Python PROGRAMMING

Piero Rivoira
Istituto Agrario Penna – Asti
piero.rivoira@yahoo.it

Creiamo il nostro profilo su Ubuntu

questo è un commento!

lanciamo il terminale per aggiungere un nuovo profilo

ctrl+alt+t

```
$ sudo adduser nome_battesimo -uid 663
$ pw di labx

# per eliminare un profilo utente (in caso di errore)
$ sudo deluser --remove-home nome_utente
$ pw di labx

# per recuperare la passwd dimenticata
# sostituire <user name> con il proprio nome utente
$ sudo passwd user name
$ pw di labx
$ nuova pw
```

Creiamo il nostro profilo su Ubuntu

acquisiamo i privilegi dell'amministratore di sistema modificando il file di configurazione del Sistema Operativo (SO) `/etc/sudoers`

```
$ sudo visudo
```

questo comando lancia l'editor di testo **nano** per creare ed aprire il file `/etc/sudoers.tmp` in modalità scrittura (per poterlo modificare); tale file è una copia di backup di `/etc/sudoers`

Creiamo il nostro profilo su Ubuntu

```
GNU nano 7.2                                  /etc/sudoers.tmp
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# This fixes CVE-2005-4890 and possibly breaks some versions of kdesu
# (#1011624, https://bugs.kde.org/show_bug.cgi?id=452532)
Defaults        use_pty

# This preserves proxy settings from user environments of root
# equivalent users (group sudo)
#Defaults:%sudo env_keep += "http_proxy https_proxy ftp_proxy all_proxy no_proxy"

# This allows running arbitrary commands, but so does ALL, and it means
# different sudoers have their choice of editor respected.
#Defaults:%sudo env_keep += "EDITOR"

# Completely harmless preservation of a user preference.
#Defaults:%sudo env_keep += "GREP_COLOR"

# While you shouldn't normally run git as root, you need to with etckeeper
#Defaults:%sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_*"

# Per-user preferences; root won't have sensible values for them.
#Defaults:%sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults:%sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
#Defaults:%sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

                                [ Lette 59 righe ]
^G Guida       ^O Salva       ^W Cerca       ^K Taglia      ^T Esegui      ^C Posizione   M-U Annulla    M-A Contrassegna M-] Parentesi  M-Q Precedente ^B Indietro
^X Esci        ^R Inserisci   ^\ Sostituisci ^U Incolla     ^J Giustifica  ^/ Vai a riga  M-E Ripeti     M-6 Copia        ^Q Cerca ind. M-W Successiva ^F Avanti
```

Creiamo il nostro profilo su Ubuntu

```
 GNU nano 7.2                                  /etc/sudoers.tmp
# This file MUST be edited with the 'visudo' command as root.
# equivalent users (group sudo)
#Defaults:%sudo env_keep += "http_proxy https_proxy ftp_proxy all_proxy no_proxy"

# This allows running arbitrary commands, but so does ALL, and it means
# different sudoers have their choice of editor respected.
#Defaults:%sudo env_keep += "EDITOR"

# Completely harmless preservation of a user preference.
#Defaults:%sudo env_keep += "GREP_COLOR"

# While you shouldn't normally run git as root, you need to with etckeeper
#Defaults:%sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_*"

# Per-user preferences; root won't have sensible values for them.
#Defaults:%sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults:%sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
#Defaults:%sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
piero   ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d

^G Guida      ^O Salva       ^W Cerca       ^K Taglia      ^T Esegui      ^C Posizione   M-U Annulla    M-A Contrassegna M-] Parentesi  M-Q Precedente ^B Indietro
^X Esci       ^R Inserisci   ^\ Sostituisci ^U Incolla     ^J Giustifica  ^/ Vai a riga  M-E Ripeti     M-6 Copia        ^Q Cerca ind. M-W Successiva ^F Avanti
```

Elenco degli amministratori di sistema
portarsi con il cursore sull'ultima riga della lista
alt-6 # copia
ctrl-u # incolla l'intera riga
inserire il proprio nome utente
ctrl-o # per salvare
cancellare l'estensione .tmp
ctrl-x # per chiudere **nano**
alt-u # in caso di errore
(per annullare l'ultimo comando inserito)

Elenco degli amministratori di sistema

Per iniziare...

```
piero@piero-XPS-9320:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 22.04.5 LTS
Release:  22.04
Codename: jammy
piero@piero-XPS-9320:~$

piero@piero-XPS-9320:~$ python3 --version
Python 3.10.12
piero@piero-XPS-9320:~$
```

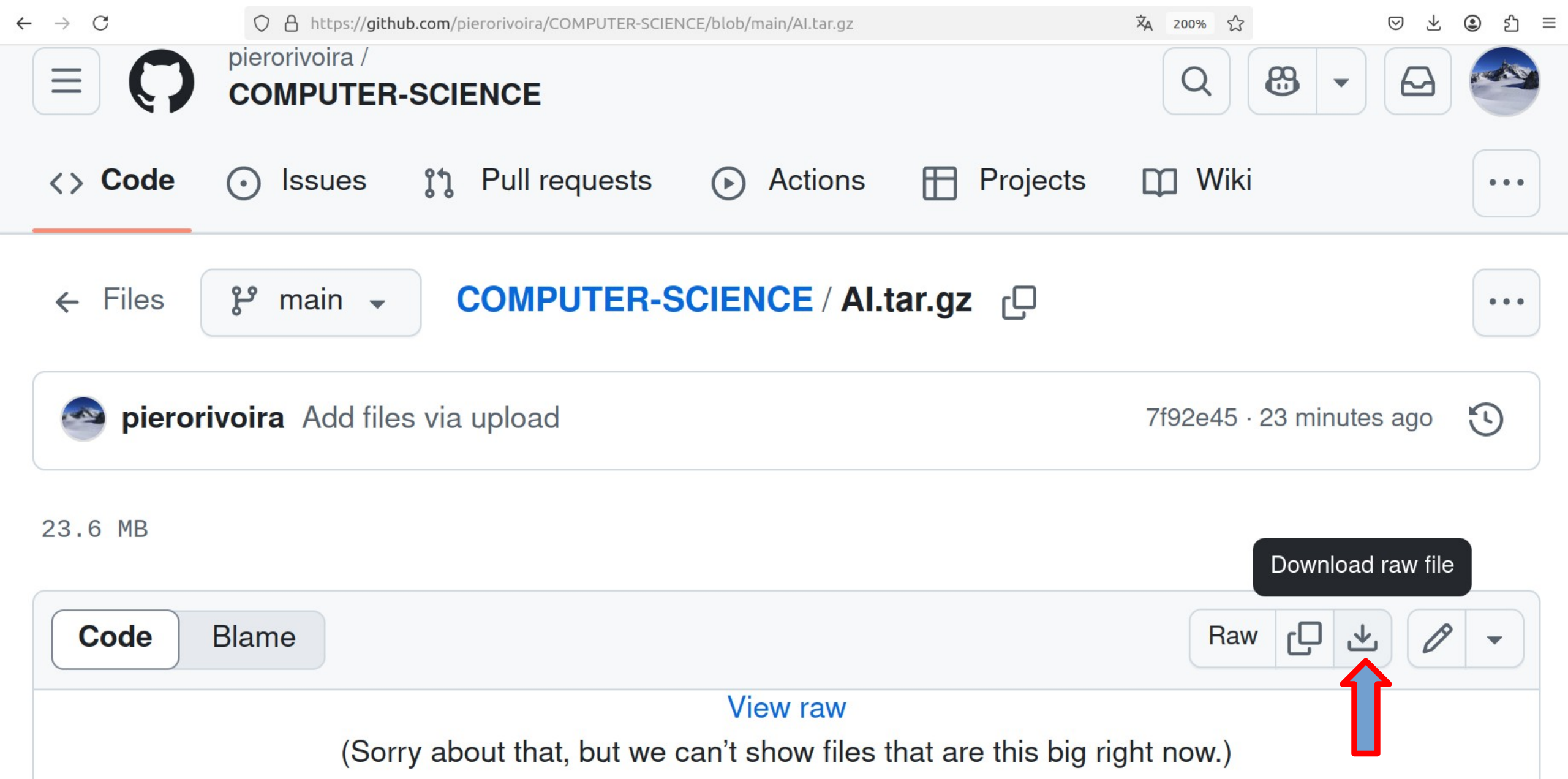
Per iniziare...

```
piero@piero-XPS-9320:~$ python3
Python 3.10.12 (main, Nov  6 2024, 20:22:13) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import dill, os, pickle
>>> import datetime
>>> now = datetime.datetime.now()
>>> now
datetime.datetime(2024, 12, 6, 9, 8, 20, 382891)
>>> day = datetime.datetime.now().strftime("%A")
>>> day
'Friday'
>>> current_time = now.strftime("%H:%M:%S")
>>> current_time
'09:08:20'
>>> user = os.getlogin()
>>> user
'piero'
>>>
```

Creare un archivio

```
piero@piero-XPS-9320:~$ tar -czvf AI.tar.gz COMPUTER_SCIENCE/AI
```

Scaricare un archivio da Github



Estrarre il contenuto dell'archivio

```
piero@piero-XPS-9320:~$ cd Scaricati/
piero@piero-XPS-9320:~/Scaricati$ file AI.tar.gz
AI.tar.gz: gzip compressed data, from Unix, original size modulo 2^32 29859840

piero@piero-XPS-9320:~/Scaricati$ tar -xvzf AI.tar.gz
COMPUTER_SCIENCE/AI/
COMPUTER_SCIENCE/AI/unpacker.py
COMPUTER_SCIENCE/AI/Leguminosa.py
COMPUTER_SCIENCE/AI/Foraggio.py
COMPUTER_SCIENCE/AI/trova_massimo.py
COMPUTER_SCIENCE/AI/blocco_di_testo_last_rows.py
COMPUTER_SCIENCE/AI/tartaglia_modulo.py
COMPUTER_SCIENCE/AI/triangolo_di_Tartaglia.ods
COMPUTER_SCIENCE/AI/blocco_di_testo_mod.py
COMPUTER_SCIENCE/AI/metabolismo_basale.xlsx
COMPUTER_SCIENCE/AI/initdata_mod.py
COMPUTER_SCIENCE/AI/script5.py
COMPUTER_SCIENCE/AI/rubrica.txt
COMPUTER_SCIENCE/AI/Sottoprodotto.py
COMPUTER_SCIENCE/AI/leggi_dati_e_trova_massimo.py
...
```

Estrarre il contenuto dell’archivio

```
piero@piero-XPS-9320:~/Scaricati$ ls
AI.tar.gz                          eap1499-sup-0004-appendixs4.txt
ALIMENTI.tar.gz                    infoalunno_prospettoassenzeorarie.xlsx
COMPUTER_SCIENCE                   vettori.txt
eap1499-sup-0002-appendixs2.pdf    wolves.csv
eap1499-sup-0002-appendixs2.txt    wolves.xlsx
eap1499-sup-0004-appendixs4.pdf
piero@piero-XPS-9320:~/Scaricati$ cd COMPUTER_SCIENCE/
piero@piero-XPS-9320:~/Scaricati/COMPUTER_SCIENCE$ ls
AI
piero@piero-XPS-9320:~/Scaricati/COMPUTER_SCIENCE$ cd AI
piero@piero-XPS-9320:~/Scaricati/COMPUTER_SCIENCE/AI$
```

Per iniziare...

```
piero@piero-XPS-9320:~/Scaricati/COMPUTER_SCIENCE/AI$ ls
alimenti_26_febbraio_25.py
alimenti.ods
alimenti.py
blocco_di_testo_last_rows.py
blocco_di_testo_mod.py
blocco_di_testo.py
CARTELLA
Cereale.py
DATA
dati_istogramma.txt
docs-pdf
eleva_a_potenza.py
Farina_di_estrazione.py
Fonte_di_fibra.py
Foraggio.py
GRAFICI
importa_fieno_prato_stabile_primo_taglio.py
initdata_mod.py
...
```

Per iniziare...

```
piero@piero-XPS-9320:~/Scaricati/COMPUTER_SCIENCE/AI$ ls *.py
alimenti_26_febbraio_25.py
alimenti.py
blocco_di_testo_last_rows.py
blocco_di_testo_mod.py
blocco_di_testo.py
Cereale.py
eleva_a_potenza.py
Farina_di_estrazione.py
Fonte_di_fibra.py
Foraggio.py
importa_fieno_prato_stabile_primo_taglio.py
initdata_mod.py
initdata.py
inserisci_gruppo.py
leggi_dati_e_trova_massimo.py
Leguminosa.py
maxarray.py
more.py
packdlg.py
packer.py
parmfile.py
```

Per iniziare...

```
piero@piero-XPS-9320:~$ pwd
/home/piero
piero@piero-XPS-9320:~$ mkdir COMPUTER_SCIENCE
piero@piero-XPS-9320:~$ cd COMPUTER_SCIENCE/
piero@piero-XPS-9320:~/COMPUTER_SCIENCE$ pwd
/home/piero/COMPUTER_SCIENCE
piero@piero-XPS-9320:~/COMPUTER_SCIENCE$ mkdir AI
piero@piero-XPS-9320:~/COMPUTER_SCIENCE$ cd AI
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ pwd
/home/piero/COMPUTER_SCIENCE/AI
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$
```

Per iniziare...

