

MANUALE LUNGO

Versione 0.0.58

By NiktorTheNat

(aggiornato al 16 febbraio 2021)

SOMMARIO

INFORMAZIONI FONDAMENTALI	3
EDITOR DI PROGRAMMAZIONE	4
CREARE UN ESEGUIBILE DEL PROGRAMMA	4
CONTROLLA PROGRAMMA PER RILEVARE ERRORI	4
SCRIVI	6
LE VARIABILI	6
MANIPOLAZIONE STRINGHE	7
LUNGHEZZA	7
CONTA	7
TROVA	8
INIZIA CON	8
FINISCE CON	8
SOSTITUISCI	8
MAIUSCOLO E MINUSCOLO	9
PRIMA MAIUSCOLA E PRIMA MAIUSCOLE	9
UNISCI	9
DIVIDI	9
ELIMINA SPAZI E ELIMINA SPAZI INIZIALI E ELIMINA SPAZI FINALI	10
UNICI	10
COMBINAZIONI UNICHE	10
COMBINAZIONI ESTESE	11
CONTA LETTERE	12
NUMERO SUCCESSIVO	13
LE LISTE	14
I DIZIONARI	16
CICLO	17
DOMANDA	17
CONSIDERA... CON	18
INTERVALLO	18

SE...ALTRIMENTISE...ALTRIMENTI	19
FUNZIONE	20
PROGRAMMAZIONE AD OGGETTI	21
GESTIONE DEGLI ERRORI	23
LEGGERE E SCRIVERE FILE	24
VERO E FALSO	25
FERMA E PASSA	25
FUNZIONI MATEMATICHE	26
MIN E MAX	26
POSITIVO	27
PI GRECO	27
NUMERO DI EULERO/NEPERO	27
GRADI E RADIANTI	27
SENO, COSENO E TANGENTE	27
LOGARITMO	28
RADICE QUADRATA	28
ARROTONDA PER ECCESSO E ARROTONDA PER DIFETTO	28
MEDIA DI NUMERI	28
NUMERO RICORRENTE	28
DEVIATIONE STANDARD	28
CASUALE	29
NUMEROCASUALE	29
SCELTACASUALE	29
MISCHIA	29
GRAFICA A FINESTRE	29
ETICHETTA	33
PULSANTE	33
CASELLADITESTO	34
CASELLAADISCESA	35
CASELLASELEZIONABILE	36
CASELLAASCELTA	37
CASELLADITESTOSCROLLABILE	38
FINESTRE POPUP MESSAGGI	40
CONTATORE	43
BARRA	44
FINESTRAFILE	44
FINESTRAFILES	45

FINESTRACARTELLA	45
GESTIONE MENU A DISCESA	45
INTEGRAZIONE CON MICROSOFT WORD	47
INTEGRAZIONE CON MICROSOFT EXCEL	50
DATABASE	54
APRIAUDIO	55
RICONOSCIMENTO VOCALE	55
SINTETIZZATORE VOCALE	56
REGISTRAZIONE DA MICROFONO	56
SCRAPING (estrapola informazioni da pagina WEB)	56
DIAGRAMMI	63
DIAGRAMMA A LINEE	63
DIAGRAMMA A BARRE	69
BOT TELEGRAM	70
AUTOMAZIONE COMPUTER	79
POSIZIONEMOUSE	79
RISOLUZIONE SCHERMO	79
MUOVIMOUSE	80
TRASCINAMOUSE	80
FAICLICK	80
PREMIPULSANTE e RILASCIAPULSANTE	81
SCROLLAMOUSE	81
USALATASTIERA	81
COMBINAZIONETASTI	82
GESTIONE GIOCHI	82
ESEMPIO RETTANGOLO DA MUOVERE	82
PROGRAMMI DI ESEMPIO	84
ESEMPIO CALCOLA CIRCONFERENZA CERCHIO	84
ESEMPIO ACCESSO PASSWORD (semplice)	85
ESEMPIO ACCESSO PASSWORD (semplice – versione grafica)	85
ESEMPIO CHE CERCA LA PAROLA 'CIAO' SCRITTA DENTRO UNA CASELLA DI TESTO	86
ESEMPIO DI SOFTWARE PER TRASCRIZIONE FILE AUDIO	86
ESEMPIO DI SOFTWARE CHE CHIEDE UNA FRASE DA RIPETERE A VOCE	86
ESEMPIO DI REGISTRAZIONE DA MICROFONO E RIPRODUZIONE DEL REGISTRATO	86
ESEMPIO DI REGISTRAZIONE DA MICROFONO E TRASCRIZIONE TESTUALE DEL REGISTRATO	87

Il linguaggio **LUNGO** si può avvalere di moduli aggiuntivi che necessita scaricare da Internet, dai canali ufficiali, quindi per usare **LUNGO** con istruzioni di base, non è necessaria nessuna connessione Internet, mentre se si utilizzano istruzioni speciali, come ad esempio l'utilizzo di istruzioni che useranno Microsoft Excel, allora è necessaria una connessione Internet attiva che permetta al sistema di prelevare i moduli di istruzioni aggiuntivi necessarie.

Quando si crea un programma con **LUNGO**, non è consentito usare come nomi di variabili, i nomi delle istruzioni di **LUNGO**. Ad esempio, la parola **altezza** è un'istruzione di **LUNGO**, quindi non può essere usata come nome di variabile, cioè non potremo scrivere istruzioni del tipo:

`altezza=10`

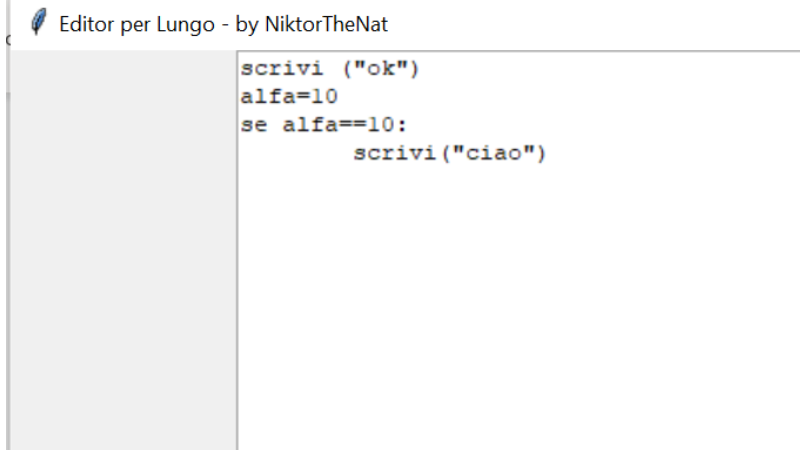
oppure, anche l'istruzione **scrivi** è un'istruzione di **LUNGO** e non può essere usata come nome di variabile, quindi non è ammesso scrivere:

`scrivi=23`

EDITOR DI PROGRAMMAZIONE

Quando viene avviato **Lungo**, si apre un editor di testo, che non ha alcuna funzionalità, se non quella di aprire e/o salvare il programma che realizzeremo.

Se abbiamo già creato un programma, questo verrà mostrato all'apertura dell'editor, mentre se non ne avevamo mai realizzato uno, allora l'editor sarà vuoto, senza testo.



```
Editor per Lungo - by NiktorTheNat

scrivi ("ok")
alfa=10
se alfa==10:
    scrivi("ciao")
```

CREARE UN ESEGUIBILE DEL PROGRAMMA

Quando si realizza un programma con **Lungo**, e si vuole creare il file eseguibile (in formato exe) del programma, è sufficiente inserire come prima istruzione, prima di qualsiasi altra istruzione, la seguente riga di codice: **creaeseguibile**

Se si crea ad esempio un programma che scrive "ciao" sullo schermo, e che quindi è costituita solo dalla seguente istruzione:

scrivi("ciao")

è possibile rendere il programma eseguibile, scrivendo il programma in questo modo:

creaeseguibile

scrivi("ciao")

Lungo creerà altri file e cartelle, tra cui la cartella **dist**, dove all'interno ci sarà il file **provola-converto.exe** che è il programma che avete generato. Volendo potete rinominarlo.

E' possibile che Microsoft Windows blocchi l'esecuzione del programma eseguibile, perché lo considera un virus. In realtà il programma che avete generato voi non è un virus, quindi dovete autorizzare il vostro sistema antivirus a far eseguire il programma.

La procedura per evitare questo problema potete vederla nel video a questo link

<https://youtu.be/MuVq2OML6Vc>

CONTROLLA PROGRAMMA PER RILEVARE ERRORI

La gestione degli errori nel programma **Lungo** non è al momento implementata. Per questo motivo si potrà tentare un debug del programma, per verificare eventuali errori, inserendo come prima

istruzione del programma, prima di qualsiasi altra istruzione, la seguente riga di codice:

controllaprogramma

Quando viene creato un programma, se mettiamo come prima istruzione l'istruzione **controllaprogramma**, potremmo poi verosimilmente rilevare errori, ma non nel linguaggio **Lungo** che abbiamo scritto noi, ma nella sua conversione in linguaggio **Python**.

Se abbiamo un programma di questo tipo, con un errore, che è evidenziato in giallo:

```
alfa=finestra()
alfa.titolo("Gestore password")
alfa.grandezza("300x300")
x=etichetta(alfa,testo="Digita la password").impacchetta()
y=caselladitesto(alfa,larghezza=10)
y.impacchetta()
z=pulsnte(alfa,testo="ACCEDI").impacchetta()
alfa.ciclico()
```

e l'errore è l'istruzione **pulsnte**, invece che **pulsante**, se proviamo ad eseguirlo normalmente, la finestra di **Lungo** si apre e si chiude senza eseguire nulla, proprio perché c'è l'errore.

Se proviamo ad eseguire lo stesso programma con lo stesso errore, aggiungendo l'istruzione per controllo degli errori **controllaprogramma**

controllaprogramma

```
alfa=finestra()
alfa.titolo("Gestore password")
alfa.grandezza("300x300")
x=etichetta(alfa,testo="Digita la password").impacchetta()
y=caselladitesto(alfa,larghezza=10)
y.impacchetta()
z=pulsnte(alfa,testo="ACCEDI",esegui=provola).impacchetta()
alfa.ciclico()
```

allora verrà aperta una finestra di comando, che attende nostre disposizioni.

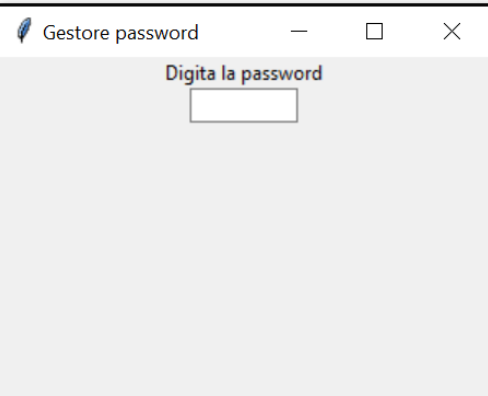
```
C:\Users\nikto\AppData\Local\Programs\Python\Python39\python.exe
> c:\users\nikto\onedrive\documenti\progetto lungo\provola-convertito.py(1)<module>()
-> from tkinter import *
(Pdb)
```

(chiaramente il testo della finestra sopra varierà da utente ad utente, in base alle proprie impostazioni del computer)

L'unico testo che sicuramente è comune a tutti gli utenti, è l'apparizione della riga **(Pdb)** e il cursore lampeggiante che aspetta che scriviamo qualcosa.

Se scriviamo la lettera **r** e premiamo INVIO, il programma verrà eseguito, e quando trova un errore lo scrive sulla riga di comando. Nel nostro caso accadrà questo:

```
C:\Users\nikto\AppData\Local\Programs\Python\Python39\python.exe
> c:\users\nikto\onedrive\documenti\progetto lungo\provola-convertito.py(1)<module>()
-> from tkinter import *
(Pdb) r
--Return--
> c:\users\nikto\onedrive\documenti\progetto lungo\provola-convertito.py(1)<module>()
-> z=pulsnte(alfa,text="ACCEDI",command=provola).pack()
(Pdb)
```



Il programma esegue le istruzioni che sono corrette, che nel caso specifico fanno apparire una finestra, ma nella riga di comando, viene scritta la riga che ha incontrato e che contiene un errore, e che ho evidenziato in rosso. Come è visibile, è stata segnalata esattamente la riga con l'errore,

di cui alcune istruzioni di **Lungo** erano già state convertite in linguaggio **Python**, come ho evidenziato in giallo.

SCRIVI

L'istruzione **scrivi** permette di scrivere testi o numeri sullo schermo, semplicemente mettendo tra parentesi tonde ciò che si vuole scrivere. Nel caso si vogliano scrivere numeri che eseguono calcoli, se i numeri vengono messi tra virgolette allora verranno ripetuti così come sono, mentre se vengono scritti senza essere racchiusi tra virgolette, verranno eseguite le operazioni.

```
scrivi("ciao")
```

```
scrivi("23+2")
```

```
scrivi(23+2)
```

la prima e seconda istruzione scriveranno esattamente quello che è scritto tra virgolette, mentre l'ultima istruzione esegue il calcolo tra i numeri, perché non è racchiusa tra virgolette.

```
ciao
23+2
25
Programma terminato
```

I testi scritti tra virgolette, ad esempio "ciao" oppure "23+2", sono definiti *stringhe*

Nelle *stringhe* possiamo scrivere caratteri speciali che eseguono determinati compiti, come ad esempio `\n` che impone il ritorno a capo, oppure `\t` che impone uno spazio di tabulazione.

```
scrivi("ciao\namico")
```

```
scrivi("ciao\tamico")
```

Nella prima istruzione il carattere `\n` farà scrivere, a capo, la parola "amico", mentre nella seconda istruzione il carattere `\t` farà spaziare di una tabulazione la parola "amico"

```
ciao
amico
ciao    amico
Programma terminato_
```

Se invece abbiamo la necessità di scrivere un testo, esattamente come l'abbiamo digitato, ma all'interno della stringa ci sono dei caratteri che potrebbero essere interpretati come caratteri speciali, come `\t` o `\n`, allora dobbiamo anteporre alla stringa la lettera **r**

```
scrivi("Uno il segno \n oppure \t per eseguire comandi")
```

```
scrivi(r"Uno il segno \n oppure \t per eseguire comandi")
```

Notare che nella seconda istruzione c'è la **r** davanti la stringa che impone di scrivere il testo esattamente come l'abbiamo digitato:

```
Uno il segno
  oppure           per eseguire comandi
Uno il segno \n oppure \t per eseguire comandi
Programma terminato_
```

Possiamo moltiplicare la scrittura di stringhe, semplicemente usando il carattere `*` (asterisco) che in informatica equivale al segno della moltiplicazione X (per)

```
scrivi("ciao"*3)
```

```
ciao ciao ciao
Programma terminato_
```

LE VARIABILI

Le variabili sono come dei contenitori dove possiamo mettere dentro dei valori o dei dati in genere.

I nomi delle variabili devono essere composte da una sola parola, e non deve essere una parola di una istruzione del linguaggio.

Se mettiamo un valore dentro una variabile, poi possiamo usarla per fare calcoli o per scrivere il suo valore.

Le variabili possono essere numeriche o tipo stringa.

Esempio di variabile numerica che ho chiamato *alfa*

alfa=3

esempio di variabile stringa che ho chiamato *parola*

parola="ciao a tutti"

esempio di variabile che contiene un calcolo numerico

prova=(3+2)*5

E' anche possibile dichiarare più variabili con la stessa istruzione, separando i vari nomi di variabili da virgola e altrettanto, separando i singoli testi o valori con la virgola:

alfa,beta,gamma=23,75,2

Una stringa è anche un oggetto, che possiamo manipolare usando le parentesi quadre. Se ad esempio abbiamo la stringa "ciao" e vogliamo scrivere solo la lettera "a" di quella parola, possiamo usare le parentesi quadre per imporre la scrittura solo di quel carattere, che si trova alla terza posizione, ma le posizioni vengono contate partendo da zero, quindi la lettera A si trova alla seconda posizione:

C I A O

0 1 2 3

Esempio:

alfa="ciao"

scrivi(alfa[2])

Viene imposto di scrivere la variabile *alfa*, ma solo il suo carattere alla posizione 2:

```
a
Programma terminato
```

Per scrivere l'ultimo carattere partendo da infondo, si usa -1, il penultimo -2.

Esempio che scrive l'ultimo carattere:

alfa="ciao"

scrivi(alfa[-1])

```
o
Programma terminato_
```

Usando il segno del : (duepunti), è possibile selezionare una determinata porzione di stringa.

Ad esempio:

alfa="ciao come stai"

scrivi(alfa[5:9])

questa istruzione farà scrivere solo dal carattere 5 al carattere 9 (escluso) della stringa *alfa*, estraendo solo la parola "come":

c i a o c o m e s t a i

0 1 2 3 4 5 6 7 8 9 10 11 12 13

```
come
Programma terminato_
```

MANIPOLAZIONE STRINGHE

Una stringa è un valore alfanumerico contenuto tra doppie virgolette " " oppure doppi apici ' '

Esempio di stringa: "ciao a tutti"

LUNGHEZZA

L'istruzione **lunghezza** permette di conoscere il numero di caratteri di cui è composta una stringa:

alfa="ciao come stai"

scrivi(lunghezza(alfa))

La stringa "ciao come stai" è composta da 14 caratteri:

```
14
Programma terminato_
```

CONTA

L'istruzione **conta** permette di contare quante volte è presente una stringa dentro un'altra. La stringa da cercare può anche essere un singolo carattere.

```
alfa="ciao a tutti"
beta=alfa.conta("a")
scrivi(beta)
```

```
2
Programma terminato
```

TROVA

L'istruzione **trova** serve a trovare una stringa dentro un'altra stringa. Se la stringa viene trovata, allora ci verrà data la posizione in cui inizia la prima occorrenza trovata, altrimenti, se non viene trovata, verrà dato come risultato **-1**.

```
alfa="ciao a tutti"
beta=alfa.trova("a")
scrivi(beta)
```

```
2
Programma terminato
```

INIZIACON

L'istruzione **iniziacon** permette di controllare se una determinata stringa, inizia con una determinata altra stringa. Se il controllo di una stringa sarà vero, allora verrà dato come risultato **True**, che in italiano significa **vero**, altrimenti se non inizia con la stringa controllata, allora avremo come risultato **False**, che in italiano significa **falso**.

Ad esempio:

```
alfa="ciao a tutti"
scrivi(alfa.iniziacon("c"))
```

darà come risultato **True**, cioè **vero**, perché è vero che la stringa "ciao a tutti" inizia con "c".

```
True
Programma terminato
```

Volendo, si può convertire il risultato in numero, in modo da avere **1** se la stringa che si è controllata è vera, oppure **0** se è falsa. Per far convertire il risultato in numero, allora si usa l'istruzione **intero**:

```
alfa="ciao a tutti"
scrivi(intero(alfa.iniziacon("c")))
```

```
1
Programma terminato
```

FINISCECON

L'istruzione **finiscecon** permette di controllare se una determinata stringa, finisce con una determinata altra stringa. La sintassi, cioè il funzionamento di questa istruzione è uguale alle regole dell'istruzione **iniziacon**, spiegata sopra.

```
alfa="ciao a tutti"
scrivi(alfa.finiscecon("c"))
```

```
False
Programma terminato
```

SOSTITUISCI

L'istruzione **sostituisci** permette di sostituire uno o più caratteri di una stringa con altro testo. Tra parentesi prima verrà messo il testo da cercare, e poi va messo il testo da sostituire al testo cercato.

```
alfa="ciao a tutti i miei amici"
```



```
scrivi(alfa)
beta=alfa.sostituisci("tutti","quasi tutti")
scrivi(beta)
```

```
ciao a tutti i miei amici
ciao a quasi tutti i miei amici
Programma terminato
```

MAIUSCOLO E MINUSCOLO

Le istruzioni **maiuscolo** e **minuscolo** permettono di convertire un testo, rispettivamente tutto in maiuscolo o tutto in minuscolo.

```
alfa="ciao"
beta="BENE"
scrivi(alfa.maiuscolo())
scrivi(beta.minuscolo())
```

```
CIAO
bene
Programma terminato
```

PRIMAMAIUSCOLA E PRIMEMAIUSCOLE

Le istruzioni **primamaiuscola** e **primemaiuscole**, permettono, rispettivamente, di convertire in maiuscolo solo la prima lettera della stringa, oppure di convertire in maiuscolo tutte le prime iniziali di ogni parola contenuta nella stringa.

```
alfa="ciao amici miei"
scrivi(alfa.primamaiuscola())
scrivi(alfa.primemaiuscole())
```

```
Ciao amici miei
Ciao Amici Mieì
Programma terminato
```

UNISCI

L'istruzione **unisci** permette di unire delle stringhe, utilizzando un o più caratteri di concatenamento.

Se ad esempio abbiamo la stringa **alfa** contenete il testo "ciao amici miei", potremmo far unire i singoli caratteri della stringa, mettendo tra l'oro un carattere, come ad esempio l'asterisco:

```
alfa="ciao amici miei"
scrivi(".*".unisci(alfa))
```

```
c*i*a*o* *a*m*i*c*i* *m*i*e*i
Programma terminato
```

Inoltre è anche possibile unire gli elementi di una lista, ad esempio:

```
alfa=["ciao","a","tutti"]
scrivi(".".unisci(alfa))
```

```
>ciao a tutti
Programma terminato
```

DIVIDI

L'istruzione **dividi** permette di dividere una stringa, specificando come dividerla.

Se ad esempio abbiamo una stringa:

```
alfa="ciao a tutti"
```

possiamo dividere la stringa, parola per parola: "ciao" "a" "tutti" con l'istruzione **dividi**, specificando tra parentesi tonde che vogliamo dividere la stringa dove trova uno spazio " ":

```
alfa="ciao a tutti"
```

```
scrivi(alfa.dividi(" "))
```

```
['ciao', 'a', 'tutti']  
Programma terminato
```

E' anche possibile specificare il massimo delle divisioni volute, ad esempio, per lo stesso programma, possiamo specificare che venga diviso una sola volta, quindi che la stringa deve essere divisa solo quando viene trovato il primo spazio e non anche per gli altri:

```
alfa="ciao a tutti"
```

```
scrivi(alfa.dividi(" ",1))
```

```
['ciao', 'a tutti']  
Programma terminato
```

ELIMINASPAZI E ELIMINASPAZIINIZIALI E ELIMINASPAZIFINALI

Le istruzioni **eliminaspazi**, **eliminaspaziiniziali** ed **eliminaspazifinali**, servono per eliminare eventuali spazi, rispettivamente, sia all'inizio che alla fine di una stringa (**eliminaspazi**) che solo all'inizio di una stringa (**eliminaspaziiniziali**) che solo alla fine di una stringa (**eliminaspazifinali**)

```
alfa="  ciao amici miei  "
```

```
scrivi(alfa)
```

```
scrivi(alfa.eliminaspazi())
```

```
scrivi(alfa.eliminaspaziiniziali())
```

```
scrivi(alfa.eliminaspazifinali())
```

```
    ciao amici miei  
ciao amici miei  
ciao amici miei  
    ciao amici miei  
Programma terminato
```

UNICI

L'istruzione **unici** permette di estrarre da una stringa o da una lista, i soli caratteri o parole uniche, anche se vengono ripetute.

Prendiamo ad esempio la parola "tutti". In questa parola ci sono 3 T ripetute, quindi dovessimo dire quali lettere formano la parola tutti, senza indicare quelle che si ripetono, dovremo rispondere "T U I":

```
alfa="tutti"
```

```
beta=unici(alfa)
```

```
scrivi(beta)
```

```
{'t', 'i', 'u'}  
Programma terminato
```

Lo stesso può essere fatto con le liste di parole.

```
alfa=["tre","ciao","lavatrice","ciao","provola","ecco","tre","ciao"]
```

```
beta=unici(alfa)
```

```
scrivi(beta)
```

```
{'ciao', 'tre', 'provola', 'ecco', 'lavatrice'}  
Programma terminato
```

COMBINAZIONIUNICHE

L'istruzione **combinazioniuniche**, permette di ottenere tutte le combinazioni possibili di elementi di una lista. Bisogna solo specificare quanti elementi vanno combinati tra loro.

Poniamo l'esempio di una lista di elementi: "niktor", "the", "nat".

Le combinazioni possibili sono:

niktor,the

niktor,nat

the,nat

nessun'altra combinazione unica è possibile.

alfa=["niktor","the","nat"]

beta=combinazioniuniche(alfa,2)

considera x con beta:

scrivi(x)

```
('niktor', 'the')
('niktor', 'nat')
('the', 'nat')
Programma terminato
```

Se ad esempio mettiamo come opzione il valore **3**, otterremo solo una combinazione unica possibile.

alfa=["niktor","the","nat"]

beta=combinazioniuniche(alfa,3)

considera x con beta:

scrivi(x)

```
('niktor', 'the', 'nat')
Programma terminato
```

E' anche possibile usare i numeri come elementi da elaborare. Proviamo questo esempio:

alfa=[6,9,3,5,4,8,2]

beta=combinazioniuniche(alfa,3)

considera x con beta:

scrivi(x)

```
(6, 3, 4)
(6, 3, 8)
(6, 3, 2)
(6, 5, 4)
(6, 5, 8)
(6, 5, 2)
(6, 4, 8)
(6, 4, 2)
(6, 8, 2)
(9, 3, 5)
(9, 3, 4)
(9, 3, 8)
(9, 3, 2)
(9, 5, 4)
(9, 5, 8)
(9, 5, 2)
(9, 4, 8)
(9, 4, 2)
(9, 8, 2)
(3, 5, 4)
(3, 5, 8)
(3, 5, 2)
(3, 4, 8)
(3, 4, 2)
(3, 8, 2)
(5, 4, 8)
(5, 4, 2)
(5, 8, 2)
(4, 8, 2)
Programma terminato
```

COMBINAZIONIESTESE

L'istruzione **combinazioniiestese** esegue lo stesso ragionamento e stesse proprietà dell'istruzione **combinazioniuniche**, solo che le combinazioni sono espansive a tutte le possibilità fattibili, anche in ordine agli elementi presi in considerazione. Vediamo la nuova istruzione e la differenza all'uso di **combinazioniuniche**:

```

alfa=["niktor","the","nat"]
beta=combinazioniestese(alfa,2)
considera x con beta:
    scrivi(x)

```

```

('niktor', 'the')
('niktor', 'nat')
('the', 'niktor')
('the', 'nat')
('nat', 'niktor')
('nat', 'the')
Programma terminato

```

Ed ecco cosa sarebbe successo usando l'istruzione **combinazioniuniche**:

```

alfa=["niktor","the","nat"]
beta=combinazioniuniche(alfa,2)
considera x con beta:
    scrivi(x)

```

```

('niktor', 'the')
('niktor', 'nat')
('the', 'nat')
Programma terminato

```

CONTALETTERE

L'istruzione **contalettere** permette di contare e/o raggruppare una lista di stringhe, in relazione al numero di lettere di cui è costituito ogni elemento.

Esempio pratico:

```

alfa=["luca","marcella","rita","franco","giorgio"]
beta=contalettere(alfa,lunghezza)
considera x,y con beta:
    scrivi(x)

```

Nell'esempio sopra, l'istruzione **contalettere** ha due opzioni, di cui una è la lista **alfa** da elaborare, mentre la seconda è l'opzione **lunghezza** che conterà la lunghezza, in caratteri, di ogni elemento della lista. Nel ciclo **considera** vanno pertanto usate due variabili, ad esempio **x** e **y**, in cui in una ci sarà il numero di lettere, e nella seconda il testo dell'elemento. Con l'esempio sopra otterremo solo il numero di caratteri di ogni elemento:

```

4
8
4
6
7
Programma terminato

```

Se invece aggiungiamo le seguenti istruzioni:

```

alfa=["luca","marcella","rita","franco","giorgio"]
beta=contalettere(alfa,lunghezza)
considera x,y con beta:
    considera z in y:
        scrivi(x,z)

```

allora otterremo la lunghezza di ogni elemento associata al suo elemento:

```

4 luca
8 marcella
4 rita
6 franco
7 giorgio
Programma terminato

```

NUMEROSUCCESSIVO

L'istruzione **numerosuccessivo** permette di elaborare una lista di numeri (ma vedremo anche di stringhe) valutando sempre l'elemento successivo di una lista.

Se non si usa alcuna opzione aggiuntiva, l'istruzione **numerosuccessivo** somma, progressivamente, l'elemento precedente con quello successivo. Ad esempio, in una lista di numeri: [1,3,2,5], il computer ragionerà valutando prima il numero 1, poi il numero 3+1 che fa 4, poi 4+2 che fa 6, poi 6+5 che fa 11.

alfa=[1,3,2,5,4,1,7]

beta=numerosuccessivo(alfa)

considera x con beta:

scrivi(x)

```
1
4
6
11
15
16
23
Programma terminato
```

Usando l'opzione **maggiore**, valuterà progressivamente, il numero più grande che sta leggendo dalla lista. Se ad esempio abbiamo una lista di questo tipo: [5,2,6,2] allora succederà questo ragionamento:

- quando legge il numero 5 per il computer 5 è il numero più grande
- quando leggerà il numero 2 per il computer 5 è il numero più grande
- quando leggerà il numero 6 per il computer 6 è il numero più grande
- quando leggerà il numero 2 per il computer 2 è il numero più grande

alfa=[5,2,6,2,5,11,9,3]

beta=numerosuccessivo(alfa,maggiore)

considera x con beta:

scrivi(x)

```
5
5
6
6
6
11
11
11
Programma terminato
```

Usando invece l'opzione **minore** verrà eseguito lo stesso tipo di ragionamento, ma considerando sempre il numero minore che trova.

alfa=[5,2,6,2,5,11,9,3]

beta=numerosuccessivo(alfa,minore)

considera x con beta:

scrivi(x)

```

5
2
2
2
2
2
2
2
2
Programma terminato

```

Si può usare l'istruzione **numerosuccessivo** anche con le stringhe, perché considera il valore numerico ASCII delle lettere, quindi:

```

alfa=["t","f","s","p","g","k","u","a","z"]
beta=numerosuccessivo(alfa,maggiore)
considera x con beta:
scrivi(x)

```

```

t
t
t
t
t
t
t
t
u
u
z
Programma terminato

```

E di conseguenza, anche:

```

alfa=["t","f","s","p","g","k","u","a","z"]
beta=numerosuccessivo(alfa,minore)
considera x con beta:
scrivi(x)

```

```

t
f
f
f
f
f
f
f
a
a
Programma terminato

```

LE LISTE

Le liste sono elenchi di dati, che possono essere numeri e/o stringhe, che poi possiamo richiamare singolarmente o in gruppo, usando le parentesi quadre, così come facciamo con le stringhe singole.

