L'ho cambiato in seconda sede per semplicità.
Dopo il commit si vede il numero identificativo del commit effettuato. Il file non è sul repositorio bisogna effettuare il **push** per sincronizzare il locale con il remoto. I commit infatti si effettuano sempre sul repositorio locale ma è necessario pusharli poi sul cloud semplicemente con
>>> git push
Se modificassi altro è sufficiente fare un nuovo commit ed è sufficiente mettere un messaggio nuovo che si noterà poi sul repositorio.
Per esempio adesso implementiamo una feture al programma hello.py prima si esegue un pull per aggiornarmi
>>> git pull
poi inizializzo il nuovo branch
>>> git branch nomedelbranch
ovviamente devo pushare la modifica se voglio visualizzarla sul cloud.
Se eseguo
>>> git branch
vedo i branch presenti e ne posso selezionare uno con il comando
>>> git checkout nomebranchdestinazione
Ho notato che è possibile cambiare branch anche da dentro komodo, in alto a destra.
Si salvano i nuovi file direttamente nella cartella poi lui capisce su quale branch in base a cosa abbiamo selezionato con il checkout nella shell. Poi bisogna fare nuovamente l'add e il commit, a questo punto lo status mi dice che è tutto ok perché il repositorio sul mio pc è a posto ma non quello nel cloud, manca il branch secondario. Alla fine dovremo rimergiare i branch nel master, eseguendo la **pull request**

[1] A fine linea di comando è possibile scrivere il nome della cartella nel quale vogliamo copiare il nostro repositorio
[2] Avendo difficoltà a comprenderlo ho scaricato atom

Capitolo 1

Version-Control Git

Credenziali
username: AndreaForesi (github forse anche git ma devo controllare)
pass: palindormo
Version control distribuito, cioè esiste un clone del repositorio sia sul web che sui vari computer che collaborano con esso. Ovviamente necessitano di essere aggiornati rispetto al server
Esistevano altre due tipologie di repositori: Local e Centralized
Cosa si intende per repositorio locale?
Programmi che tengono traccia della versione di aggiornamento dei propri file testo(latex, word) di programmazione(python, c, c++, ...) in questa maniera è possibile ritrovare il proprio programmino ad uno stato magari più arretrato però funzionante su un secondo pc per vattelo a pesca te quale motivo.
Permettono di tenere traccia di tutti i commit eseguiti sui file del repositorio aggiornando di conseguenza tutto il magazzino. Per il momento conosco "git" ed è a parere di alcuni qua a Pisa il migliore. Git dispone anche di un repositorio online, un cloud al quale puoi accedere da internet in qualsiasi posto.

1.1 installazione

Dato che vorrei capirci di più su alcune cose del pc e delle macchine in generale volevo seguire passo passo l'installazione del programma. Alcune di queste cose nemmeno il prof ha tenuto di conto lasciando comunque quelle predefinite e consigliate

1. Accettazione dei termini di utilizzo
2. Creazione cartella in c/Program Files/git
3. selezionare i componenti: icona desktop, integrazione windows ex(NO), Git Support, associare i file testo in arrivo e i file .git, no alla ricerca degli aggiornamenti quotidiani.
4. selezione editor di testo. Al momento ho tenuto Vim, impostazione predefinita di git e windows **Assolutamente da cambiare**
5. command line??? raccomandato non so cosa sia
6. server open ssrl library
7. line ending conversion (Bo)
8. terminal emulator
9. default behavior
10. credential helper

tortoise git

interfaccia grafica per git che mi permette di visualizzare graficamente se la copia è in sync con il repositorio locale

git desktop

git bash

shell di git esclusiva di windows. Gli stessi comandi funzionano anche dalla shell win

1.1.1 comandi

I comandi principali sono quelli in figura 1.1 e sono quelli utilizzati poi per i passi del Workflow, vedi 0.1

Comandi Git Bash

```
start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Join two or more development histories together
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects
```

Figura 1.1: Comandi presi dalle note del prof. Per usarli su git bash bisogna anteporre sempre 'git'

Cambio configurazioni

Ho cambiato l'editor di default con: >>>git config.editor "editorname –wait" Ho selezionato atom

Capitolo 2

Ex Cap Python

2.1 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

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.

2.2 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 == 2.0)
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 è unhashable.
```

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:

Nuovi ordini con la hash table

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)

2.3 Algoritmi

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

2.3.1 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.*

2.3.2 Analisi di un algoritmo, complessità

Ricerca di un massimo in python

```python
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 2.1: 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 2.1 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).