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ cd
piero@piero-XPS-9320:~$ pwd
/home/piero
piero@piero-XPS-9320:~$ python3
Python 3.10.12 (main, Feb  4 2025, 14:57:36) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.getcwd()
'/home/piero'
>>> os.getlogin()
'piero'
>>> user = os.getlogin()
>>> user
'piero'
>>> os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
>>> cartella = os.getcwd()
>>> cartella
'/home/piero/COMPUTER_SCIENCE/AI'
>>> os.chdir('/home/%s' % user)
>>> os.getcwd()
'/home/piero'
```

Elencare i file in una dir: the hard way

```
>>> os.chdir('/home/%s/Scaricati/COMPUTER_SCIENCE/AI' % user)
>>> os.getcwd()
'/home/piero/Scaricati/COMPUTER_SCIENCE/AI'
>>> for line in os.popen('ls *.py'): print(line, end='')
...
alimenti_26_febbraio_25.py
alimenti.py
blocco_di_testo_last_rows.py
blocco_di_testo_mod.py
blocco_di_testo.py
Cereale.py
eleva_a_potenza.py
Farina_di_estrazione.py
Fonte_di_fibra.py
Foraggio.py
importa_fieno_prato_stabile_primo_taglio.py
initdata_mod.py
initdata.py
inserisci_gruppo.py
leggi_dati_e_trova_massimo.py
Leguminosa.py
...
>>>
```

Salvare un oggetto in un file esterno

```
piero@piero-XPS-9320:~$ sudo apt install python3-dill
piero@piero-XPS-9320:~$ python3
>>> import dill, os, pickle
>>> species_1 = ['Tachyglossus aculeatus', 'Monotremata', 2.404, 2725, 30.7, 19.7]
>>> species_2 = ['Plecotus auritus', 'Chiroptera', 0.082, 10.25, 35.5, 4.4]
>>> taxa = (species_1, species_2)
>>> file_name = 'piero.pkl'
>>> with open(file_name, 'wb') as file:
...     dill.dump(taxa, file)
...
>>> exit()
```

Salvare un oggetto in un file esterno

```
>>> import dill, os, pickle
>>> os.getcwd()
'/home/piero'
>>> user = os.getlogin()
>>> os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
>>> os.getcwd()
'/home/piero/COMPUTER_SCIENCE/AI'
>>> with open("piero.pkl", "rb") as file:
...     taxa = pickle.load(file)
...
>>> taxa
(['Tachyglossus aculeatus', 'Monotremata', 2.404, 2725, 30.7, 19.7], ['Plecotus
auritus', 'Chiroptera', 0.082, 10.25, 35.5, 4.4])
>>> taxa[0]
['Tachyglossus aculeatus', 'Monotremata', 2.404, 2725, 30.7, 19.7]
>>> taxa[1]
['Plecotus auritus', 'Chiroptera', 0.082, 10.25, 35.5, 4.4]
>>>
```

Salvare un oggetto in un file esterno

```
>>> NAME, ORDER, METABOLIC_RATE, BODY_MASS, BODY_TEMP, AMBIENT_TEMP = range(6)
>>> taxa[0][NAME]
'Tachyglossus aculeatus'
>>> taxa[0][ORDER]
'Monotremata'
>>>
```

Il ciclo for

```
>>> for i in taxa:
...     print(i)
...
['Tachyglossus aculeatus', 'Monotremata', 2.404, 2725, 30.7, 19.7]
['Plecotus auritus', 'Chiroptera', 0.082, 10.25, 35.5, 4.4]
>>>
```

La gerarchia concettuale di Python

1. I programmi sono composti da moduli

2. I moduli contengono affermazioni ("*statements*")

3. Le affermazioni contengono espressioni

4. Le espressioni creano e modificano gli oggetti

Il nostro primo programma in Python: script1.py

```
# A first Python script
# script1.py

import os, sys
```

Importa due diversi **moduli** di Python (**`os`** e **`sys`**) per identificare il nome dell'utente e del sistema operativo

```
print(os.getlogin())
print(sys.platform)
print(2**100)
x = 'Python è una figata!'
```

Crea la variabile **`x`** e le assegna una stringa di caratteri

```
print(x * 8)
```

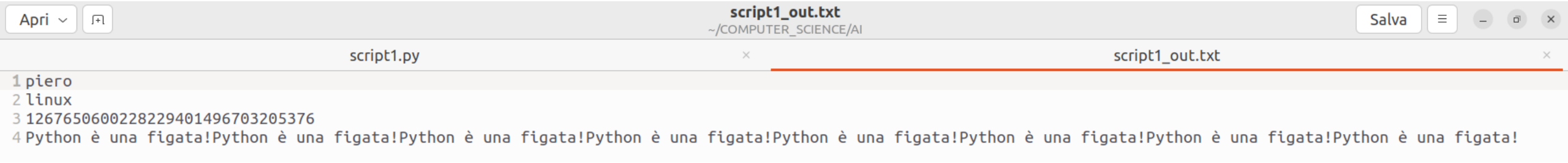
Esegue la chiamata della funzione **`print`** per 4 volte, per inviare i risultati all'output standard (lo schermo)

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 script1.py
piero
linux
1267650600228229401496703205376
Python è una figata!Python è una figata!Python è una figata!Python è una figata!Python
è una figata!Python è una figata!Python è una figata!Python è una figata!
```

Stream redirection

```
$ python3 script1.py > script1_out.txt
```

- L’output viene scritto nel file di testo <**script1_out.txt**>



Importare i moduli

Ogni file di codice sorgente scritto in linguaggio Python, il cui nome abbia l'estensione *.py*, è un **modulo** e può essere importato

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3
Python 3.10.12 (main, Nov  6 2024, 20:22:13) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import script1
piero
linux
126765060022822940149670320537 6
Python è una figata!Python è una figata!Python è una figata!Python è una
figata!Python è una figata!Python è una figata!Python è una figata!Python è
una figata!
```

Importare i moduli

- I moduli vengono utilizzati come *librerie* di strumenti
- Un modulo è un pacchetto (*package*) di nomi di variabili, noto come *namespace*
- I nomi (attributi) designano un oggetto specifico
- Importando un modulo si ottiene accesso a tutti i nomi assegnati all’inizio del file del modulo stesso

Da dove vengono importati i moduli?

```
>>> import sys
>>> sys.path
['', '/usr/lib/python310.zip', '/usr/lib/python3.10', '/usr/lib/python3.10/lib-
dynload', '/usr/local/lib/python3.10/dist-packages', '/usr/lib/python3/dist-packages']
>>>
```

Scaricare un file da internet usando il terminale

```
piero@piero-XPS-9320:~/Scaricati$ wget
https://raw.githubusercontent.com/pierorivoira/COMPUTER-SCIENCE/refs/heads/main/taxa.py
--2025-03-27 08:34:49--
https://github.com/pierorivoira/COMPUTER-SCIENCE/raw/refs/heads/main/taxa.py
Risoluzione di github.com (github.com)... 140.82.121.4
Connessione a github.com (github.com)|140.82.121.4|:443... connesso.
Richiesta HTTP inviata, in attesa di risposta... 302 Found
Posizione: https://raw.githubusercontent.com/pierorivoira/COMPUTER-SCIENCE/refs/heads/
main/taxa.py [segue]
--2025-03-27 08:34:50--  https://raw.githubusercontent.com/pierorivoira/COMPUTER-
SCIENCE/refs/heads/main/taxa.py
Risoluzione di raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connessione a raw.githubusercontent.com (raw.githubusercontent.com)|
185.199.111.133|:443... connesso.
Richiesta HTTP inviata, in attesa di risposta... 200 OK
Lunghezza: 35851 (35K) [text/plain]
Salvataggio in: ‘taxa.py’

taxa.py             100%[=================>]  35,01K  --.-KB/s    in 0,08s

2025-03-27 08:34:50 (460 KB/s) - ‘taxa.py’ salvato [35851/35851]

piero@piero-XPS-9320:~/Scaricati$
```

Importare i moduli: il file taxa.py

Ogni file di codice sorgente scritto in linguaggio Python, il cui nome abbia l'estensione <*.py*>, è un **modulo** e può essere importato

```
NA = 'Not Available'
species_1 = ['Tachyglossus aculeatus', 2.404, 2725, 30.7, 19.7]
species_2 = ['Zaglossus bruijni',6.778,10300,30.8,16.2]
species_3 = ['Ornithorhynchus anatinus',1.082,693,32.1,12.9]
species_4 = ['Caluromys derbianus',1.255,329,35,19.3]
species_5 = ['Chironectes minimus',2.549,935,35,21.5]
species_6 = ['Didelphis marsupialis',3.185,1165,35,23.5]
species_7 = ['Didelphis virginiana',4.641,2488,35,15.4]
species_8 = ['Gracilinanus microtarsus',0.106,13,35,16.8]
species_9 = ['Lutreolina crassicaudata',2.265,812,35.8,19.1]
species_10 = ['Marmosa robinsoni',0.547,122,34,19.3]
...
```

Importare i moduli: il file taxa.py

import, from, reload

```
>>> import os
>>> user = os.getlogin()
>>> os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
>>> os.getcwd()
'/home/piero/COMPUTER_SCIENCE/AI'
>>> import taxa
>>> taxa.species_1
['Tachyglossus aculeatus', 2.404, 2725, 30.7, 19.7]
>>> taxa.species_2
['Zaglossus bruijni', 6.778, 10300, 30.8, 16.2]
>>> taxa.species_3
['Ornithorhynchus anatinus', 1.082, 693, 32.1, 12.9]
>>> taxa.species_3[1]
1.082
>>>
```

Importare i moduli: il file taxa.py

import, from, reload

```
>>> NAME, METABOLIC_RATE, BODY_MASS, BODY_TEMP, AMBIENT_TEMP = range(5)
>>> taxa.species_3[NAME]
'Ornithorhynchus anatinus'
>>> taxa.species_3[BODY_MASS]
693
>>>
```

Importare i moduli: il file taxa.py

Quali sono la temperatura corporea ed il tasso metabolico basale del coyote?

```
>>> from taxa import species_582
>>> species_582
['Canis latrans', 14.99, 10000, 37, 5.5]
>>> species_582[3]
37
>>> print('The body temperature of %s is %f' % (species_582[0],
species_582[3]))
The body temperature of Canis latrans is 37.000000
>>> print('The body temperature of %s is %d' % (species_582[0],
species_582[3]))
The body temperature of Canis latrans is 37
>>> print('The body temperature of %s is %f°C' % (species_582[0],
species_582[3]))
The body temperature of Canis latrans is 37.000000°C
>>> print('The coyote (%s) has a basal metabolic rate of %f watt' %
(species_582[0], species_582[1]))
The coyote (Canis latrans) has a basal metabolic rate of 14.990000 watt
>>>
```

Il programma initdata.py

```
# initialize data to be stored in files, pickles, shelves
import os
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
# records
species_1 = {'NAME': 'Tachyglossus aculeatus', 'ORDER': 'Monotremata', 'METABOLIC RATE
(Watt)': 2.404, 'BODY MASS': 2725, 'BODY TEMPERATURE': 30.7, 'AMBIENT TEMPERATURE':
19.7}
species_2 = {'NAME': 'Plecotus auritus', 'ORDER': 'Chiroptera', 'METABOLIC RATE
(Watt)': 0.082, 'BODY MASS': 10.25, 'BODY TEMPERATURE': 35.5, 'AMBIENT TEMPERATURE':
4.4}
# database
db = {}
db['species_1'] = species_1
db['species_2'] = species_2

if __name__ == '__main__':
# se questo modulo è il programma principale, ossia viene eseguito e non importato da un altro
# programma o dall'interprete dei comandi di python
    for key in db:
        print(key, '=>\n   ', db[key])
```

Il programma initdata.py

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 initdata.py
species_1 =>
    {'NAME': 'Tachyglossus aculeatus', 'ORDER': 'Monotremata', 'METABOLIC RATE (Watt)':
2.404, 'BODY MASS': 2725, 'BODY TEMPERATURE': 30.7, 'AMBIENT TEMPERATURE': 19.7}
species_2 =>
    {'NAME': 'Plecotus auritus', 'ORDER': 'Chiroptera', 'METABOLIC RATE (Watt)': 0.082,
'BODY MASS': 10.25, 'BODY TEMPERATURE': 35.5, 'AMBIENT TEMPERATURE': 4.4}


>>> exec(open('initdata.py').read())
species_1 =>
    {'NAME': 'Tachyglossus aculeatus', 'ORDER': 'Monotremata', 'METABOLIC RATE (Watt)':
2.404, 'BODY MASS': 2725, 'BODY TEMPERATURE': 30.7, 'AMBIENT TEMPERATURE': 19.7}
species_2 =>
    {'NAME': 'Plecotus auritus', 'ORDER': 'Chiroptera', 'METABOLIC RATE (Watt)': 0.082,
'BODY MASS': 10.25, 'BODY TEMPERATURE': 35.5, 'AMBIENT TEMPERATURE': 4.4}
>>>
```

Il programma initdata.py

```
>>> from initdata import db
>>> db
{'species_1': {'NAME': 'Tachyglossus aculeatus', 'ORDER': 'Monotremata', 'METABOLIC
RATE (Watt)': 2.404, 'BODY MASS': 2725, 'BODY TEMPERATURE': 30.7, 'AMBIENT
TEMPERATURE': 19.7}, 'species_2': {'NAME': 'Plecotus auritus', 'ORDER': 'Chiroptera',
'METABOLIC RATE (Watt)': 0.082, 'BODY MASS': 10.25, 'BODY TEMPERATURE': 35.5, 'AMBIENT
TEMPERATURE': 4.4}}
>>>
```

Il programma initdata_mod.py

```
# initialize data to be stored in files, pickles, shelves
import os
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
# records
species_1 = {'NAME': 'Tachyglossus aculeatus', 'ORDER': 'Monotremata', 'METABOLIC RATE
(Watt)': 2.404, 'BODY MASS': 2725, 'BODY TEMPERATURE': 30.7, 'AMBIENT TEMPERATURE':
19.7}
species_2 = {'NAME': 'Plecotus auritus', 'ORDER': 'Chiroptera', 'METABOLIC RATE
(Watt)': 0.082, 'BODY MASS': 10.25, 'BODY TEMPERATURE': 35.5, 'AMBIENT TEMPERATURE':
4.4}
# database
db = {}
db['species_1'] = species_1
db['species_2'] = species_2

# if __name__ == '__main__':
# questa riga verrà ignorata, quindi il successivo ciclo for sarà eseguito anche in caso di importazione del
# modulo initdata_mod
for key in db:
   print(key, '=>\n   ', db[key])
```