Per creare una lista, si procede come quando creiamo una variabile, solo che i dati vanno scritti tra parentesi quadre:

```

alfa=["ciao","ultimo","casa",7,23,"lavatrice","provola"]
scrivi(alfa)

```

l'istruzione **scrivi(alfa)** scriverà la lista completa:

```

['ciao', 'ultimo', 'casa', 7, 23, 'lavatrice', 'provola']
Programma terminato_

```

Se voglio estrarre solo la stringa "casa", che si trova alla terza posizione, ma devo ricordare che il conteggio viene fatto da 0 (zero), allora la stringa "casa" si trova alla posizione 2:

```
alfa=["ciao","ultimo","casa",7,23,"lavatrice","provola"]
scrivi(alfa[2])
```

```
casa
Programma terminato_
```

Con il metodo **aggiungi** possiamo aggiungere elementi alla lista *alfa*

```
alfa=["ciao","ultimo","casa",7,23,"lavatrice","provola"]
alfa.aggiungi("niktor")
scrivi(alfa)
```

In questo caso aggiungo la stringa "niktor" alla lista *alfa*:

```
['ciao', 'ultimo', 'casa', 7, 23, 'lavatrice', 'provola', 'niktor']
Programma terminato_
```

Con il metodo **posizione** puoi eliminare un elemento dalla lista, specificando la sua posizione.

Nell'esempio sotto, elimino l'elemento 1, che però ricordiamo che sarà eliminato il secondo della lista, perché il programma inizia a contare da zero.

```
alfa=["ciao","ultimo","casa",7,23,"lavatrice","provola"]
alfa.posizione(1)
scrivi(alfa)
```

Con questo programma, verrà eliminato l'elemento "ultimo"

```
['ciao', 'casa', 7, 23, 'lavatrice', 'provola']
Programma terminato
```

Con il metodo **rimuovi** si può invece rimuovere un elemento attraverso il suo contenuto.

Nell'esempio sotto verrà rimosso l'elemento con testo "lavatrice"

```
alfa=["ciao","ultimo","casa",7,23,"lavatrice","provola"]
alfa.rimuovi("lavatrice")
scrivi(alfa)
```

```
['ciao', 'ultimo', 'casa', 7, 23, 'provola']
Programma terminato
```

Usando l'istruzione **lunghezza** possiamo sapere di quanti elementi è costituita una lista:

```
alfa=["ciao","ultimo","casa",7,23,"lavatrice","provola"]
scrivi(lunghezza(alfa))
```

```
7
Programma terminato
```

Il metodo **conta** permette di contare quante volte viene trovato un determinato elemento nella lista, se ad esempio vogliamo cercare quante volte è presente l'elemento "ultimo" nella lista, scriviamo:

```
alfa=["ciao","ultimo","casa",7,"ultimo","lavatrice","provola"]
scrivi(alfa.conta("ultimo"))
```

```
2
Programma terminato
```

Il metodo **indice** permette invece di sapere in quale posizione si trova un determinato elemento nella lista. Nell'esempio sotto vediamo in quale posizione si trova l'elemento "lavatrice":

```
alfa=["ciao","ultimo","casa",7,"ultimo","lavatrice","provola"]  
scrivi(alfa.indice("lavatrice"))
```

Il risultato sarà 5, perché partendo a contare da zero, l'elemento "lavatrice" è il quinto elemento della lista:

```
5  
Programma terminato
```

Il metodo **alfabetico** permette di ordinare alfabeticamente gli elementi di una lista.

```
alfa=["ciao","ultimo","casa","ultimo","lavatrice","provola"]  
alfa.alfabetico()  
scrivi(alfa)
```

Il seguente programma darà un ordine alfabetico agli elementi:

```
['casa', 'ciao', 'lavatrice', 'provola', 'ultimo', 'ultimo']  
Programma terminato
```

ATTENZIONE che non è possibile ordinare alfabeticamente elementi che contengono sia elementi numerici che elementi stringa, come ad esempio quello sotto che ha all'interno l'elemento numerico 7:

```
alfa=["ciao","ultimo",7,"casa","ultimo","lavatrice","provola"]
```

Il metodo **inverso** permette di ordinare una lista al contrario, non alfabeticamente in modo inverso, ma letteralmente in modo inverso a come è stata scritta la lista. Per intenderci il primo elemento sarà ultimo, il secondo sarà penultimo e così via.

```
alfa=["ciao","ultimo","casa","ultimo","lavatrice","provola"]  
alfa.inverso()  
scrivi(alfa)
```

```
['provola', 'lavatrice', 'ultimo', 'casa', 'ultimo', 'ciao']  
Programma terminato
```

I DIZIONARI

I **dizionari** sono come le liste, cioè contengono elementi, ma ogni elemento ha un indice, così come avviene, ad esempio, per i database.

I **dizionari** vanno scritti tra parentesi graffe e gli elementi e gli indici, vanno divisi tra loro, dal segno del duepunti (:)

INFORMAZIONE: Le parentesi graffe si ottengono, premendo contemporaneamente sulla tastiera il **tasto per le maiuscole** più il tasto **ALT GR** più il tasto **[** oppure **]** in relazione a se si deve aprire o chiudere la parentesi graffa.

```
alfa={"fiat":2,"volvo":5,"renault":8,"citroen":3,"bmw":6}  
scrivi(alfa)
```

L'esempio sopra, crea una lista dove l'indice sono i nomi di aiuto e ogni indice è associato ad un numero, che potremmo, per esempio aver scritto, per indicare quanti veicoli abbiamo nella nostra concessionaria.

```
{'fiat': 2, 'volvo': 5, 'renault': 8, 'citroen': 3, 'bmw': 6}  
Programma terminato
```

Se vogliamo ottenere l'informazione associata ad un indice, dobbiamo scrivere il nome del dizionario, seguito dal nome dell'indice racchiuso tra parentesi quadre.

Nell'esempio sotto, estraiamo il numero di veicoli associati all'indice "volvo":

```
alfa={"fiat":2,"volvo":5,"renault":8,"citroen":3,"bmw":6}  
scrivi(alfa["volvo"])
```



```
5
Programma terminato
```

Per cambiare un valore associato ad un indice, gli si assegna il nuovo valore, tramite il segno uguale, come nell'esempio sotto:

```
alfa={"fiat":2,"volvo":5,"renault":8,"citroen":3,"bmw":6}
alfa["volvo"]=9
scrivi(alfa["volvo"])
```

In questo caso ho cambiato il valore associato all'indice "volvo" da 5 a 9

```
9
Programma terminato
```

Per aggiungere un nuovo indice con il suo valore al dizionario, si usa la stessa sintassi spiegata sopra per la modifica, scrivendo il nome di indice nuovo ed il suo relativo valore. Nell'esempio sotto aggiungo al dizionario l'indice "audi" con il valore 4

```
alfa={"fiat":2,"volvo":5,"renault":8,"citroen":3,"bmw":6}
alfa["audi"]=4
scrivi(alfa)
```

```
{'fiat': 2, 'volvo': 5, 'renault': 8, 'citroen': 3, 'bmw': 6, 'audi': 4}
Programma terminato
```

CICLO

L'istruzione **ciclo** permette di eseguire istruzioni cicliche in base a determinate condizioni.

L'esempio mostra meglio l'utilizzo di questa istruzione. Le prime due righe vengono scritte attaccate al margine sinistro, mentre le ultime due vanno scritte spostandosi con il tabulatore (tasto TAB della tastiera), per essere rientrate dal margine sinistro. Lo spostamento con la tabulazione viene chiamato **indentazione**:

```
alfa=10
ciclo alfa<20:
    scrivi(alfa)
    alfa=alfa+1
```

la prima riga crea una variabile di nome *alfa* a cui viene assegnato il numero 10

la seconda riga inizia l'istruzione **ciclo** in cui si dice al programma di eseguire le istruzioni che troverà sotto, indentate (indentate significa "rientrate dal margine"), fino a che la variabile *alfa* avrà un valore minore di 20. Inizialmente ricordiamo che *alfa* ha un valore di 10, quindi inferiore a 20

la terza riga fa scrivere il valore di *alfa* sullo schermo

la quarta riga aumenta di una unità il valore di *alfa*

```
10
11
12
13
14
15
16
17
18
19
Programma terminato_
```

DOMANDA

L'istruzione **domanda** è una istruzione di input, che permette di ottenere informazioni dall'utente.

L'istruzione deve essere associata ad una variabile, dove andrà a finire la risposta.

```
alfa=domanda("Come ti chiami")
scrivi(alfa)
```

la prima riga pone la domanda, e la risposta verrà inserita nella variabile *alfa*
la seconda riga scrive sullo schermo il contenuto della variabile *alfa*

```
Come ti chiami nik
nik
Programma terminato_
```

Per ottenere risposte di valori numerici, dobbiamo racchiudere l'istruzione *domanda*, dentro le parentesi tonde di una istruzione **intero**, che è una istruzione che serve a impostare i numeri a valore intero, senza decimali:

```
alfa=intero(domanda("Digita un numero"))
scrivi(alfa+3)
```

la prima riga pone la domanda "digita un numero" e la risposta, viene considerata un numero intero, che poi viene messa nella variabile *alfa*
la seconda riga scrive sullo schermo il valore della variabile *alfa* sommandoci il numero 3

```
Digita un numero23
26
Programma terminato_
```

CONSIDERA...CON

L'istruzione **considera con** è un'istruzione ciclica che permette di eseguire il ciclo un tot di volte, usando un indice. Vediamo un esempio pratico:

```
alfa=["uno","due","tre","quattro"]
considera x con alfa:
```

```
    scrivi(x)
```

la prima riga crea una lista di elementi

la seconda riga prende in considerazione la variabile *x*, che viene creata in quel momento, processando la variabile *alfa*, che contiene la lista di dati e da quel momento inizia un ciclo, per cui ogni elemento presente nella lista *alfa*, verrà messo nella variabile *x*, e quindi quel ciclo verrà ripetuto per tutte le parole della lista, cioè 4 volte

la terza riga scrive sullo schermo il contenuto della variabile *x*. Dato che ad ogni giro del ciclo alla variabile *x* viene passato un elemento della lista *alfa*, succede che verranno scritti tutti gli elementi sullo schermo

```
uno
due
tre
quattro
Programma terminato_
```

Altro esempio in cui vengono scritte gli stessi elementi della lista precedente, ma con a fianco la lunghezza in caratteri di ogni stringa:

```
alfa=["uno","due","tre","quattro"]
considera x con alfa:
```

```
    scrivi(x,lunghezza(x))
```

```
uno 3
due 3
tre 3
quattro 7
Programma terminato_
```

INTERVALLO

L'istruzione **intervallo** permette di indicare un determinato intervallo di numeri che potrà essere oggetto di elaborazione. Un utilizzo tipico è con l'istruzione **considera con**, come nell'esempio sotto:

```
considera x con intervallo(6):
    scrivi(x)
```

la prima riga considera la variabile *x* per contenere tutti i valori del ciclo, che sono presi dall'**intervallo** di 6, cioè da 0 a 5 (il linguaggio parte sempre a contare da zero e così sono comunque 6 elementi). Questo significa che il ciclo sarà eseguito 6 volte
la seconda riga scrive man mano il valore della variabile *x* ad ogni ciclo

```
0
1
2
3
4
5
Programma terminato_
```

SE...ALTRIMENTISE...ALTRIMENTI

L'istruzione **se** permette di eseguire dei ragionamenti.

Faccio un esempio pratico in cui chiedo un numero all'utente, e se il numero che scriverà è maggiore di 10 allora verrà scritto "Numero troppo grande"

```
alfa=intero(domanda("Dammi un numero minore di 10 "))
```

```
se alfa>10:
```

```
    scrivi("Numero troppo grande")
```

la prima riga pone la domanda all'utente e il numero digitato dall'utente lo inserisce nella variabile *alfa*

la seconda riga controlla se la variabile *alfa* contiene un numero maggiore di 10, e in caso è così, allora esegue l'istruzione indentata che avvisa "Numero troppo grande"

```
Dammi un numero minore di 10 32
Numero troppo grande
Programma terminato
```

L'istruzione **se** prevede anche l'uso opzionale di **altrimentise** che permette di fare una o più ulteriori ragionamenti. Nell'esempio sotto aggiungo l'istruzione **altrimentise** per verificare anche se il numero inserito dall'utente è minore di 1, ed in caso sia così, faccio scrivere "Numero troppo piccolo"

```
alfa=intero(domanda("Dammi un numero minore di 10 "))
```

```
se alfa>10:
```

```
    scrivi("Numero troppo grande")
```

```
altrimentise alfa<1:
```

```
    scrivi("Numero troppo piccolo")
```

Anche l'istruzione **altrimentise** prevede di indentare le relative istruzioni da eseguire

```
Dammi un numero minore di 10 -3
Numero troppo piccolo
Programma terminato
```

Infine esiste la possibilità di usare con l'istruzione **se**, anche l'istruzione **altrimenti**, che viene eseguita se tutti gli altri ragionamenti non corrispondono al ragionamento.

Nell'esempio sotto ho infatti aggiunto l'istruzione **altrimenti** che scriverà sullo schermo il numero digitato dall'utente, che pertanto non sarà né troppo grande né troppo piccolo:

```
alfa=intero(domanda("Dammi un numero minore di 10 "))
```

```
se alfa>10:
```

```
    scrivi("Numero troppo grande")
```

```
altrimentise alfa<1:
```

```
    scrivi("Numero troppo piccolo")
```

```
altrimenti:
```

```
    scrivi("Hai scritto il numero",alfa)
```

Ed ecco il risultato:

```
Dammi un numero minore di 10 4
Hai scritto il numero 4
Programma terminato
```

Con l'istruzione **se**, è possibile usare gli operatori booleani, che permettono di creare ragionamenti più complessi.

Gli operatori booleani sono **anche** e **oppure**.

Nell'esempio sotto chiedo il nome dell'utente, e se il nome è "nicola" oppure "niktor", allora gli faccio scrivere "ciao amico", altrimenti gli faccio scrivere "non ti conosco"

```
alfa=domanda("Come ti chiami? ")
se alfa=="nicola" oppure alfa=="niktor":
    scrivi("Ciao amico")
altrimenti:
    scrivi("Non ti conosco")
```

```
Come ti chiami? nicola
Ciao amico
Programma terminato
```

Vediamo ora un esempio dell'uso dell'operatore booleano **anche**.

Nel programma sotto, chiedo all'utente il suo nome utente e la password. Se il nome utente è "niktor" e anche la password è "123" allora faccio scrivere "accesso consentito", altrimenti faccio scrivere "accesso negato"

```
alfa=domanda("Nome utente? ")
beta=domanda("Password? ")
se alfa=="niktor" anche beta=="123":
    scrivi("Accesso consentito")
altrimenti:
    scrivi("Accesso negato")
```

```
Nome utente? niktor
Password? 143
Accesso negato
Programma terminato
```

FUNZIONE

La **funzione** permette di racchiudere un piccolo o grande pezzo di codice di programmazione, che può essere richiamato dal programma, anche più volte.

Le istruzioni nella **funzione** devono essere elencate in modo indentato.

Mostro un esempio in cui creo una funzione di nome *provola* che contiene l'istruzione per scrivere "ciao" sullo schermo.

```
funzione provola():
    scrivi("ciao")
```

Questo programma non esegue nulla, perché il linguaggio riconosce la funzione, con le sue istruzioni, ma non la esegue, perché va chiamata dal resto del programma. Per chiamare una funzione e farle eseguire le sue istruzioni, va scritto il nome della funzione. Le parentesi tonde sono obbligatorie.

Ecco una chiamata alla funzione che farà eseguire le sue istruzioni.

```
funzione provola():
    scrivi("ciao")
```

```
provola()
```

La prima riga crea la funzione *provola*

La seconda riga fa scrivere "ciao" sullo schermo, ma solo se verrà eseguita la funzione
La terza riga chiama la funzione *provola*, che quindi farà eseguire la scrittura di "ciao" sullo schermo

Con la **funzione** si possono usare anche parametri che sono delle variabili che inviamo quando chiamiamo la funzione.

Ho creato l'esempio sotto per spiegare questo concetto.

funzione provola(alfa):

 scrivi("ciao",alfa)

beta=domanda("Come ti chiami? ")

provola(beta)

La prima riga crea la funzione *provola* e tra parentesi ho messo il nome di una variabile *alfa* che si occuperà di ricevere dati dalla chiamata

La seconda riga scriverà "ciao " e il nome dell'utente che arriverà dalla chiamata

La terza riga crea un input per ricevere il nome dell'utente, mettendolo in una variabile *beta*

La quarta riga chiama la funzione *provola* inviando il contenuto della variabile *beta*.

Una variabile dentro una funzione, non può essere usata fuori dalla funzione e viceversa.

```
Come ti chiami? marco
ciao marco
Programma terminato
```

PROGRAMMAZIONE AD OGGETTI

La programmazione ad oggetti è una modalità di programmazione che migliora la scrittura e gestione del codice di programmazione, usando la possibilità di usare come degli stampini che permettono di realizzare più oggetti.

La **classe** rappresenta la base di partenza della programmazione ad oggetti.

Per creare una classe, ad esempio, di nome *veicolo*, si deve scrivere:

classe veicolo:

motore=1500

La classe è come lo stampino. Ora possiamo creare oggetti.

Nell'esempio sotto creo un oggetto *alfa*:

classe veicolo:

motore=1500

alfa=veicolo()

scrivi(alfa.motore)

La prima riga inizializza la classe *veicolo*

La seconda riga assegna il valore 1500 alla proprietà *motore*

La terza riga crea un oggetto che si chiama *alfa*, richiamando lo stampino *veicolo*

La quarta riga scrive sullo schermo il valore della proprietà *motore* dell'oggetto *alfa*

La funzione **__inizializza__** è una funzione direi necessaria, quando si programma ad oggetti, perché è una funzione che si esegue all'inizio della classe, e serve ad assegnare valori agli oggetti o comunque eseguire operazioni all'inizio della classe.

Con l'utilizzo di **__inizializza__** si usa anche il parametro *mestesso* che è una istanza che viene utilizzata per accedere alle variabili che appartengono alla classe

classe veicolo:

funzione __inizializza__(mestesso,marca,motore):

mestesso.marca=marca

mestesso.motore=motore

alfa=veicolo("fiat",1500)

scrivi(alfa.marca)

scrivi(alfa.motore)

La prima riga inizializza la classe *veicolo*

La seconda riga crea la funzione **__inizializza__** che verrà eseguita per prima, e che dichiara le variabili che vengono usate nella classe. Le parole *mestesso*, *marca* e *motore*, sono parametri

La terza e quarta riga dichiarano le variabili della classe

La quinta riga crea un oggetto *alfa* assegnando i valori ai metodi *marca* e *motore*
La sesta e settima riga scrivono sullo schermo i valori

```
fiat
1500
Programma terminato
```

Nella classe si possono inserire più funzioni, per fargli eseguire più compiti e per essere richiamate dall'oggetto.

Vediamo un esempio, aggiungendo una funzione alla classe precedente:

classe veicolo:

```
funzione __inizializza__(mestesso,marca,motore):
```

```
    mestesso.marca=marca
```

```
    mestesso.motore=motore
```

```
funzione descrivi(mestesso):
```

```
    scrivi("La mia auto è una",mestesso.marca,"con motore",mestesso.motore)
```

```
alfa=veicolo("fiat",1500)
```

```
alfa.descrivi()
```

Da notare in particolare l'ultima riga, dove non inserisco l'istruzione dentro una istruzione **scrivi**, perché la funzione *descrivi* usa già un'istruzione *scrivi*

L'ereditarietà, nella programmazione ad oggetti, consiste semplicemente nella possibilità di creare una classe che eredita tutte le proprietà di un'altra classe.

Seguendo l'esempio della classe precedente, che si occupa di veicoli, supponiamo, a 4 ruote, possiamo creare un'altra classe, che si occupa di veicoli a 2 ruote, che però eredita le stesse proprietà della classe veicoli a 4 ruote, dato che entrambi i tipi di veicoli hanno una marca ed un motore.

Per ereditare una classe, si crea una nuova classe mettendo tra parentesi tonde il nome della classe che si vuole ereditare. Fate attenzione che viene usata una nuova istruzione, che si chiama **passa**, e serve a dire al programma di passare oltre, cioè di non eseguire nulla in concreto. E' un'istruzione di ripiego, dato che non possiamo creare una classe, senza dargli delle istruzioni da eseguire, quindi si scrive **passa** in modo che c'è un'istruzione, che però di fatto non esegue alcun compito.

classe veicolo:

```
funzione __inizializza__(mestesso,marca,motore):
```

```
    mestesso.marca=marca
```

```
    mestesso.motore=motore
```

```
funzione descrivi(mestesso):
```

```
    scrivi("La mia auto è una",mestesso.marca,"con motore",mestesso.motore)
```

classe moto(veicolo):

```
    passa
```

```
alfa=moto("aprilia",1500)
```

```
alfa.descrivi()
```

```
La mia auto è una aprilia con motore 1500
Programma terminato
```

Per fare in modo di mantenere l'ereditarietà dei metodi della classe *veicolo*, possiamo usare la funzione **__inizializza__** che richiama i metodi della classe *veicolo*

classe moto(veicolo):

```
funzione __inizializza__(self,marca,motore):
```

inoltre, attraverso l'istruzione **super**, possiamo far ereditare tutti i metodi e proprietà della classe *veicolo*:

classe veicolo:

```
funzione __inizializza__(mestesso,marca,motore):
```

```
    mestesso.marca=marca
```

```

    mestesso.motore=motore
funzione descrivi(mestesso):
    scrivi("La mia auto è una",mestesso.marca,"con motore",mestesso.motore)

```

```

classe moto(veicolo):
    funzione __inizializza__(self,marca,motore):
        super().__inizializza__(marca,motore)

```

```

alfa=moto("aprilia",1500)
alfa.descrivi()

```

Se poi vogliamo aggiungere una nuova proprietà alla sola classe *moto*, la aggiungiamo nella funzione `__inizializza__` della classe *moto* in questo modo:

```

classe veicolo:
    funzione __inizializza__(mestesso,marca,motore):
        mestesso.marca=marca
        mestesso.motore=motore
    funzione descrivi(mestesso):
        scrivi("La mia auto è una",mestesso.marca,"con motore",mestesso.motore)

```

```

classe moto(veicolo):
    funzione __inizializza__(self,marca,motore):
        super().__inizializza__(marca,motore)
        mestesso.ruote="due"

```

```

alfa=moto("aprilia",1500)
scrivi(alfa.ruote)

```

```

due
Programma terminato

```

Nell'esempio sopra, però, ho forzato la proprietà **ruote** con il valore "due". Se vogliamo poter far gestire quel valore all'utente, in programmazione, allora bisogna aggiungere il parametro **ruote** alla funzione *inizializza*

```

classe moto(veicolo):
    funzione __inizializza__(self,marca,motore,ruote):

```

E ora possiamo riscrivere il codice nel seguente modo:

```

classe veicolo:
    funzione __inizializza__(mestesso,marca,motore):
        mestesso.marca=marca
        mestesso.motore=motore
    funzione descrivi(mestesso):
        scrivi("La mia auto è una",mestesso.marca,"con motore",mestesso.motore)

```

```

classe moto(veicolo):
    funzione __inizializza__(self,marca,motore,ruote):
        super().__inizializza__(marca,motore)
        mestesso.ruote=ruote

```

```

alfa=moto("aprilia",1500,"tre")
scrivi(alfa.ruote)

```

```

tre
Programma terminato

```

GESTIONE DEGLI ERRORI

E' possibile gestire gli errori di programmazione, inserendo il codice tra le istruzioni **prova** e **problema**.

In pratica, si indenta del codice dentro l'istruzione **prova**. Se vengono rilevati errori, allora il programma passa all'istruzione **problema**.

Vediamo un esempio, in cui dichiaro una variabile di nome *alfa*, quindi con l'istruzione **scrivi**, dico di scrivere la variabile sullo schermo, ma sbaglio a scrivere il nome della variabile anziché *alfa* scrivo *alfas* e così verrà rilevato l'errore.

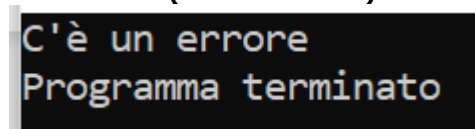
prova:

```
alfa="ciao"
```

```
scrivi(alfas)
```

problema:

```
scrivi("C'è un errore")
```



```
C'è un errore
Programma terminato
```

Usare l'istruzione **problema** da solo, rileva qualsiasi errore che si dovesse verificare, ma è anche possibile controllare il verificarsi di un determinato tipo di errore, in modo da agire di conseguenza. E' anche possibile sapere che tipo di errore si è verificato. Al momento (dicembre 2020) l'avviso del tipo di errore viene scritto in lingua inglese.

Per capire il tipo di errore bisogna usare il parametro **TipoErrore** facendolo diventare, con l'istruzione **diventa**, una variabile nostro piacimento. Esempio:

prova:

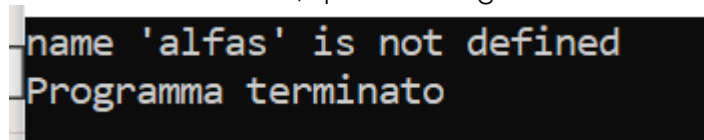
```
alfa="ciao"
```

```
scrivi(alfas)
```

problema TipoErrore diventa beta:

```
scrivi(beta)
```

La terza riga tenta di scrivere la variabile di nome *alfas* mentre la variabile che abbiamo creato si chiama *alfa* senza la *s*, quindi viene generato l'errore con la descrizione del tipo di errore:



```
name 'alfas' is not defined
Programma terminato
```

Per errori di nomi di variabili sbagliati o variabili inesistenti si può verificare l'errore

VariabileNonDefinita. ATTENZIONE che questi errori vanno scritti rispettando maiuscole e minuscole, quindi non verrebbe rilevato l'errore e il programma va in crash se si scrive **variabilenondefinita** anziché **VariabileNonDefinita**

prova:

```
alfa="ciao"
```

```
scrivi(alfas)
```

problema VariabileNonDefinita:

```
scrivi("C'è un errore")
```

LEGGERE E SCRIVERE FILE

Per scrivere un file di testo sul computer, si usano le seguenti istruzioni:

```
alfa=apri("esempio.txt",scrittura)
```

```
alfa.memorizza("ciao a tutti")
```

```
alfa.chiudi()
```

La prima riga apre un file sul disco, con l'istruzione **apri**, di nome "esempio.txt" e lo crea in scrittura su disco, grazie all'istruzione **scrittura**

La seconda riga, con l'istruzione **memorizza**, non fa altro che memorizzare dentro il file la frase "ciao a tutti". Volendo potevamo memorizzare il contenuto di una variabile

La terza riga, con l'istruzione **chiudi()**, chiude la comunicazione con il file

Sullo schermo non apparirà nulla, ma nel vostro hard disk, nella stessa cartella in cui state realizzando il programma, viene generato il file "esempio.txt"

OneDrive > Documenti > progetto lungo					Cerca in progetto	
	Nome	Stato	Ultima modifica	Tipo		
	esempio.txt	✓	27/12/2020 09.46	Docu		
	lungo.py	✓	27/12/2020 09.46	Pytho		
	MANUALE LUNGO.docx	↻	27/12/2020 09.43	Docu		
	provola.txt	✓	27/12/2020 09.44	Docu		
	provola-convertito.py	✓	27/12/2020 09.46	Pytho		

Per leggere quel file, si usano le seguenti istruzioni:

```
alfa=apri("esempio.txt",lettura)  
scrivi(alfa.leggi())
```

La prima riga, con l'istruzione **apri**, legge il file "esempio.txt" in modalità **lettura**

La seconda riga, scrive sullo schermo quello che viene letto dal file, grazie all'istruzione **leggi()**

VERO E FALSO

Le istruzioni **vero** e **falso** sono costanti, che possono essere utili in determinate occasioni.

Determinano lo stato di una determinata condizione, appunto, se vera o falsa.

Di norma, ogni condizione, se non espressamente elaborata, è vera.

ciclo vero:

```
scrivi("ciao")
```

La prima riga inizia un ciclo fino a che la condizione è vera. Ho detto prima che una condizione è sempre vera a meno che non viene elaborato il contrario, quindi questo ciclo sarà infinito.

Per questo motivo la parola "ciao" verrà scritta all'infinito.