Complessità algoritmo

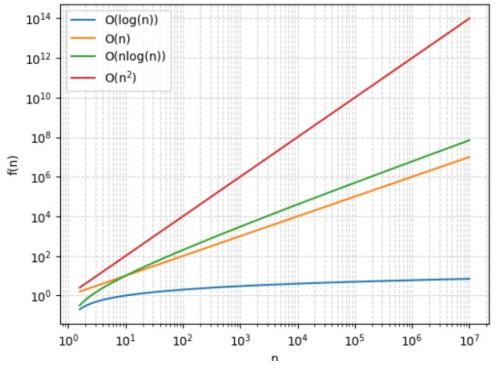

Figura 2.2: Confronto tra difficoltà di algoritmi con ordini di complessità differente

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 ordine n^2

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.

Capitolo 3

Lezioni da inserire

Assegnamento 1

Seguo la lezione scrivendo il codice e commentando direttamente quello. poi aggiungerò determinate cose alle sezioni librerie di python. tenere sempre a mente che la documentazione online è sempre la cosa migliore e più istruttiva si imparare le cose.

```
# !/usr/bin/env python 3 serve su linux per avvertire che si vuole usare python 3 come environment di run

import argparse
import logging
import time

logging.basicConfig(level=logging.DEBUG) #setto il livello di logging a debug


def process(file_path): definizione funzioni
""" readds a text file and compile the letter statistics """
start_time =time.time()
logging.info("Reading input file %s...", file_path) #non serve il percento. è inutile formattare la stringa m
with open(file_path) as input_file:
text = input_file.read()
num_chars=lenght(text)
logging.info("Done, %d characters found", num_chars)
print(text) # tipo stringa quindi lista sono da evitare i print preferire i logging.info
elapsed_time= time.time() - start_time
logging.info("Done in %f second", elapsed_time)


if __name__== "__main__": # metto dentro l'if tutte le cose che vogliio siano eseguite se il file è stato ape

parser = argparse.ArgumentParser #inizializzo un parser, analizzatore
#argomenti di argparse
parser.add_argument('infile', type=str, help_"path to input file") #nome argomento: infile input file

args = parser.parse_args() # analizza

file_path = args.infile
process(file_path)
```

Con l'inserimento dell'argomento input file il nostro programmino.py non può essere runnato perché io tramite argparse ho indicato che è necessario dargli il path del file da far runnare. perciò inserisco file_path. Inseriamo logging
É inutile formattare una stringa che poi non viene stampata su output, non utilizzo del %
IMMAGINE Schema_base.jpg
Tutto ciò che c'è in più serve per eseguire ciò che serve al fine del programma. leggiamo quindi alcune cose nella lettura del testo :

- processare tutto il file in memori oppure no
- with per aprire file. **'context manager**' quella riga è uguale a scrivere:

  ```
  input_file = open(file_path)
  ```

  con il **with** quando esci dal blocco di codice dove utilizzi input file sei sicuro che il file viene chiuso. se qualcosa va storto viene ugualmente chiuso

modulo timeit o time
Possiamo iterare sulle linee al posto di metterla in memoria
Passiamo al goal dell'assignement. Segno qua i pezzetti di codice

```
# creiamo il dizionario vuoto e inseriamo le chiavi dando un valore a tutte le lettere lower case
char_dict={chr(x): 0 for x in range(ord('a'),ord(z)+1)} #nonnostante sia spettacolare non è pythonico

#serve import string
char_dict = ch: 0for ch in string.ascii_lowercase

#riempiamo il dizionario
for ch in text:
try
char_dict[ch.lower()] += 1 #bisogna
except keyError:
pass
OPPURE
ch=ch.lower
if ch in char_dict:
char_dict[ch.lower()] += 1

elapsed_time=time.time()- start_time
logging.info("Done in %.3f seconds", elapsed_time)
num_letters = sum(char_dict.values())
for ch, num in char_dict-items():
print(f"{ch} -> {num/num_letters:.3%}") #il formato % trasforma direttamente in percentuale il numero da usar
print(char_dict)
{}
```