Il programma initdata_mod.py

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 initdata_mod.py
species_1 =
    {'NAME': 'Tachyglossus aculeatus', 'ORDER': 'Monotremata', 'METABOLIC RATE (Watt)':
2.404, 'BODY MASS': 2725, 'BODY TEMPERATURE': 30.7, 'AMBIENT TEMPERATURE': 19.7}
species_2 =
    {'NAME': 'Plecotus auritus', 'ORDER': 'Chiroptera', 'METABOLIC RATE (Watt)': 0.082,
'BODY MASS': 10.25, 'BODY TEMPERATURE': 35.5, 'AMBIENT TEMPERATURE': 4.4}
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$
```

Il funzionamento del programma, quando viene eseguito come programma principale (`main`) non cambia ma…

```
>>> from initdata_mod import db
species_1 =
    {'NAME': 'Tachyglossus aculeatus', 'ORDER': 'Monotremata', 'METABOLIC RATE (Watt)':
2.404, 'BODY MASS': 2725, 'BODY TEMPERATURE': 30.7, 'AMBIENT TEMPERATURE': 19.7}
species_2 =
    {'NAME': 'Plecotus auritus', 'ORDER': 'Chiroptera', 'METABOLIC RATE (Watt)': 0.082,
'BODY MASS': 10.25, 'BODY TEMPERATURE': 35.5, 'AMBIENT TEMPERATURE': 4.4}
>>>
```

Importare i moduli: il file parmfile.py

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ wget
https://raw.githubusercontent.com/pierorivoira/COMPUTER-SCIENCE/refs/heads/
main/parmfile.py
...
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ gedit parmfile.py
```

```
################################################################################
#_Parmfile_for_Migrate_5.0.4(git:distribution_version)_May_09_2022
#_DO_NOT_REMOVE_THE_FIRST_TWO_LINES!
#
#_generated_automatically_on
#_Thu_Apr__6_08:44:47_2023
#
#_please_report_problems_to_Peter_Beerli
#__email:_beerli@fsu.edu
#__http://popgen.sc.fsu.edu/migrate.html
################################################################################
#
################################################################################
#_General_options
################################################################################
#
#_Interactive_or_batch_job_usage
#___Syntax:_menu=<_YES_|_NO_>_
#_For_batch_runs_it_needs_to_be_set_to_NO
#_menu='YES'
#
#_Specification_of_length_of_names_of_individuals_(default=10)
#____Syntax:_nmlength=<INTEGER_between_0_..._30>
nmlength=12
#
#
#_If_you_long_runs_fails_because_of_timelimits
#_AND_you_have_specified_bayes_allfile='YES'....
#_then_you_can_recover_and_continue
#_by_setting_recover_to_yes,_default_is_NO
#_This_option_fails_if_your_bayesallfile_is_complete!
#_At_the_moment,_I_have_no_experience_about_failure_rate_etc._[March_30_2015]
#__________recover=<YES_|_NO>_
#
################################################################################
#_Data_options
################################################################################
#
#_Several_different_main_datatypes_are_possible:
#_INFINITE_ALLELE:_usable_for_electrophoretic_markers,
#___________________other_markers_with_unknown_mutation_model
#_STEPWISE_MUTATION:_usable_for_microsatellite_data_or
#___________________other_markers_with_stepwise_change
```

Importare i moduli: il file parmfile.py

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ gedit parmfile.py
#######################################################################
#_Parmfile_for_Migrate_5.0.4(git:distribution_version)_May_09_2022
#_DO_NOT_REMOVE_THE_FIRST_TWO_LINES!
#
#_generated_automatically_on
#_Thu_Apr__6_08:44:47_2023
#
#_please_report_problems_to_Peter_Beerli
#__email:_beerli@fsu.edu
#__http://popgen.sc.fsu.edu/migrate.html
#######################################################################
#
#######################################################################
#_General_options
#######################################################################
#
#_Interactive_or_batch_job_usage
#___Syntax:_menu=<_YES_|_NO_>_
#_For_batch_runs_it_needs_to_be_set_to_NO
menu='YES'
#
#_Specification_of_length_of_names_of_individuals_(default=10)
#____Syntax:_nmlength=<INTEGER_between_0_.._30>
nmlength=12
```

```
datatype='BrownianMicrosatelliteData'
#include_unknown='NO'
datamodel='BM (BROWNIAN_MOTION)'
#
...
inheritance_scalars='NO'
#
...
haplotyping='NO'
#
#
#_____population_relabel=={assignment_for_each_location_in_the_infile}
#_____________example_is_population_relabel=={1_2_2}
population_relabel='{1,_2,_2,_2,_2,_2,_2,_2,_2,_2,_3,_3,_3}'
#
#__#random_subset=number:#random_numberseed
#_____allows_to_subset_the_dataset_#randomly,_if_number_>_sample_in_population
#_____all_samples_are_taken,_if_number_is_smaller_then_the_pop_sample_is_shuffled_and
#_____and_the_first_number_samples_are_taken
#random_subset=10:3176820
#
#############################################################################
#_Input_options
#############################################################################
#
#_input_file_location
#___Syntax_infile=FILEPATH
infile='infile.msat.Myotis.myotis_13_pop'
...
```

Importare i moduli: il file parmfile.py

import, from, reload

```
>>> import parmfile
>>> parmfile.nmlength
12
>>> parmfile.datamodel
'BM (BROWNIAN_MOTION)'
>>> parmfile.infile
'infile.msat.Myotis.myotis_13_pop'
>>> parmfile.burn_in
100
>>> parmfile.outfile
'outfile_msat_13_pop_27_mag'
>>> parmfile.datatype
'BrownianMicrosatelliteData'
>>> from parmfile import datatype
>>> datatype
'BrownianMicrosatelliteData'
>>> print('My data model is %s' % parmfile.datamodel)
My data model is BM (BROWNIAN_MOTION)
>>>
```

Oggetti “built-in”: lists

L’oggetto <**list**> è il tipo di sequenza più generale in Python. È una collezione di oggetti di tipo arbitrario, ordinati in base alla loro posizione.

A differenza delle stringhe, le lists sono mutabili: ogni elemento di una list può essere modificato conoscendone la posizione.

Oggetti “built-in”: lists

```
>>> address_1 = ['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it']
>>> address_1
['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it']
>>> len(address_1)
5
>>> for x in address_1: print(x)
...
piero
rivoira
istitutopennaasti
edu
it
>>>
>>> address_1[0]
'piero'
>>> address_1[0][0]
'p'
>>> school_name = address_1[2][8:13]
>>> school_name
'penna'
```

Oggetti “built-in”: lists

```
>>> 'the name of my school is ' +  school_name
'the name of my school is penna'
>>> address_1.append('Via della Resistenza, 18/D')
>>> len(address_1)
6
>>> address_1
['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it', 'Via della Resistenza,
18/D']
>>> address_1.append('Saluzzo')
>>> len(address_1)
7
>>> address_1
['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it', 'Via della Resistenza,
18/D', 'Saluzzo']
>>> address_1.append('CN')
>>> len(address_1)
8
>>> address_1
['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it', 'Via della Resistenza,
18/D', 'Saluzzo', 'CN']
>>>
```

Oggetti “built-in”: lists

```
>>> address_1.remove('CN')
>>> address_1
['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it', 'Via della Resistenza,
18/D', 'Saluzzo']
>>> address_1.remove('edu')
>>> address_1
['piero', 'rivoira', 'istitutopennaasti', 'it', 'Via della Resistenza, 18/D',
'Saluzzo']
>>> address_1 = ['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it', 'Via
della Resistenza, 18/D', 'Saluzzo']
>>> address_1[0] + '.' + address_1[1] + '@' + address_1[2] + '.' +
address_1[3] + '.' + address_1[4]
'piero.rivoira@istitutopennaasti.edu.it'
>>>
```

Oggetti “built-in”: lists

```
>>> address_1 = ['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it', 'Via della Resistenza, 18/D', 'Saluzzo']
>>> address_1[2]
'istitutopennaasti'
>>> address_1[2] = 'yahoo'
>>> address_1[2]
'yahoo'
>>> address_1
['piero', 'rivoira', 'yahoo', 'edu', 'it', 'Via della Resistenza, 18/D', 'Saluzzo']
>>> address_1.remove('edu')
>>> address_1
['piero', 'rivoira', 'yahoo', 'it', 'Via della Resistenza, 18/D', 'Saluzzo']
>>>
```

Oggetti “built-in”: comprehensions

When applied to strings, the `%` operator provides a simple way to format values as strings according to a format definition, coding multiple substitutions all at once, instead of building and concatenating parts individually.

```
>>>
```

Oggetti “built-in”: tuples

When applied to strings, the `%` operator provides a simple way to format values as strings according to a format definition, coding multiple substitutions all at once, instead of building and concatenating parts individually.

```
>>>
```

Oggetti “built-in”: stringhe

Strings can be indexed (subscripted), with the first character having index 0. There is no separate character type; a character is simply a string of size one:

```
>>> word = 'Python'
>>> word[0] # character in position 0
'P'
>>> word[5] # character in position 5
'n'
>>> word[-1] # last character
'n'
>>> word[-6] # first character
'P'
>>>
```

Oggetti “built-in”: stringhe

```
>>> S = 'Migrate-n'
>>> S
'Migrate-n'
>>>
>>> S[-1]
'n'
>>> S[-2]
'-'
>>> S[-9]
'M'
>>> S[len(S)-1]
'n'
>>> S[0:7]
'Migrate'
>>> S[1:]
'igrate-n'
>>> S[:-2]
'Migrate'
```

Oggetti “built-in”: stringhe

```
>>> S + ' is a great program!'
'Migrate-n is a great program!'
>>> L = list(S)
>>> L
['M', 'i', 'g', 'r', 'a', 't', 'e', '-', 'n']
>>> ''.join(L)
'Migrate-n'
```

Oggetti “built-in”: stringhe

```
>>> S = S.replace('Migrate-n', 'Lamarc')
>>> S
'Lamarc'
>>> S = ('Migrate-n')
>>> L = S.replace('Migrate-n', 'lamarc')
>>> L
'lamarc'
>>> L.capitalize()
'Lamarc'
>>> L
'lamarc'
>>> L = L.capitalize()
>>> L
'Lamarc'
>>> L.upper()
'LAMARC'
>>> L.isalpha()
True
>>>
```

Oggetti “built-in”: stringhe

```
>>> S1 = 'Migrate-n '
>>> S1
'Migrate-n '
>>> S2 = 'uses genetic data (microsatellite data, sequences) to infer
populaiton genetic parameters'
>>> S12 = S1 + S2
>>> S12
'Migrate-n uses genetic data (microsatellite data, sequences) to infer
populaiton genetic parameters'
>>> S12 * 2
'Migrate-n uses genetic data (microsatellite data, sequences) to infer
populaiton genetic parametersMigrate-n uses genetic data (microsatellite data,
sequences) to infer populaiton genetic parameters'
>>> S1[1]
'i'
>>>
```

Oggetti “built-in”: stringhe

```
>>> S3 = 'MIGRATE uses genetic data (microsatellite data ,sequences) to infer populaiton genetic parameters, such a population size, immigration rates, and population divergence times.\n This is our flagship product! There is also a google-support group (migrate-support@google.com)'
>>> S3
'MIGRATE uses genetic data (microsatellite data ,sequences) to infer populaiton genetic parameters, such a population size, immigration rates, and population divergence times.\n This is our flagship product! There is also a google-support group (migrate-support@google.com)'
>>> print(S3)
MIGRATE uses genetic data (microsatellite data ,sequences) to infer populaiton genetic parameters, such a population size, immigration rates, and population divergence times.
 This is our flagship product! There is also a google-support group (migrate-support@google.com)
>>>
```

Oggetti “built-in”: stringhe

```
>>> 'Py' 'thon'
'Python'
text = ('Put several strings within parentheses '
... 'to have them joined together.')
>>> text
'Put several strings within parentheses to have them joined together.'
>>> prefix = 'Py'
>>> prefix + 'thon'
'Python'
```

Oggetti “built-in”: stringhe

```
>>> address_1 = ['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it']
>>> address_1
['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it']
>>> address_2 = ['piero', 'rivoira', 'yahoo', '', 'it']
>>> address_2
['piero', 'rivoira', 'yahoo', '', 'it']
>>> address_book = (address_1, address_2)
>>> address_book
(['piero', 'rivoira', 'istitutopennaasti', 'edu', 'it'], ['piero', 'rivoira',
'yahoo', '', 'it'])
>>> NAME, SURNAME, INSTITUTION, DOMAIN, EXTENSION = range(5)
>>> address_book[0][NAME]
'piero'
>>> whole_address_1 = address_book[0][NAME] + '.' + address_book[0][SURNAME] +
'@' + address_book[0][INSTITUTION] + '.' + address_book[0][DOMAIN] + '.' +
address_book[0][EXTENSION]
>>> whole_address_1
'piero.rivoira@istitutopennaasti.edu.it'
>>>
```

Oggetti “built-in”: stringhe

```
>>> address_1 = ['piero', '.', 'rivoira', '@', 'istitutopennaasti', '.',
'edu', '.', 'it']
>>> type(address_1)
<class 'list'>
>>> import string
>>> ''.join(address_1)
'piero.rivoira@istitutopennaasti.edu.it'
>>>
```

Scriviamo il nostro indirizzo e-mail istituzionale