Per bloccare l'esecuzione del programma bisognerà quindi premere sulla tastiera i tasti **CTRL** e il tasto **C** contemporaneamente.

```
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao  
ciao
```

FERMA E PASSA

Quando si eseguono istruzioni cicliche, può essere utile uscire dal ciclo, magari al verificarsi di una condizione. Per uscire dal ciclo si può usare l'istruzione **ferma**.

considera x con intervallo(5000):

```
scrivi(x)
```

```
se x==30:
```

```
ferma
```

Il programma qui sopra, esegue un ciclo per 5000 volte, e fa scrivere ogni numero del ciclo sullo schermo, ma arrivato a 30, con l'istruzione **ferma**, blocca il ciclo ed esce dal programma.

```

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
Programma terminato

```

L'istruzione **passa** permette di non far terminare il programma, nonostante non sia stato determinato alcun compito. Se ad esempio scriviamo:

```
x=domanda("Dimmi il tuo nome ")
```

```
se x=="nik":
```

```
altrimenti:
```

```
    scrivi("non ti conosco")
```

ed eseguiamo il programma, avremmo un errore, perché mancano le istruzioni da eseguire se il nome che viene scritto è uguale a "nik". Ma ipotizzando che non sappiamo ancora che istruzioni mettere, ma vogliamo comunque testare il programma, senza avere errori, allora possiamo inserire in quel punto, l'istruzione **passa** che dice a **Lungo** di passare all'eventuale istruzione successiva, senza andare in errore.

```
x=domanda("Dimmi il tuo nome ")
```

```
se x=="nik":
```

```
    passa
```

```
altrimenti:
```

```
    scrivi("non ti conosco")
```

FUNZIONI MATEMATICHE

Per sommare dei numeri si usa il segno +

Per sottrarre dei numeri si usa il segno -

Per moltiplicare dei numeri si usa il segno *

Per fare la divisione dei numeri si usa il segno /

Per elevare a potenza un numero si usa ** e ad esempio:

```
scrivi(5**5)
```

equivale a 5*5*5*5*5

MIN E MAX

Le istruzioni **min()** e **max()** permettono di estrarre rispettivamente il numero più piccolo e il numero più grande, di un elenco di numeri.

```
alfa=min(10,32,7,26,90,41)
```

```
beta=max(10,32,7,26,90,41)
```

```
scrivi(alfa)
```

```
scrivi(beta)
```

La prima riga assegna alla variabile *alfa* il numero più piccolo fra quelli elencati

La seconda riga assegna alla variabile *beta* il numero più grande fra quelli elencati

La terza e quarta riga scrivono sullo schermo il risultato.

POSITIVO

L'istruzione **positivo()** permette di scrivere il valore assoluto, positivo, di un numero.

alfa=-23

scrivi(alfa)

scrivi(positivo(alfa))

```
-23
23
Programma terminato
```

L'istruzione **potenza()** permette di calcolare un valore elevato a potenza, come nell'esempio sotto:

alfa=3

beta=2

scrivi(potenza(alfa,beta))

```
9
Programma terminato
```

PI GRECO

Per avere la numerazione esatta del valore del pi greco, si usa l'istruzione **pigreco**.

Esempio di calcolo della circonferenza di un cerchio, che come sappiamo si ottiene moltiplicando il valore del diametro per pi greco:

diametro=10

circonferenza=diametro*pigreco

scrivi("La circonferenza di un cerchio con diametro",diametro,"misura",circonferenza)

```
La circonferenza di un cerchio con diametro 10 misura 31.41592653589793
Programma terminato
```

NUMERO DI EULERO/NEPERO

Per usare il numero di Eulero/Nepero si usa l'istruzione **eulero** oppure **nepero**.

scrivi(eulero)

```
2.718281828459045
Programma terminato
```

GRADI E RADIANTI

Per convertire un angolo da gradi a radianti si usa l'istruzione **radianti**, mentre per convertire un angolo in radianti in gradi, si usa **gradi**.

scrivi(radianti(30))

scrivi(gradi(30))

```
0.5235987755982988
1718.8733853924696
Programma terminato
```

SENO, COSENO E TANGENTE

Per ottenere il **seno**, **coseno** e **tangente** di un numero, si usano le parole appena evidenziate in grassetto.

```
scrivi(seno(30))
scrivi(coseno(30))
scrivi(tangente(30))
```

```
-0.9880316240928618
0.15425144988758405
-6.405331196646276
Programma terminato
```

LOGARITMO

Per ottenere il **logaritmo** di un numero, si usa l'istruzione appena evidenziata in grassetto.

```
scrivi(logaritmo(30))
```

```
3.4011973816621555
Programma terminato
```

RADICE QUADRATA

Per ottenere la radice quadrata di un numero si usa l'istruzione **radicequadrata**.

```
scrivi(radicequadrata(25))
```

```
5.0
Programma terminato
```

ARROTONDA PER ECCESSO E ARROTONDA PER DIFETTO

Per arrotondare un numero decimale al numero intero per eccesso, si usa l'istruzione **arrotondaper eccesso**, mentre per arrotondare un numero decimale al numero intero per difetto, si usa l'istruzione **arrotondaper difetto**.

```
scrivi(arrotondaper eccesso(7.2))
```

```
scrivi(arrotondaper difetto(7.8))
```

```
8
7
Programma terminato
```

MEDIA DI NUMERI

Per calcolare la media tra diversi numeri di una lista, si usa l'istruzione **media**.

```
alfa=[5,7,6,9,6,9,6]
```

```
scrivi(media(alfa))
```

```
6.857142857142857
Programma terminato
```

NUMERO RICORRENTE

Per ottenere il numero più usato in una lista di numeri, si usa l'istruzione **numeroricorrente**.

```
alfa=[5,7,6,9,6,9,6]
```

```
scrivi(numeroricorrente(alfa))
```

```
6
Programma terminato
```

Come risultato è stato ottenuto il numero 6, perché è il numero che viene ripetuto più frequentemente nella lista **alfa**.

DEVIATION STANDARD

Per calcolare la deviazione standard (usata in statistica) di una popolazione di numeri, si usa l'istruzione **deviazionestandard**.

```
alfa=[5,7,6,9,6,9,6]
scrivi(deviazionestandard(alfa))
1.5735915849388864
Programma terminato
```

CASUALE

Per generare un numero casuale, decimale, tra 0 e 1, si usa l'istruzione **casuale**.

```
scrivi(casuale())
0.1744099166676003
Programma terminato
```

NUMEROCASUALE

Per generare un numero casuale tra un minimo ed un massimo, si usa l'istruzione **numerocasuale**.

```
scrivi(numerocasuale(1,10))
6
Programma terminato
```

L'esempio sopra genera un numero casuale tra 1 e 10.

SCELTACASUALE

L'istruzione **sceltacasuale** permette di eseguire la scelta tra elementi di una lista, o tra caratteri di una stringa.

```
alfa=[3,21,7,45,34,12,9]
beta=["marco","elisa","giulio","marcella","franco"]
gamma="NiktorTheNat"
scrivi(sceltacasuale(alfa))
scrivi(sceltacasuale(beta))
scrivi(sceltacasuale(gamma))
9
marcella
a
Programma terminato
```

MISCHIA

L'istruzione **mischia**, permette di mischiare una lista di elementi numerici o stringa.

```
alfa=[3,21,7,45,34,12,9]
scrivi(alfa)
[3, 21, 7, 45, 34, 12, 9]
mischia(alfa)
scrivi(alfa)
[3, 9, 21, 7, 34, 45, 12]
Programma terminato
```

Nell'esempio sopra, prima vengono scritti i numeri della lista così come inseriti nella lista, e poi vengono scritti gli stessi numeri, ma mischiati tra loro.

GRAFICA A FINESTRE

La grafica a finestre, si intende l'uso della grafica con finestre tipo Windows.

Per usare le finestre grafiche, si deve prima di tutto creare un oggetto finestra, che conterrà i vari componenti, come pulsanti, caselle di testo, etichette, ecc.. Per creare una finestra, si inserisce nel nome di un oggetto a piacere, l'istruzione **finestra()**

```
alfa=finestra()
```

se ad esempio vogliamo inserire una etichetta, cioè un testo nella finestra, usiamo un altro nome di oggetto a piacere, a cui assegniamo l'istruzione **etichetta**, dove dentro dobbiamo scrivere a quale finestra è associata quella etichetta (nel nostro caso alla finestra che ha oggetto *alfa*) e poi possiamo aggiungere, separate da virgola, le varie proprietà utilizzabili con quel determinato componente. Ad esempio le etichette hanno la proprietà **testo** che permette di specificare un testo da scrivere all'interno dell'etichetta.

```
X=etichetta(alfa,testo="ciao a tutti")
```

Per poter poi mostrare l'etichetta sullo schermo, dobbiamo impacchettare il componente con l'istruzione **impacchetta()**

```
x.impacchetta()
```

infine, come ultima istruzione, dobbiamo dire al programma che la finestra deve rimanere attiva sullo schermo fino ad eventuale nuova istruzione, perché altrimenti la finestra scomparirebbe subito dopo essere stata disegnata. L'istruzione per far rimanere la finestra sullo schermo è **ciclico()** che determina appunto che la visualizzazione della finestra rimane attiva in un ciclo.

```
alfa.ciclico()
```

Riepilogando, per creare una finestra con una etichetta con testo "ciao a tutti" si scrivono queste istruzioni:

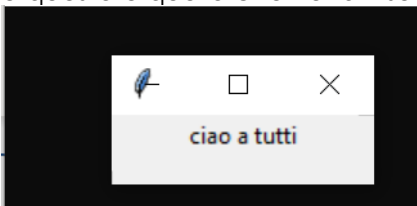
```
alfa=finestra()
```

```
x=etichetta(alfa,testo="ciao a tutti")
```

```
x.impacchetta()
```

```
alfa.ciclico()
```

e questo è quello che verrà visualizzato sullo schermo:



Se vogliamo aggiungere un'altra etichetta con del testo, sarà sufficiente aggiungere un nuovo oggetto etichetta, con il suo testo, ed impacchettarlo sullo schermo:

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="ciao a tutti")
```

```
x.impacchetta()
```

```
y=etichetta(alfa,testo="sono NiktorTheNat")
```

```
y.impacchetta()
```

```
alfa.ciclico()
```

L'istruzione **impacchetta**, può essere inserita anche sulla stessa riga del componente grafico da inserire, in questo modo:

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="ciao a tutti").impacchetta()
```

```
y=etichetta(alfa,testo="sono NiktorTheNat").impacchetta()
```

```
alfa.ciclico()
```

N.B.: In alcune istruzioni sono supportate diverse proprietà. Si possono inserire più proprietà, separate da una virgola.

Proprietà testo

Come visto nell'esempio precedente, nella **Label** è possibile inserire la proprietà **testo** che permette di scrivere del testo dentro l'etichetta.

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="ciao a tutti").impacchetta()
```

```
alfa.ciclico()
```

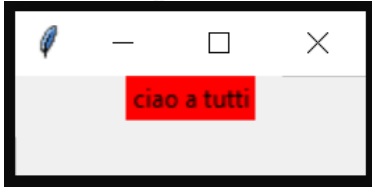
Proprietà **coloresfondo**

La proprietà **coloresfondo** permette di colorare lo sfondo del testo. Il colore va scritto senza virgolette ma anticipato da segno punto esclamativo. Esempio: **coloresfondo=!giallo**.

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="ciao a tutti",coloresfondo=!rosso).impacchetta()
```

```
alfa.ciclico()
```



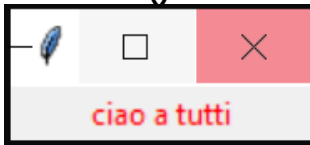
Proprietà **coloretesto**

La proprietà **coloretesto** permette di impostare il colore del testo. Il colore va messo senza virgolette ma anticipato da punto esclamativo. Esempio: **coloretesto=!giallo**

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="ciao a tutti",coloretesto=!rosso).impacchetta()
```

```
alfa.ciclico()
```



Proprietà **margin**

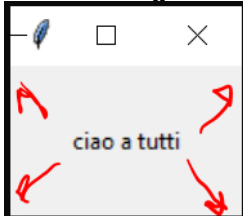
La proprietà **margin** permette di spaziare il componente da altri componenti e/o bordo della finestra. Il numero che verrà impostato per il bordo viene considerato per tutti e 4 i lati. Esempio:

```
margin=30
```

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="ciao a tutti",margin=30).impacchetta()
```

```
alfa.ciclico()
```



30 PIXEL

Altro esempio:

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="ciao a tutti",margin=30).impacchetta()
```

```
v=etichetta(alfa,testo="ciao a tutti",margin=90).impacchetta()
```

```
alfa.ciclico()
```



30 PIXEL

90 PIXEL

Proprietà tipocarattere

La proprietà **tipocarattere** serve ad impostare il font del carattere ed eventualmente anche la sua dimensione. Il tipo di carattere va scritto in modo corretto, anche tutto minuscolo, e soprattutto, se composto da più parole, va scritto tutto attaccato. Infine se si vuole impostare anche la dimensione del carattere, va inserito un numero che indica la dimensione del carattere. **Esempio1:**

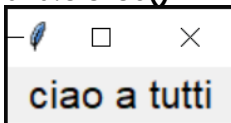
tipocarattere="courier" – Esempio2: tipocarattere="timesnewroman" – Esempio3:

tipocarattere="courier 8"

alfa=finestra()

x=etichetta(alfa,testo="ciao a tutti",tipocarattere="timesnewroman 16").impacchetta()

alfa.ciclico()



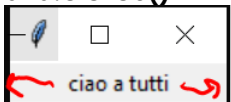
Proprietà spziox e spazioy

Le proprietà **spziox** e **spazioy** permettono di impostare solo una spaziatura su asse orizzontale con **spziox** oppure su asse verticale con **spazioy**. Può essere considerato come la proprietà **margin**, che però impostava la distanza da bordo o altri componenti sia su asse x che y.

alfa=finestra()

x=etichetta(alfa,testo="ciao a tutti",spziox=20).impacchetta()

alfa.ciclico()

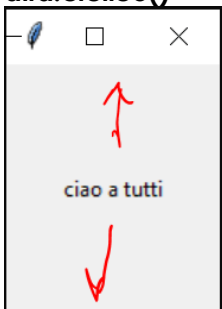


PIXEL 20

alfa=finestra()

x=etichetta(alfa,testo="ciao a tutti",spazioy=60).impacchetta()

alfa.ciclico()

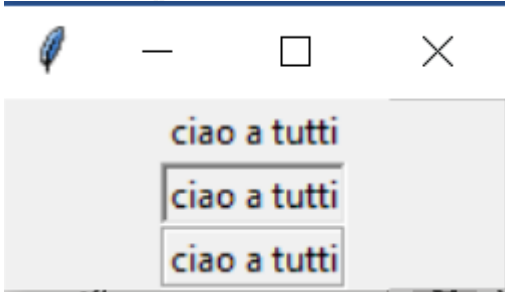


60 PIXEL

Proprietà rilievo

La proprietà **rilievo** permette di impostare che tipo di rilievo deve avere il componente. Ci sono solo tre proprietà impostabili che sono: **piatto**, **incassato**, **bordato**.

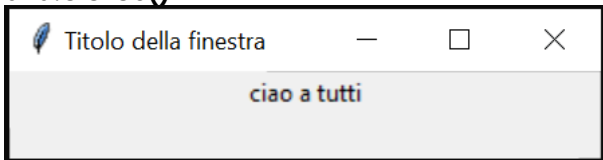
```
alfa=finestra()  
x=etichetta(alfa,testo="ciao a tutti",rilievo=piatto).impacchetta()  
y=etichetta(alfa,testo="ciao a tutti",rilievo=incassato).impacchetta()  
z=etichetta(alfa,testo="ciao a tutti",rilievo=bordato).impacchetta()  
alfa.ciclico()
```



Proprietà titolo

La proprietà **titolo** si usa con la finestra, per scrivere un testo nella barra del titolo.

```
alfa=finestra()  
alfa.titolo("Titolo della finestra")  
x=etichetta(alfa,testo="ciao a tutti").impacchetta()  
alfa.ciclico()
```



Proprietà grandezza

La proprietà **grandezza** si usa con la finestra, e serve a determinare la dimensione di apertura della finestra, usando la sintassi larghezzaXaltezza. Se vogliamo realizzare una finestra larga 300 pixel e alta 500 pixel useremo **grandezza("300x500")**

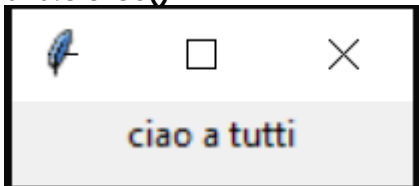
```
alfa=finestra()  
alfa.grandezza("300x500")  
x=etichetta(alfa,testo="ciao a tutti").impacchetta()  
alfa.ciclico()
```

I COMPONENTI

ETICHETTA

L'**etichetta** permette di inserire del testo nella finestra.

```
alfa=finestra()  
x=etichetta(alfa,testo="ciao a tutti").impacchetta()  
alfa.ciclico()
```

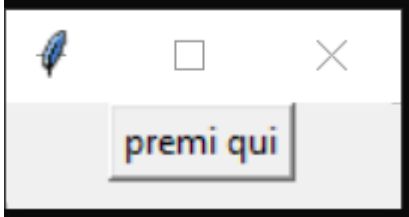


PULSANTE

Il **pulsante** inserisce un pulsante nella finestra.

```
alfa=finestra()  
x=pulsante(alfa,testo="premi qui").impacchetta()
```

alfa.ciclico()



Per far eseguire qualcosa quando il pulsante viene premuto, **è IMPORTANTE ricordare** che l'impacchettamento dei componenti da modificare, va fatto su una riga a parte e non sulla stessa riga. Inoltre bisogna usare la proprietà **esegui** per chiamare una funzione che deve essere eseguita quando si preme sul pulsante. La proprietà **configura** serve invece per modificare le proprietà di un componente che era già stato creato.

funzione premuto():

```
x.configura(testo="mi hai premuto")
```

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="non ancora premuto")
```

```
x.impacchetta()
```

```
y=pulsante(alfa,testo="premi qui",esegui=premuta).impacchetta()
```

```
alfa.ciclico()
```

ISTRUZIONI CORRETTE	ISTRUZIONI INCORRETTE
da grafica importa * funzione premuto(): x.configura(testo="mi hai premuto") alfa=finestra() x=etichetta(alfa,testo="non ancora premuto") x.impacchetta() y=pulsante(alfa,testo="premi qui",esegui=premuta).impacchetta() alfa.ciclico()	da grafica importa * funzione premuto(): x.configura(testo="mi hai premuto") alfa=finestra() x=etichetta(alfa,testo="non ancora premuto").impacchetta() y=pulsante(alfa,testo="premi qui",esegui=premuta).impacchetta() alfa.ciclico()

NO

Per prelevare il testo di una casella di testo, dopo che è stato premuto sul pulsante, si usa l'istruzione **prendi**, che prende il testo contenuto nella casella di testo. Anche in questo caso, ricordarsi di impacchettare il componente **caselladitesto** su un'altra riga e non nella stessa riga.

funzione premuto():

```
esempio=z.prendi()
```

```
x.configura(testo=esempio)
```

```
alfa=finestra()
```

```
x=etichetta(alfa,testo="Scrivi il tuo nome")
```

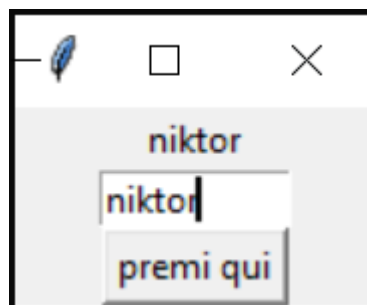
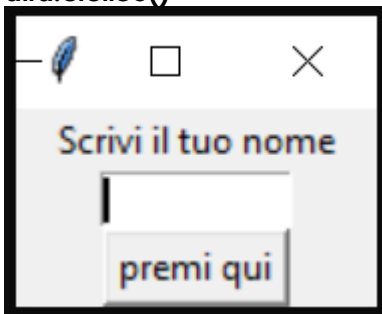
```
x.impacchetta()
```

```
z=caselladitesto(alfa,larghezza=10)
```

```
z.impacchetta()
```

```
y=pulsante(alfa,testo="premi qui",esegui=premuta).impacchetta()
```

```
alfa.ciclico()
```



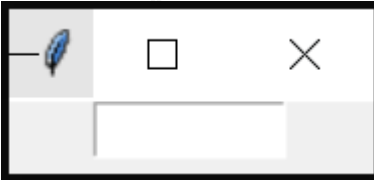
CASELLADITESTO

La **caselladitesto** crea una casella di testo dove poter inserire del testo o dei numeri. Per determinare la larghezza della casella di testo, si usa la proprietà **larghezza**.

```

alfa=finestra()
z=caselladitesto(alfa,larghezza=10).impacchetta()
alfa.ciclico()

```



Per vedere come ottenere ed elaborare il testo scritto dentro la **caselladitesto**, guarda le istruzioni sul componente **pulsante**.

Per forzare la scrittura nella **caselladitesto** si usa settare il focus al suo interno, che equivale a far apparire il cursore dentro la casella per permettere all'utente di scrivere. L'istruzione da proprietà da usare è **focus()**

```

alfa=finestra()
x=etichetta(alfa,testo="Scrivi il tuo nome")
x.impacchetta()
z=caselladitesto(alfa,larghezza=10)
z.impacchetta()
z.focus()
alfa.ciclico()

```

CASELLAADISCESA

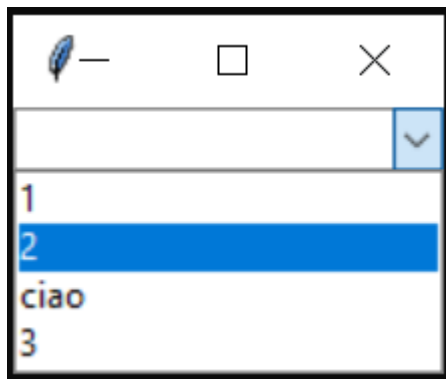
La **casellaadiscesa** serve a creare una casella con degli elementi che vengono mostrati come un elenco a discesa.

Per inserire i valori nella **casellaadiscesa** si usa la proprietà **valori** dove tra parentesi quadre, inseriamo i valori numeri o stringa (alfanumerici) da mostrare nell'elenco, separandoli da virgola.

```

alfa=finestra()
x=casellaadiscesa(alfa,valori=[1,2,"ciao",3]).impacchetta()
alfa.ciclico()

```



Per poter forzare la selezione di un elemento specifico dell'elenco, si usa la proprietà **selezionaelemento** indicando tra parentesi tonde quale elemento selezionare, ricordando che il conteggio deve partire da zero. **IMPORTANTE** è anche ricordare che l'uso della proprietà **selezionaelemento** va fatta prima di impacchettare l'istruzione **casellaadiscesa**.

Riprendendo l'esempio precedente, per selezionare l'elemento "ciao", dobbiamo indicare il numero 2.

```

alfa=finestra()
x=casellaadiscesa(alfa,valori=[1,2,"ciao",3])
x.selezionaelemento(2)
x.impacchetta()
alfa.ciclico()

```



Per ottenere l'elemento selezionato dalla **casellaadiscesa** si usa la proprietà **prendi**, ma va usato necessariamente un **pulsante** per poter elaborare la richiesta.

funzione controlla():

```
x.configure(testo=y.prendi())
alfa=finestra()
x=etichetta(alfa,testo="non selezionato")
x.impacchetta()
y=casellaadiscesa(alfa,valori=[1,2,"ciao",3])
y.impacchetta()
z=pulsante(alfa,testo="PREMI QUI",esegui=controlla).impacchetta()
alfa.ciclico()
```

Nell'esempio sopra viene creata un'**etichetta** con scritto "non selezionato", quindi una **casellaadiscesa** con i vari elementi ed un **pulsante** che eseguirà la funzione **controlla()** quando verrà premuto. Dopo che abbiamo selezionato l'elemento dalla **casellaadiscesa**, premiamo sul **pulsante** e lui farà eseguire la funzione **controlla()** che preleverà il testo dell'elemento selezionato con l'istruzione **y.prendi()**

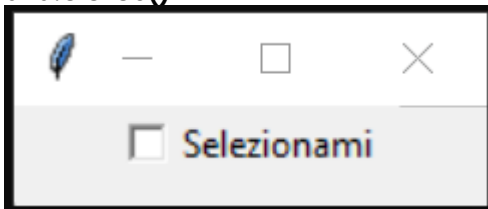
CASELLASELEZIONABILE

La **casellaselezionabile** crea un quadratino, dove possiamo mettere o togliere una spunta di selezione.

alfa=finestra()

x=casellaselezionabile(alfa,testo="Selezionami").impacchetta()

alfa.ciclico()



Per forzare la spunta sulla **casellaselezionabile**, bisogna prima dichiarare una variabile di tipo booleano, usando l'istruzione **variabilebooleana**, quindi bisogna impostare quella variabile su **vero**, infine bisogna impostare la variabile della **casellaselezionabile** sul valore della **variabilebooleana**.

alfa=finestra()

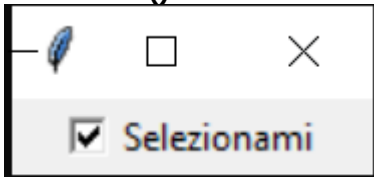
pp=variabilebooleana

pp.imposta(vero)

x=casellaselezionabile(alfa,testo="Selezionami",variabile=pp)

x.impacchetta()

alfa.ciclico()



Per ottenere l'informazione se la **casellaselezionabile** è flaggata o meno, si deve utilizzare un **pulsante** per avviare una funzione, dove bisogna leggere lo stato del flag nella variabile attraverso la proprietà **prendi()**.

funzione controlla():

```
x.configura(testo=pp.prendi())
```

```

alfa=finestra()
pp=variabilebooleana
pp.imposta(vero)
x=etichetta(alfa,testo="nessun controllo")
x.impacchetta()
y=casellaselezionabile(alfa,testo="Selezionami",variabile=pp)
y.impacchetta()
z=pulsante(alfa,testo="PREMI QUI",esegui=controlla).impacchetta()
alfa.ciclico()

```

Nell'esempio sopra, quando si preme sul **pulsante**, nell'**etichetta** verrà scritto **1** se la casella è flaggata, oppure **0** se la casella non è flaggata.

CASELLAASCELTA

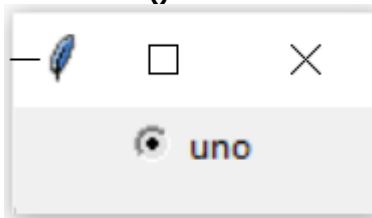
La **casellaascelta** si differenzia dalla **casellaselezionabile**, perché permette di flaggare una sola opzione, mentre la **casellaselezionabile** permette di flaggare più caselle.

L'istruzione della **casellaascelta** prevede l'uso della proprietà **valore**, per differenziare quale **casellaascelta** è stata flaggata.

```

alfa=finestra()
x=casellaascelta(alfa,testo="uno",valore=1)
x.impacchetta()
alfa.ciclico()

```

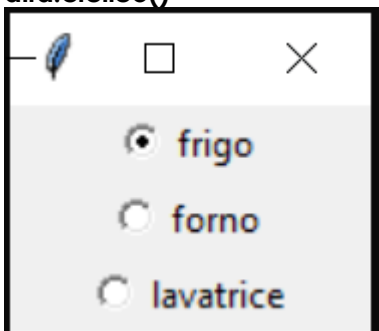


Se si devono inserire più scelte, vanno inserite più **casellaascelta**, modificando la proprietà **valore** per ognuno di loro, in modo poi da sapere quale casella è stata flaggata.

```

alfa=finestra()
x1=casellaascelta(alfa,testo="frigo",valore=1)
x1.impacchetta()
x2=casellaascelta(alfa,testo="forno",valore=2)
x2.impacchetta()
x3=casellaascelta(alfa,testo="lavatrice",valore=3)
x3.impacchetta()
alfa.ciclico()

```



Per verificare quale opzione è stata selezionata, si deve usare un **pulsante** per avviare una funzione, che elabora i dati. Inoltre bisogna creare una **variabileintera** che conterrà il **valore** assegnato alla **casellaascelta**.

```

funzione comando():
    x.configura(testo=pp.prendi())
alfa=finestra()
pp=variabileintera
x=etichetta(alfa,testo="scelta non selezionata")

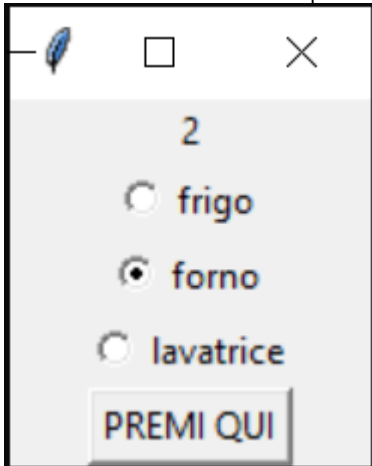
```

```

x.impacchetta()
x1=casellaascelta(alfa,testo="frigo",valore=1,variabile=pp)
x1.impacchetta()
x2=casellaascelta(alfa,testo="forno",valore=2,variabile=pp)
x2.impacchetta()
x3=casellaascelta(alfa,testo="lavatrice",valore=3,variabile=pp)
x3.impacchetta()
y=pulsante(alfa,testo="PREMI QUI",esegui=comando).impacchetta()
alfa.ciclico()

```

Quando premeremo su una opzione, e poi premiamo sul **pulsante**, allora verrà scritto nella etichetta il valore dell'opzione che è stata selezionata.