IDEE costruisco un dizionario e per ogni carattere faccio salire di uno un counter, il dizionario perché posso impostarlo. potrei fare anche una lista contenete solo contatori e poi solo in fase di stampa attribuire la lettera. il dizionario è più ovvio perché è autoesplicativo. proviamo con il dizionario. **Non fate economia dei nomi e delle lettere devono essere chiari ed espressivi**.
Noi vogliamo looppare su tutti i caratteri nel testo, se la version minuscola del carattere. lower considera solo le minuscole. aumenta di uno altrimenti niente. potevamo mettere if char.lower in chardict allor char dicht + =1 dict comprehension utilizzo di chr e ord
chr e ord due funzioni. ord è il numero d'ordine di un carattre riporta il numero del carattere in codice ascii

```
ord('a')
97
ord('b')
98
chr(97)
a
```

guarda ascii in python
Possiamo suddividere in capitoli e leggere i vari capitoli, le righe, solo la parte del testo fare un istogramma con matplotlib. Tutto quello che può venire in mente
Ho messo tra gli argomenti del parser la possibilità di non printare gli info del programma.

try-except prova ad aggiungere 1 alla chiave ch.lower se mi lascia un eccezione keyError allora passa avanti

Matches Adesso legge il file e mi restituisce tutte le lettere con 0 giustamente.
Userò il modulo re

3.1 Lezione giovedì 8 ottobre

3.1.1 Definizioni

Programmazione di oggetti L' oggetto è un'entità di codice, come una lista, che ha uno stato rappresentato dai suoi dati ed un comportamento implementato tramite funzioni che può essere attivo o passivo.
Le variabili sono chiamate attributi. Le funzioni sono chiamate metodi o membri dell'oggetto
Alla base c'è l'idea che le variabili, i dati e il codice che manipola questi devono stare insieme non solo nel file ma dal punto di vista logico.

Il codice che manipola i dati e i dati devono stare insieme non solo nel file ma anche nel codice. Per esempio in C troviamo solo dati e se accediamo in un array di 4 elementi al quinto non solo riceviamo un segmentation fault all'inizio ma bisogna scoprire dove.

La lista è una classe e una lista è un oggetto simile alla relazione tipo e variabile. In python non esiste questa differenza, tutti i tipi sono classi tutte le variabili sono oggetti.

Esempio: Televisione Uno stato on/off, il canale, volume luminosità.
ha un comportamento dato il telecomando che cambia volume o canali.
come potremmo salvarlo in codice:
Per uno stato si usano le variabili; con una variabile boleana per on off, alcuni interi o decimali per il canale o il volume.
Il comportamento si implementa tramite funzioni, metodi se vivono in una classe, metodo turn off, turn on, per cambiare canale e cose varie che fai con la tv.

Attributi e metodi sono chiamati membri di una classe.
Ogni oggetto, come una lista, diventano un'istanza d quella lista.

Come fare una classe su python? con def

my_television e your_television sono due istanze della classe Television definita tramite main.television vuol dire che la classe è definita nel main, dove è definito non è così importante. rig 14 è un'istanza e nella 17 l'espresione boleana dimostra che sono due oggetti separati() a utilizato l'espressione is

Aggiungiamo un primo metodo. Turn on alla televisione
per definire un metodo possiamo definirlo nel blocco indentato della classe

```
class Television: #define class

\par def turn_on(elf,channel=1):
```

Il primo argomento di ogni argomento, in questo caso turn_on è per convenzione chiamato self , questo primo argomento contiene l'istanza corrente della classe, contiene l'oggetto sul quale applicheremo il metodo.
Si accede al metodo tramite l'operatore '.' ("dot") e l'operatore self non lo devo passare

Aggiungiamo un attributo
assegniamo una variabile specificando che questa vive nel nostro oggetto alla riga 6 stiamo assegnando a self la variabile current_channel=1. si può fare anche all'esterno riga 9. accedo agli attributi sempre con il dot $'\cdot'$.

Aggiungere attributi in una di queste due maniere è **ASSURDO**, non ti da maniera di tenere traccia degli attributi aggiunti se non leggendo il codice sorgente.
Invece con il **costruttore** è possibile, una funzione automaticamente chiamata quando creiamo un'istanza dell'oggetto.
Quando creiamo un'istanza della televisione si autodefinisce il metodo con un 'dunder init'.
__init__ è un costruttore, una specie di inizializzatore, e appunto inizializza un oggetto e aggiunge gli attributi. questo init viene chiamato sempre quando definisco una nuova televisione e posso quindi assegnare i valori agli attributi

Namespace Dizionario di nomi al quale è associato un oggetto che può essere qualsiasi cosa, funzione, variabile o classe.

Questo serve per differenziare attributi della classe e attributi dell'istanza
Il namespace della classe, di livello più alto è visibile da tutte le istanze della classe.