```
# $ sudo apt install python3-tk
# script5.py

import os, sys
from tkinter.simpledialog import askstring

NOME = askstring('Entry', 'inserisci il tuo nome')
print(NOME)
COGNOME = askstring('Entry', 'inserisci il tuo cognome')
print(COGNOME)
ISTITUZIONE = askstring('Entry', 'inserisci il nome della tua istituzione')
print(ISTITUZIONE)
DOMINIO = askstring('Entry', "inserisci il dominio della tua istituzione, es. 'edu'")
print(DOMINIO)
ESTENSIONE = askstring('Entry', "inserisci l'estensione, es. 'it'")
print(ESTENSIONE)
indirizzo_1 = [NOME, COGNOME, ISTITUZIONE, DOMINIO, ESTENSIONE]
print(indirizzo_1)
NOME, COGNOME, ISTITUZIONE, DOMINIO, ESTENSIONE = range(5)
RUBRICA = (indirizzo_1)
indirizzo_completo_1 = RUBRICA[NOME] + '.' + RUBRICA[COGNOME] + '@' + RUBRICA[ISTITUZIONE] + '.' +
RUBRICA[DOMINIO] + '.' +  RUBRICA[ESTENSIONE]
print(indirizzo_completo_1)
print(os.getlogin())
print(sys.platform)
```

Scaricare un file da internet

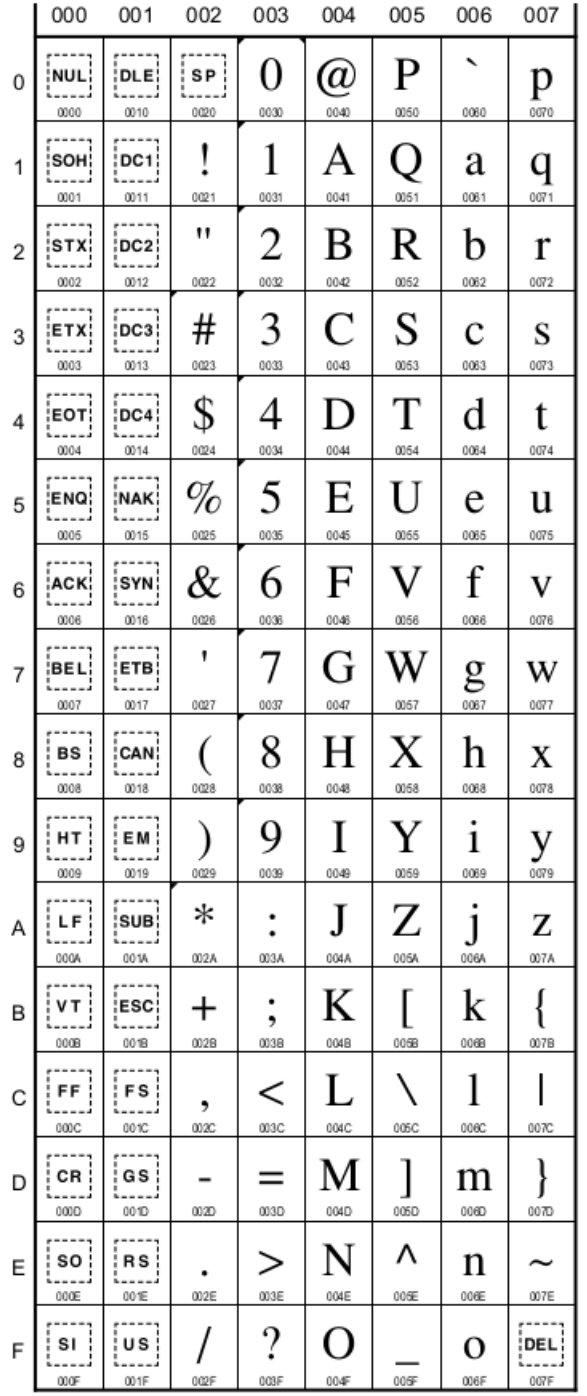
```
piero@piero-XPS-9320:~$ wget
https://github.com/pierorivoira/COMPUTER-SCIENCE/raw/refs/heads/main/parmfile.txt
--2025-04-01 18:35:34--
https://github.com/pierorivoira/COMPUTER-SCIENCE/raw/refs/heads/main/parmfile.txt
Risoluzione di github.com (github.com)... 140.82.121.4
Connessione a github.com (github.com)|140.82.121.4|:443... connesso.
Richiesta HTTP inviata, in attesa di risposta... 302 Found
Posizione: https://raw.githubusercontent.com/pierorivoira/COMPUTER-SCIENCE/refs/heads/
main/parmfile.txt [segue]
--2025-04-01 18:35:34--  https://raw.githubusercontent.com/pierorivoira/COMPUTER-
SCIENCE/refs/heads/main/parmfile.txt
Risoluzione di raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.110.133, 185.199.108.133, 185.199.109.133, ...
Connessione a raw.githubusercontent.com (raw.githubusercontent.com)|
185.199.110.133|:443... connesso.
Richiesta HTTP inviata, in attesa di risposta... 200 OK
Lunghezza: 25960 (25K) [text/plain]
Salvataggio in: ‘parmfile.txt’

parmfile.txt        100%[================>]  25,35K  --.-KB/s    in 0,05s

2025-04-01 18:35:34 (495 KB/s) - ‘parmfile.txt’ salvato [25960/25960]

piero@piero-XPS-9320:~$
```

Scaricare un file da internet

```
>>> import os
>>> os.system('wget
https://github.com/pierorivoira/COMPUTER-SCIENCE/raw/refs/heads/main/parmfile.txt')
--2025-04-01 18:41:17--
https://github.com/pierorivoira/COMPUTER-SCIENCE/raw/refs/heads/main/parmfile.txt
Risoluzione di github.com (github.com)... 140.82.121.4
Connessione a github.com (github.com)|140.82.121.4|:443... connesso.
Richiesta HTTP inviata, in attesa di risposta... 302 Found
...

>>>
```

Files di testo

Contengono stringhe di caratteri (tipo **`str`**), codificati secondo il sistema **Unicode** (https://www.unicode.org/charts/)



	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
1	SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
3	ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
4	EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
5	ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
6	ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
7	BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
8	BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
9	HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
A	LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
B	VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
C	FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
D	CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
E	SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
F	SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

Leggere un file di testo

```
>>> for line in open('parmfile.txt'): print(line)
...
##############################################################################

# Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022

# DO NOT REMOVE THE FIRST TWO LINES!

#

# generated automatically on

# Thu Apr  6 08:44:47 2023
```

Leggere un testo in un oggetto di tipo stringa

```
>>> import os
>>> user = os.getlogin()
>>> os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
>>> with open('parmfile.txt', 'r') as file:
...     text = file.read()
...
>>> text
"#####################################################################\n#
Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022\n# DO NOT REMOVE THE
FIRST TWO LINES!\n#\n# generated automatically on\n# Thu Apr  6 08:44:47 2023\n#\n#
please report problems to Peter Beerli\n#  email: beerli@fsu.edu\n#
http://popgen.sc.fsu.edu/migrate.html\n
...

>>> type(text)
<class 'str'>
>>> text.find('ALLELE')
1439
>>>
```

Leggere un testo in un oggetto di tipo stringa

```
>>> text.find('RANDOM')
12443
>>> text = text.replace('RANDOM', 'CASUALE')
>>> text.find('RANDOM')
-1
>>> text.find('CASUALE')
12443
>>> 'RANDOM' in text
False
>>> 'CASUALE' in text
True
>>> string = '# DO NOT REMOVE THE FIRST TWO LINES!\n'
>>> string
'# DO NOT REMOVE THE FIRST TWO LINES!\n'
>>> string.strip()
'# DO NOT REMOVE THE FIRST TWO LINES!'
>>>
```

ctrl-maiusc-c

piero@piero-XPS-9320: ~/COMPUTER_SCIENCE/AI

piero@piero-XPS-9320: ~

```
d}\n#     Syntax: heating=< NO | <YES | ADAPTIVE>:SKIP:TEMPERATURES\n#
 NO    No heating\n#          YES   heating using TEMPERATURES\n#          ADAPT
IVE adaptive heating using start TEMPERATURES [fails sometimes!!]\n#
SKIP skip that many comparisons, this lengthens the run by SKIP\n#
  TEMPERATURES    { 1.0, temp1, temp2, temp3 .. tempn}\n#     Example: heati
ng=YES:1:{1.0, 1.2, 3.0,1000000.0}\n# Heating:  swapping chains\n#     Synta
x: heated-swap=< YES | NO >\n#          YES  swapping of chains enabled [DEFA
ULT]\n#          NO   swapping of chains disabled\n#      Example: heated-swa
p=YES\nheating=YES:1:{1.0,1.071,1.154,1.25,1.364,1.5,1.667,1.875,2.143,2.5,3
.0,3.75,5.0,7.5,15.0,1000000.0}\nheated-swap=YES\n#\n# Lengthening chain sch
emes\n#    Syntax: moving-steps=< NO | YES:VALUE>\n#       VALUE   frequency
 is between 0..1\nmoving-steps=NO\n#\n#\n#    Convergence statistic [Gelman
and Rubin]\n#    Syntax: gelman-convergence=< YES:Pairs|Summary | NO >\n#
    NO      do not use Gelman's convergence criterium\n#       YES     use G
elman's convergence criteria between chain i, and i-1\n#                PAIRS
 reports all replicate pairs\n#                    SUM   reports only mean and ma
xima\ngelman-convergence=No\n#\n#    REPLICATON:\n#    Syntax: replicate=< N
O | YES:<VALUE> >\n#       NO     no replication of run\n#       YES    repl
icate run\n#          VALUE     number between 2 and many, complete replica
tes\nreplicate=YES:10\n#\n#-------------------------------------------------
-----------------------------\n#\nend\n"
>>>
```

ctrl-alt-t

piero@piero-XPS-9320:~$ **gedit**

ctrl-v

*Documento 1 senza nome

Salva

```
##################################\n# Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022\n# DO NOT
tically on\n# Thu Apr  6 08:44:47 2023\n#\n# please report problems to Peter Beerli\n#  email: beerli@fsu.edu\n#

##############################################\n#\n#############################################################################
##############################################\n#\n# Interactive or batch job usage\n#   Syntax: menu= < YES | NO
u=YES\n#\n# Specification of length of names of individuals (default=10)\n#    Syntax: nmlength=<INTEGER between
0 .. 30>\nnmlength=12\n#\n#\n# If you long runs fails because of timelimits\n# AND you have specified bayes-allfile=YES....\n# then you can recover and continue\n#
by setting recover to yes, default is NO\n# This option fails if your bayesallfile is complete!\n# At the moment, I have no experience about failure rate etc. [March
30 2015]\n#          recover=<YES | NO> \n#\n#############################################################################\n# Data
options\n#############################################################################\n#\n# Several different main datatypes are possible:\n# INFINITE ALLELE:
usable for electrophoretic markers,\n#                   other markers with unknown mutation model\n# STEPWISE MUTATION: usable for microsatellite data
or\n#                   other markers with stepwise change\n#                   from one allele to another\n#                   [singlestep versus multistep model, see
micro-submodel option]\n# FINITE SITES MUTATION: standard DNA/RNA sequence mutation\n#                   model, usable for DNA or RNA contiguous\n#
sequences or variable sites only (SNP)\n#-----------------------------------------------------------------------------\n# INFINITE ALLELE\n#  Syntax:
datatype=ALLELICDATA \n#          include-unknown=<YES | NO> with YES unknown alleles\n#                   are included into analysis, NO is the
default\n#\n#-----------------------------------------------------------------------------\n#\n# STEPWISE MUTATION\n#  Syntax: datatype=<MICROSATELLITEDATA |
BROWNIANDATA\n#                MICRO specifies the standard stepwise mutation\n#                   model, the BROWNIAN is an approximation to this\n#          FOR 99%
of all cases BROWNIAN WILL BE FASTEST AND 'BEST'\n#          for MICRO several suboptions are available, again use BROWNIAN! \n#          micro-submodel=<1|2:
{tune,pinc}>\n#                 1 means singlestep mutation model (this is the default and the standard\n#                 2 is the Multistep model (see Watkins 2007
TPB, section 4.2) it needs\n#                   two parameters: tune specifies how close the model is to a singlestep model\n#                   so tune=0 -->
singlestep, tune=1 --> infinite allele model;\n#                   the second parameter defines the probability that the repeat number\n#                   is
increasing, this value cannot be larger than 0.666, I suggest 0.5.\n#                   Example: micro-submodel=2:{0.5,0.5}\n#          micro-threshold=<INTEGER>
Default is 10 [MICRO only, NEEDS TO BE EVEN!],\n#                smaller values speed up analysis, but might also\n#                crash, large values slow down
analysis considerably.\n#                Change this value only when you suspect that your\n#                data has huge gaps in repeat length.\n#
include-unknown=<YES | NO> with YES unknown alleles\n#                are included into analysis, NO is the
default\n#\n#-----------------------------------------------------------------------------\n#\n# FINITE SITES MUTATION\n#  Syntax: datatype=<SEQUENCEDATA |
NUCLEOTIDE\n#          SEQENCEDATA: typical linked stretches of DNA, for example mtDNA\n#          NUCLEOTIDE: linked DNA stretches, all invariable sites
removed\n#\n#          freqs-from-data=<YES | NO: freq(A), freq(C), freq(G), freq(T)>\n#                calculate the prior base frequencies from the data,
\n#                or specify the frequencies\n#          ttratio=<RATIO1 RATIO2 ....> Default is 2.0,\n#                ratio between transitions and transversions.
\n#          seqerror-rate=<{VALUE,VALUE,VALUE,VALUE}|Estimate:1|4> Default is 0.0, typical values for ABI 3700 \n#                sequencers after base calling are
around 0.001 (1/650)\n#          categories=<VALUE:CATFILE> The categories are integers or letters\n#                specified in file called CATFILE, this assumes
that all\n#                sites belong to known categories, this can be used to\n#                weight third positions etc.\n#          rates=<VALUE1 VALUE2 ...>
the rates are specified arbitrarily or\n#                then are from a Gamma distribution with alpha=x, currently\n#                the alpha value gets lost and
is not recorded in the parmfile\n#          prob-rates=<RATE2 RATE1 ... > These rates can be arbitrary or \n#                generated with gamma-deviated rates and
then are derived\n#                using Laguerre's quadrature, this should get better\n#                results than equal probability methods.\n#
autocorrelation=<NO | YES:VALUE> Default is NO\n#                autocorrelation makes only sense with rates,\n#                VALUE should be >1.0\n#
weights=<NO | YES:WEIGHTFILE> The weights are specified\n#                in file called WEIGHTFILE, this assumes that all sites\n#                belong to known
weights, this can be used to weight\n#                portions of the sequence etc.\n#          fast-likelihood=<YES | NO> Default is
NO\n#-----------------------------------------------------------------------------\n#          population-relabel={assignment for each location in the infile}
\n#                example is population-relabel={1 2 2}\n#          random-subset=number<:seed>\n#                allows to subset the dataset randomly, if number >
sample in population\n#                all samples are taken, if number is smaller then the pop sample is shuffled and\n#                and the first number samples
are taken.\n#                the random number seed guarantees that the\n#                same subset is chosen in different
runs\ndatatype=BrownianMicrosatelliteData\ninclude-unknown=NO\ndatamodel=BM (BROWNIAN MOTION)\n#\n#  inheritance-scalars=<NO|YES:{values for each locus}>\n#
these values are multiplied with Theta, for example having\n#          two autosomal and a locus on X- and one on Y-chromosome we would give \n#          inheritance-
scalars=YES:{1 1 1.33 4.0}\n#          # this will scale Theta = 4*N*mu\n#          # We can scale differently:\n#          inheritance-scalars=YES:{0.25 0.25 0.33
1.00}\n#          # this will scale Theta = N*mu all is in scale for Y or mtDNA\n#          the first locus is the reference and the combined estimate has its
inheritance-scalar\n#          if loci are {mtDNA,nucDNA} then {1.0 0.25} --> Theta=N*mu for all\ninheritance-scalars=NO\n#\n#\n#    Haplotyping: \n#    Syntax:
haplotyping=<YES:<report|no-report> | NO > \n#    If you have diplotype data input two lines per individual marking\n#    individual names name:other where name must
be on both lines, it will\n#    will be used to build the indiviudal database for the haplotyping.\n#    If you are not interested in the haplotypes, refrain from
```

Perché gedit trova un solo <\n>?

piero@piero-XPS-9320:~$ **gedit ~/COMPUTER_SCIENCE/AI/parmfile.txt**

Perché gedit trova 526 <\n>?

parmfile.txt
~/COMPUTER_SCIENCE/AI

Salva

\n 1 di 526