CASELLADITESTOSCROLLABILE

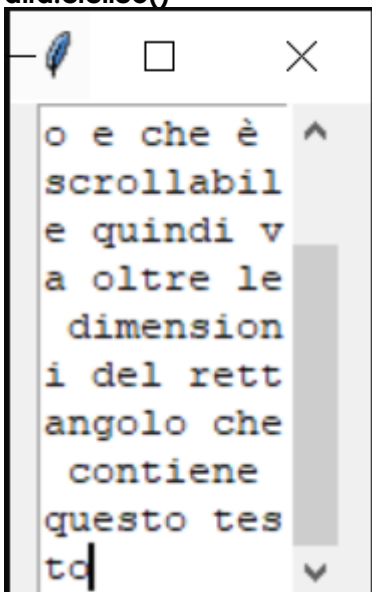
La **caselladitestoscrollabile** permette di creare una casella di testo, con specifiche dimensioni, che se contiene molto testo, sarà scrollabile con una barra laterale.

Per indicare le dimensioni della **caselladitestoscrollabile** bisogna usare le proprietà **larghezza** e **altezza**, che vanno considerate in unità di misura di caratteri e non in pixel, quindi **larghezza=10**, non equivale a 10 pixel di larghezza, ma a 10 caratteri di larghezza.

```

alfa=finestra()
x=caselladitestoscrollabile(alfa,larghezza=10,altezza=10)
x.impacchetta()
alfa.ciclico()

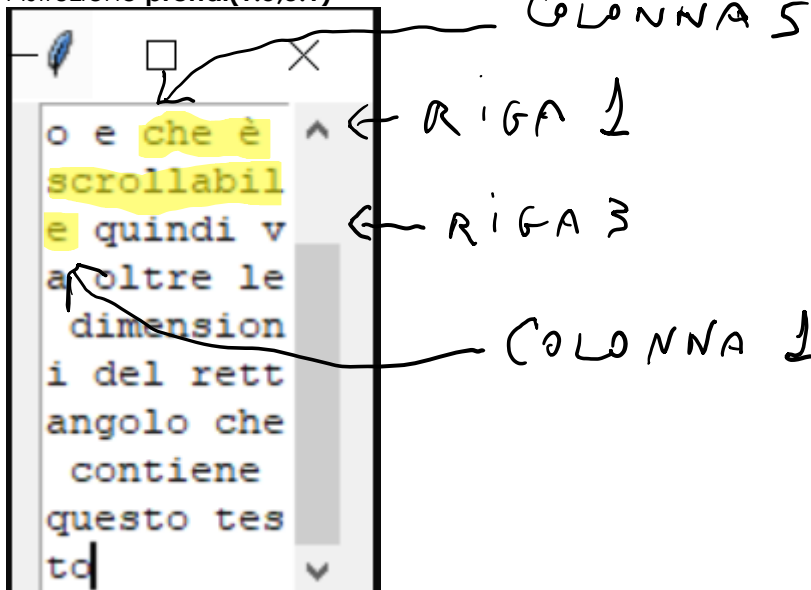
```



Per prelevare del testo dalla **caselladitestoscrollabile** bisogna usare un pulsante, e quando si preleva il testo, va messo dentro una variabile usando la proprietà **prendi()** dove tra parentesi va

scritto l'inizio della riga e colonna da prelevare, separato da virgola da riga e colonna di fine prelievo.

Nel caso dell'immagine sopra, se si vuole prelevare il testo "che è scrollabile", bisogna usare l'istruzione **prendi(1.5,3.1)**



funzione controlla():

esempio=**y.prendi(1.5,3.1)**

x.configura(testo=esempio)

alfa=finestra()

x=etichetta(alfa,testo="nessun testo")

x.impacchetta()

y=caselladitestoscrollabile(alfa,larghezza=10,altezza=10)

y.impacchetta()

z=pulsante(alfa,testo="PREMI QUI",esegui=controlla).impacchetta()

alfa.ciclico()

Per prelevare tutto il testo della **caselladitestoscrollabile** bisogna usare l'istruzione **prendi(1.0,FINE)**. Il numero **1.0** indica la riga 1 colonna 0, mentre **FINE** indica la fine del testo. IMPORTANTE che la parola FINE sia scritta tutta in maiuscolo.

funzione controlla():

esempio=**y.prendi(1.0,FINE)**

x.configura(testo=esempio)

alfa=finestra()

x=etichetta(alfa,testo="nessun testo")

x.impacchetta()

y=caselladitestoscrollabile(alfa,larghezza=10,altezza=10)

y.impacchetta()

z=pulsante(alfa,testo="PREMI QUI",esegui=controlla).impacchetta()

alfa.ciclico()

Per inserire del testo dentro la **caselladitestoscrollabile** si usa l'istruzione **inserisci**, indicando da quale riga/colonna iniziare a scrivere, seguito dal testo da scrivere.

alfa=finestra()

x=caselladitestoscrollabile(alfa,larghezza=10,altezza=10)

x.inserisci(1.0,"LAVATRICE")

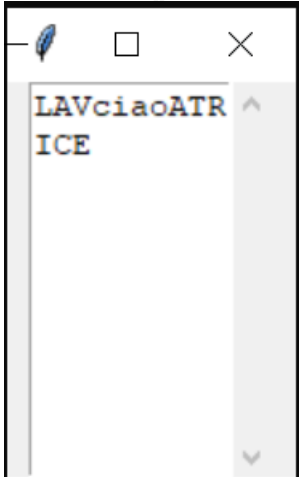
x.impacchetta()

alfa.ciclico()

Nell'esempio sopra, verrà scritta la parola "LAVATRICE", partendo dalla riga 1 colonna 0. Se non c'è altro testo dentro la **caselladitestoeditabile** allora qualsiasi riga/colonna scriviamo, il computer partirà a scrivere sempre dalla riga 1 colonna 0, mentre se c'è altro testo già scritto dentro, allora possiamo far inserire del testo nella riga/colonna che desideriamo.

Ecco un esempio in cui viene aggiunta una istruzione che inserisce del testo alla riga 1 colonna 3.

```
alfa=finestra()  
x=caselladitestoscrollabile(alfa,larghezza=10,altezza=10)  
x.inserisci(1.0,"LAVATRICE")  
x.inserisci(1.3,"ciao")  
x.impacchetta()  
alfa.ciclico()
```



Per eliminare del testo dalla **caselladitestoscrollabile** si usa lo stesso sistema di conteggio righe/colonne ma si usa l'istruzione **elimina**.

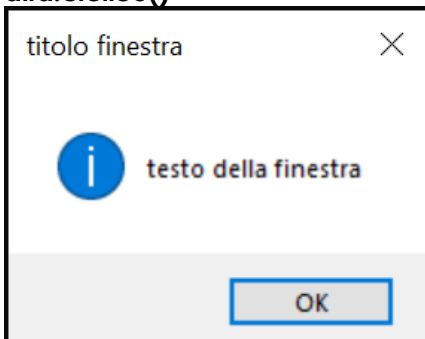
```
alfa=finestra()  
x=caselladitestoscrollabile(alfa,larghezza=10,altezza=10)  
x.inserisci(1.0,"LAVATRICE")  
x.inserisci(1.3,"ciao")  
x.elimina(1.3,1.7)  
x.impacchetta()  
alfa.ciclico()
```

Nell'esempio sopra, viene inserito il testo "ciao" che viene eliminato con l'istruzione **elimina**.

FINESTRE POPUP MESSAGGI

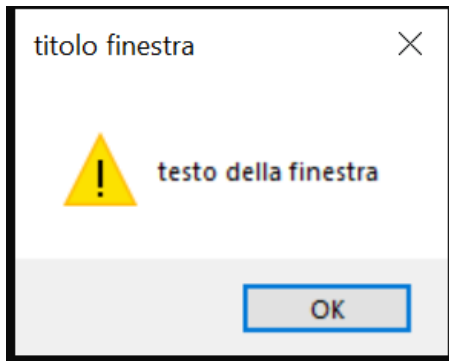
L'istruzione **finestramessaggio** permette di far visualizzare una finestra popup che accetta un titolo e un testo del messaggio. Deve essere usata come messaggio informativo per l'utente.

```
alfa=finestra()  
finestramessaggio("titolo finestra","testo della finestra")  
alfa.ciclico()
```



Ci sono altri tipi di finestre utilizzabili, come la **finestraattenzione**, che permette di mostrare un'icona di avviso.

```
alfa=finestra()  
finestraattenzione("titolo finestra","testo della finestra")  
alfa.ciclico()
```

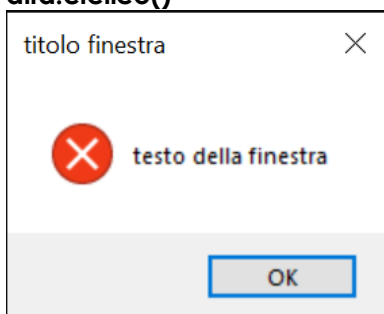



La **finestraerrore** permette di mostrare una finestra con icòna di avviso errore.

```
alfa=finestra()
```

```
finestraerrore("titolo finestra","testo della finestra")
```

```
alfa.ciclico()
```



Si possono creare anche finestre che permettono una scelta tramite pulsante, per cui ad esempio possiamo rispondere Sì, NO, o dare risposte di questo tipo. Queste finestre sono diverse e per diversi utilizzi. Vediamole.

La **finestradomanda** mostra un messaggio dove possiamo rispondere premendo SÌ oppure NO. La risposta possiamo poi metterla in una variabile per essere elaborata successivamente. Per verificare se è stato risposto SÌ oppure NO si usano le speciali istruzioni **_si_** oppure **_no_**

```
alfa=finestra()
```

```
xy=finestradomanda("titolo finestra","testo della finestra")
```

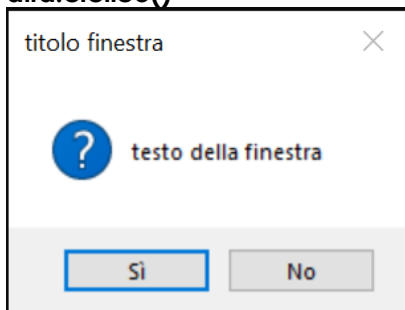
```
se xy==_si_:
```

```
    finestramessaggio("Risposta","Hai risposto sì")
```

```
altrimenti:
```

```
    finestramessaggio("Risposta","Hai risposto no")
```

```
alfa.ciclico()
```



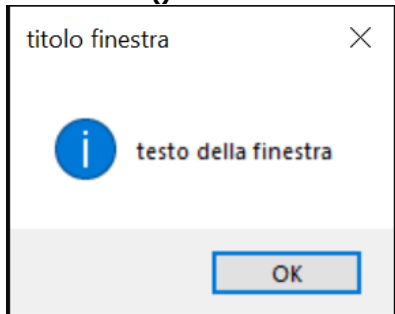
La **finestrasino** mostra un messaggio dove possiamo rispondere SÌ oppure NO, esattamente come avviene per la **finestradomanda** spiegata qui sopra, solo che cambia il tipo di icòna visualizzato e inoltre, in caso di risposta SÌ la variabile conterrà il numero **1** mentre se viene risposto NO la variabile conterrà il numero **0**

```
alfa=finestra()
```

```
xy=finestrasino("titolo finestra","testo della finestra")
```

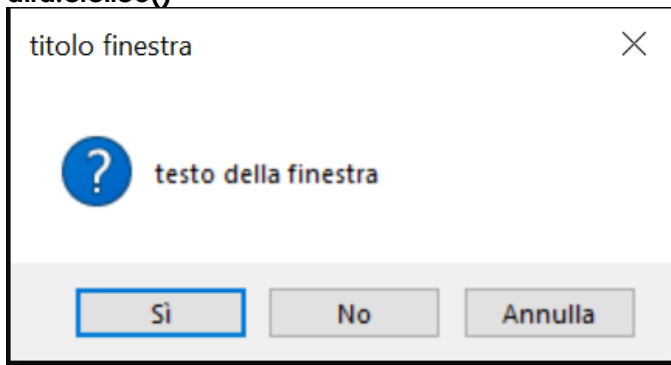
```
se xy==1:
```

```
finestramessaggio("Risposta","Hai risposto sì")
altrimenti:
    finestramessaggio("Risposta","Hai scritto no")
alfa.ciclico()
```



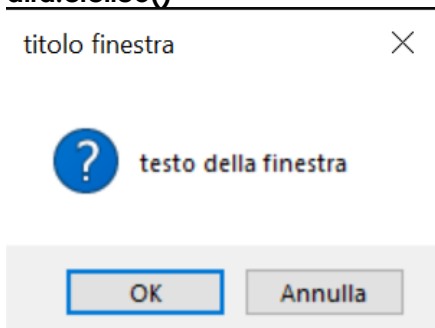
La **finestrasinoannulla** mostra una finestra come quelle precedenti, in cui vengono mostrati i pulsanti SI, NO e ANNULLA. In questo caso la variabile associata conterrà il numero **1** se viene risposto SI, mentre non conterrà nessun valore se viene risposto NO oppure ANNULLA.

```
alfa=finestra()
xy=finestrasinoannulla("titolo finestra","testo della finestra")
finestramessaggio("Risposta",xy)
alfa.ciclico()
```



La **finestraokannulla** mostra una finestra come quelle precedenti, in cui vengono mostrati i pulsanti OK e ANNULLA. Se viene premuto sul pulsante OK la variabile associata conterrà il numero **1**, mentre se viene premuto sul pulsante ANNULLA non viene generato alcun numero.

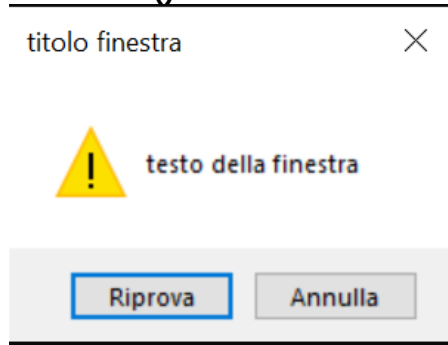
```
alfa=finestra()
xy=finestraokannulla("titolo finestra","testo della finestra")
finestramessaggio("Risposta",xy)
alfa.ciclico()
```



La **finestrariprovaannulla** mostra una finestra come quelle precedenti, in cui appariranno i pulsanti RIPROVA e ANNULLA. Se si preme sul pulsante RIPROVA la variabile associata conterrà il numero **1**, altrimenti premendo sul pulsante ANNULLA la variabile non conterrà alcun valore.

```
alfa=finestra()
xy=finestrariprovaannulla("titolo finestra","testo della finestra")
finestramessaggio("Risposta",xy)
```

alfa.ciclico()



CONTATORE

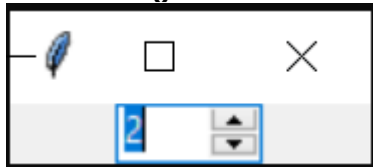
Il **contatore** è un componente che permette di inserire un contatore che ha un inizio dato dalla proprietà **dalnumero** ed un limite che arriva alla proprietà **alnumero**.

alfa = finestra()

x = contatore(alfa, dalnumero=0, alnumero=100, larghezza=5)

x.impacchetta()

alfa.ciclico()



Il contatore potrà essere selezionato tra il numero 0 e il numero 100.

Con la proprietà **imposta** è possibile impostare un numero preimpostato.

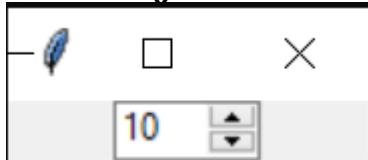
alfa = finestra()

x = contatore(alfa, dalnumero=0, alnumero=100, larghezza=5)

x.imposta(10)

x.impacchetta()

alfa.ciclico()



Per ottenere il numero che è stato selezionato, magari usando un **pulsante**, si usa l'istruzione **prendi()**, che nell'esempio sopra sarebbe **x.prendi()**

alfa = finestra()

x = contatore(alfa, dalnumero=0, alnumero=100, larghezza=5)

x.imposta(10)

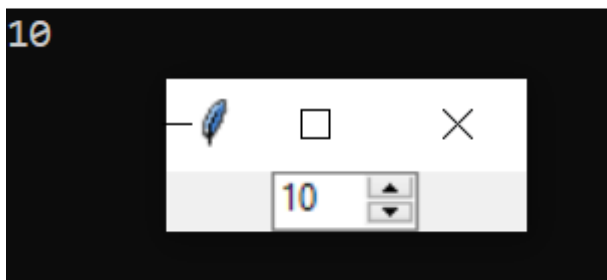
x.impacchetta()

print(x.prendi())

alfa.ciclico()

C:\Users\nikto\AppData\Local\Pr

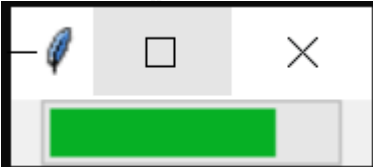
10



BARRA

La **barra** permette di realizzare una barra di avanzamento, in cui dobbiamo dare le proprietà **estensione** che imposta la larghezza della barra, e la proprietà **valore** che imposta il valore di avanzamento della barra.

```
alfa = finestra()  
x=barra(alfa,estensione=100,valore=80).impacchetta()  
alfa.ciclico()
```



Se aumentiamo il valore della proprietà **estensione**, avremo una barra con selettore uguale a prima, ma la larghezza della barra sarà maggiore.

```
alfa = finestra()  
x=barra(alfa,estensione=200,valore=80).impacchetta()  
alfa.ciclico()
```



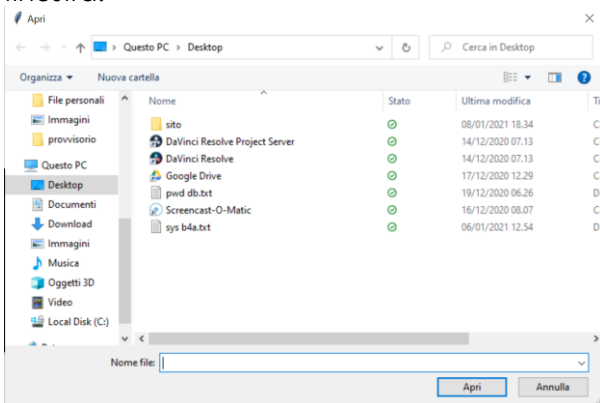
FINESTRAFILE

La **finestrafile** permette di aprire una finestra di dialogo che ci permette di selezionare un file dal nostro computer scegliendolo dalla gestione file.

```
nome = finestrafile()  
print(nome)
```

```
x=domanda("termine")
```

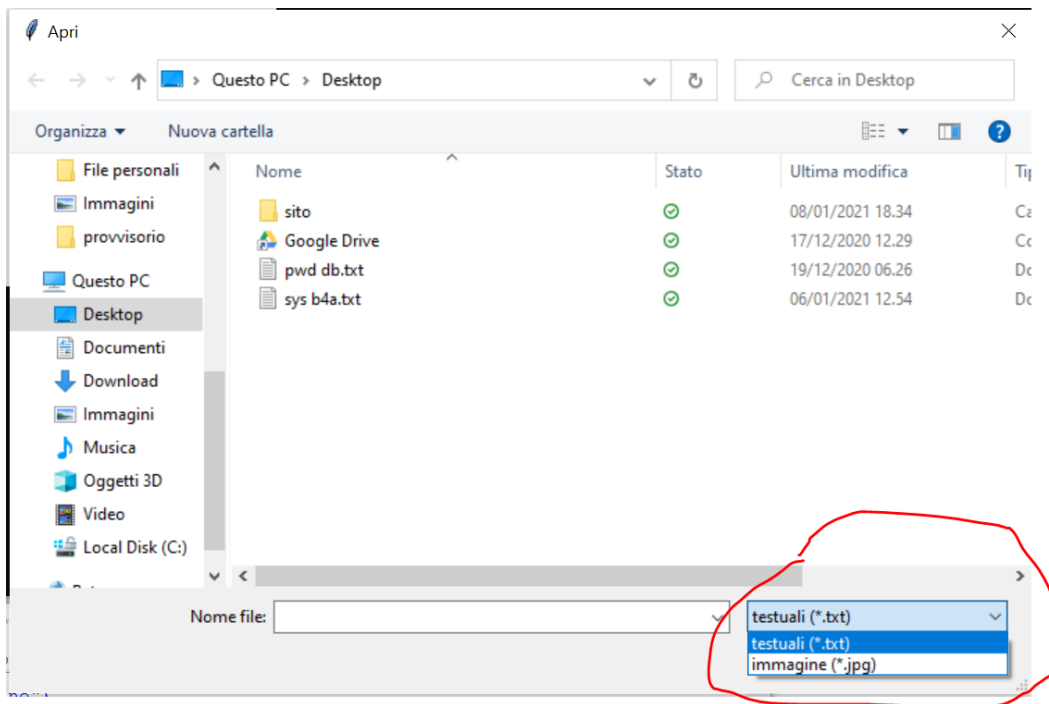
L'istruzione **print(nome)** scriverà sul terminale il percorso e nome del file appena selezionato dalla finestra.



```
C:\Users\nikto\AppData\Local\Programs\Python\Python39\p  
C:/Users/nikto/OneDrive/Desktop/sys b4a.txt  
termine
```

E' possibile usare la proprietà **tipofile** per filtrare solo i file con un determinato nome e/o estensione. Nell'esempio sotto vengono filtrati i file con estensione TXT o estensione JPG.

```
nome = finestrafile(tipofile=(("testuali","*.txt"),("immagine","*.jpg")))  
print(nome)  
x=domanda("termine")
```



FINESTRAFILES

La **finestrafiles** funziona esattamente come **finestrafile**, solo che si possono selezionare più files.

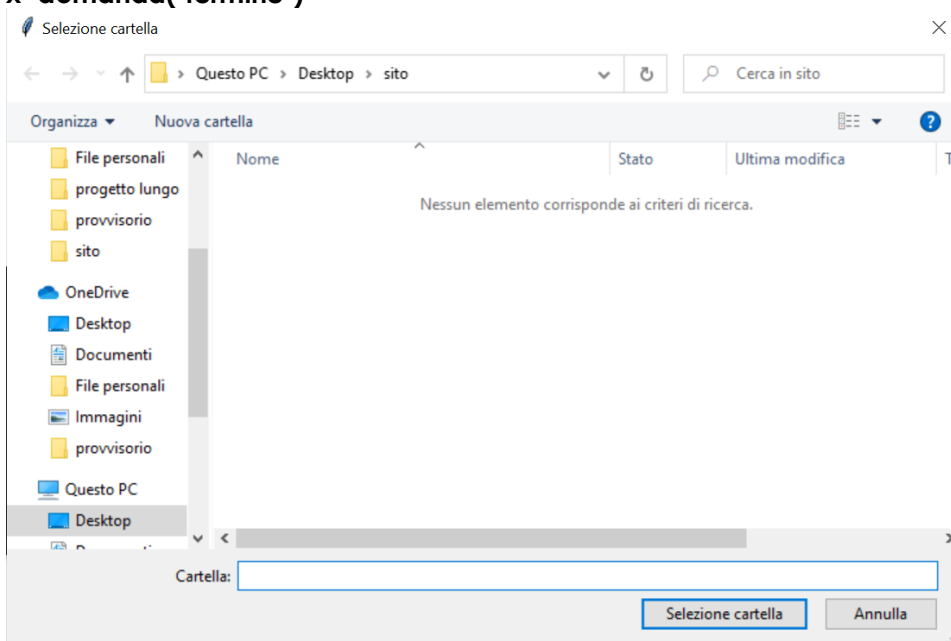
FINESTRACARTELLA

La **finestracartella** funziona come la **finestrafile** solo che permette di selezionare una cartella anziché un file.

cartella = finestracartella()

print(cartella)

x=domanda("termine")



GESTIONE MENU A DISCESA

La gestione dei menu di una finestra è particolarmente contorta, ma non difficile da apprendere.

Questo è un codice di esempio, in cui viene creato un menu principale **File** e due sottomenu **Nuovo** e **Modifica**.

```
alfa=finestra()
beta=menu(alfa)
gamma=menu(beta)
gamma.aggiungimensecondario(nomemenusecondario="Nuovo")
gamma.aggiungimensecondario(nomemenusecondario="Modifica")
beta.aggiungimenu(menuprimario="File",menusecondario=gamma)
alfa.configura(menufinestra=beta)
alfa.ciclico()
```

Prima di tutto si crea un contenitore di tutti i menu che intendiamo creare, e che abbiamo chiamato, ad esempio, *beta*, utilizzando l'istruzione **menu** e associandolo alla finestra *alfa*

```
beta=menu(alfa)
```

poi creiamo un contenitore di menu secondari (o se preferite "sotto menu") che abbiamo chiamato *gamma*, sempre con l'istruzione **menu**, che colleghiamo al contenitore principale *beta*

```
gamma=menu(beta)
```

quindi aggiungiamo al menu secondario (o se preferite "sotto menu") *gamma* i vari nomi di menu secondari, come ad esempio:

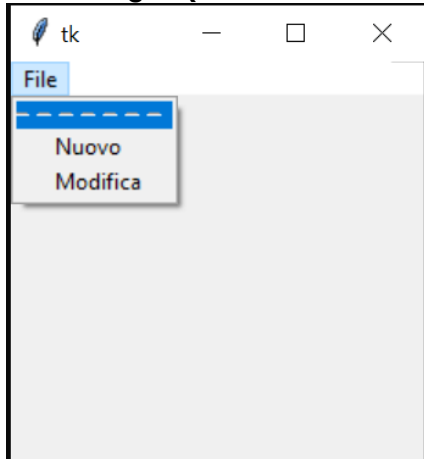
```
gamma.aggiungimensecondario(nomemenusecondario="Nuovo")
```

poi al contenitore principale *beta*, aggiungiamo un nome di menu primario, che nel caso abbiamo chiamato "File", al quale saranno collegati i menu secondari *gamma*

```
beta.aggiungimenu(menuprimario="File",menusecondario=gamma)
```

infine, con l'istruzione **configura**, aggiungiamo materialmente tutto il menu *beta* nella finestra *alfa*

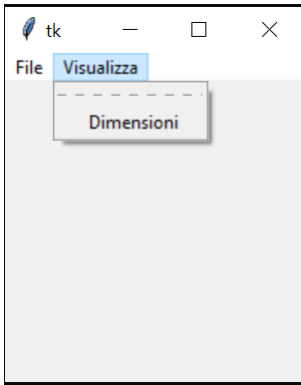
```
alfa.configura(menufinestra=beta)
```



Se intendiamo aggiungere un altro menu, che chiamiamo ad esempio "Visualizza", a cui è collegato, ad esempio, un solo menu secondario con scritto "Dimensioni", allora:

- aggiungeremo un nuovo contenitore per i menu secondari, che ad esempio chiamiamo *delta*,
- quindi al menu secondario *delta* aggiungiamo il nome di menu secondario "Dimensioni",
- ed infine aggiungiamo il menu secondario *delta* al contenitore di menu principale *beta*, con testo sul menu "Visualizza"

```
alfa=finestra()
beta=menu(alfa)
gamma=menu(beta)
delta=menu(beta)
gamma.aggiungimensecondario(nomemenusecondario="Nuovo")
gamma.aggiungimensecondario(nomemenusecondario="Modifica")
delta.aggiungimensecondario(nomemenusecondario="Dimensioni")
beta.aggiungimenu(menuprimario="File",menusecondario=gamma)
beta.aggiungimenu(menuprimario="Visualizza",menusecondario=delta)
alfa.configura(menufinestra=beta)
alfa.ciclico()
```



Per far eseguire delle istruzioni ai vari menu, si deve aggiungere l'istruzione **esegui** alla proprietà **aggiungimenusecondario**, dove indicheremo la funzione da eseguire nel caso il menu venisse premuto.

funzione istruzione1():

`scrivi("Hai premuto Nuovo")`

funzione istruzione2():

`scrivi("Hai premuto Modifica")`

`alfa=finestra()`

`beta=menu(alfa)`

`gamma=menu(beta)`

`gamma.aggiungimenusecondario(nomemenusecondario="Nuovo", esegui=istruzione1)`

`gamma.aggiungimenusecondario(nomemenusecondario="Modifica", esegui=istruzione2)`

`beta.aggiungimenu(menuprimario="File", menusecondario=gamma)`

`alfa.configura(menufinestra=beta)`

`alfa.ciclico()`

INTEGRAZIONE CON MICROSOFT WORD

Con **Lungo** è possibile interagire con i documenti di Microsoft Word.

Vediamo qui di seguito, passo dopo passo, come gestire un documento di Microsoft Word.