Non si definiscono gli attributi nella definizione di classe perché sarebbero condivise tra tutte le televisioni, per esempio l'attributo può essere chiamato senza definire un'istanza della classe e quando definisco another_tv quest'ultima contiene già l'attributo definito nella class Television.
Se modifico l'argomento della classe allora viene cambiato a tutte le televisioni, come un aggiornamento ai dispositivi rilasciati dai produttori. Mentre e modifico l'argomento di una specifica istanza allora banalmente non la cambierò per another_tv, in pratica io individuo ho fatto modifiche alla mia televisione (tv) senza determinate istruzioni dei distributori, si chiama "shadowing", perché maschera un attributo di una sola istanza. Io sono responsabile delle azioni sulla mia tv e il distributore (definizioni tramite modifiche nella classe è responsabile di modifiche su tutte le television)

3.1.2 Come fare le nostre classi

É sufficiente conoscere come funziona l'interfaccia dei nostri oggetti

Incapsulamento Lo stato di un oggetto deve essere modificato e letto solo tramite un'interfaccia pubblica, e non è necessario sapere com'è fatto il codice sorgente con il quale è stato scritto.
Non mi interessa, almeno a grandi linee come è implementata la classe television o la classe lista ma ci interessa sapere come funziona l'interfaccia.
Inoltre possiamo imporre dei determinati comportamenti, cioè si evita il cambiare canale a televisione spenta.
L'utilizzo dell'oggetto televisione è utile sia per il lettore dvd ma anche per la console dei videogiochi che altre cose, quindi ho parti di codice ben separate.
In altro linguaggi funziona leggermente diverso, in C++ e java potete definire alcuni attributi come privati, cioè nessuno esterno può modificarlo, per conoscere dall'esterno il valore dell'argomento devo crearmi dei getter, get channel_number o setter set "current_channel"

In python non esiste questo concetto di "privato". ma quando un attributo o un metodo ha il nome che inizia con uno o due underscore quello è da considerarsi privato; è un accordo informale, una convenzione.

Da pythonico a molto pythonico In realtà l'interprete python aiuta questa convenzione facendo sì che se un metodo o attributo inizia con due underscore, python dietro le quinte ci aggiunge underscore_nome della classe (_nome__x) quindi rende più difficile chiamarlo per errore, vale solo per due underscore. esempio slide 25. Per chiamare l'owner di una televisione allora devo assicurarmi di scrivere _television__owner.
pagina 27 vecchio sistema senza incapsulamento e senza argomenti privati, e diventa pericoloso. slide 28 rendo privati alcuni argomenti ma necessito dei getter e dei setter. a pagina 29 c'è un metodo nuovo e migliore con l'utilizzo delle property, il vantaggio è che poso accedere ad owner con il semplice dot e rende tutto più leggibile
Quindi io posso aver chiamato owner con una sola underscore e lo rendo privato con una property, mentre se uso il vecchio sistema obbligo l'utilizzatore della tv ad usare una sintassi diversa invece con la property l'utilizzatore non nota differenze.
Python per venire incontro ai pigri ha creato la @property oppure @owner.setter per definire un setter. con questa sintassi posso anche eseguire più funzioni come il print "do you wanna stole my tv?"

Da buoni fisici bisogna pensare anche alle altre persone che useranno il codice, quindi bisogna occuparci per bene dell'interfaccia del nostro codice. Ogni funzione ha la sua interfaccia, ogni classe. L'interfaccia è costituita dagli argomenti non privati senza underscore.

Ereditrietà Don't repeat yourself
Esempi sllide 36
la classe base si chiama anche genitore o superclass mentre quelle che ereditano si chiamano figlie o derivate, seguono le regole di Aristotele $A- > B- > C \Rightarrow A- > C$.
L'elettrone, protone etc sono specializzazioni della classe particle.
Si può fare l'overload di un metodo, cioè al momento della definizione del figlio si può fissare un argomento. cioè per un animale definisco il metodo sound e poi quando definisco la classe dog fisso l'argomento woof.
Si può ereditare da più classi. La televisione può ereditare dalle classi audio device e video device, si viene a presentare un problema se sono definiti argomenti con lo stesso nome. esiste un ordine di priorità e si può visualizzare tramite il comando .mro. Quindi bisogna stare molto attenti a questa ereditarietà multipla.
Non si abusa dell'ereditarietà