```
# Thu Apr  6 08:44:47 2023
#
# please report problems to Peter Beerli
#  email: beerli@fsu.edu
#  http://popgen.sc.fsu.edu/migrate.html
#######################################################################################
#
#######################################################################################
# General options
#######################################################################################
#
# Interactive or batch job usage
#   Syntax: menu= < YES | NO >
# For batch runs it needs to be set to NO
menu=YES
#
# Specification of length of names of individuals (default=10)
#    Syntax: nmlength=<INTEGER between 0 .. 30>
nmlength=12
#
#
# If you long runs fails because of timelimits
# AND you have specified bayes-allfile=YES....
# then you can recover and continue
# by setting recover to yes, default is NO
# This option fails if your bayesallfile is complete!
# At the moment, I have no experience about failure rate etc. [March 30 2015]
#          recover=<YES | NO>
#
#######################################################################################
# Data options
#######################################################################################
#
# Several different main datatypes are possible:
# INFINITE ALLELE: usable for electrophoretic markers,
#                  other markers with unknown mutation model
# STEPWISE MUTATION: usable for microsatellite data or
#                  other markers with stepwise change
#                  from one allele to another
#                  [singlestep versus multistep model, see micro-submodel option]
# FINITE SITES MUTATION: standard DNA/RNA sequence mutation
#                  model, usable for DNA or RNA contiguous
#                  sequences or variable sites only (SNP)
#---------------------------------------------------------------------------
# INFINITE ALLELE
```

Apri

parmfile.txt
~/COMPUTER_SCIENCE/AI

Salva

Trova e sostituisci

Trova: \n

Sostituisci con: Niente

Maiuscole/Minuscole — Cercare all'indietro
Solo parole intere — Ricominciare dall'inizio
Espressione regolare

Sostituisci tutti | Sostituisci | Trova

```
#                                                                    ##########
#                                                                    May-09-2022
#
#
#
#
#
#                                                                    ##########
#                                                                    ##########
# General options
##############################################################################
#
# Interactive or batch job usage
#   Syntax: menu= < YES | NO >
# For batch runs it needs to be set to NO
menu=YES
#
# Specification of length of names of individuals (default=10)
#    Syntax: nmlength=<INTEGER between 0 .. 30>
nmlength=12
#
#
# If you long runs fails because of timelimits
# AND you have specified bayes-allfile=YES....
# then you can recover and continue
# by setting recover to yes, default is NO
# This option fails if your bayesallfile is complete!
# At the moment, I have no experience about failure rate etc. [March 30 2015]
#          recover=<YES | NO>
#
##############################################################################
# Data options
##############################################################################
#
# Several different main datatypes are possible:
# INFINITE ALLELE: usable for electrophoretic markers,
#                  other markers with unknown mutation model
# STEPWISE MUTATION: usable for microsatellite data or
#                  other markers with stepwise change
#                  from one allele to another
#                  [singlestep versus multistep model, see micro-submodel option]
# FINITE SITES MUTATION: standard DNA/RNA sequence mutation
#                  model, usable for DNA or RNA contiguous
#                  sequences or variable sites only (SNP)
#-----------------------------------------------------------------------------
# INFINITE ALLELE
```

Da quante righe risulta formato il file di testo <parmfile.txt> dopo l'ultima modifica?

Trova le differenze

```
>>> text
"#####################################################################\n#
Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022\n# DO NOT REMOVE THE
FIRST TWO LINES!\n#\n# generated automatically on\n# Thu Apr  6 08:44:47 2023\n#\n#
please report problems to Peter Beerli\n#  email: beerli@fsu.edu\n#
http://popgen.sc.fsu.edu/migrate.html\
n#####################################################################\n#\
n#####################################################################\n#
General options\
n#####################################################################\n#\n#
Interactive or batch job usage\n#   Syntax:
```

Trova le differenze

```
>>> lines = text.splitlines()

>>> lines
['#######################################################################', '#
Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022', '# DO NOT REMOVE THE
FIRST TWO LINES!', '#', '# generated automatically on', '# Thu Apr  6 08:44:47 2023',
'#', '# please report problems to Peter Beerli', '#  email: beerli@fsu.edu', '#
http://popgen.sc.fsu.edu/migrate.html',
'#######################################################################',
'#',
'#######################################################################', '#
General options',
'#######################################################################',
'#', '# Interactive or batch job usage', '#   Syntax:
```

<lines> è una list!

```
>>> type(lines)
<class 'list'>
>>> lines[0]
'##################################################################################'
>>> lines[1]
'# Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022'
>>> lines[2]
'# DO NOT REMOVE THE FIRST TWO LINES!'
>>> lines[3]
'#'
>>> lines[4]
'# generated automatically on'
>>> lines[5]
'# Thu Apr  6 08:44:47 2023'
>>> lines[500]
'heating=YES:1:
{1.0,1.071,1.154,1.25,1.364,1.5,1.667,1.875,2.143,2.5,3.0,3.75,5.0,7.5,15.0,1000000.0}'
>>>
```

Leggere un testo in un oggetto di tipo stringa

```
>>> numlines = 19
>>> chunk = lines[:numlines]
>>> chunk
['#####################################################################', '#
Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022', '# DO NOT REMOVE THE
FIRST TWO LINES!', '#', '# generated automatically on', '# Thu Apr  6 08:44:47 2023',
'#', '# please report problems to Peter Beerli', '#  email: beerli@fsu.edu', '#
http://popgen.sc.fsu.edu/migrate.html',
'#####################################################################',
'#',
'#####################################################################', '#
General options',
'#####################################################################',
'#', '# Interactive or batch job usage', '#   Syntax: menu= < YES | NO > ', '# For
batch runs it needs to be set to NO']
>>>
```

Apri | *Documento 1 senza nome | Salva

parmfile.txt | *Documento 1 senza nome

```
1['################################################################################', '# Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022', '# DO NOT
 REMOVE THE FIRST TWO LINES!', '#', '# generated automatically on', '# Thu Apr  6 08:44:47 2023', '#', '# please report problems to Peter Beerli', '#  email:
 beerli@fsu.edu', '#  http://popgen.sc.fsu.edu/migrate.html', '################################################################################', '#',
 '################################################################################', '# General options',
 '################################################################################', '#', '# Interactive or batch job usage', '#   Syntax: menu= < YES | NO > ', '#
 For batch runs it needs to be set to NO']
```

Trova e sostituisci

Trova: #

Sostituisci con: \n#

- [ ] Maiuscole/Minuscole
- [ ] Cercare all'indietro
- [ ] Solo parole intere
- [x] Ricominciare dall'inizio
- [ ] Espressione regolare

Sostituisci tutti | Sostituisci | Trova

Apri | *Documento 1 senza nome | Salva

parmfile.txt | *Documento 1 senza nome

```
['#################################################################################', '
# Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022', '
# DO NOT REMOVE THE FIRST TWO LINES!', '#', '
# generated automatically on', '
# Thu Apr  6 08:44:47 2023', '#', '
# please report problems to Peter Beerli', '
#  email: beerli@fsu.edu', '
#  http://popgen.sc.fsu.edu/migrate.html', '#########################################################################################', '#',
'#################################################################################', '
# General options', '#########################################################################################', '#', '
# Interactive or batch job usage', '
#   Syntax: menu= < YES | NO > ', '
# For batch runs it needs to be set to NO']
```

Trova e sostituisci

Trova: #

Sostituisci con: \n#

- [ ] Maiuscole/Minuscole
- [ ] Cercare all'indietro
- [ ] Solo parole intere
- [x] Ricominciare dall'inizio
- [ ] Espressione regolare

Sostituisci tutti | Sostituisci | Trova

Leggere un testo in un oggetto di tipo stringa

```
>>> text = open('parmfile.txt').read()
>>> type(text)
<class 'str'>
>>> lines = open('parmfile.txt').readlines()
>>> type(lines)
<class 'list'>
>>> lines[0]
'##################################################################\n'
>>> lines[7]
'# please report problems to Peter Beerli\n'
>>> lines[500]
'heating=YES:1:
{1.0,1.071,1.154,1.25,1.364,1.5,1.667,1.875,2.143,2.5,3.0,3.75,5.0,7.5,15.0,1000000.0}\
n'
>>>
```

Common os.path Tools

```
piero@piero-XPS-9320:~$ python3
Python 3.10.12 (main, Feb  4 2025, 14:57:36) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.getcwd()
'/home/piero'
>>> os.path.isdir('COMPUTER_SCIENCE')
True
>>> os.path.isdir('COMPUTER_SCIENCE/AI')
True
>>> os.path.isdir('COMPUTER_SCIENCE/AI/PARMFILE')
False
>>>
```

‘PARMFILE’ è un file o una directory?

```
>>> import dill, os, pickle
>>> name = 'PARMFILE'
>>> def apri(name):
...     if os.path.isdir('COMPUTER_SCIENCE/AI/%s' % name):
...             print('The object %s is a directory' % name)
...     else:
...             print('The object %s is not a directory' % name)
...
>>> apri(name)
The object PARMFILE is not a directory
>>> name = 'CARTELLA'
>>> apri(name)
The object CARTELLA is a directory
>>>
```

Cos'è la shell?

Shell: system that reads and runs command-line strings on your computer

Shell command: command-line string that you would normally enter at your computer's shell prompt

Eseguire un comando della shell

```
>>> os.getcwd()
'/home/piero/COMPUTER_SCIENCE/AI'
>>> os.system('ls')
1_Python_PROGRAMMING_01.03.2025_non_eliminare.odp
1_Python_PROGRAMMING_09.03.2025.odp
ALGORITMO_18.01.2022.odt
alimenti_26_febbraio_25.py
alimenti.ods
alimenti.py
blocco_di_testo_last_rows.py
blocco_di_testo_mod.py
blocco_di_testo.py
CARTELLA
Cereale.py
DATA
dati_istogramma.txt
docs-pdf
eleva_a_potenza.py
Farina_di_estrazione.py
Fonte_di_fibra.py
Foraggio.py
...
>>>
```

Home / COMPUTER_SCIENCE / AI

- Recenti
- Preferiti
- Home
- Documenti
- Immagini
- Musica
- Scaricati
- Video
- Cestino
- Altre posizioni

Nome	Dimensione	Modifica
CARTELLA	0 oggetti	Ieri
DATA	5 oggetti	9 feb
docs-pdf	35 oggetti	23 gen
GRAFICI	1 oggetto	11 feb
__pycache__	32 oggetti	lun
1_Python_PROGRAMMING_01.03.2025_non_eliminare.odp	1,8 MB	1 mar
1_Python_PROGRAMMING_09.03.2025.odp	4,8 MB	08:24
ALGORITMO_18.01.2022.odt	31,1 kB	27 gen
alimenti.ods	2,7 MB	1 feb
alimenti.py	72,7 kB	26 feb
alimenti_26_febbraio_25.py	72,7 kB	26 feb
blocco_di_testo.py	592 byte	lun
blocco_di_testo_last_rows.py	673 byte	mer
blocco_di_testo_mod.py	592 byte	lun
Cereale.py	790 byte	2 feb
dati_istogramma.txt	3,1 MB	1 mar
eleva_a_potenza.py	632 byte	14 feb
Farina_di_estrazione.py	1,2 kB	2 feb
Fonte_di_fibra.py	2,7 kB	3 feb
Foraggio.py	2,4 kB	1 feb

Eseguire un comando della shell

```
>>> os.system('lsb_release -a')
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 22.04.5 LTS
Release:  22.04
Codename: jammy
0
>>>
```

Formatting Expression Basics