Per gestire un nuovo documento, bisogna usare l'istruzione **nuovodocumento**

`alfa=nuovodocumento()`

poi per aggiungere del testo nel documento, si usa **aggiungiparagrafo**:

`alfa=nuovodocumento()`

`paragrafo=alfa.aggiungiparagrafo("Questa è una frase")`

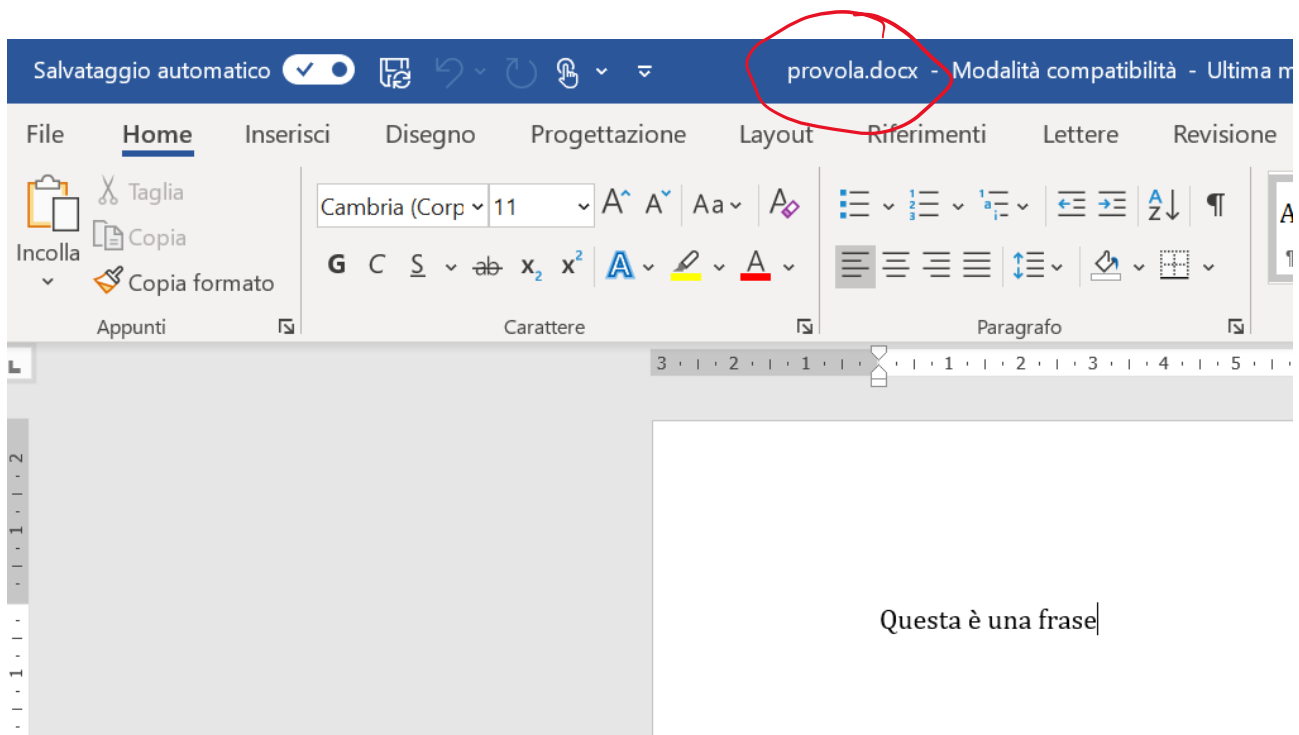
Infine per salvare il documento, si usa l'istruzione **salva**, specificando il nome che vogliamo dare al file.

`alfa=nuovodocumento()`

`paragrafo=alfa.aggiungiparagrafo("Questa è una frase")`

`alfa.salva("provola.docx")`

E questo è il documento che è stato creato:

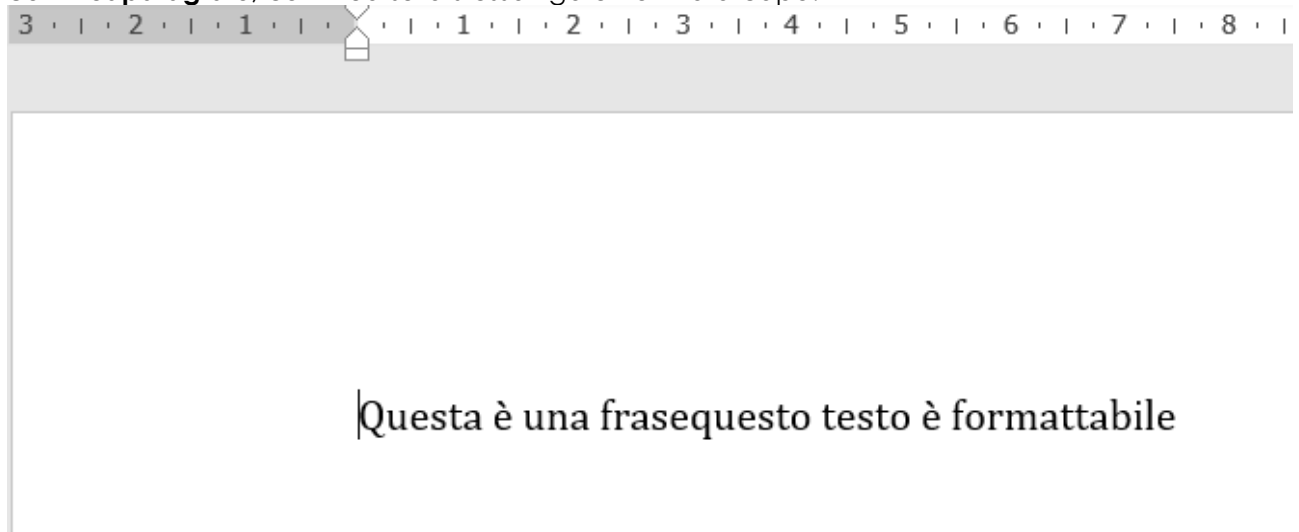


Se invece volevamo aggiungere del testo ad un documento esistente, invece di usare l'istruzione: **alfa=nuovodocumento()** dovevamo specificare il nome del file da modificare: **alfa=nuovodocumento("lavatrice.docx")**

Per cambiare la formattazione del testo da scrivere, si usa l'istruzione **continuaparagrafo**, specificando che tipo di formattazione vogliamo dare, mentre se vogliamo scrivere un nuovo paragrafo, useremo sempre l'istruzione **nuovoparagrafo**.

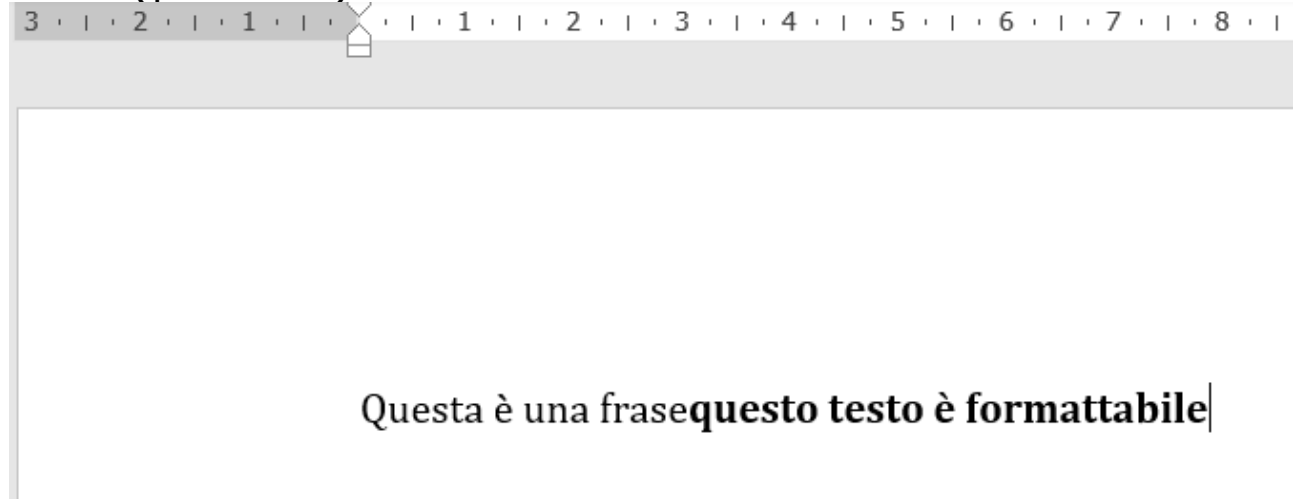
```
alfa=nuovodocumento()
paragrafo=alfa.aggiungiparagrafo("Questa è una frase")
paragrafo.continuaparagrafo("questo testo è formattabile")
alfa.salva("provola.docx")
```

Con il programma sopra, vengono scritti entrambi i testi, ma sulla stessa riga, perché l'istruzione **continuaparagrafo**, continua sulla stessa riga e non va a capo.



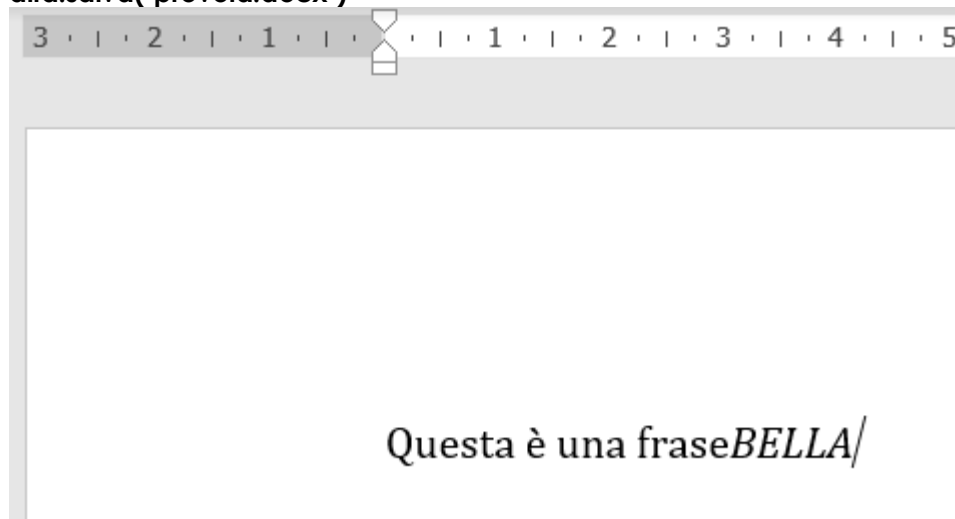
Ora, con l'istruzione **continuaparagrafo**, possiamo formattare il testo. Il grassetto, si ottiene usando l'istruzione **grassetto** impostata su **vero** o **falso**: **alfa=nuovodocumento()** **paragrafo=alfa.aggiungiparagrafo("Questa è una frase")** **paragrafo.continuaparagrafo("questo testo è formattabile").grassetto=vero**


```
alfa.salva("provola.docx")
```



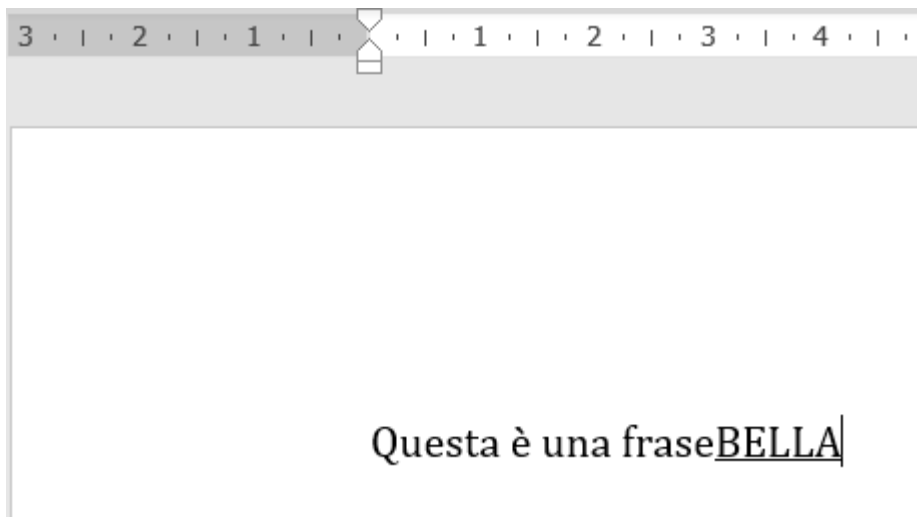
Il corsivo si ottiene con l'istruzione **corsivo** impostata su **vero** o **falso**

```
alfa=nuovodocumento()  
paragrafo=alfa.aggiungiparagrafo("Questa è una frase")  
paragrafo.continuaparagrafo("BELLA").corsivo=vero  
alfa.salva("provola.docx")
```



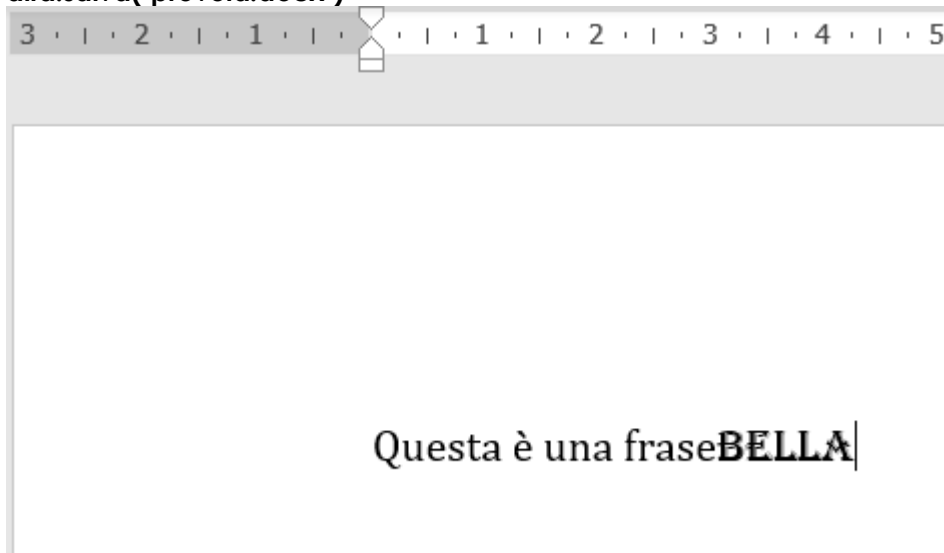
Il sottolineato si ottiene con l'istruzione **sottolineato** impostata su **vero** o su **falso**:

```
alfa=nuovodocumento()  
paragrafo=alfa.aggiungiparagrafo("Questa è una frase")  
paragrafo.continuaparagrafo("BELLA").sottolineato=vero  
alfa.salva("provola.docx")
```



Per cambiare il tipo di carattere di scrittura, si usa l'istruzione **nomecarattere**:

```
alfa=nuovodocumento()  
paragrafo=alfa.aggiungiparagrafo("Questa è una frase")  
paragrafo.continuaparagrafo("BELLA").nomecarattere="Algerian"  
alfa.salva("provola.docx")
```



INTEGRAZIONE CON MICROSOFT EXCEL

Con **Lungo** è possibile interagire con i file di Microsoft Excel.

Prima di tutto bisogna ricordare i termini tecnici di Microsoft Excel, che permetteranno di usare al meglio le istruzioni di **Lungo**.

Cella = ogni singolo rettangolo dove vengono scritti i dati

Foglio di lavoro = ogni singolo foglio, etichettato in basso al foglio, in cui scriviamo i dati

Cartella di lavoro = è il file di Microsoft Excel che contiene tutti i fogli di lavoro

Per creare una nuova cartella di lavoro (che equivarrebbe ad un nuovo file, ma in realtà inizialmente non viene creato alcun file), si usa l'istruzione **nuovacartella()** assegnata ad un oggetto, come ad esempio:

```
alfa=nuovacartella()
```

Per poter operare in scrittura o lettura sul foglio attivo (di norma una nuova cartella di lavoro genera un solo foglio che chiama "Foglio1" o "Sheet1"), si usa la proprietà **foglioattivo** che va associato ad un oggetto:

```
beta=alfa.foglioattivo
```

A questo punto è possibile scrivere i dati all'interno delle celle. Il modo più semplice per scrivere un dato dentro una cella è conoscere il nome della cella su cui scrivere, ad esempio A1, e poi assegnargli un valore o dato. Bisogna fare riferimento al foglio su cui si vuole operare.

Considerato che abbiamo realizzato un oggetto **beta**, che è relativo al foglio attivo, possiamo scrivere nella sua cella A1 in questo modo:

```
beta["a1"]=10
```

Se eseguiamo un programma che usa solo queste tre istruzioni, non verrà creato alcun documento di Microsoft Excel, perché è come aprire Microsoft Excel, scrivere un dato all'interno del foglio di lavoro e poi chiudere il file senza salvarlo, quindi è essenziale salvare il file con un nome.

L'istruzione per salvare il file è:

```
alfa.salva("nomefile.xlsx")
```

Ora la cartella di lavoro è stata salvata con il nome che abbiamo messo tra virgolette e parentesi tonde.

Ecco l'elenco completo delle istruzioni per creare una nuova cartella di lavoro, scrivendo all'interno della cella A1 il dato "ciao a tutti"

```
alfa=nuovacartella()
```

```
beta=alfa.foglioattivo
```

```
beta["a1"]="ciao a tutti"
```

```
alfa.salva("provola.xlsx")
```

Il programma qui sopra, crea una nuova cartella **alfa**, crea un oggetto **beta** che sarà il foglio di lavoro attivo, quindi nella cella A1 del foglio **beta**, inserisce il dato "ciao a tutti", ed infine salva il file con il nome **provola.xlsx**.

Per cambiare il nome del foglio di lavoro attivo, si usa la proprietà **titolo**:

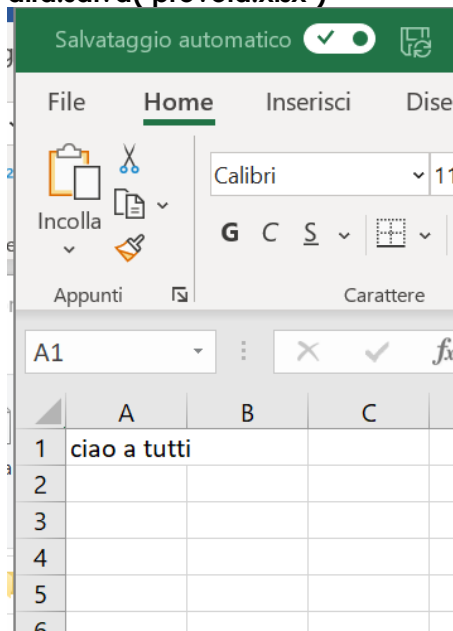
```
alfa=nuovacartella()
```

```
beta=alfa.foglioattivo
```

```
beta["a1"]="ciao a tutti"
```

```
beta.titolo="conto corrente"
```

```
alfa.salva("provola.xlsx")
```



Per scrivere all'interno di una cella, si può usare anche la proprietà **cella**, indicando tra parentesi la riga, la colonna ed il valore da assegnare a quella determinata cella.

```
alfa=nuovacartella()
```

```
beta=alfa.foglioattivo
```

```
beta.cella(riga=4,colonna=3,valore=76)
```

```
alfa.salva("provola.xlsx")
```

	A	B	C	D
1				
2				
3				
4			76	
5				
6				
7				

Per poter scrivere dati in un intervallo di celle, si possono usare le istruzioni cicliche, come ad esempio **considera...con**, come nell'esempio qui sotto in cui scriviamo nelle celle da A1 ad A10, la il dato "ciao"

alfa=nuovacartella()

beta=alfa.foglioattivo

considera x con intervallo(10):

beta.cella(riga=x+1,colonna=1,valore="ciao")

alfa.salva("provola.xlsx")

	A	B	C
1	ciao		
2	ciao		
3	ciao		
4	ciao		
5	ciao		
6	ciao		
7	ciao		
8	ciao		
9	ciao		
10	ciao		
11			

Per aprire una cartella di lavoro di Microsoft Excel che ad esempio si chiama provola.xlsx, si usa l'istruzione **apricartella**, mettendo tra parentesi tonde e virgolette il nome del file da aprire:

alfa=apricartella("provola.xlsx")

Il file provola.xlsx che sto usando per questo esempio è costituito così:

	A	B	C	D
1	MESI	ENTRATE	USCITE	
2	gennaio	300,00 €		
3	febbraio		45,00 €	
4	marzo	700,00 €		
5	aprile		22,00 €	
6		1.000,00 €	67,00 €	
7				
8				

conto corrente

Come si può notare, la cartella contiene solo un foglio di lavoro che si chiama **contro corrente**, mentre nelle varie celle ci sono dei dati.

Per leggere il nome di tutti i fogli della cartella, si usa la proprietà **nomefogli**, e nell'esempio sotto, apriamo il file e scriviamo su schermo il nome di tutti i fogli di lavoro della cartella provola.xlsx:

alfa=apricartella("provola.xlsx")

```
['conto corrente']  
Programma terminato.
```

Per leggere il contenuto delle celle di un foglio di lavoro già compilato, si usa il formato seguente, chiamando in causa la proprietà **valore**, qualora il foglio di lavoro è associato all'oggetto **beta**:

L'esempio sopra serve a leggere il valore della cella A1. Il programma completo è:

```
scrivi(beta["a1"].valore)
```

	A	B	C
1	MESI	ENTRATE	USCITE
2	gennaio	300,00 €	
3	febbraio		45,00 €
4	marzo	700,00 €	
5	aprile		22,00 €
6		1.000,00 €	67,00 €
7			
8			

←

→

conto corrente

+

```
MESI
Programma terminato.
```

Considerando il solito foglio di esempio:

	A	B	C	D
1	MESI	ENTRATE	USCITE	
2	gennaio	300,00 €		
3	febbraio		45,00 €	
4	marzo	700,00 €		
5	aprile		22,00 €	
6		1.000,00 €	67,00 €	
7				
8				

conto corrente

e volendo estrarre solo tutte le entrate e tutte le uscite con i loro totali, dovremo dire a **Lungo** di estrarre i dati dalla riga 2 colonna 2, fino alla riga 5 colonna 3:

5 2 3 4

	1	2	3	4
	A	B	C	D
1	MESI	ENTRATE	USCITE	
2	gennaio	300,00 €		
3	febbraio		45,00 €	
4	marzo	700,00 €		
5	aprile		22,00 €	
6		1.000,00 €	67,00 €	
7				
8				

conto corrente

scriveremo il seguente programma:

```
alfa=apricartella("provola.xlsx")
```

```
beta=alfa.foglioattivo
```

```
considera x con beta.righefoglio(rigainiziale=2,colonnainiziale=2,colonnafinale=3,rigafinale=5,solovalori=vero):
```

```
scrivi(x)
```

la proprietà **solovalori** serve a specificare se vogliamo estrarre i valori delle celle o visualizzare la denominazione delle celle. Mettendo **vero**, specifichiamo che vogliamo estrarre i valori delle celle.

Ecco il risultato del programma:

```
(300, None)
(None, 45)
(700, None)
(None, 22)
Programma terminato_
```

La parola **none** viene visualizzata quando la cella non contiene dati.

DATABASE

La gestione dei database con **Lungo**, è relativamente semplice, e permette di gestire database ospitati in locale o su server MySQL.

L'istruzione per connettersi ad un database è **connettidatabase**.

E' possibile quindi connettersi al database e ottenere un messaggio di connessione corretta con il seguente programma. Le informazioni che servono per connettersi al database sono l'**ospite** (cioè il server che fa da host al database), il nome **utente** (che in inglese è definito "user") e la **password** che avete dato al vostro database.

```
alfa=connettidatabase(ospite="miohost.net",utente="niktor",password="lavatrice%02%LAVATRICE")
```

```
scrivi(alfa)
```

```
<mysql.connector.connection.MySQLConnection object at 0x000001BD70FF1FD0>
Programma terminato
```

Per creare un nuovo database nello spazio server che abbiamo aperto sopra, dobbiamo usare una istruzione **gestore** che ci permette di gestire le comunicazioni con l'host, e poi usiamo l'istruzione **lancia**, per inviare qualsiasi comando MySQL all'host.

Ecco come creare un nuovo database che chiameremo *provola*

```
alfa=connettidatabase(ospite="miohost.net",utente="niktor",password="lavatrice%02%LAVATRICE")
```

```
pippo=alfa.gestore()
```

```
pippo.lancia("CREATE DATABASE provola")
```

Per vedere se ci sono database già registrati nel nostro host, e quindi ottenere il nome dei database, usiamo l'istruzione MySQL SHOW DATABASES:

```
alfa=connettidatabase(ospite="miohost.net",utente="niktor",password="lavatrice%02%LAVATRICE")
```

```
pippo=alfa.gestore()
pippo.lancia("SHOW DATABASES")
considera x con pippo:
    scrivi(x)
```

```
('information_schema',)
('lavatrice',)
Programma terminato
```

Per poter creare una tabella nel database **provola**, si lancia semplicemente l'istruzione MySql per creare la tabella che è CREATE TABLE, ma nell'istruzione **connettidatabase** va aggiunto il nome del database che vogliamo gestire:

```
alfa=connettidatabase(ospite="miohost.net",utente="niktor",password="lavatrice%02%LAVATRICE",database="provola")
pippo=alfa.gestore()
pippo.lancia("CREATE TABLE nomi (nome VARCHAR(100),cognome VARCHAR(100))")
```

Per vedere quali tabelle sono presenti in un determinato database, ad esempio nel database **provola**, si usa l'istruzione MySql SHOW TABLES:

```
alfa=connettidatabase(ospite="miohost.net",utente="niktor",password="lavatrice%02%LAVATRICE",database="provola")
pippo=alfa.gestore()
pippo.lancia("SHOW TABLES")
considera x con pippo:
    scrivi(x)
```

Per inserire dati all'interno di una tabella, ad esempio alla tabella **nomi**, si usa l'istruzione di MySql **INSERT INTO VALUES**, ed inoltre si deve usare l'istruzione **aggiornadatabase** perché i dati siano memorizzati all'interno.

```
alfa=connettidatabase(ospite="miohost.net",utente="niktor",password="lavatrice%02%LAVATRICE",database="provola")
pippo=alfa.gestore()
istruzione1="INSERT INTO nomi(nome,cognome) VALUES (%s,%s)"
istruzione2=("niktor","thenat")
pippo.lancia(istruzione1,istruzione2)
alfa.aggiornadatabase()
```

APRIAUDIO

Per far riprodurre un file audio, si usa semplicemente l'istruzione **apriaudio** indicando tra parentesi il nome del file audio da aprire.

```
apriaudio("esempio.mp3")
```

RICONOSCIMENTO VOCALE

Il riconoscimento vocale, è quella tecnica che permette di ascoltare una voce, un parlato, e il computer ne trascrive in testo quanto ascoltato.

Con **Lungo** è possibile leggere file audio in formato WAV, che verranno trascritti in testo.

Per prima cosa bisogna creare un oggetto collegato all'istruzione **riconoscimentovocale**

```
alfa=riconoscimentovocale()
```

Ora che **alfa** è l'oggetto di riconoscimento vocale, è possibile, ad esempio, creare una variabile in cui inserire il nome del file da leggere.

```
beta="esempio.wav"
```

Poi, usando obbligatoriamente l'istruzione **usa...come**, bisogna catturare il file audio con l'istruzione **fileaudio** e il tutto va assegnato ad una variabile a piacere, che nel caso specifico è **gamma**. Sotto l'istruzione **usa...come**, scritto in modo indentato (cioè rientrato dal margine), dobbiamo creare una variabile a piacere, dove far leggere la registrazione audio, attraverso l'istruzione **registrazione**, ed infine possiamo far scrivere il testo del parlato, lanciando l'audio della **registrazione** al motore di riconoscimento vocale di Google, attraverso l'istruzione **riconoscimentogoogle**, e specificando sia

l'audio da trascrivere, sia la lingua in cui è registrato l'audio e pertanto dovrà essere trascritto il testo.

usa fileaudio(beta) come gamma:

voce=alfa.registrazione(gamma)

scrivi(alfa.riconoscimentogoogle(voce,linguaggio="it-IT"))

Altri linguaggi supportati sono "en-EN" per l'inglese, "fr-FR" per il francese, "es-ES" per lo spagnolo.

SINTETIZZATORE VOCALE

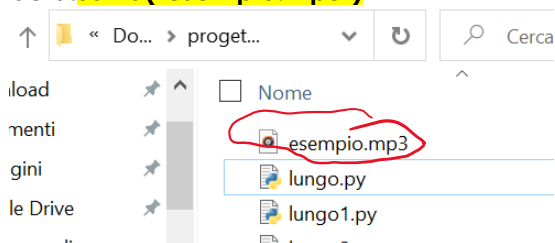
Il sintetizzatore vocale, è quella funzionalità che permette al computer di dire con una voce sintetizzata, ciò che legge da un testo scritto.

La lettura del testo, verrà registrata con la voce in un file a nostro piacere, come nell'esempio sotto. Le istruzioni fondamentali da usare sono l'istruzione **parla**, che serve a dire al computer quale testo vogliamo far dire con la voce, e la lingua con la quale cogliamo che sia pronunciata. Infine l'istruzione **salva**, permette di salvare il file, che nominiamo a piacere, ma con estensione **mp3**, che conterrà l'audio del testo parlato.

alfa="Questo è quello che voglio dire a voi tutti"

beta=parla(alfa,italiano)

beta.salva("esempio.mp3")



Lingue supportate e relativa istruzione di esempio:

- italiano
 - `parla(alfa,italiano)`
- inglese
 - `parla(alfa,inglese)`
- francese
 - `parla(alfa,francese)`

REGISTRAZIONE DA MICROFONO

Per poter registrare dell'audio dal microfono, si può usare l'istruzione **microfono**, indicando le proprietà **frequenza** di registrazione e **secondi** di apertura microfono per la registrazione.

Nell'esempio qui sotto, viene creata una variabile **fr** a cui assegniamo la frequenza di registrazione, quindi viene creata una variabile **sec** a cui assegnare i secondi di registrazione, poi creiamo un oggetto **alfa** che aprirà il **microfono**, e daremo come opzioni: la moltiplicazione tra secondi e frequenza, poi la frequenza, usando l'opzione **frequenza**, ed infine i canali di registrazione (si consiglia l'uso del numero 2) attraverso l'opzione **canali**.

Poi si deve usare l'istruzione **attendifineregistrazione** per attendere i secondi impostati di apertura microfono, ed infine si può memorizzare in un file WAV l'audio che abbiamo registrato, attraverso l'istruzione **memorizza**, a cui daremo, come proprietà, il nome del file da salvare, la frequenza, e l'oggetto di registrazione audio dal microfono.

fr=44100

sec=3

alfa=microfono(intero(sec*fr),frequenza=fr,canali=2)

attendifineregistrazione()

memorizza("esempio.wav",fr,alfa)

SCRAPING (ESTRAPOLA INFORMAZIONI DA PAGINA WEB)

Lo **scraping** è quell'attività di automazione, attraverso codice di programmazione, per leggere informazioni da pagine Web.

Con **Lungo** si possono estrarre informazioni da tag HTML, in modo relativamente semplice, a patto di avere le informazioni necessarie per farlo.