3.2 Ottobre 12 2020,mattina

Lecture basic 5_6

3.2.1 Ihnerithance-Ereditarietà

le classi figlie ereditano classi dai genitori, basta non abusarne

Composizione

Tecnica alternativa, o meglio complementare, all'ereditarietà, più sicura, meno problematica. Consiste nell'avere un oggetto di una classe come attributo di un'altra classe, cioè al posto di una variabile ho anche un oggetto. Per esempio la classe automobile ha l'oggetto motore e 4 oggetti ruote. Si chiama **Classe aggregata**.[1]

Differenze La composizione corrisponde a proprietà: la macchina ha il motore. L'ere: modellizza una relazione di uguaglianza "is" la classe derivata dovrebbe essere una specializzazione di quella dal quale lo eredita l'elettrone è una speciale particella.

[1] Slide:42/45 lecture basic 5

Un problema è quello tra rettangolo e quadrato con la classe ChangeHeight() se il quadrato eredita questo modello allora non sarò più un quadrato. Overload vuol dire ridefinire uno stesso metodo nella classe derivata.

3.2.2 Esempi

non saranno esempi di ereditarietà ma con la composizione

Vettore2d [2]Ammette self.x e self.y e li definisce utilizzando float(x) per assicurarci di lavorare con float fallisce se portiamo un stringa o un dizionario
'.format' = è di python3 sintassi delle fstring
Però non è bello scrivere t=v.add(z) oppure t.nicePrint(). ma possiamo scriverli in maniera più chiara usando i dunder (page 4). definisco un metodo tramite i doppi underscore così quando richiamo abs python prima legge quelle che definisco nel mio codice. altri possono essere:
__str__ e __repr__ restituiscono una stringa.

- esiste sempre un repr di default (reTURNprOPERIES) è come un print
- str trasforma la tupla di coordinate in stringa

il comando !r richiama la funzione repr
entrambi sono implementabili con nostro gusto
Con rmul intendo la moltiplicazione a destra e iadd quella interna $x+=y$
Python sa lavorare con le tuple.

Hashable vector per renderlo hashabile bisogna implementare una maniera per verificare che sia uguale per poter definire che quell'oggetto deve avere la stessa chiave oppure no e necessita della funzione di hash
Con le parentesi graffe indichiamo un set, un gruppo { set, contenitore di dati } rispetto agli altri prevede il fatto che gli elementi devono essere tutti diversi, devono avere hash diversi.
usiamo la definizione di vettori non usando stringhe e tuple perché fanno uso di elementi uguale un array no. Typecode mi definisce il "placeholder"

[2]Esiste già numpy per questi oggetti

Appendice A

Cose da fare a casa, no assignement

legger licensa gpl
guardiamo come funziona merenne wister generatore di numeri casuali
Conversioni tra numeri. vedi rappresentazione numeri e tra potenze guardare la visione standard di un repositorio per python nelle slides
Antipattern, pattern in generale sono le cose che si fanno tipicamente, le buone pratiche. gli antipattern sono le cose da non fare, o che generalmente si fanno ma è meglio evitare, linkata nelle slides
def=?
for item in list
.format()??
Namespace
NOMI MAIUSCOLI=COSTANTI
property?

Appendice B

Casualità

GNU=GNU's not unix
unix=
imacs=
leap second: meccanismo per cui una volta ogni t0ot anni si aggiunge o si toglie un secondo dall'orario per calcoli eseguiti sull'orbita terrestre.
Donald Kluft, complexity of song se introduco un ritornello diminuisco la complessità delle canzoni

Appendice C

Editor di testo

Un editor dal nome editore significa proprio dirigere modificare. Esistono editor di grafica, scrittura, modellazione 3d o codice di programmazione.
ciò che in questo momento mi potrebbe essere utile è un editor di testo per poter gestire i file di programmazione, al momento python, e approfittare della connessione diretta con il version control git 1

C.1 komodo ide

Per il momento l'ho disinstallato mi sembrava troppo incasinato. Ho preferito AtomC.2 Programmi Collegati:

- Python 3
- Latex
- !!! Git

Attualmente non so come abbia fatto ma è possibile selezionare il branch direttamente dal programma proprio come si fa eseguendo 'git checkout nomebranch'

ActiveState

azienda distributrice di komodo-ide
É un sito che permette di controllare facilmente le installazioni aggiornamenti di python e di komodo ide
Sul sito sono occasionalmente disponibili gli aggiornamenti da poter apportare al linguaggio python

C.2 Atom

Ultimo installato al posto di komodo.