```
>>> 'These are %d %s bones' % (2, 'animal')
'These are 2 animal bones'
>>>
```

When applied to strings, the `%` operator provides a simple way to format values as strings according to a format definition, coding multiple substitutions all at once, instead of building and concatenating parts individually.

To format strings:

- To the left `%` operator, provide one or more conversion targets, each of which starts with a `%` (e.g., `%d`)
- To the right `%` operator, provide the object (or objects, embedded in a **tuple**) that you want Python to insert into the format string on the left in place of the conversion target (or targets)

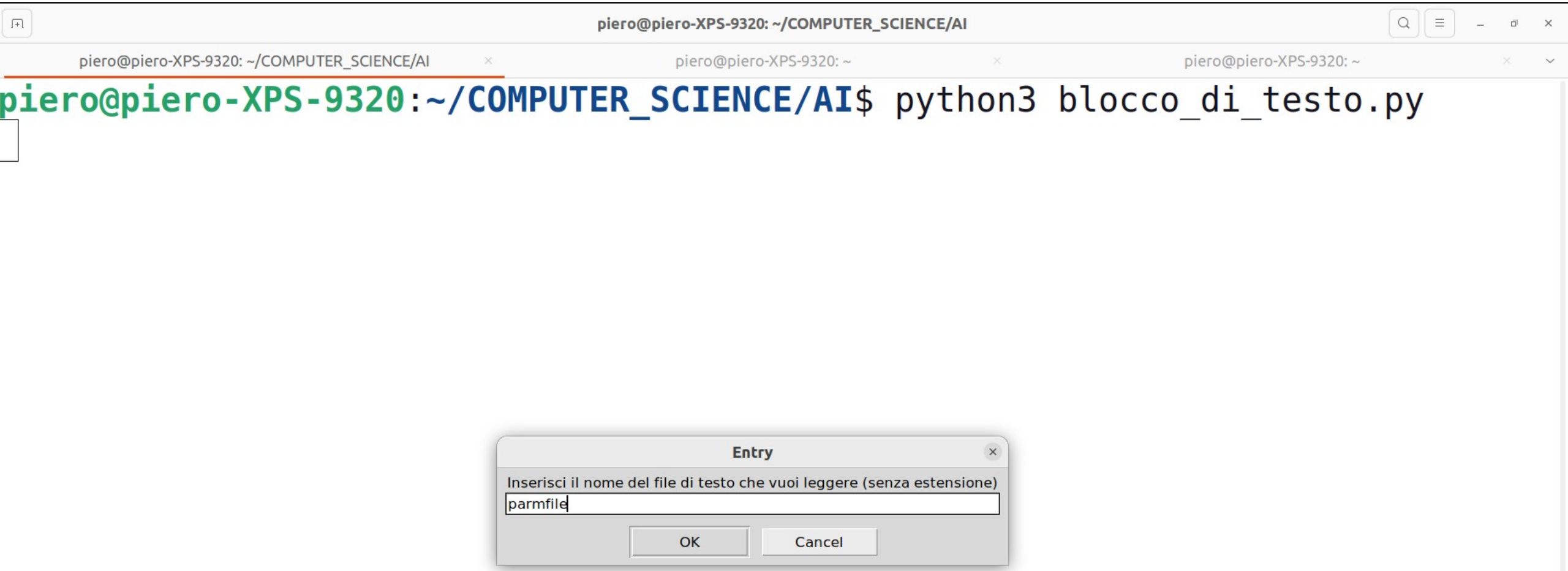
Formatting Expression Basics

```
>>> import math
>>> 'The base of the natural logarithms is %f' % math.e
'The base of the natural logarithms is 2.718281828459045'
>>>
```

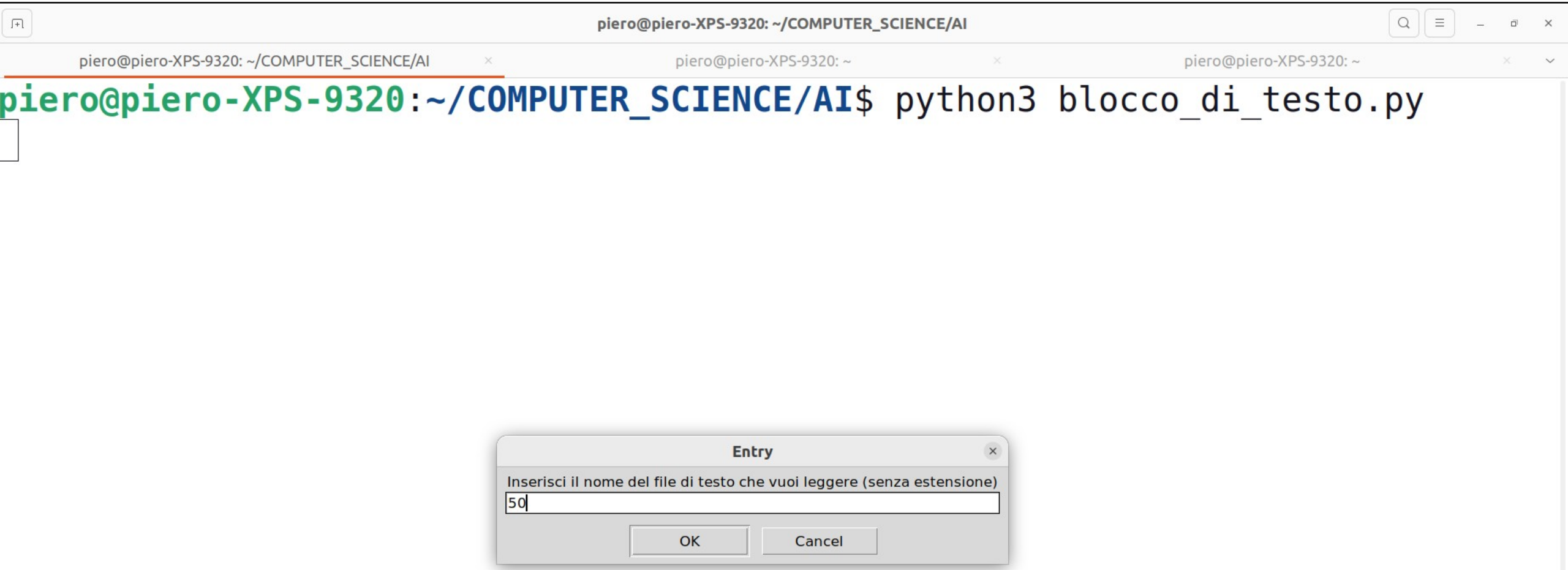
Il programma blocco_di_testo.py

```
"""
print a chunk of text
"""

import os
from tkinter.simpledialog import askstring
from tkinter.simpledialog import askinteger
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
filename = askstring('Entry', 'Inserisci il nome del file di testo che vuoi leggere
(senza estensione)')
numlines = askinteger('Entry', 'Inserisci il numero di righe del blocco')
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
    chunk = lines[:numlines]
    print(chunk)

blocco = more(text, numlines)
print(blocco)
```

Il programma blocco_di_testo.py



Il programma blocco_di_testo.py



Il programma blocco_di_testo.py

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 blocco_di_testo.py
['##################################################################', '# Parmfile
for Migrate 5.0.4(git:distribution-version)-May-09-2022', '# DO NOT REMOVE THE FIRST TWO LINES!',
'#', '# generated automatically on', '# Thu Apr  6 08:44:47 2023', '#', '# please report problems
to Peter Beerli', '#  email: beerli@fsu.edu', '#  http://popgen.sc.fsu.edu/migrate.html',
'##################################################################', '#',
'##################################################################', '# General
options', '##################################################################', '#',
'# Interactive or batch job usage', '#   Syntax: menu= < YES | NO > ', '# For batch runs it needs
to be set to NO', 'menu=YES', '#', '# Specification of length of names of individuals
(default=10)', '#    Syntax: nmlength=<INTEGER between 0 .. 30>', 'nmlength=12', '#', '#', '# If
you long runs fails because of timelimits', '# AND you have specified bayes-allfile=YES....', '#
then you can recover and continue', '# by setting recover to yes, default is NO', '# This option
fails if your bayesallfile is complete!', '# At the moment, I have no experience about failure
rate etc. [March 30 2015]', '#          recover=<YES | NO> ', '#',
'##################################################################', '# Data
options', '##################################################################', '#',
'# Several different main datatypes are possible:', '# INFINITE ALLELE: usable for electrophoretic
markers,', '#                    other markers with unknown mutation model', '# STEPWISE MUTATION:
usable for microsatellite data or', '#                    other markers with stepwise change', '#
             from one allele to another', '#                    [singlestep versus multistep model,
see micro-submodel option]', '# FINITE SITES MUTATION: standard DNA/RNA sequence mutation', '#
            model, usable for DNA or RNA contiguous', '#                    sequences or variable
sites only (SNP)',
'#-----------------------------------------------------------------', '# INFINITE
ALLELE']
None
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$
```

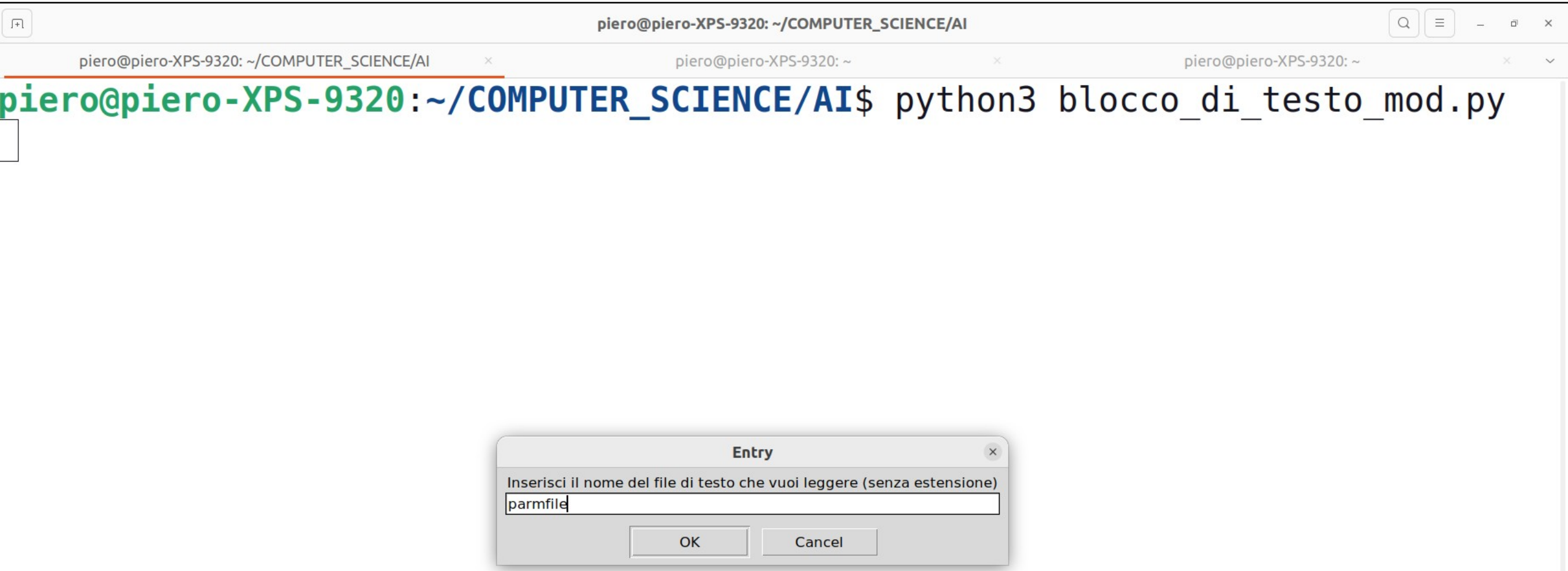
Il programma blocco_di_testo.py

ESERCIZIO: modificare il codice sorgente <blocco_di_testo.py> affinchè il programma visualizzi in output tutto il testo del file **tranne le prime <`numlines`> righe**

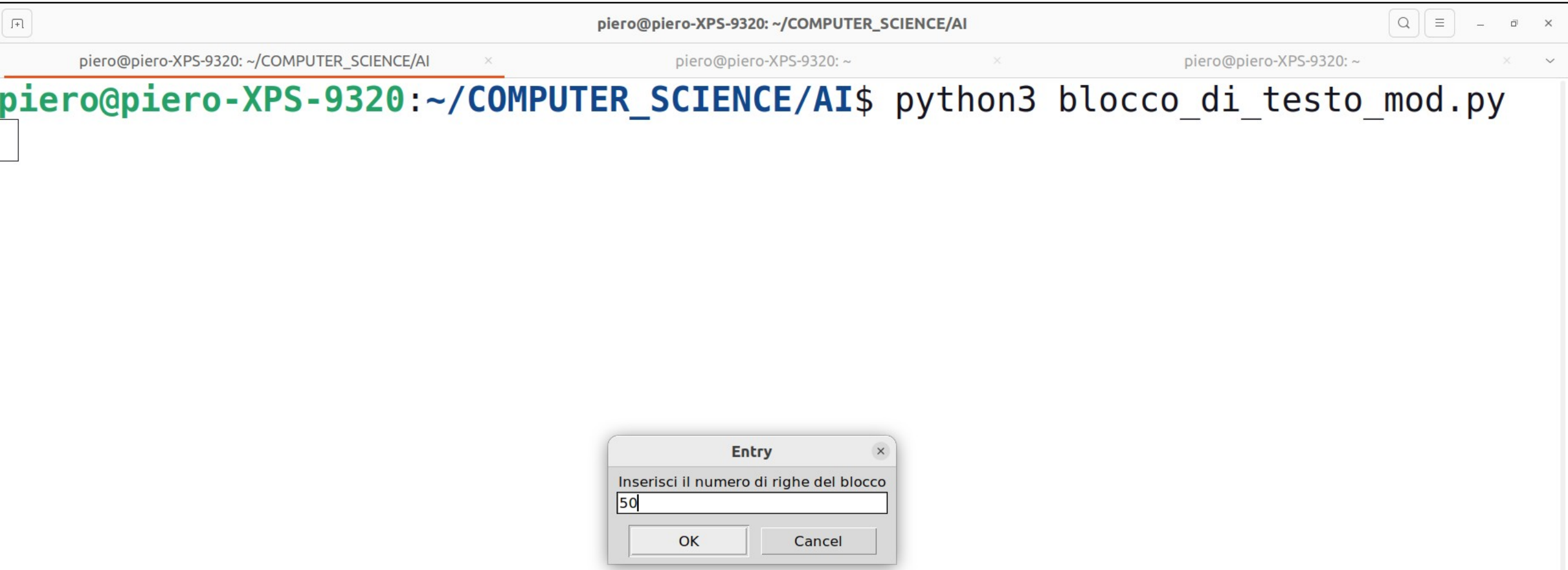
Il programma blocco_di_testo_mod.py

```
"""
print a chunk of text
"""

import os
from tkinter.simpledialog import askstring
from tkinter.simpledialog import askinteger
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
filename = askstring('Entry', 'Inserisci il nome del file di testo che vuoi leggere
(senza estensione)')
numlines = askinteger('Entry', 'Inserisci il numero di righe del blocco')
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
    chunk = lines[numlines:]
    print(chunk)

blocco = more(text, numlines)
print(blocco)
```