Prendiamo ad esempio questa pagina Web:

← → ↻ zooplus.it/shop/cani/cibo_secco_cani/royal_canin_size/1010017

CONSEGNA GRATUITA da 45€ | Solo 0,99 a partire da 35€*


Servizio clienti My zooplus ✓

zooplus Oltre 8.000 articoli in stock

Carrello

Cani Gatti Diete & Antiparassitari Piccoli animali Uccelli Acquaristica Cavalli Top brands

← Torna zooplus.it Cani | Alimenti secchi per cani | Royal Canin Size | 15 + 3 kg - 18 kg Royal Canin Size Puppy/Junior Ove...



15 + 3 kg - 18 kg Royal Canin Size Puppy/Junior Overfill


Offerta limitata: secco Royal Canin Size Puppy/Junior per cuccioli e cani giovani di taglie diverse, con i nutrienti essenziali per una crescita armonica, **in bonus bag 15 + 3 kg a prezzo speciale!** [leggi tutto >](#)

Componenti analitici

★★★★★ 224 valutazioni |

Happy Spotty : "Arrivato oggi!! Servizio eccellente non pensavo arrivasse prima dell'anno nuovo!! Per fortuna anche perchè stamattina era finito!!"

100 Foto di clienti



Consegna in 2-4 gg. lav. [Approfondisci >](#)
Tutti i prezzi IVA incl., Più info sulla [Spedizione](#)

Nella pagina ci sono numerose informazioni, come le immagini, il titolo del prodotto "15 + 3 kg – 18 kg Royal Canin ecc..." o la scritta sotto il titolo "Offerta limitata: secco Royal Canin ecc...", o ancora le valutazioni dei clienti "Happy Spotty: 'Arrivato oggi!!! Servizio eccellente ecc...". Tutte queste informazioni sono trascritte dentro il codice di programmazione HTML della pagina, e più precisamente dentro dei TAG.

Se ad esempio vogliamo estrarre le informazioni relative al titolo del prodotto, dobbiamo esaminare il codice HTML del titolo, e possiamo farlo attraverso il browser che stiamo utilizzando per navigare su Internet:

lus

Oltre 8.000 articoli in stock

Carrello

Diete & Antiparassitari Piccoli animali Uccelli Acquaristica Cavalli Top brands

Canil Alimenti secchi per cani Royal Canin Size 15 + 3 kg - 18 kg Royal Canin Size Puppy/Junior Overfill



15 + 3 kg - 18 kg Royal Canin Size Puppy/Junior Overfill

Offerta limitata: secco Royal Canin Size Puppy/Junior per cuccioli e cani giovani di taglie diverse, con i nutrienti essenziali per una crescita armonica, in **bonus bag 15 + 3 kg a prezzo speciale!** [leggi tutto >](#)

Componenti analitici

★★★★★ 224 valutazioni

Happy Spotty: "Arrivato oggi!! Servizio eccellente non pensavo arrivasse prima dell'anno nuovo!! Per fortuna anche perché stamattina era finito!!"

100 Foto di clienti

```

::before
▶<div class="col-sm-4 col-md-6">...</div>
▼<div class="col-sm-8 col-md-6">
  <div class="pd_title">
    <meta itemprop="name" content="15 + 3 kg - 18 kg Royal Canin
    unior Overfill">
    <meta itemprop="brand" content="Royal Canin Size">
    <h1 class="producttitle">
      15 + 3 kg - 18 kg Royal Canin Size Puppy/Junior Over
    </h1> == $0
  </div>
  <div class="product_description clearfix" data-zta="product
  </div>
  <div class="customer__section_wrapper clearfix">
    ::before
    <div class="product_rating_teaser" data-zta="productRati
    itemprop="aggregateRating" itemscope itemType="http://schem
    eRating">...</div>
  </div>
  <div id="div_product_comment" class="product_comment hide
  ... main_page div.row.pd_title div.col-sm-8.col-md-6 div.pd_title h1.product
  Styles Computed Layouts Event Listeners DOM Breakpoints Properties Acces
  Filter
  element.style {
  }
  .product_main_page .producttitle {
    margin-top: 0
  }
  @media screen and (min-width: 768px) {
    .h1, h1 {
      font-size: 32px;
    }
  }
  @media screen and (min-width: 544px) {
    .h1, .h2, .h3, .h4, .h5, .h6, .pd_reco_similar_v .slider-
    title, h1, h2, h3, h4, h5, h6 {
      margin-bottom: 15px;
    }
  }

```

Come si può notare nel codice di programmazione a destra, evidenziato in azzurro, il titolo del prodotto è contenuto nel tag **h1** che ha **class** "producttitle".

Per estrarre queste informazioni, programmiamo in questo modo:

Prima di tutto mettiamo in una ~~variabile~~ il link della pagina:

alfa=sito("https://www.zooplus.it/shop/cani/cibo_secco_cani/royal_canin_size/1010017")

poi mettiamo in un'altra variabile, l'elaborazione del testo della pagina, usando l'istruzione **elaborasito**, indicando la proprietà **testo** al link, e specificando la proprietà **paginaweb**:

beta=elaborasito(alfa.testo,paginaweb)

dato che il titolo del prodotto si trova nel tag **h1**, mettiamo in un'altra variabile, la ricerca di tutti i tag **h1**, e leggiamo il nome della **class** "producttitle" contenuta nel tag **h1**. Per leggere nome "producttitle" della **class**, dobbiamo usare la sintassi dei dizionari **{"class":"producttitle"}**

gamma=beta.cercatutti("h1",{"class":"producttitle"})

ora che **gamma** contiene tutti i tag **h1** con **class** "producttitle", eseguiamo un ciclo per far scrivere la proprietà **testo** di tutta la lista di tag **h1**. In questo esempio ce n'è solo uno e difatti viene estratto solo il titolo del prodotto:

considera provola con gamma:
scrivi(provola.testo)

Ecco il programma completo:

alfa=sito("https://www.zooplus.it/shop/cani/cibo_secco_cani/royal_canin_size/1010017")

beta=elaborasito(alfa.testo,paginaweb)

gamma=beta.cercatutti("h1",{"class":"producttitle"})

considera provola con gamma:
scrivi(provola.testo)

```

15 + 3 kg - 18 kg Royal Canin Size Puppy/Junior Overfill
Programma terminato

```

IMPORTANTE: lo **scraping** di pagine web, equivale alla consultazione automatizzata del sito, che viene offerto, a noi utenti, per essere consultato e non per essere oggetto di lettura automatizzata con download di informazioni a danno del sito che le propone. Alcuni siti, inoltre, hanno informazioni coperte da copyright, e per questo motivo va fatta attenzione sull'uso di questa particolare tecnica di estrapolazione informazioni.

Ipotizziamo di avere questa pagina di eBay dove abbiamo cercato la parola "scarpe", e il risultato del browser è questo:

← → ↻ [ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc](https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc)

Abbigliamento e accessori

- Uomo
- Donna
- Occasioni speciali
- Bambini
- Neonati
- Sport e viaggi
- Auto e moto: ricambi e accessori
- Collezionismo
- Fumetti e memorabilia
- Mostra di più ▾

Tema

- ☐ Colorato (26.876)
- ☐ Classico (23.646)
- ☐ Sports (15.341)
- ☐ Retro (10.866)
- ☐ USA (7.266)
- [Vedi tutti](#)

Su misura

- ☐ Sì (2.606)
- ☐ No (16.796)
- ☐ Non specificato (913.581)
- [Vedi tutti](#)

Stile

- ☐ Sneaker (146.667)
- ☐ Piatto (1.397)
- ☐ Platform (1.095)
- ☐ Oxford (1.035)
- ☐ Mocassino (884)
- [Vedi tutti](#)

Compra in base al prezzo

Inferiore a EUR 10,00

da EUR 10,00 a EUR 41,00

Superiore a EUR 41,00

A destra, sono elencati tutti i prodotti "scarpe" presenti su eBay.

Il link a questa ricerca, copiato dalla barra degli indirizzi, è

https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc

(è chiaro che se provate a collegarvi a questo link quando leggerete questo manuale, le informazioni potranno essere diverse, perché i prodotti cambiano continuamente)

Se vogliamo ottenere tutto il codice di programmazione con cui è stata realizzata la pagina, usiamo l'istruzione **sito** in questo modo:

alfa=sito("https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc")

Ora **alfa** contiene tutto il codice di programmazione. Se vogliamo stampare tutto il codice, dobbiamo stampare la sua proprietà **contenuto**:

scrivi(alfa.contenuto)

Quindi le istruzioni complete sono:

alfa=sito("https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc")

scrivi(alfa.contenuto)

e questo è il risultato:


```

nt","renderBody"],"r":{"type":"NOOP"}},{1":["w",311,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",311,2,"con
tent","renderBody"],"r":{"type":"NOOP"}},{1":["w",312,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",312,2,"c
ontent","renderBody"],"r":{"type":"NOOP"}},{1":["w",313,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",313,2,
"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",314,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",314,
2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",315,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",31
5,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",321,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",
321,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",322,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",
322,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",323,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",
323,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",324,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",
324,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",325,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1["
w",325,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",326,2,"trigger","renderBody"],"r":{"type":"NOOP"}},
{"1":["w",326,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",327,2,"trigger","renderBody"],"r":{"type":"NOOP"}},
{"1":["w",327,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",328,2,"trigger","renderBody"],"r":{"type":"NOOP"}},
{"1":["w",328,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",329,2,"trigger","renderBody"],"r":{"type":"NO
OP"}},{1":["w",329,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",330,2,"trigger","renderBody"],"r":{"type":"
NOOP"}},{1":["w",330,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",331,2,"trigger","renderBody"],"r":{"type":
"NOOP"}},{1":["w",331,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",337,2,"trigger","renderBody"],"r":{"typ
e":"NOOP"}},{1":["w",337,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",338,2,"trigger","renderBody"],"r":{"t
ype":"NOOP"}},{1":["w",338,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",339,2,"trigger","renderBody"],"r":{"
type":"NOOP"}},{1":["w",339,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",1,"renderBody"],"r":{
"type":"NOOP"}},{1":["w",341,2,"items",2,"renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",3,"renderBody"],"r
":{"type":"NOOP"}},{1":["w",341,2,"items",4,"renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",5,"renderBody"]
,"r":{"type":"NOOP"}},{1":["w",341,2,"items",6,"renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",7,"renderBody
"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",8,"renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",9,"renderBo
dy"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",10,"renderBody"],"r":{"type":"NOOP"}},{1":["w",342,2,"items",0,"rende
rBody"],"r":{"type":"NOOP"}},{1":["w",342,2,"items",1,"renderBody"],"r":{"type":"NOOP"}},{1":["w",342,2,"items",2,"ren
derBody"],"r":{"type":"NOOP"}},{1":["w",357,3,"w"],"r":["w",357,2]]]}</script></body></html><!-- RcmdId srpnweb,RlogId
t6pwwit%60d%3D9iptpwwit%60d*as0es(rbpv6762-17763a9f9b1-0x1905 --><!-- SiteId: 101, Environment: production, AppName: sr
pnweb, PageId: 2334524 -->
Programma terminato

```

Per ottenere il codice HTML del codice estratto sopra, dobbiamo usare l'istruzione **elaborasito**, utilizzando l'opzione **paginahtml**:

alfa=sito("https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&t=nc")

beta=elaborasito(alfa.contenuto,paginahtml)

scrivi(beta.mostrahtml())


e questo è il risultato:

```

ent","renderBody"],"r":{"type":"NOOP"}},{1":["w",315,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",315,2,"co
ntent","renderBody"],"r":{"type":"NOOP"}},{1":["w",321,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",321,2,"c
ontent","renderBody"],"r":{"type":"NOOP"}},{1":["w",322,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",322,2
,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",323,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",323
,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",324,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",3
24,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",325,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1":["w",
325,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",326,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1["
w",326,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",327,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1["
w",327,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",328,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1["
w",328,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",329,2,"trigger","renderBody"],"r":{"type":"NOOP"}},{1["
w",329,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",330,2,"trigger","renderBody"],"r":{"type":"NOOP"}},
{"1":["w",330,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",331,2,"trigger","renderBody"],"r":{"type":"NOOP"}},
{"1":["w",331,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",337,2,"trigger","renderBody"],"r":{"type":"NOO
P"}},{1":["w",337,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",338,2,"trigger","renderBody"],"r":{"type":"N
OOP"}},{1":["w",338,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",339,2,"trigger","renderBody"],"r":{"type":
"NOOP"}},{1":["w",339,2,"content","renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",1,"renderBody"],"r":{"type
":"NOOP"}},{1":["w",341,2,"items",2,"renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",3,"renderBody"],"r":{"ty
pe":"NOOP"}},{1":["w",341,2,"items",4,"renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",5,"renderBody"],"r":{"
type":"NOOP"}},{1":["w",341,2,"items",6,"renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",7,"renderBody"],"r":{
"type":"NOOP"}},{1":["w",341,2,"items",8,"renderBody"],"r":{"type":"NOOP"}},{1":["w",341,2,"items",9,"renderBody"],"r
":{"type":"NOOP"}},{1":["w",341,2,"items",10,"renderBody"],"r":{"type":"NOOP"}},{1":["w",342,2,"items",0,"renderBody"]
,"r":{"type":"NOOP"}},{1":["w",342,2,"items",1,"renderBody"],"r":{"type":"NOOP"}},{1":["w",342,2,"items",2,"renderBody
"],"r":{"type":"NOOP"}},{1":["w",357,3,"w"],"r":["w",357,2]]]}</script>
</body>
</html>
<!-- RcmdId srpnweb,RlogId t6pwwit%60d%3D9vjdpwwit%60d*eqqwt(rbpv6775-17763ac5d3d-0x1904 -->
<!-- SiteId: 101, Environment: production, AppName: srpnweb, PageId: 2334524 -->
Programma terminato

```


I titoli di ogni prodotto, si trovano nel tag **H3**, come si può notare qui sotto:




EUR 24,90 a EUR 29,90
 Compralo Subito
 Spedizione gratis
Più di 6 venduti

RAPIDO E GRATUITO

Consegna stimata ven. 5 feb.
 Servizio eBay Premium



FILA scarpe sportive uomo Donna Disruptor Scarpe da ginnastica bianche nero
 Nuovo (Altro)
EUR 10,49 a EUR 28,71
 Compralo Subito
 +EUR 1,00 di spedizione
Più di 17 venduti



Kappa Scarpe Sneakers Uomo Donna LOGO OLMER Camminata Basso
 Nuovo
EUR 19,99
 Prezzo di listino EUR 35,99 43% di sconto
 Altri colori
 theGigastore

Elements

▼

data-track="("eventFamily":"LST","eventAction":"ACTN","actionKind":"NAVSRC","actionKinds":["NAVSRC"],"operationId":"2351460","flushImmediately":false,"eventProperty":{"module": "mi:1686|iid:9|li:7400|uid:1|scen:Listings","parentrq": "639f35cb1770a7b21f6c1d98ffe04220","pageci":"8eac088b-6575-11eb-bd21-822c119c7541")"

sp="p2351460.m1686.17400" class="s-item_link" href="https://www.ebay.it/itm/Scarpe-fitness-rassoda-glutei-benessere-DIMAGR-basculanti-EGLEMTEK/122418981110?hash=item1c9b0d68f6:g:1kAAOSw15h81XU")

h3 class="s-item_title" == \$0

Scarpe fitness rassoda glutei benessere DIMAGRANTI sportive basculanti EGLEMTEK"

h3>

div class="s-item_subtitle">#1 PIU' VENDUTO! Risparmio Garantito. Spedizione veloce</div>

div class="s-item_subtitle"></div>

div class="s-item_details clearfix"></div>

em. divs-item_wrapper.clearfix divs-item_info.clearfix as-item_link h3.s-item_title

Styles

Computed

Layout

Event Listeners

DOM Breakpoints

Properties

Accessibility

Filter

element.style {

font-size: 16px;

margin-bottom: 2px;

line-height: 22px;

Se voglio stampare tutti i tag **H3** della pagina, uso semplicemente la proprietà **cercatutti**:

alfa=sito("https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc")

beta=elaborasito(alfa.contenuto,paginahtml)

scrivi(beta.cercatutti("h3"))

```
>, <h3 class="s-item_title">GUCCI SCARPE SNEAKERS ACE UNISEX UOMO DONNA</h3>, <h3 class="s-item_title">Scarpe Shoes Co
nverse x Comme des Garçons Play alte nere black high Original </h3>, <h3 class="s-item_title">AUSTRALIAN scarpe uomo gi
nnastica strappi velcro regolabili riabilitazione legge</h3>, <h3 class="s-item_title">sneakers NIKE AIR MAX 97 SILVER
col. NERO uomo Spedizione CORRIERE 24H</h3>, <h3 class="s-item_title">Scarpe uomo e donna sneakers alte da ginnastica i
n tela con stringhe</h3>, <h3 class="s-item_title">Scarpe uomo/donna SAUCONY Jazz Original Vintage / Trainer / Azura sn
eakers</h3>, <h3 class="s-item_title">s-item_title s-item_title--has-tags>Uomo donna Scarpe Da Corsa Walking Ginnastica Sportive Ca
sual Sneakers D</h3>, <h3 class="s-item_title s-item_title--has-tags">Scarpe antinfortunistica Uomo S1P SRC Sorpasso S
carpe da lavoro basse</h3>, <h3 class="s-item_title">Scarpe Sneaker Uomo DIADORA Modello Robin 4 Colori</h3>, <h3 clas
s="s-item_title">SCARPE UOMO BASCULANTI DIMAGRANTI massaggianti POSTURALI GINNASTICA SPORTIVE </h3>, <h3 class="s-item
_title">Scarpe sportive uomo sportive NEW BALANCE in nabuk cammello ML574XAA</h3>, <h3 class="s-item_title">Scarpe Tip
o Alexander McQueen 36 37 38 39 40 41 42 43 44 Nero Spedizione 48/72</h3>, <h3 class="s-item_title">Scarpe antinfortuni
stiche UPower Real S1P SRC da lavoro alte leggere</h3>, <h3 class="s-item_title">Scarpe Uomo Nike Revolution 5 Sneaker
Sportive Running Nero Antracite Full Black</h3>, <h3 class="s-item_title">SUADEEX Scarpe antinfortunistica Uomo Scarpe
da lavoro S3 Con tappo in acciaio</h3>, <h3 class="s-item_title">SCARPE UOMO sneakers LIBERO CLASSICHE SPORTIVE BLU ne
re 39 40 41 42 43 44 45</h3>, <h3 class="s-item_title">NIKE AIR FORCE 1 ONE AF1 BLACK&WHITE Low MEN taglia 38/3
9/40/41/42/43/44/5/45</h3>, <h3 class="s-item_title">AUSTRALIAN - SOTTOCOSTO scarpe donna ginnastica palestra corsa spo
rtive tessuto</h3>, <h3 class="s-item_title">Scarpe antinfortunistiche Foxcot R038 S3 SRC invernali impermeabili in pe
lle</h3>, <h3 class="s-item_title">Scarpe Nike Shox R4 EU Bianco White Nero 40, 41, 42, 43, 44, 45 - SALDI -40%</h3>,
<h3 class="s-item_title">Scarpe antinfortunistica Uomo Scarpe da lavoro leggere S1P SRC antiscivolo</h3>, <h3 class="s-
item_title">SCARPE ADIDAS UOMO DONNA UNISEX VS PACE AW4594 SKATEBOARD BIANCO PELLE ORIGINALI</h3>, <h3 class="s-item_t
itle"><span class="LIGHT_HIGHLIGHT">Nuova inserzione</span> Converse Scarpe Sneakers STAR PLAYER OX Nero Tessuto</h3>,
<h3 class="s-item_title">Scarpe Da Ginnastica Air Jordan1 Retro High OG PS Satin Mid Chicago Uomo - Donna</h3>, <h3 cla
ss="s-item_title">Adidas Scarpe Sportive Sneakers ROGUERA Uomo Nero Vera pelle</h3>, <h3 class="s-item_title">Scarpe
antinfortunistiche U-Power Safe S3 SRC alte in pelle impermeabile</h3>, <h3 class="s-item_title">SCARPE UOMO pelle ti
po timberland classiche sportive casual tempo libero -50%</h3>, <h3 class="s-item_title">ENRICO COVERI offerta Sneakers
scarpe sportive casual da uomo comode con lacci</h3>, <h3 class="s-item_title">Scarpe da basket uomo NIKE Precision IV
in tela rosso e bianco CK1069-600</h3>, <h3 class="s-item_title s-item_title--has-tags">Scarpe antinfortunistica Uomo
Scarpe da lavoro leggere nero S1P SRC Miller Steel</h3>, <h3 class="s-item_title s-item_title--has-tags">Scarpe antin
fortunistica basse Uomo S1P SRC Sorpasso Scarpe da lavoro leggere</h3>]
Programma terminato
```

però in questo modo ottengo una lista di stringhe, contenenti i tag **H3**. E' chiaro che per mostrare l'elenco di tutti i tag **H3** della lista, dovrò usare un ciclo, come quello qui sotto:

alfa=sito("https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc")

beta=elaborasito(alfa.contenuto,paginahtml)

gamma=beta.cercatutti("h3")

considera provola con gamma:

scrivi(provola)


```

<h3 class="s-item_title s-item_title--has-tags">Uomo donna Scarpe Da Corsa Walking Ginnastica Sportive Casual Sneakers
D</h3>
<h3 class="s-item_title s-item_title--has-tags">Scarpe antinfortunistica Uomo S1P SRC Sorpasso Scarpe da lavoro basse
</h3>
<h3 class="s-item_title">Scarpe Sneaker Uomo DIADORA Modello Robin 4 Colori</h3>
<h3 class="s-item_title">SCARPE UOMO BASCULANTI DIMAGRANTI massaggianti POSTURALI GINNASTICA SPORTIVE </h3>
<h3 class="s-item_title">Scarpe sportive uomo sportive NEW BALANCE in nabuk cammello ML574XAA</h3>
<h3 class="s-item_title">SUADEEX Scarpe antinfortunistica Uomo Scarpe da lavoro S3 Con tappo in acciaio</h3>
<h3 class="s-item_title">Scarpe Tipo Alexander McQueen 36 37 38 39 40 41 42 43 44 Nero Spedizione 48/72</h3>
<h3 class="s-item_title">Scarpe antinfortunistiche UPower Real S1P SRC da lavoro alte leggere</h3>
<h3 class="s-item_title">Scarpe Uomo Nike Revolution 5 Sneaker Sportive Running Nero Antracite Full Black</h3>
<h3 class="s-item_title">SCARPE UOMO sneakers LIBERO CLASSICHE SPORTIVE BLU nere 39 40 41 42 43 44 45</h3>
<h3 class="s-item_title">NIKE AIR FORCE 1 ONE AF1 BLACK&WHITE Low MEN taglia 38/39/40/41/42/43/44/5/45</h3>
<h3 class="s-item_title">AUSTRALIAN - SOTTOCOSTO scarpe donna ginnastica palestra corsa sportive tessuto </h3>
<h3 class="s-item_title">Scarpe antinfortunistiche Foxcot R038 S3 SRC invernali impermeabili in pelle</h3>
<h3 class="s-item_title">Scarpe Nike Shox R4 EU Bianco White Nero 40, 41, 42, 43, 44, 45 - SALDI -40%</h3>
<h3 class="s-item_title">Scarpe antinfortunistica Uomo Scarpe da lavoro leggere S1P SRC antiscivolo</h3>
<h3 class="s-item_title">SCARPE ADIDAS UOMO DONNA UNISEX VS PACE AW4594 SKATEBOARD BIANCO PELLE ORIGINALI</h3>
<h3 class="s-item_title"><span class="LIGHT_HIGHLIGHT">Nuova inserzione</span> Converse Scarpe Sneakers STAR PLAYER OX
Nero Tessuto </h3>
<h3 class="s-item_title">Scarpe Da Ginnastica Air Jordan1 Retro High OG PS Satin Mid Chicago Uomo - Donna</h3>
<h3 class="s-item_title">Adidas Scarpe Sportive Sneakers ROGUERA Uomo Nero Vera pelle </h3>
<h3 class="s-item_title">Scarpe antinfortunistiche U-Power Safe S3 SRC alte in pelle impermeabile</h3>
<h3 class="s-item_title">Scarpe antinfortunistica basse Uomo S1P SRC Sorpasso Scarpe da lavoro leggere </h3>
<h3 class="s-item_title">SCARPE UOMO pelle tipo timberland classiche sportive casual tempo libero -50%</h3>
<h3 class="s-item_title">ENRICO COVERI offerta Sneakers scarpe sportive casual da uomo comode con lacci</h3>
<h3 class="s-item_title s-item_title--has-tags">Scarpe antinfortunistica Uomo Scarpe da lavoro leggere nero S1P SRC Mi
ller Steel</h3>
<h3 class="s-item_title s-item_title--has-tags">Saucony Jazz Shadow Scarpe Uomo Donna Adulto 100% Originali 2021</h3>
Programma terminato

```

Quello che però abbiamo ottenuto, è meramente il codice HTML di ogni tag **H3**. Se vogliamo stampare solo il testo contenuto in ogni singolo tag **HTML**, dobbiamo usare la proprietà **testo**, come mostrato qui sotto:

alfa=sito("https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc")

beta=elaborasito(alfa.contenuto,paginahtml)

gamma=beta.cercatutti("h3")

considera provola con gamma:

scrivi(provola.testo)

```

AUSTRALIAN scarpe uomo ginnastica strappi velcro regolabili riabilitazione legge
sneakers NIKE AIR MAX 97 SILVER col. NERO uomo Spedizione CORRIERE 24H
Scarpe uomo e donna sneakers alte da ginnastica in tela con stringhe
Scarpe uomo/donna SAUCONY Jazz Original Vintage / Trainer / Azura sneakers
Scarpe antinfortunistica Uomo S1P SRC Sorpasso Scarpe da lavoro basse
SCARPE ANTINFORTUNISTICHE S3 SRC IMPERMEABILI LOTTO WORKS SPRINT 301 Q8359
Scarpe Sneaker Uomo DIADORA Modello Robin 4 Colori
SCARPE UOMO BASCULANTI DIMAGRANTI massaggianti POSTURALI GINNASTICA SPORTIVE
Scarpe antinfortunistiche UPower Real S1P SRC da lavoro alte leggere
Scarpe Uomo Nike Revolution 5 Sneaker Sportive Running Nero Antracite Full Black
Scarpe Tipo Alexander McQueen 36 37 38 39 40 41 42 43 44 Nero Spedizione 48/72
SUADEEX Scarpe antinfortunistica Uomo Scarpe da lavoro S3 Con tappo in acciaio
SCARPE UOMO sneakers LIBERO CLASSICHE SPORTIVE BLU nere 39 40 41 42 43 44 45
Scarpe sportive uomo sportive NEW BALANCE in nabuk cammello ML574XAA
NIKE AIR FORCE 1 ONE AF1 BLACK&WHITE Low MEN taglia 38/39/40/41/42/43/44/5/45
AUSTRALIAN - SOTTOCOSTO scarpe donna ginnastica palestra corsa sportive tessuto
Scarpe da basket uomo NIKE Precision IV in tela rosso e bianco CK1069-600
Scarpe antinfortunistiche Foxcot R038 S3 SRC invernali impermeabili in pelle
Scarpe antinfortunistica Uomo Scarpe da lavoro leggere S1P SRC antiscivolo
SCARPE ADIDAS UOMO DONNA UNISEX VS PACE AW4594 SKATEBOARD BIANCO PELLE ORIGINALI
Nuova inserzione Converse Scarpe Sneakers STAR PLAYER OX Nero Tessuto
Scarpe Da Ginnastica Air Jordan1 Retro High OG PS Satin Mid Chicago Uomo - Donna
Adidas Scarpe Sportive Sneakers ROGUERA Uomo Nero Vera pelle
Scarpe antinfortunistiche U-Power Safe S3 SRC alte in pelle impermeabile
Scarpe antinfortunistica basse Uomo S1P SRC Sorpasso Scarpe da lavoro leggere
SCARPE UOMO pelle tipo timberland classiche sportive casual tempo libero -50%
ENRICO COVERI offerta Sneakers scarpe sportive casual da uomo comode con lacci
Scarpe antinfortunistica Uomo Scarpe da lavoro leggere nero S1P SRC Miller Steel
Saucony Jazz Shadow Scarpe Uomo Donna Adulto 100% Originali 2021
Programma terminato

```

Esaminando il codice HTML della pagina che stiamo elaborando, noteremo che i prezzi sono inseriti in questi tag **EUR 19,99**, ma se chiediamo a **Lungo**, di scrivere

tutti i tag **span**, otterremo anche informazioni di altri testi, tra tag **span**, che non c'entrano con il prezzo. Allora possiamo finalizzare, cioè filtrare meglio la selezione, indicando anche la **class** precisa del tag **span**, che nel nostro caso è "**s-item__price**":

```
alfa=sito("https://www.ebay.it/sch/i.html?_from=R40&_nkw=scarpe&_sacat=0&_dmd=1&rt=nc")
```

```
beta=elaborasito(alfa.contenuto,paginahtml)
```

```
gamma=beta.cercatutti("span",htmlclasse="s-item__price")
```

considera provola con gamma:

```
scrivi(provola.testo)
```

```
EUR 29,90
EUR 79,99
EUR 11,99
EUR 79,99
EUR 17,90
EUR 39,00
EUR 29,99
EUR 25,00
EUR 32,99
EUR 49,90
EUR 89,99
EUR 25,79 a EUR 35,79
EUR 25,00
EUR 69,00
EUR 69,99
EUR 19,90 a EUR 24,90
EUR 19,90
EUR 34,99
EUR 49,90
EUR 51,99 a EUR 62,99
EUR 59,95
EUR 34,76
EUR 25,99
EUR 33,90
EUR 25,00
EUR 34,99
EUR 79,00
EUR 33,99
EUR 59,00 a EUR 125,00
Programma terminato
```

Le possibili ricerche che possiamo fare nel codice HTML, sono sia quelle già spiegate che altre, che vediamo qui di seguito:

```
gamma=beta.cercatutti("span")
```

che cerca solo i tag specifici (in questo caso il tag ``)

Oppure possiamo cercare un tag specifico con un determinato nome di **classe**:

```
gamma=beta.cercatutti("span",htmlclasse="s-item__price")
```

Oppure possiamo cercare un tag specifico con un determinato nome di **id**:

```
gamma=beta.cercatutti("span",htmlid="s-item__price")
```

Oppure possiamo cercare solo un determinato nome di **classe**:

```
gamma=beta.cercatutti(htmlclasse="s-item__price")
```

Oppure possiamo cercare solo un determinato nome di **id**:

```
gamma=beta.cercatutti(htmlid="s-item__price")
```

DIAGRAMMI

Con **Lungo** è possibile generare dei diagrammi in vari formati.