Il programma blocco_di_testo_mod.py



Il programma blocco_di_testo_mod.py



Il programma blocco_di_testo_mod.py

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 blocco_di_testo_mod.py
['#  Syntax: datatype=ALLELICDATA ', '#          include-unknown=<YES | NO> with YES unknown
alleles', '#                 are included into analysis, NO is the default', '#',
'#-----------------------------------------------
...
tive heating using start TEMPERATURES [fails sometimes!!]', '#        SKIP skip that many
comparisons, this lengthens the run by SKIP', '#             TEMPERATURES    { 1.0, temp1, temp2,
temp3 .. tempn}', '#     Example: heating=YES:1:{1.0, 1.2, 3.0,1000000.0}', '# Heating:  swapping
chains', '#     Syntax: heated-swap=< YES | NO >', '#          YES  swapping of chains enabled
[DEFAULT]', '#          NO   swapping of chains disabled', '#      Example: heated-swap=YES',
'heating=YES:1:
{1.0,1.071,1.154,1.25,1.364,1.5,1.667,1.875,2.143,2.5,3.0,3.75,5.0,7.5,15.0,1000000.0}', 'heated-
swap=YES', '#', '# Lengthening chain schemes', '#    Syntax: moving-steps=< NO | YES:VALUE>', '#
    VALUE   frequency is between 0..1', 'moving-steps=NO', '#', '#', '#    Convergence statistic
[Gelman and Rubin]', '#    Syntax: gelman-convergence=< YES:Pairs|Summary | NO >', "#       NO
 do not use Gelman's convergence criterium", "#       YES     use Gelman's convergence criteria
between chain i, and i-1", '#                PAIRS reports all replicate pairs', '#
SUM   reports only mean and maxima', 'gelman-convergence=No', '#', '#    REPLICATON:', '#
Syntax: replicate=< NO | YES:<VALUE> >', '#       NO     no replication of run', '#       YES
replicate run', '#           VALUE     number between 2 and many, complete replicates',
'replicate=YES:10', '#',
'#-------------------------------------------------------------------', '#', 'end']
None
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$
```

Leggere un testo in un oggetto di tipo stringa

```
>>> lines[1:19]
['# Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022', '# DO NOT REMOVE
THE FIRST TWO LINES!', '#', '# generated automatically on', '# Thu Apr  6 08:44:47
2023', '#', '# please report problems to Peter Beerli', '#  email: beerli@fsu.edu', '#
http://popgen.sc.fsu.edu/migrate.html',
'#######################################################################',
'#',
'#######################################################################', '#
General options',
'#######################################################################',
'#', '# Interactive or batch job usage', '#   Syntax: menu= < YES | NO > ', '# For
batch runs it needs to be set to NO']
>>>
```

Leggere un testo in un oggetto di tipo stringa

```
>>> righe_successive = lines[numlines:]
>>> righe_successive
YES   heating using TEMPERATURES', '#       ADAPTIVE adaptive heating using start
TEMPERATURES [fails sometimes!!]', '#       SKIP skip that many comparisons, this
lengthens the run by SKIP', '#           TEMPERATURES    { 1.0, temp1, temp2, temp3 ..
tempn}', '#     Example: heating=YES:1:{1.0, 1.2, 3.0,1000000.0}', '# Heating:
swapping chains', '#     Syntax: heated-swap=< YES | NO >', '#         YES  swapping of
chains enabled [DEFAULT]', '#         NO   swapping of chains disabled', '#
Example: heated-swap=YES', 'heating=YES:1:
{1.0,1.071,1.154,1.25,1.364,1.5,1.667,1.875,2.143,2.5,3.0,3.75,5.0,7.5,15.0,1000000.0}'
, 'heated-swap=YES', '#', '# Lengthening chain schemes', '#    Syntax: moving-steps=<
NO | YES:VALUE>', '#       VALUE   frequency is between 0..1', 'moving-steps=NO', '#',
'#', '#    Convergence statistic [Gelman and Rubin]', '#    Syntax: gelman-
convergence=< YES:Pairs|Summary | NO >', "#       NO      do not use Gelman's
convergence criterium", "#       YES     use Gelman's convergence criteria between
chain i, and i-1", '#              PAIRS reports all replicate pairs', '#
 SUM   reports only mean and maxima', 'gelman-convergence=No', '#', '#    REPLICATON:',
'#    Syntax: replicate=< NO | YES:<VALUE> >', '#       NO     no replication of run',
'#       YES    replicate run', '#          VALUE     number between 2 and many,
complete replicates', 'replicate=YES:10', '#',
'#-------------------------------------------------------------',
'#', 'end']
>>>
```

Il programma blocco_di_testo.py

ESERCIZIO: modificare il codice sorgente <blocco_di_testo.py> affinchè il programma visualizzi in output **le ultime <`numlines`> righe**

SUGGERIMENTO:

```
>>> nrows = len(list(open("parmfile.txt")))
>>> nrows
527
>>>
```

```
long-chains=1
long-inc=100
long-sample=10000
burn-in=100
auto-tune=YES:0.440000
assign=NO
#
#-------------------------------------------------------------------------------
# Schemes to improve MCMC searching and/or thermodynamic integration
#
# Heating schemes {MCMCMC = MC cubed}
#    Syntax: heating=< NO | <YES | ADAPTIVE>:SKIP:TEMPERATURES
#        NO    No heating
#        YES   heating using TEMPERATURES
#        ADAPTIVE adaptive heating using start TEMPERATURES [fails sometimes!!]
#        SKIP skip that many comparisons, this lengthens the run by SKIP
#            TEMPERATURES    { 1.0, temp1, temp2, temp3 .. tempn}
#     Example: heating=YES:1:{1.0, 1.2, 3.0,1000000.0}
# Heating:  swapping chains
#     Syntax: heated-swap=< YES | NO >
#         YES  swapping of chains enabled [DEFAULT]
#         NO   swapping of chains disabled
#      Example: heated-swap=YES
heating=YES:1:{1.0,1.071,1.154,1.25,1.364,1.5,1.667,1.875,2.143,2.5,3.0,3.75,5.0,7.5,15.0,1000000.0}
heated-swap=YES
#
# Lengthening chain schemes
#    Syntax: moving-steps=< NO | YES:VALUE>
#       VALUE   frequency is between 0..1
moving-steps=NO
#
#
#    Convergence statistic [Gelman and Rubin]
#    Syntax: gelman-convergence=< YES:Pairs|Summary | NO >
#       NO      do not use Gelman's convergence criterium
#       YES     use Gelman's convergence criteria between chain i, and i-1
#               PAIRS reports all replicate pairs
#               SUM   reports only mean and maxima
gelman-convergence=No
#
#    REPLICATON:
#    Syntax: replicate=< NO | YES:<VALUE> >
#       NO     no replication of run
#       YES    replicate run
#           VALUE     number between 2 and many, complete replicates
replicate=YES:10
#
#-------------------------------------------------------------------------------
#
end
```

Il programma blocco_di_testo_last_rows.py

```
"""
print a chunk of text
"""
import os
from tkinter.simpledialog import askstring
from tkinter.simpledialog import askinteger
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
filename = askstring('Entry', 'Inserisci il nome del file di testo che vuoi leggere
(senza estensione)')
numlines = askinteger('Entry', 'Inserisci il numero di righe del blocco')
nrows = len(list(open('%s.txt', % filename)))
lines_2_skip = nrows - numlines
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
    chunk = lines[lines_2_skip:]
    print(chunk)

blocco = more(text, numlines)
print(blocco)
```

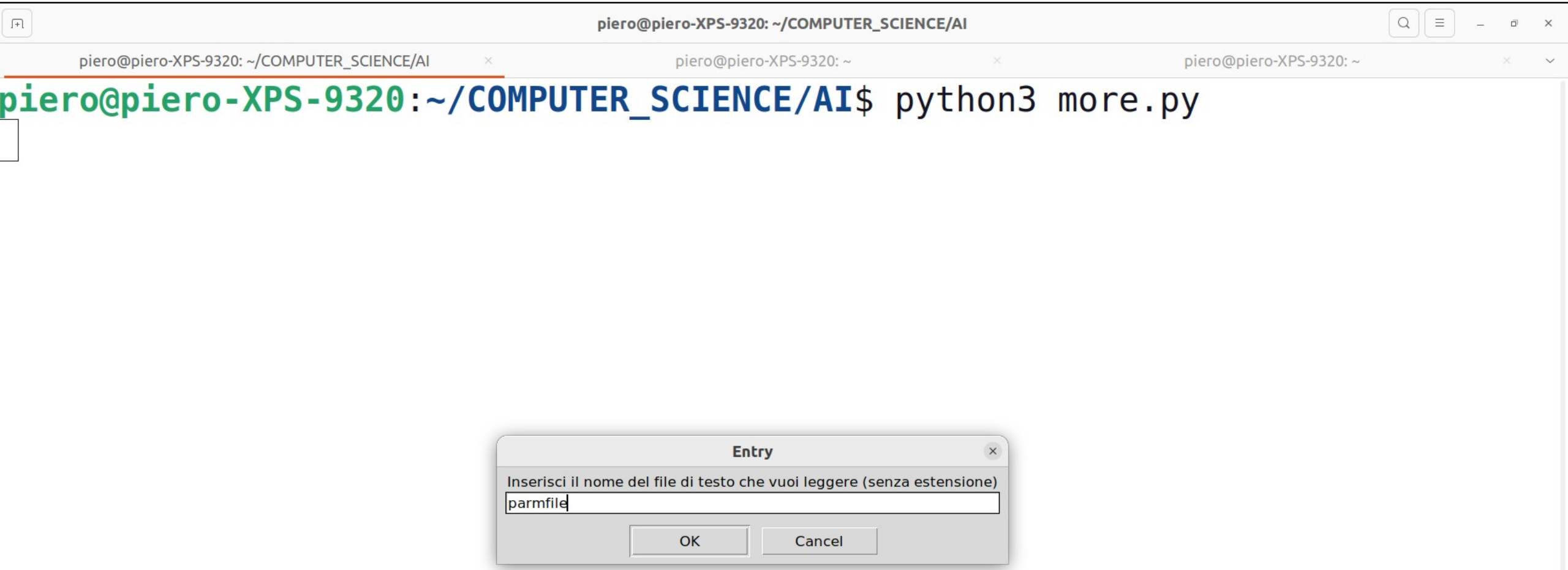
Il programma blocco_di_testo_last_rows.py

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 blocco_di_testo_last_rows.py
['#    REPLICATON:', '#    Syntax: replicate=< NO | YES:<VALUE> >', '#       NO     no
replication of run', '#       YES    replicate run', '#           VALUE     number
between 2 and many, complete replicates', 'replicate=YES:10', '#',
'#-------------------------------------------------------------',
'#', 'end']
None
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$
```

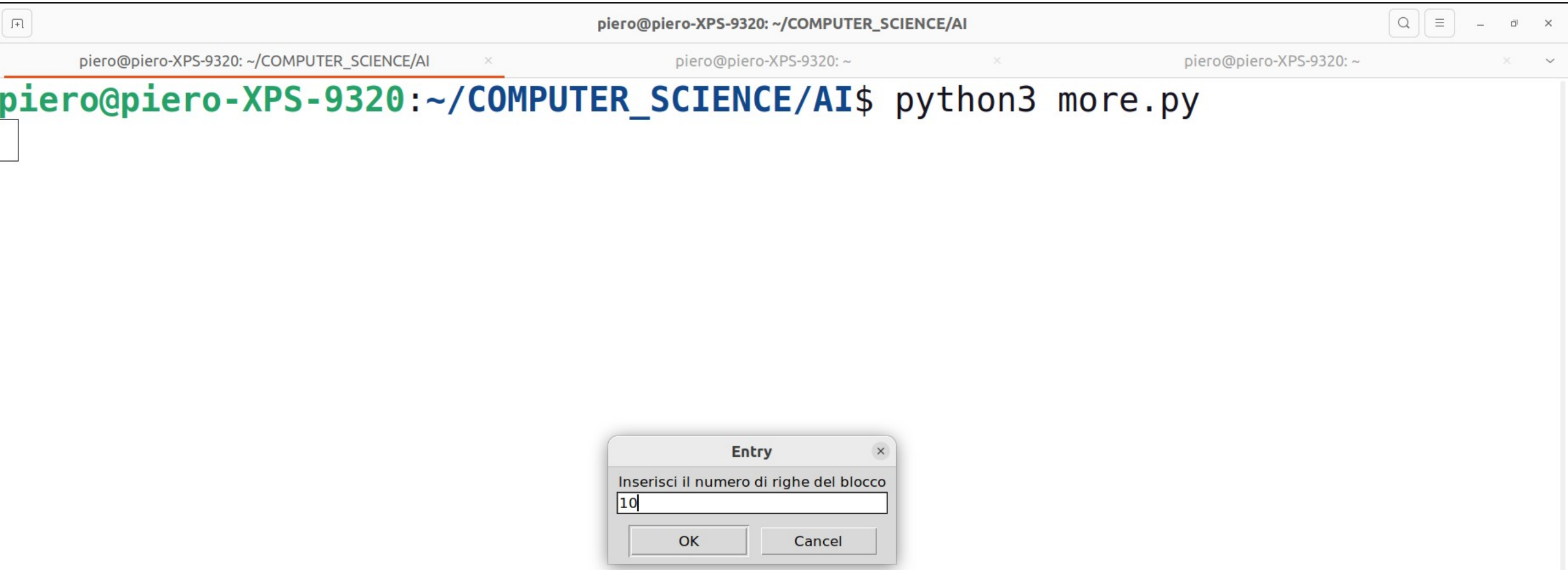
Il programma more.py

```
"""
split and interactively page a string or file of text
"""

import os
from tkinter.simpledialog import askinteger, askstring
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
filename = askstring('Entry', 'Inserisci il nome del file di testo che vuoi leggere
(senza estensione)')
numlines = askinteger('Entry', 'Inserisci il numero di righe del blocco')
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
    while lines:
       chunk = lines[:numlines]
       lines = lines[numlines:]
       for line in chunk: print(line)
       if lines and input('More?') not in ['y', 'Y']: break

more(text, numlines)
```