La realizzazione è molto basilare, ma semplice.

DIAGRAMMA A LINEE

Per creare un diagramma a linee, bisogna avere dei dati da visualizzare, e per questo supponiamo di avere i seguenti dati:

- temperature clima di un paese, ad esempio gradi: 23, 25, 27, 26, 20, 27, tutti presi mese per mese, quindi questi sono relativi ai mesi da gennaio a giugno;
- etichetta su asse x con scritto "mesi" ed etichetta su asse y con scritto "temperature"
- titolo del diagramma "Temperature dell'anno"

```
x = ["gen","feb","mar","apr","mag","giu"]
```

```

y = [23,25,27,26,20,27]
diagrammaalinee(x, y)
diagrammaetichettax('mesi')
diagrammaetichettay('temperature')
diagrammatitolo("Temperature dell'anno")
mostradiagramma()

```

La prima riga contiene la lista dei mesi messe dentro la variabile **x**

La seconda riga contiene le temperature rilevate per ogni mese messe dentro **y**

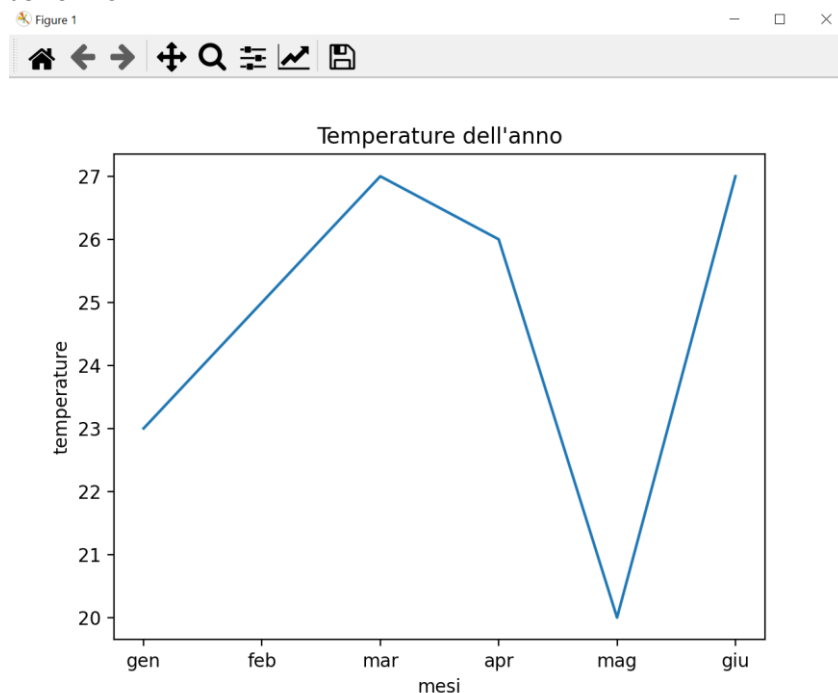
La terza riga con l'istruzione **diagrammaalinee** prepara il diagramma a linee

La quarta riga con l'istruzione **diagrammaetichettax** scrive l'etichetta sul lato **x**

La quinta riga con l'istruzione **diagrammaetichettay** scrive l'etichetta sul lato **y**

La sesta riga con l'istruzione **diagrammatitolo** scrive il titolo del diagramma

La settima ed ultima riga, con l'istruzione **mostradiagramma()** fa visualizzare il diagramma sullo schermo

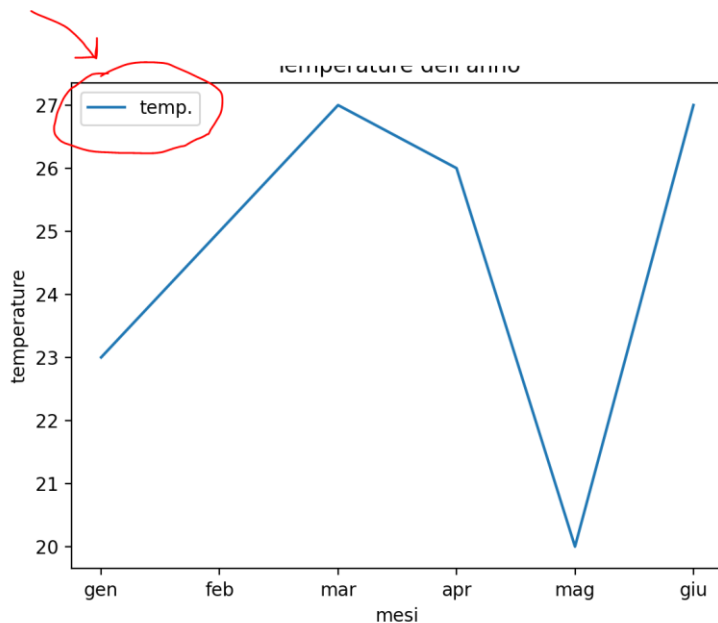


Se vogliamo aggiungere una legenda al grafico, ad esempio la legenda "temp.", dobbiamo aggiungere la proprietà **legenda** all'istruzione **diagrammaalinee**, e l'istruzione **mostralegenda** per farla visualizzare sul diagramma:

```

x = ["gen","feb","mar","apr","mag","giu"]
y = [23,25,27,26,20,27]
diagrammaalinee(x, y,legenda="temp.")
diagrammaetichettax('mesi')
diagrammaetichettay('temperature')
diagrammatitolo("Temperature dell'anno")
mostralegenda()
mostradiagramma()

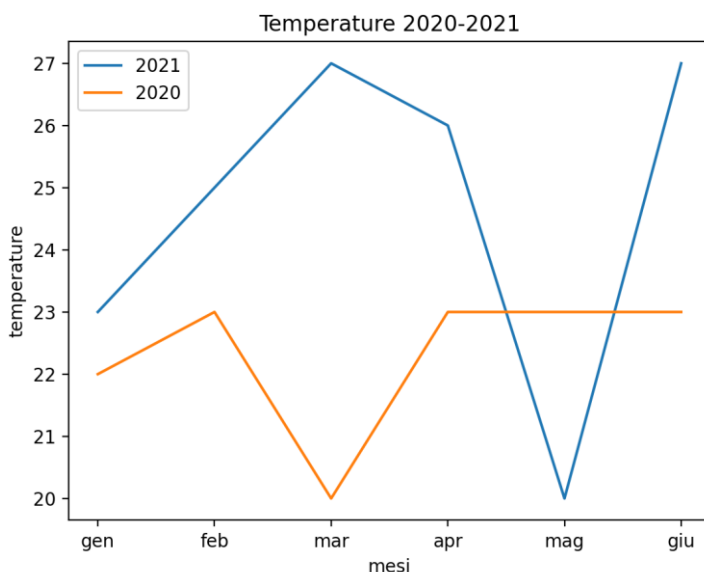
```

Se ad esempio vogliamo modificare il progetto sopra, ipotizzando che le temperature rilevate sono relative all'anno 2021, possiamo cambiare la legenda con il testo "2021".

E se volessimo aggiungere un'altra rilevazione di temperature relative all'anno 2020, possiamo aggiungere i nuovi dati, aggiungendo la legenda per i nuovi dati in questo modo:

```
x = ["gen","feb","mar","apr","mag","giu"]
y = [23,25,27,26,20,27]
x1 = ["gen","feb","mar","apr","mag","giu"]
y1 = [22,23,20,23,23,23]
diagrammaalinee(x, y,legenda="2021")
diagrammaalinee(x1, y1,legenda="2020")
diagrammaetichettax('mesi')
diagrammaetichettay('temperature')
diagrammatitolo("Temperature 2020-2021")
mostralegenda()
mostradiagramma()
```



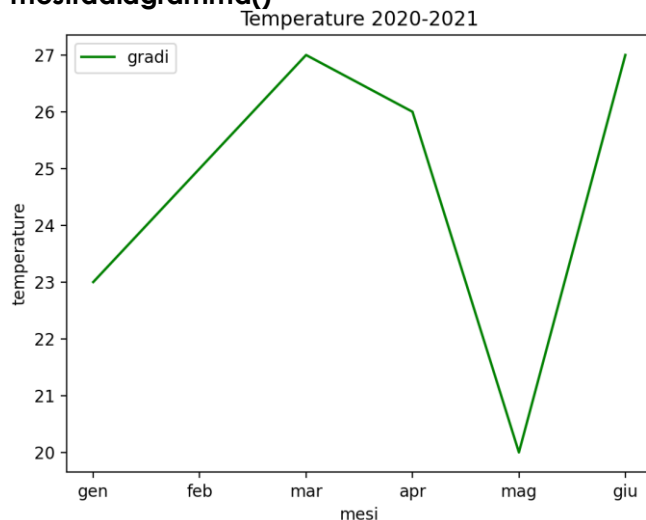
Per impostare il colore di un digramma lineare, si usa la proprietà **colore** e i colori devono essere preceduti da punto esclamativo, come ad esempio nel programma sotto:

```
x = ["gen","feb","mar","apr","mag","giu"]
```

```

y = [23,25,27,26,20,27]
diagrammaalinee(x, y,legenda="gradi",colore=!verde)
diagrammaetichettax('mesi')
diagrammaetichettay('temperature')
diagrammatitolo("Temperature 2020-2021")
mostralegenda()
mostradiagramma()

```



Per impostare un tipo di disegno di linea, si usa la proprietà **stilelinea** impostandola solo su una di queste modalità:

lineacontinua



lineaattrattini



lineaapuntini



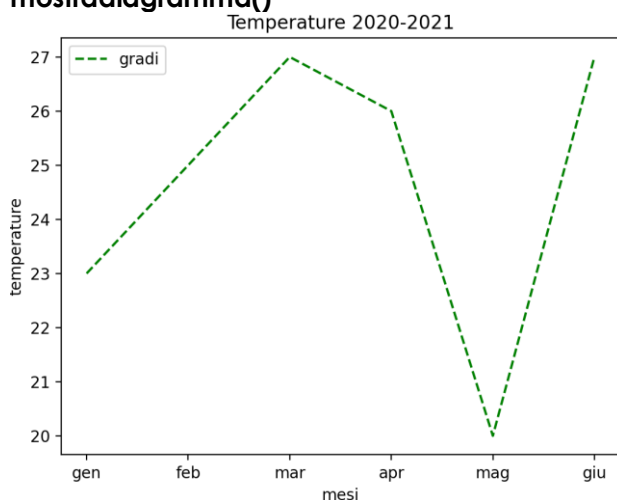
lineaapuntinietrattini oppure **lineaattrattiniepuntini**



```

x = ["gen","feb","mar","apr","mag","giu"]
y = [23,25,27,26,20,27]
diagrammaalinee(x, y,legenda="gradi",colore=!verde,stilelinea=lineaattrattini)
diagrammaetichettax('mesi')
diagrammaetichettay('temperature')
diagrammatitolo("Temperature 2020-2021")
mostralegenda()
mostradiagramma()

```



Per impostare lo spessore della linea del diagramma, si usa la proprietà **spessorelinea** con un numero che indica i pixel di spessore:

```

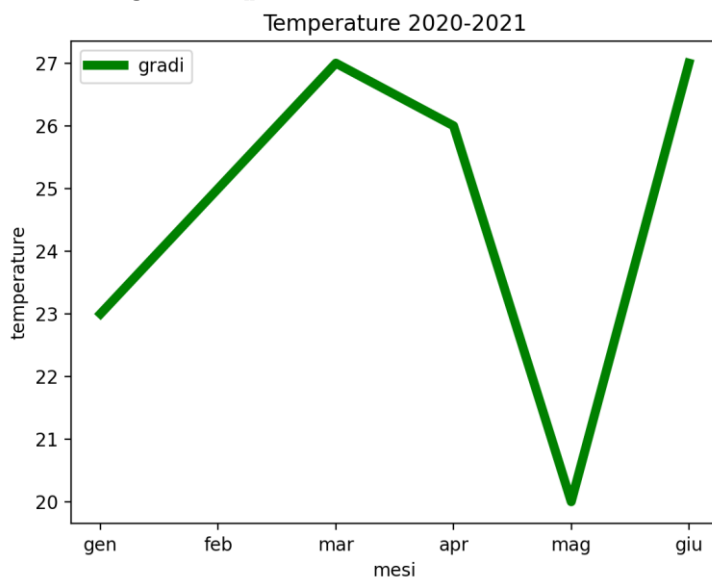
x = ["gen","feb","mar","apr","mag","giu"]

```

```

y = [23,25,27,26,20,27]
diagrammaalinee(x, y,legenda="gradi",colore=!verde,spessorelinea=5)
diagrammaetichettax('mesi')
diagrammaetichettay('temperature')
diagrammatitolo("Temperature 2020-2021")
mostralegenda()
mostradiagramma()

```



Per impostare un disegno sul vertice in cui si spezzano le varie linee, si usa la proprietà **verticelinea**, usando una delle possibili alternative qui sotto elencate:



```

x = ["gen","feb","mar","apr","mag","giu"]
y = [23,25,27,26,20,27]
diagrammaalinee(x, y,legenda="gradi",colore=!verde,verticelinea="o")
diagrammaetichettax('mesi')
diagrammaetichettay('temperature')
diagrammatitolo("Temperature 2020-2021")
mostralegenda()
mostradiagramma()

```

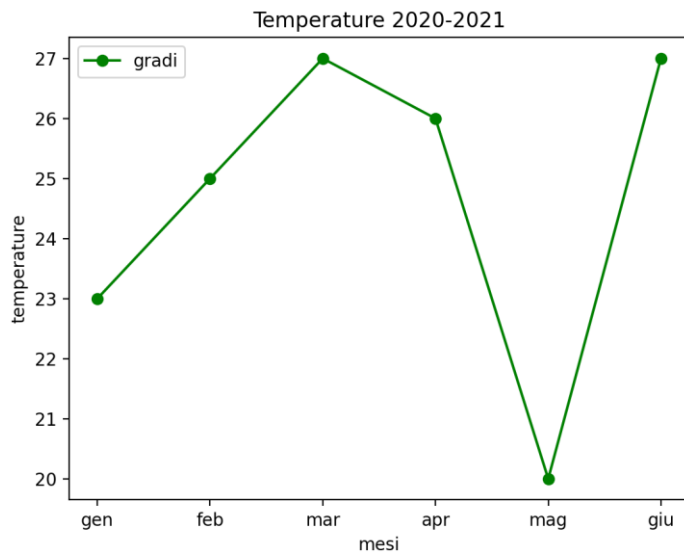


DIAGRAMMA A BARRE

Per creare un grafico diagramma a barre, si usano le stesse istruzioni e proprietà viste per l'istruzione **diagrammaalinee**, solo che l'istruzione da usare è **diagrammaabarre**:

```
x = ["gen","feb","mar","apr","mag","giu"]
```

```
y = [23,25,27,26,20,27]
```

```
diagrammaabarre(x, y, legenda="gradi")
```

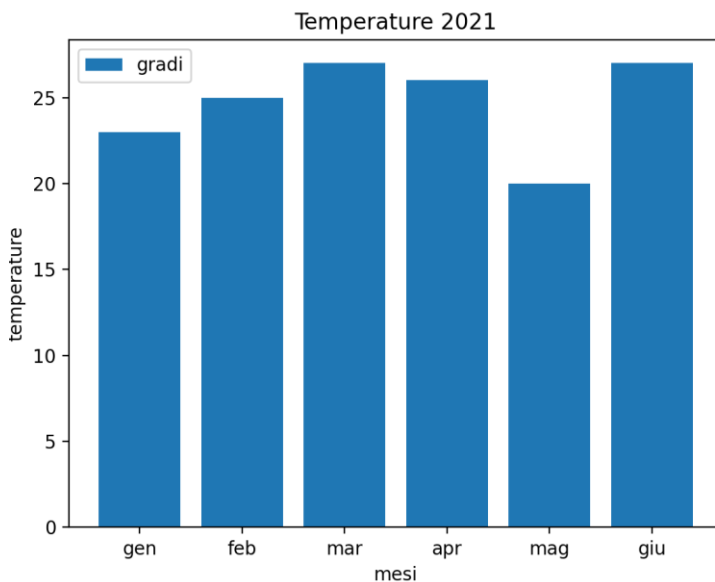
```
diagrammaetichettax('mesi')
```

```
diagrammaetichettay('temperature')
```

```
diagrammatitolo('Temperature 2021')
```

```
mostralegenda()
```

```
mostradiagramma()
```



Se si vogliono alternare i colori delle barre, è possibile usare una lista di due colori, sempre usando la proprietà **colore**:

```
x = ["gen","feb","mar","apr","mag","giu"]
```

```
y = [23,25,27,26,20,27]
```

```
diagrammaabarre(x, y, legenda="gradi", colore=["verde","rosso"])
```

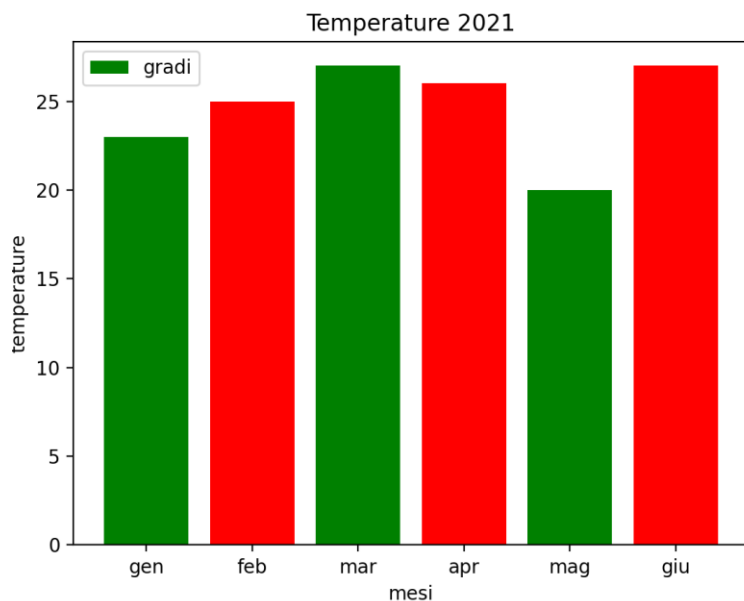
```
diagrammaetichettax('mesi')
```

```
diagrammaetichettay('temperature')
```

```
diagrammatitolo('Temperature 2021')
```

```
mostralegenda()
```

```
mostradiagramma()
```



BOT TELEGRAM

Con **Lungo** è possibile programmare, in modo relativamente semplice, dei bot per il sistema di messaggistica Telegram.

(Cos'è un BOT? Un BOT non è altro che usa sistema automatico che fa da risponditore a richieste e quesiti. Se gestisco un'azienda di ricambi auto, potrei voler rispondere alle richieste di clienti che chiedono la disponibilità di determinati prodotti o servizi. Per dargli risposta, dovrei leggere tutti i loro messaggi e rispondere ad ognuno, oppure posso creare un BOT che capisce le domande e sa dare risposte automatiche)

Il programma che realizzeremo con **Lungo** avrà però bisogno di un particolare server, che farà funzionare il programma anche quando il nostro computer sarà spento, così che gli utenti di Telegram, quando usano in nostro bot, possono conversare con lui.

Prima di tutto, aprite Telegram, sul cellulare o Telegram Web sul computer e cercate l'utente **BotFather** che deve avere la spunta blu, perché ci sono altri utenti di nome BotFather, senza spunta blu, che non sono quello ufficiale:



Scrivete a questo "utente" il messaggio:

/start

e lui risponderà con una lunga lista di istruzioni:



BotFather

I can help you create and manage Telegram bots. If you're new to the Bot API, please [see the manual](#).

You can control me by sending these commands:

[/newbot](#) - create a new bot

[/mybots](#) - edit your bots **[beta]**

Edit Bots

[/setname](#) - change a bot's name

[/setdescription](#) - change bot description

[/setabouttext](#) - change bot about info

[/setuserpic](#) - change bot profile photo

[/setcommands](#) - change the list of commands

[/deletebot](#) - delete a bot

Bot Settings

[/token](#) - generate authorization token

[/revoke](#) - revoke bot access token

[/setinline](#) - toggle [inline mode](#)

[/setinlinegeo](#) - toggle [inline location requests](#)

[/setinlinefeedback](#) - change [inline feedback](#) settings

[/setjoingroups](#) - can your bot be added to groups?

[/setprivacy](#) - toggle [privacy mode](#) in groups

Games

[/mygames](#) - edit your [games](#) **[beta]**

[/newgame](#) - create a new [game](#)

[/listgames](#) - get a list of your games

[/editgame](#) - edit a game

[/deletegame](#) - delete an existing game

Ora scrivete il messaggio:

[/newbot](#)

che serve a creare un nuovo bot.

Lui risponderà con il messaggio:



BotFather

Alright, a new bot. How are we going to call it? Please choose a name for your bot.

Con questo messaggio ci sta chiedendo che nome vogliamo dare al nostro bot. Nel mio esempio lo chiamo "provolo" e quindi lo scrivo come messaggio:

provolo

Ci verrà data la seguente risposta:



BotFather

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot.

Con la risposta sopra, ci chiede di dare un username al bot, usando la parola "bot" come finale e attaccando un'altra parola con underscore (sottolineato) alla parola "bot".

provola3_bot

Come si può notare, non ho usato lo stesso nome del bot, ma una parola simile, l'importante è che ci sia una parola, un underscore (sottolineato) e la parola "bot".

Lui risponderà:



BotFather

1

Done! Congratulations on your new bot. You will find it at t.me/provola3_bot. You can now add a description, about section and profile picture for your bot, see [/help](#) for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:

1687365487:AAExemhsowGV11YL7Yy9MSI3[REDACTED]nFzk

CODICE
↙

Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page:

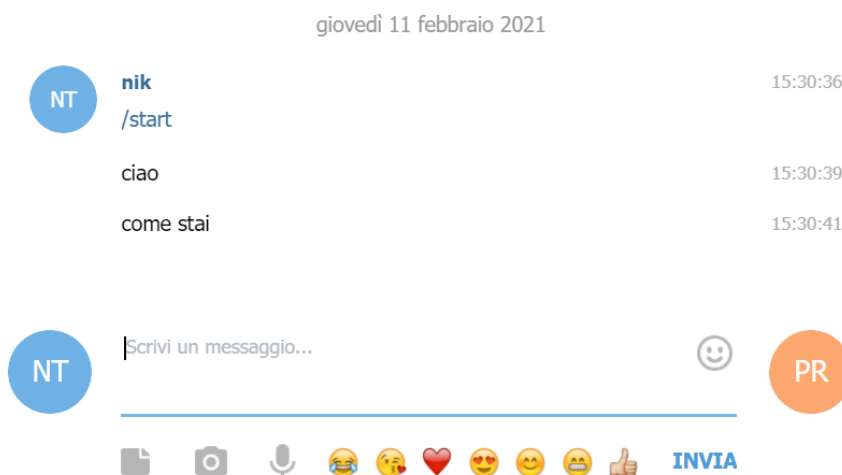
<https://core.telegram.org/bots/api>

Il messaggio dice che il bot è stato creato. Inoltre ci viene dato un lungo codice, scritto in rosso (che ho cancellato parzialmente per la privacy) con cui potrete gestire il vostro bot, e che ci sarà utile quando programmeremo il bot con **Lungo**.

Per vedere il vostro bot che è attivo, cercate su Telegram l'utente **provola3_bot** che è il nome che avete creato, e vedrete che sarà un utente presente. Questo utente è il vostro bot:



Premendo su questo vostro bot, potrete conversare con lui, ma non saprà rispondere ad alcuna richiesta perché non lo abbiamo programmato.



Ora dobbiamo registrarci sul sito web <https://www.pythonanywhere.com/> che è un sito, con un server, cioè con un hard disk virtuale, in cui possiamo far funzionare dei software in linguaggio Python, che funzioneranno anche quando il nostro computer è spento. Come sapete, **Lungo** sviluppa software in linguaggio **Python**, quindi su questo sito copieremo il file generato da **Lungo**.

Quando sarete registrati su <https://www.pythonanywhere.com/> sappiate che potrebbe passare anche un'ora prima che possiate avere tutte le funzioni attive, senza errori.

Ora creiamo il bot in linguaggio **Lungo**.

Prima di tutto creiamo una variabile dove inserire il lungo codice che ci è stato fornito da Telegram e che vi ho spiegato poco più sopra.

```
x="1623790016AAGm2gCSMdeeXpTjZP278trCSd6eo2o"
```

Poi usiamo l'istruzione **botinizio** a cui indichiamo la variabile del codice Telegram, e mettiamo tutto dentro un'altra variabile, ad esempio di nome **alfa**

alfa=botinizio(x)

Poi, usando la chiocciola iniziale, inseriamo il nome di variabile dove abbiamo inserito il bot (nel nostro caso è **alfa**) e usiamo la proprietà **gestionemessaggi** a cui inseriamo l'istruzione **comandidibase**. L'istruzione **comandidibase** si occupa di gestire gli help di sistema.

@alfa.gestionemessaggi(comandidibase)

Poi creiamo una funzione che dobbiamo chiamare obbligatoriamente **invia_benvenuto** e che deve contenere obbligatoriamente il metodo **messaggio**. In questa funzione inseriamo al nostro bot, la proprietà **rispondi**, in cui inseriremo il metodo **messaggio** e tra virgolette il messaggio di benvenuto da dare:

funzione invia_benvenuto(messaggio):

alfa.rispondi(messaggio,"Ciao sono LungoBot. Chiedimi cosa ti serve")

Poi, usando la chiocciola iniziale, inseriamo il nome di variabile dove abbiamo inserito il bot (nel nostro caso è **alfa**) e usiamo la proprietà **gestionemessaggi** a cui inserire l'istruzione **comandimessaggi**. L'istruzione **comandimessaggi** si occupa di gestire tutti i messaggi del nostro bot, che programmeremo ora.

@alfa.gestionemessaggi(comandimessaggi)

Poi creiamo una funzione che dobbiamo chiamare obbligatoriamente **invia_messaggio** e che deve contenere obbligatoriamente il metodo **messaggio**, quindi possiamo programmare come meglio vogliamo la gestione delle risposte da dare alle domande degli utenti. Ricordiamo che le risposte degli utenti saranno contenute nell'istruzione **messaggio.testo**, e le risposte che daremo, saranno processate dall'istruzione **rispondi**:

funzione invia_messaggio(messaggio):

se messaggio.testo=="ciao":

alfa.rispondi(messaggio,"ciao a te")

altrimentise messaggio.testo=="buongiorno":

alfa.rispondi(messaggio,"buona giornata a te")

altrimenti:

alfa.rispondi(messaggio,"Non capisco")

Infine, dobbiamo inserire come ultima istruzione la seguente:

alfa.botfine

che specifica a **Lungo** che il bot è terminato.

Ed ecco tutto il programma di esempio completo:

x="1623790016AArGm2gCSMdeeXTjZP27N8rCSd6Geo2o"

alfa=botinizio(x)

@alfa.gestionemessaggi(comandidibase)

funzione invia_benvenuto(messaggio):

alfa.rispondi(messaggio,"Ciao sono LungoBot. Chiedimi cosa ti serve")

@alfa.gestionemessaggi(comandimessaggi)

funzione invia_messaggio(messaggio):

se messaggio.testo=="ciao":

alfa.rispondi(messaggio,"ciao a te")

altrimentise messaggio.testo=="buongiorno":

alfa.rispondi(messaggio,"buona giornata a te")

altrimenti:

alfa.rispondi(messaggio,"Non capisco")

alfa.botfine

Quando avvieremo il programma, verrà generato il file **provola-convertito.py** che contiene il programma in linguaggio **Python** del nostro bot. Questo semplice bot saprà rispondere al messaggio "ciao" con "ciao a te" e al messaggio "buongiorno" con il messaggio "buona giornata a te".

Inoltre, verrà scritto nella finestra di sistema, tutta la serie di istruzioni per caricare ed avviare il bot sul sito <https://www.pythonanywhere.com/>

```
===== RESTART: C:/Users/nikto/OneDrive/Documenti/progetto lungo/lungo3.py ==  
Ho creato il file provola-convertito.py  
Ora vai sul sito https://www.pythonanywhere.com/  
e premi sul pulsante 'Browse Files'  
Premi su 'Upload a file' e carica il file provola-convertito.py  
Poi premi sul link 'Open Bash console here' che trovi sulla stessa pagina  
Si aprirà un terminale dove devi scrivere la seguente istruzione:  
mkvirtualenv --python=/usr/bin/python3 mysite-virtualenv  
Premi il tasto INVIO per lanciare quell'istruzione  
Al termine scrivi la seguente istruzione:  
pip install pyTelegramBotAPI  
premi il tasto INVIO sulla tastiera  
Al termine della procedura, lancia il bot con la seguente istruzione:  
python provola-convertito.py  
Premi il tasto INVIO sulla tastiera e il bot sarà attivo  
Buon divertimento...
```

Infatti, ora dobbiamo procedere così:

Vai sul sito <https://www.pythonanywhere.com/> dove ci eravamo già registrati precedentemente, e premiamo sul pulsante **Browse Files**:

Dashboard

Welcome, [niktorthenat](#)CPU Usage: 31% used – 31.26s of 100s. Resets in 19 hours, 23 minutes [More Info](#)[Upgrade Account](#)

File storage: 5% full – 24.5 MB of your 512.0 MB quota

Recent
Consoles

+ 5 -

[Bash console 18957474](#)

New console:

[\\$ Bash](#) [>>> Python](#) [More...](#)Recent
Files

+ 5 -

You have no recently edited files.

[+ Open another file](#)[Browse files](#)Recent
Notebooks

+ 5 -

Your account does not support
Jupyter Notebooks. [Upgrade your
account](#) to get access!All
Web apps

You don't have any web apps.

[Open Web tab](#)

Si aprirà una pagina come questa, in cui dovremo premere sul pulsante **Upload a file**, e caricare il file **provola-convertito.py** che hai appena realizzato:

[/home/](#) [niktorthenat](#)[Open Bash console](#)

Directories

Enter new directory name

[New directory](#)[.cache/](#)[.local/](#)[.virtualenvs/](#)

Files

Enter new file name, eg hello.py

[.bashrc](#)

2021-02-10 09:55 559 bytes

[.gitconfig](#)

2021-02-10 09:55 266 bytes

[.profile](#)

2021-02-10 09:55 79 bytes

[.pythonstartup.py](#)

2021-02-10 09:55 77 bytes

[.vimrc](#)

2021-02-10 09:55 4.6 KB

[README.txt](#)

2021-02-10 09:55 232 bytes

[Upload a file](#)

100MiB maximum size

Dopo che avete caricato il file, lo troveremo tra l'elenco dei file caricati:

Directories

Enter new directory name

New directory

[.cache/](#)



[.local/](#)



[.virtualenvs/](#)



Files

Enter new file name, eg hello.py

 [.bashrc](#)



2021-02-10

 [.gitconfig](#)



2021-02-10

 [.profile](#)



2021-02-10

 [.pythonstartup.py](#)



2021-02-10

 [.vimrc](#)



2021-02-10

 [README.txt](#)



2021-02-10

 [provola-convertito.py](#)



2021-02-11

 Upload a file

100MiB maximum size

Ora il bot è caricato, ma non è attivo su Telegram. Per attivarlo su Telegram prima di tutto dobbiamo premere sul link **Open bash console here**

 [Open Bash console here](#) 5% full

Directories

Enter new directory name

New directory

[.cache/](#)



[.local/](#)




[.virtualenvs/](#)



Files

Enter new file name, eg hello.py

 [.bashrc](#)



2021-02-10 09:55 559 bytes

 [.gitconfig](#)



2021-02-10 09:55 266 bytes

 [.profile](#)



2021-02-10 09:55 79 bytes

 [.pythonstartup.py](#)



2021-02-10 09:55 77 bytes

 [.vimrc](#)



2021-02-10 09:55 4.6 KB

 [README.txt](#)



2021-02-10 09:55 232 bytes

 [provola-convertito.py](#)



2021-02-11 15:15 594 bytes

 Upload a file

100MiB maximum size

Si aprirà un terminale



Bash console 18958645

15:18 ~ \$

Qui dobbiamo scrivere la seguente istruzione che carica una versione virtuale del linguaggio Python **mkvirtualenv --python=/usr/bin/python3 mysite-virtualenv**



Bash console 18958645

```
15:18 ~ $ mkvirtualenv --python=/usr/bin/python3 mysite-virtualenv
Running virtualenv with interpreter /usr/bin/python3
Already using interpreter /usr/bin/python3
Using base prefix '/usr'
New python executable in /home/niktorthenat/.virtualenvs/mysite-virtualenv/bin/python
Not overwriting existing python script /home/niktorthenat/.virtualenvs/mysite-virtualenv/bin/python3
Installing setuptools, pip, wheel...
```

Quando il terminale avrà finito di processare quelle istruzioni, scriviamo l'istruzione **pip install pyTelegramBotAPI** che si occupa di importare la libreria Python per gestire le istruzioni di Telegram:



Bash console 18958645

Share with others

```
15:18 ~ $ mkvirtualenv --python=/usr/bin/python3 mysite-virtualenv
Running virtualenv with interpreter /usr/bin/python3
Already using interpreter /usr/bin/python3
Using base prefix '/usr'
New python executable in /home/niktorthenat/.virtualenvs/mysite-virtualenv/bin/python
Not overwriting existing python script /home/niktorthenat/.virtualenvs/mysite-virtualenv/bin/python (you must use /home/niktorthenat/.virtualenvs/mysite-virtualenv/bin/python3)
Installing setuptools, pip, wheel...
done.
(mysite-virtualenv) 15:19 ~ $ pip install pyTelegramBotAPI
DEPRECATION: Python 3.5 reached the end of its life on September 13th, 2020. Please upgrade your Python as Python 3.5 is no longer maintained. pip 21.0 will drop support for Python 3.5 in January 2021. pip 21.0 will remove support for this functionality.
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: pyTelegramBotAPI in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (3.7.6)
Requirement already satisfied: requests in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from pyTelegramBotAPI) (2.25.1)
Requirement already satisfied: chardet<5,>=3.0.2 in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from requests->pyTelegramBotAPI) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from requests->pyTelegramBotAPI) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from requests->pyTelegramBotAPI) (1.26.3)
Requirement already satisfied: certifi>=2017.4.17 in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from requests->pyTelegramBotAPI) (2020.12.5)
(mysite-virtualenv) 15:20 ~ $
```

Al termine anche di questa procedura, scriviamo la seguente istruzione che avvia il nostro bot su Telegram: **python provola-converto.py**

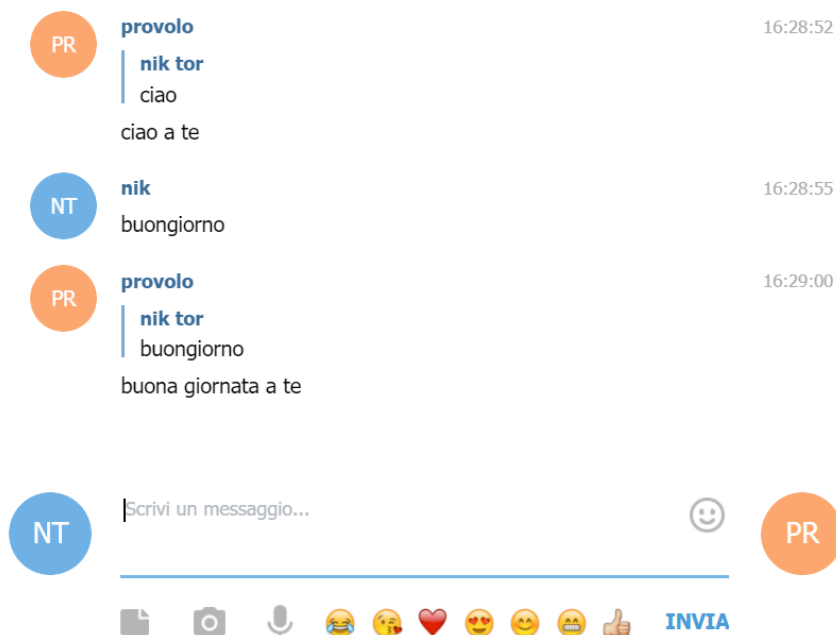


Bash console 18958645

Share with

```
15:18 ~ $ mkvirtualenv --python=/usr/bin/python3 mysite-virtualenv
Running virtualenv with interpreter /usr/bin/python3
Already using interpreter /usr/bin/python3
Using base prefix '/usr'
New python executable in /home/niktorthenat/.virtualenvs/mysite-virtualenv/bin/python
Not overwriting existing python script /home/niktorthenat/.virtualenvs/mysite-virtualenv/bin/python (you must use /home/niktorthenat/.virtualenvs/mysite-virtualenv/bin/python3)
Installing setuptools, pip, wheel...
done.
(mysite-virtualenv) 15:19 ~ $ pip install pyTelegramBotAPI
DEPRECATION: Python 3.5 reached the end of its life on September 13th, 2020. Please upgrade your Python as Python 3.5 is no longer maintained. pip 21.0 will drop support for Python 3.5 in January 2021. pip 21.0 will remove support for this functionality.
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: pyTelegramBotAPI in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (3.7.6)
Requirement already satisfied: requests in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from pyTelegramBotAPI) (2.25.1)
Requirement already satisfied: chardet<5,>=3.0.2 in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from requests->pyTelegramBotAPI) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from requests->pyTelegramBotAPI) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from requests->pyTelegramBotAPI) (1.26.3)
Requirement already satisfied: certifi>=2017.4.17 in ./virtualenvs/mysite-virtualenv/lib/python3.5/site-packages (from requests->pyTelegramBotAPI) (2020.12.5)
(mysite-virtualenv) 15:20 ~ $ python provola-converto.py
```

Non apparirà più nessun messaggio, perché il nostro bot è attivo e sta funzionando. Provate a scrivere "ciao" o "buongiorno" sul vostro bot, e vedrete che risponderà come da vostro programma:



AUTOMAZIONE COMPUTER

Con **Lungo** è possibile gestire l'automazione del computer in modo davvero semplice. Ci sono diverse istruzioni che permettono il pieno controllo di tastiera e mouse.

POSIZIONEMOUSE

Se vogliamo conoscere le coordinate x e y del puntatore del mouse, possiamo usare l'istruzione **posizionemouse** che otterrà quelle informazioni e le potremo inserire in una variabile:

```
x=posizionemouse()
```

```
scrivi(x)
```

ed avremo come risultato:

```
Point(x=1027, y=1472)
Programma terminato
```

Dato che abbiamo ottenuto una lista di valori, possiamo accedere ad ogni singolo valore usando la sintassi `x[0]` per ottenere la posizione x, oppure `x[1]` per ottenere la posizione y:

```
x=posizionemouse()
```

```
scrivi("La coordinata x: ",x[0])
```

```
scrivi("La coordinata y: ",x[1])
```

```
La coordinata x: 1524
La coordinata y: 1487
Programma terminato
```

RISOLUZIONE SCHERMO

Per conoscere qual è la risoluzione dello schermo, quindi avere la larghezza in pixel e l'altezza in pixel, si usa l'istruzione **risoluzioneschermo**, che restituisce una lista di due valori per la larghezza in pixel e altezza in pixel:

```
x=risoluzioneschermo()
```

scrivi(x)

```
Size(width=2256, height=1504)  
Programma terminato
```

Oppure possiamo accedere ai singoli valori di larghezza e altezza, usando la seguente sintassi:

```
x=risoluzioneschermo()  
scrivi("Larghezza in pixel: ",x[0])  
scrivi("Altezza in pixel: ",x[1])
```

```
Larghezza in pixel: 2256  
Altezza in pixel: 1504  
Programma terminato
```

MUOVIMOUSE

L'istruzione **muovimouse** permette di spostare il mouse ad una determinata posizione dello schermo, inserendo le coordinate x e y che sono, rispettivamente, larghezza e altezza schermo. Se ad esempio voglio spostare il mouse alla posizione x=200 e y=600, scriverò:

```
muovimouse(200,600)
```

e il mouse si sposterà, o meglio, salterà immediatamente alla posizione 200,600.

Se invece voglio spostare il mouse, dalla posizione attuale, alla posizione 200,600 in un determinato tempo, aggiungerò l'opzione **insecondi**, indicando un valore che sarà considerato in secondi di tempo per spostarsi lentamente dalla posizione attuale a quella impostata. I numeri decimali equivalgono ai centesimi, quindi **2** equivale a due secondi, mentre, ad esempio **3.5** equivale a 3 secondi e mezzo:

```
muovimouse(200,600,insecondi=3)
```

Quando eseguirete l'istruzione sopra, vedrete il mouse spostarsi alla posizione 200,600 nel tempo di tre secondi.

TRASCINAMOUSE

L'istruzione **trascinamouse** permette di spostare il mouse verso una posizione, ma trascinando o selezionando qualcosa, è come se stessi spostando il mouse premendo contemporaneamente il tasto sinistro del mouse. Spostare il mouse tenendo premuto il tasto sinistro del mouse, equivale o a spostare un oggetto sul desktop, o a selezionare eventuale testo. Questa istruzione, quando attivata, rimane attiva, cioè il tasto sinistro del mouse rimarrà premuto anche a fine istruzione.

```
trascinamouse(0,0,insecondi=2)
```

Come per le istruzioni sopra, è possibile usare l'opzione **insecondi** per determinare un limite di tempo in cui eseguire il trascinamento.

FAICLICK

L'istruzione **faiclick** permette di premere uno dei pulsanti del mouse, in una determinata posizione dello schermo. Se non si specificano dei parametri, allora verrà sottointeso che il pulsante da premere sarà il pulsante sinistro del mouse.

Ad esempio, se vogliamo premere il pulsante sinistro del mouse alla posizione 10,20 scriveremo:

```
faiclick(10,20)
```

Se invece vogliamo che alla posizione 10,20 venga premuto il pulsante destro del mouse, scriveremo:

faiclick(10,20,pulsantesinistro)

Ricordiamo che possiamo usare le opzioni **pulsantesinistro**, oppure **pulsantedestro**, oppure **pulsantedimezzo**

E' anche possibile indicare quante volte vogliamo che il pulsante venga premuto. Se ad esempio vogliamo che venga premuto in rapida successione, per due volte, il pulsante sinistro del mouse alla posizione 10,20, useremo la proprietà **volte**:

faiclick(10,20,volte=2,pulsantesinistro)

Se infine vogliamo distanziare il tempo tra le due pressioni del pulsante sinistro del mouse, cioè vogliamo che venga premuto il pulsante sinistro del mouse una volta e poi, dopo 3 secondi, venga premuto un'altra volta, allora useremo la proprietà **volte** su **2** e la proprietà **insecondi** su **3**:

faiclick(10,20,volte=2,insecondi=3,pulsantesinistro)

PREMIPULSANTE E RILASCIAPULSANTE

L'istruzione **premipulsante** e **rilasciapulsante**, permettono, rispettivamente, di simulare la pressione di un pulsante del mouse, che pertanto rimane premuto, oppure rilasciare il pulsante del mouse, che equivale a togliere il dito dal pulsante.

Come anche le altre istruzioni, bisogna specificare in quale posizione x,y premere o rilasciare il pulsante.

premipulsante(10,10,pulsantesinistro)

oppure:

rilasciapulsante(10,10,pulsantesinistro)

E' possibile usare le proprietà **pulsantesinistro**, oppure **pulsantedimezzo** oppure **pulsantedestro**.

SCROLLAMOUSE

L'istruzione **scrollamouse** permette di fare uno scrolling, cioè di far ruotare la rotellina centrale del mouse, per tot pixel. Il numero da usare con questa istruzione sarà positivo, se voglio scrollare verso l'alto o negativo se voglio scrollare verso il basso.

E' opportuno usare l'istruzione **muovimouse** per posizionare il mouse nel punto in cui vogliamo iniziare lo scrolling. Se ad esempio voglio scrollare dalla posizione 200,1300 per 200 pixel verso l'alto, scriverò:

muovimouse(200,1300)

scrollamouse(200)

Se invece volevo scrollare verso il basso, allora userò un numero negativo:

muovimouse(200,1300)

scrollamouse(-200)

USALATASTIERA

L'istruzione **usalatastiera** permette di far usare la tastiera in modo automatizzato. E' sufficiente scrivere tra virgolette il testo da scrivere. E' consigliabile usare l'istruzione **faiclick** per indicare il punto dove bisogna iniziare a scrivere. Se ad esempio abbiamo aperto un editor di testo, che ha il foglio da scrivere magari anche in posizione x=40 e y=80, possiamo usare la seguente istruzione:

faiclick(60,80,pulsantesinistro)

usalatastiera("ciao a tutti")

In questo modo, il mouse cliccherà alla posizione 68,80 e poi inizierà a scrivere "ciao a tutti".

La scrittura sarà immediata. Sembrerà quasi un copia/incolla, ma in realtà scrive velocissimamente.

Per rallentare la scrittura del testo, si può usare la proprietà **faipausa**, indicando i secondi ed eventuali decimi di secondo, tra un carattere e l'altro.

faiclick(60,80,pulsantesinistro)

usalatastiera("ciao",faipausa=2)

Con l'esempio sopra scriverò la lettera c, poi fa una pausa di due secondi, poi scrive la lettera i, poi fa una pausa di due secondi, poi scrive la lettera a, poi fa una pausa di due secondi, poi scrive la lettera o.

Con l'istruzione **usalatastiera** è anche possibile impartire comandi di tasti speciali, come il tasto INVIO, oppure il tasto ELIMINA, ed altri, ma quando di usano questi comandi, bisogna metterli obbligatoriamente tra parentesi quadre.

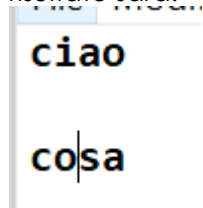
Si possono usare i comandi **tastodestra** (sposta il cursore a destra), **tastosinistra** (sposta il cursore a sinistra), **tastoinvio** (fa andare a capo), **tastoelimina** (preme il pulsante di eliminazione carattere alla sinistra del cursore).

Ricordarsi che i comandi di tastiera vanno scritti tra parentesi quadre.

Ecco un esempio:

```
faiclick(60,80,pulsantesinistro)
usalatastiera("ciao",faipausa=1)
usalatastiera([tastoinvio,tastoinvio])
usalatastiera("casa",faipausa=1)
usalatastiera([tastosinistra,tastosinistra,tastoelimina],faipausa=1)
usalatastiera("o",faipausa=1)
```

L'esempio sopra, prima scrive "ciao", poi va a capo di due righe, poi scrive "casa", poi sposta il cursore due volte a sinistra, cancella la lettera "a" di "casa", e poi scrive la lettera "o", così il risultato sarà:



```
ciao
cosa
```

COMBINAZIONETASTI

L'istruzione **combinazionetasti** permette di eseguire un comando combinato, tra più tasti, come ad esempio la tipica combinazione ctrl+c che serve per copiare una selezione o ctrl+v che serve ad incollare una selezione. Per combinare due tasti, vanno scritti i nomi dei tasti tra virgolette, separati da virgola.

Nell'esempio sotto eseguo la combinazione di tasti ctrl+v:

```
faiclick(60,80,pulsantesinistro)
combinazionetasti("ctrl","v")
```

GESTIONE GIOCHI

La gestione dei giochi grafici con **Lungo**, necessita la comprensione di diversi aspetti tecnici, non semplici ed immediati.

ESEMPIO RETTANGOLO DA MUOVERE

Per realizzare giochi, o azioni grafiche con questo tipo di sistema, prima di tutto bisogna segnalare a **Lungo** che si vuole procedere con l'esecuzione di istruzioni di grafica di gioco, e si usa l'istruzione **giocoinizio**. Si tenga presente che tutte le istruzioni di gioco, iniziano con la parola **gioco**:

```
giocoinizio()
```

poi si può impostare la grandezza della finestra in pixel, indicato una tuple contenente larghezza e altezza finestra. Una tuple di dati va scritta tra parentesi tonde, quindi, qui sotto, vedremo una tupla dentro altre due parentesi, usando l'istruzione **giocodimensionefinestra**:

```
alfa = giocodimensionefinestra((500, 500))
```

è possibile anche dare un titolo alla finestra nella barra del titolo con l'istruzione **giocotitolo**finestra:

```
giocotitolo("Sposta il rettangolo con le frecce")
```

poi possiamo impostare delle variabili che sono, ad esempio:

- **x** per la posizione x in cui disegnare l'oggetto da muovere;
- **y** per la posizione y in cui disegnare l'oggetto da muovere;
- **lg** per indicare la larghezza in pixel dell'oggetto che disegneremo da muovere;
- **at** per indicare l'altezza in pixel dell'oggetto che disegneremo da muovere;
- **vel** per indicare la velocità in pixel di spostamento dell'oggetto;

```
x = 200
y = 200
lg = 20
at = 20
vel = 10
```

poi si esegue un ciclo di istruzioni, impostate su **vero**, che determina un ciclo infinito, dove, con l'istruzione **giocofrequenza** indicheremo la frequenza di refresh, (cioè aggiornamento in fotogrammi) della grafica. Maggiore è il numero e minore è la velocità di movimento (anche se viene da pensare il contrario), si aggiunge un'istruzione **giococontrollaevento** che si occupa di verificare cosa accade durante il gioco, come ad esempio la pressione dei tasti. Poi si usa l'istruzione **tastopremuto** per catturare il tasto che è stato premuto dall'utente e in questo caso viene inserito nella variabile **tasto**; poi si controlla quale tasto freccia della tastiera è stato premuto, grazie alle istruzioni **giocotastosinistro**, **giocotastodestro**, **giocotastoinalto**, **giocotastoinbasso**; e nel caso di pressione di uno di quei tasti si eseguono le operazioni aritmetiche per far spostare il disegno sullo schermo, in base al valore della variabile **vel**, controllando che il disegno non esca dalla finestra; poi con l'istruzione **giocoriempimentofinestra** viene impostato il colore di sfondo della finestra attraverso i valori RGB (red, green, blue); poi si usa l'istruzione **giocodisegnarerettangolo** per dire a **Lungo** di disegnare un rettangolo, dentro la finestra **alfa**, del colore RGB tra parentesi tonde, e nella posizione **x,y** e di dimensione **lg** (che sta per larghezza) e **at** (che sta per altezza). Infine si deve inserire l'istruzione **giocoaggiornafinestra** per aggiornare e fare il refresh della finestra, che equivale a ridisegnare tutta la finestra con l'oggetto eventualmente spostato di posizione.

ciclo vero:

```
    giocofrequenza(60)
    giococontrollaevento()
    tasto = giocotastopremuto()
    se tasto[giocotastosinistro] e x>0:
        x -= vel
    se tasto[giocotastodestro] e x<500-lg:
        x += vel
    se tasto[giocotastoinalto] e y>0:
        y -= vel
    se tasto[giocotastoinbasso] e y<500-at:
        y += vel
    alfa.giocoriempimentofinestra((100, 50, 0))
    giocodisegnarerettangolo(alfa, (255, 0, 0), (x, y, lg, at))
    giocoaggiornafinestra()
```

Ecco il programma completo:

```
giocoinizio()
alfa = giocodimensionefinestra((500, 500))
giocotitolofinestra("Sposta il rettangolo con le frecce")
x = 200
y = 200
lg = 20
at = 20
vel = 10
ciclo vero:
    giocofrequenza(60)
    giococontrollaevento()
    tasto = giocotastopremuto()
    se tasto[giocotastosinistro] e x>0:
        x -= vel
```

```

se tasto[giocotastodestro] e x<500-lg:
    x += vel
se tasto[giocotastoinalto] e y>0:
    y -= vel
se tasto[giocotastoinbasso] e y<500-at:
    y += vel
alfa.giocoriempimentofinestra((100, 50, 0))
giocodisegnarettangolo(alfa, (255, 0, 0), (x, y, lg, at))
giocoaggiornafinestra()

```



PROGRAMMI DI ESEMPIO

ESEMPIO CALCOLA CIRCONFERENZA CERCHIO

Questo programma permette di calcolare la circonferenza di un cerchio, conoscendo la misura del raggio o del diametro.

Il programma prima chiederà se conosciamo la misura del raggio o del diametro, poi, in base alla scelta, passerà al calcolo della circonferenza, chiedendo la misura, rispettivamente, del raggio o del diametro.

Ricordiamo che la formula della circonferenza di un cerchio si ha:

- $(\text{raggio} * 2) * 3.14$
- $\text{Diametro} * 3.14$

funzione calcoladaraggio():

```

r=intero(domanda("Misura del raggio in centimetri: "))
scrivi("La circonferenza del cerchio misura:",((r*2)*pigreco))

```

funzione calcoladadiametro():

```

d=intero(domanda("Misura del diametro in centimetri: "))
scrivi("La circonferenza del cerchio misura:",(d*pigreco))

```

```

scrivi("Formule sul cerchio")
scrivi("-----")
scrivi("\n")
scrivi("1. calcola circonferenza dal raggio")
scrivi("2. calcola circonferenza dal diametro")
scrivi("\n")
scelta=domanda("Fai la tua scelta...")
se scelta=="1":
    calcoladaraggio()
altrimentise scelta=="2":
    calcoladadiametro()
altrimenti:
    scrivi("Scelta sbagliata")

```

ESEMPIO ACCESSO PASSWORD (SEMPLICE)

Questo programma chiede il nome utente e password e se verrà scritto come nome utente "nik" e password "123" allora verranno mostrate le password personali dei vari social, altrimenti non si vedranno le password:

```

scrivi("ACCESSO ALLE PASSWORD PERSONALI")
scrivi("\n")
nome=domanda("Nome utente: ")
pwd=domanda("Password: ")
se nome=="nik" anche pwd=="123":
    scrivi("Accesso consentito")
    scrivi("Le tue password personali sono:")
    scrivi("Facebook: 123ciccio4")
    scrivi("Twitter: 677h&7")
    scrivi("Google: 47djj8")
altrimenti:
    scrivi("Nome utente o password sbagliate")

```

ESEMPIO ACCESSO PASSWORD (SEMPLICE – VERSIONE GRAFICA)

Questo programma, a finestre, chiede la password e se verrà scritta la password "1234" allora verrà mostrata una finestra con le varie password dei social:

```

funzione provola():
    aa=y.prendi()
    se aa=="1234":
        beta=finestra()
        beta.titolo("Mie password")
        beta.grandezza("300x300")
        g1=etichetta(beta,testo="Password di Google 232321").impacchetta()
        g2=etichetta(beta,testo="Password di Facebook fgwojij").impacchetta()
    altrimenti:
        finestramessaggio("ATTENZIONE","Password sbagliata")

alfa=finestra()
alfa.titolo("Gestore password")
alfa.grandezza("300x300")
x=etichetta(alfa,testo="Digita la password").impacchetta()
y=caselladitesto(alfa,larghezza=10)
y.impacchetta()
z=pulsante(alfa,testo="ACCEDI",esegui=provola).impacchetta()
alfa.ciclico()

```

ESEMPIO CHE CERCA LA PAROLA 'CIAO' SCRITTA DENTRO UNA CASELLA DI TESTO

```
funzione provola():
    frase=y.prendi(1.0,FINE)
    se "ciao" in frase:
        x.configura(testo="c'è la parola ciao")
    altrimenti:
        x.configura(testo="la parola ciao non c'è")

alfa=finestra()
alfa.titolo("Generatore di password")
alfa.grandezza("300x300")
x=etichetta(alfa,testo="Digita la password")
x.impacchetta()
y=caselladitestoscrollabile(alfa,larghezza=25,altezza=15)
y.impacchetta()
z=pulsante(alfa,testo="CERCA ciao",esegui=provola).impacchetta()
alfa.ciclico()
```

ESEMPIO DI SOFTWARE PER TRASCRIZIONE FILE AUDIO

Questo programma legge un file audio denominato "esempio.wav", che deve trovarsi nella stessa cartella in cui si trova il programma, e lo trascrive in testo.

```
alfa=riconoscimentovocale()
beta="esempio.wav"
usa fileaudio(beta) come gamma:
    voce=alfa.registrazione(gamma)
    scrivi(alfa.riconoscimentogoogle(voce,linguaggio="it-IT"))
```

ESEMPIO DI SOFTWARE CHE CHIEDE UNA FRASE DA RIPETERE A VOCE

Con questo programma ci viene chiesto di digitare un testo. Quando premeremo il tasto INVIO, il testo digitato sarà memorizzato nel file "esempio.mp3" e letto a voce dal computer.

```
alfa=domanda("Cosa vuoi che ripeto con la mia voce? ")
beta=parla(alfa,italiano)
beta.salva("esempio.mp3")
apriaudio("esempio.mp3")
```

ESEMPIO DI REGISTRAZIONE DA MICROFONO E RIPRODUZIONE DEL REGISTRATO

Con questo programma è possibile registrare la propria voce dal microfono del computer, per 5 secondi, quindi a fine registrazione verrà riprodotto il file audio registrato.

```
fr=44100
sec=5
scrivi("Ho aperto il microfono...parla...")
alfa=microfono(intero(sec*fr),frequenza=fr,canali=2)
attendifineregistrazione()
scrivi("Chiudo il microfono e ripeto cosa hai detto")
memorizza("esempio.wav",fr,alfa)
apriaudio("esempio.wav")
```

ESEMPIO DI REGISTRAZIONE DA MICROFONO E TRASCRIZIONE TESTUALE DEL REGISTRATO

Con questo programma è possibile registrare la propria voce dal microfono del computer, per 5 secondi, quindi a fine registrazione verrà trascritto in testo il file audio registrato.

```
fr=44100
```

```
sec=5
```

```
scrivi("Ho aperto il microfono...parla...")
```

```
alfa=microfono(intero(sec*fr),frequenza=fr,canali=2)
```

```
attendifineregistrazione()
```

```
scrivi("Chiudo il microfono e ripeto cosa hai detto")
```

```
memorizza('esempio.wav', alfa, fr, 'PCM_24')
```

```
alfa1=riconoscimentovocale()
```

```
beta="esempio.wav"
```

```
usa fileaudio(beta) come gamma:
```

```
    voce=alfa1.registrazione(gamma)
```

```
    scrivi(alfa1.riconoscimentogoogle(voce,linguaggio="it-IT"))
```