Il programma more.py



Il programma more.py



Il programma more.py

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 more.py
#####################################################################
# Parmfile for Migrate 5.0.4(git:distribution-version)-May-09-2022
# DO NOT REMOVE THE FIRST TWO LINES!
#
# generated automatically on
# Thu Apr  6 08:44:47 2023
#
# please report problems to Peter Beerli
#  email: beerli@fsu.edu
#  http://popgen.sc.fsu.edu/migrate.html
More?
```

Il programma more.py step by step

```
"""
split and interactively page a string or file of text
"""
```

un altro modo per inserire un commento, anche su più righe

Il programma more.py step by step

```
"""
split and interactively page a string or file of text
"""

import os
from tkinter.simpledialog import askinteger, askstring
```

importiamo il modulo `os` (operating system) e le funzioni necessarie per creare una
finestra di dialogo che chieda all’utente di inserire un numero intero, ossia da quante
righe ogni blocco debba essere formato (`askinteger`) ed una finestra di dialogo che
consenta all'utente di inserire il nome del file (`askstring`); quest'ultimo, infatti, è una
stringa di caratteri

Il programma more.py step by step

```
"""
split and interactively page a string or file of text
"""

import os
from tkinter.simpledialog import askinteger, askstring
user = os.getlogin()
```

usiamo l’attributo `getlogin()` del modulo `os` per ottenere il nome dell'utente e lo
salviamo nella variabile `user`

```
>>> dir(os)
['CLD_CONTINUED', 'CLD_DUMPED', 'CLD_EXITED', 'CLD_KILLED', 'CLD_STOPPED', 'CLD_TRAPPED', 'DirEntry', 'EFD_CLOEXEC', 'EFD_NONBLOCK',
'EFD_SEMAPHORE', 'EX_CANTCREAT', 'EX_CONFIG', 'EX_DATAERR', 'EX_IOERR', 'EX_NOHOST', 'EX_NOINPUT', 'EX_NOPERM', 'EX_NOUSER', 'EX_OK',
'EX_OSERR', 'EX_OSFILE', 'EX_PROTOCOL', 'EX_SOFTWARE', 'EX_TEMPFAIL', 'EX_UNAVAILABLE', 'EX_USAGE', 'F_LOCK', 'F_OK', 'F_TEST',
'F_TLOCK', 'F_ULOCK', 'GRND_NONBLOCK', 'GRND_RANDOM', 'GenericAlias', 'MFD_ALLOW_SEALING', 'MFD_CLOEXEC', 'MFD_HUGETLB',
'MFD_HUGE_16GB', 'MFD_HUGE_16MB', 'MFD_HUGE_1GB', 'MFD_HUGE_1MB', 'MFD_HUGE_256MB', 'MFD_HUGE_2GB', 'MFD_HUGE_2MB', 'MFD_HUGE_32MB',
'MFD_HUGE_512KB', 'MFD_HUGE_512MB', 'MFD_HUGE_64KB', 'MFD_HUGE_8MB', 'MFD_HUGE_MASK', 'MFD_HUGE_SHIFT', 'Mapping', 'MutableMapping',
'NGROUPS_MAX', 'O_ACCMODE', 'O_APPEND', 'O_ASYNC', 'O_CLOEXEC', 'O_CREAT', 'O_DIRECT', 'O_DIRECTORY', 'O_DSYNC', 'O_EXCL', 'O_FSYNC',
'O_LARGEFILE', 'O_NDELAY', 'O_NOATIME', 'O_NOCTTY', 'O_NOFOLLOW', 'O_NONBLOCK', 'O_PATH', 'O_RDONLY', 'O_RDWR', 'O_RSYNC', 'O_SYNC',
'O_TMPFILE', 'O_TRUNC', 'O_WRONLY', 'POSIX_FADV_DONTNEED', 'POSIX_FADV_NOREUSE', 'POSIX_FADV_NORMAL', 'POSIX_FADV_RANDOM',
'POSIX_FADV_SEQUENTIAL', 'POSIX_FADV_WILLNEED', 'POSIX_SPAWN_CLOSE', 'POSIX_SPAWN_DUP2', 'POSIX_SPAWN_OPEN', 'PRIO_PGRP',
'PRIO_PROCESS', 'PRIO_USER', 'P_ALL', 'P_NOWAIT', 'P_NOWAITO', 'P_PGID', 'P_PID', 'P_PIDFD', 'P_WAIT', 'PathLike', 'RTLD_DEEPBIND',
'RTLD_GLOBAL', 'RTLD_LAZY', 'RTLD_LOCAL', 'RTLD_NODELETE', 'RTLD_NOLOAD', 'RTLD_NOW', 'RWF_APPEND', 'RWF_DSYNC', 'RWF_HIPRI',
'RWF_NOWAIT', 'RWF_SYNC', 'R_OK', 'SCHED_BATCH', 'SCHED_FIFO', 'SCHED_IDLE', 'SCHED_OTHER', 'SCHED_RESET_ON_FORK', 'SCHED_RR',
'SEEK_CUR', 'SEEK_DATA', 'SEEK_END', 'SEEK_HOLE', 'SEEK_SET', 'SPLICE_F_MORE', 'SPLICE_F_MOVE', 'SPLICE_F_NONBLOCK', 'ST_APPEND',
'ST_MANDLOCK', 'ST_NOATIME', 'ST_NODEV', 'ST_NODIRATIME', 'ST_NOEXEC', 'ST_NOSUID', 'ST_RDONLY', 'ST_RELATIME', 'ST_SYNCHRONOUS',
'ST_WRITE', 'TMP_MAX', 'WCONTINUED', 'WCOREDUMP', 'WEXITED', 'WEXITSTATUS', 'WIFCONTINUED', 'WIFEXITED', 'WIFSIGNALED', 'WIFSTOPPED',
'WNOHANG', 'WNOWAIT', 'WSTOPPED', 'WSTOPSIG', 'WTERMSIG', 'WUNTRACED', 'W_OK', 'XATTR_CREATE', 'XATTR_REPLACE', 'XATTR_SIZE_MAX',
'X_OK', '_Environ', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__',
'__spec__', '_check_methods', '_execvpe', '_exists', '_exit', '_fspath', '_fwalk', '_get_exports_list', '_spawnvef', '_walk',
'_wrap_close', 'abc', 'abort', 'access', 'altsep', 'chdir', 'chmod', 'chown', 'chroot', 'close', 'closerange', 'confstr',
'confstr_names', 'copy_file_range', 'cpu_count', 'ctermid', 'curdir', 'defpath', 'device_encoding', 'devnull', 'dup', 'dup2',
'environ', 'environb', 'error', 'eventfd', 'eventfd_read', 'eventfd_write', 'execl', 'execle', 'execlp', 'execlpe', 'execv', 'execve',
'execvp', 'execvpe', 'extsep', 'fchdir', 'fchmod', 'fchown', 'fdatasync', 'fdopen', 'fork', 'forkpty', 'fpathconf', 'fsdecode',
'fsencode', 'fspath', 'fstat', 'fstatvfs', 'fsync', 'ftruncate', 'fwalk', 'get_blocking', 'get_exec_path', 'get_inheritable',
'get_terminal_size', 'getcwd', 'getcwdb', 'getegid', 'getenv', 'getenvb', 'geteuid', 'getgid', 'getgrouplist', 'getgroups',
'getloadavg', 'getlogin', 'getpgid', 'getpgrp', 'getpid', 'getppid', 'getpriority', 'getrandom', 'getresgid', 'getresuid', 'getsid',
'getuid', 'getxattr', 'initgroups', 'isatty', 'kill', 'killpg', 'lchown', 'linesep', 'link', 'listdir', 'listxattr', 'lockf', 'lseek',
'lstat', 'major', 'makedev', 'makedirs', 'memfd_create', 'minor', 'mkdir', 'mkfifo', 'mknod', 'name', 'nice', 'open', 'openpty',
'pardir', 'path', 'pathconf', 'pathconf_names', 'pathsep', 'pidfd_open', 'pipe', 'pipe2', 'popen', 'posix_fadvise', 'posix_fallocate',
'posix_spawn', 'posix_spawnp', 'pread', 'preadv', 'putenv', 'pwrite', 'pwritev', 'read', 'readlink', 'readv', 'register_at_fork',
'remove', 'removedirs', 'removexattr', 'rename', 'renames', 'replace', 'rmdir', 'scandir', 'sched_get_priority_max',
'sched_get_priority_min', 'sched_getaffinity', 'sched_getparam', 'sched_getscheduler', 'sched_param', 'sched_rr_get_interval',
'sched_setaffinity', 'sched_setparam', 'sched_setscheduler', 'sched_yield', 'sendfile', 'sep', 'set_blocking', 'set_inheritable',
'setegid', 'seteuid', 'setgid', 'setgroups', 'setpgid', 'setpgrp', 'setpriority', 'setregid', 'setresgid', 'setresuid', 'setreuid',
'setsid', 'setuid', 'setxattr', 'spawnl', 'spawnle', 'spawnlp', 'spawnlpe', 'spawnv', 'spawnve', 'spawnvp', 'spawnvpe', 'splice', 'st',
'stat', 'stat_result', 'statvfs', 'statvfs_result', 'strerror', 'supports_bytes_environ', 'supports_dir_fd', 'supports_effective_ids',
'supports_fd', 'supports_follow_symlinks', 'symlink', 'sync', 'sys', 'sysconf', 'sysconf_names', 'system', 'tcgetpgrp', 'tcsetpgrp',
'terminal_size', 'times', 'times_result', 'truncate', 'ttyname', 'umask', 'uname', 'uname_result', 'unlink', 'unsetenv', 'urandom',
'utime', 'wait', 'wait3', 'wait4', 'waitid', 'waitid_result', 'waitpid', 'waitstatus_to_exitcode', 'walk', 'write', 'writev']
>>>
```

Il modulo os di Python

Contiene funzioni utili ad interagire con il sistema operativo. Qualche esempio:

```
>>> import os
>>> info = os.uname()
>>> type(info)
<class 'posix.uname_result'>
>>> info
posix.uname_result(sysname='Linux', nodename='piero-XPS-9320', release='6.8.0-52-
generic', version='#53~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Wed Jan 15 19:18:46 UTC 2',
machine='x86_64')
>>> info = "{0}".format(info)
>>> type(info)
<class 'str'>
>>> info
"posix.uname_result(sysname='Linux', nodename='piero-XPS-9320', release='6.8.0-52-
generic', version='#53~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Wed Jan 15 19:18:46 UTC 2',
machine='x86_64')"
>>> with open("uname.txt", "w") as text_file:
...     text_file.write("%s" % info)
...
184
>>>
```

Il programma more.py step by step

```
"""
split and interactively page a string or file of text
"""

import os
from tkinter.simpledialog import askinteger, askstring
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
```

impostiamo la dir di lavoro usando l'operatore di formattazione %s

Il programma more.py step by step

```
"""
split and interactively page a string or file of text
"""

import os
from tkinter.simpledialog import askinteger, askstring
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
filename = askstring('Entry', 'Inserisci il nome del file di testo che vuoi leggere
(senza estensione)')
```

creiamo una finestra di dialogo per chiedere all'utente il nome del file di testo da
leggere
il nome del file viene salvato nella variabile, di tipo stringa, <`filename`>

Il programma more.py step by step

```
"""
split and interactively page a string or file of text
"""

import os
from tkinter.simpledialog import askinteger, askstring
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
filename = askstring('Entry', 'Inserisci il nome del file di testo che vuoi leggere
(senza estensione)')
numlines = askinteger('Entry', 'Inserisci il numero di righe del blocco')
```

creiamo una finestra di dialogo per chiedere all'utente da quante righe dev'essere
formato ogni blocco di testo
tale numero viene salvato nella variabile numerica (di tipo intero) <`numlines`>

Il programma more.py step by step

```
with open('%s.txt' % filename, 'r') as file:
    text = file.read()
```

leggiamo il file di testo, usando l'operatore di formattazione <%s>, a cui viene sostituito
il nome del file, memorizzato nella variabile di tipo stringa <`filename`>
il contenuto testuale del file viene letto nella variabile di tipo stringa <`text`>

Il programma more.py step by step

```
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
```

definiamo la funzione <more>, che ha come argomenti:
1) la stringa <text>, che rappresenta il testo del file
2) la variabile numerica <numlines>, il cui valore è il numero di righe di ogni blocco di
testo

Il programma more.py step by step

```
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
```

suddividiamo il testo nelle singole righe, creando la **list di righe** <`lines`>

Il programma more.py step by step

```
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
    while lines:
       chunk = lines[:numlines]
       lines = lines[numlines:]
```

finché non si sia arrivati alla fine della list di righe <`lines`>, prendi le prime <`numlines`>
righe e memorizzale nella nuova list <`chunk`>
<`:numlines`> significa "le prime `numlines` righe", per es. le prime 50 righe del file
(dalla riga 0 alla riga 49)
la list <`lines`> viene aggiornata: le vengono assegnate le righe dalla 50 all'ultima

nel ciclo successivo, a <`chunk`> vengono assegnate le righe dalla 50 alla 99
la list <`lines`> viene nuovamente aggiornata: le vengono assegnate le righe dalla 100
all'ultima

Il programma more.py step by step

```
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
    while lines:
       chunk = lines[:numlines]
       lines = lines[numlines:]
       for line in chunk: print(line)
```

stampa (invia all'output standard, lo schermo) le righe del blocco

Il programma more.py step by step

```
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
    while lines:
       chunk = lines[:numlines]
       lines = lines[numlines:]
       for line in chunk: print(line)
       if lines and input('More?') not in ['y', 'Y']: break
```

se ci sono ancora righe ma alla domanda <More?> l'utente non risponde sì, interrompi
l'esecuzione del programma

Il programma more.py step by step

```
with open('%s.txt' % filename, 'r') as file:
    text = file.read()

def more(text, numlines):
    lines = text.splitlines()
    while lines:
       chunk = lines[:numlines]
       lines = lines[numlines:]
       for line in chunk: print(line)
       if lines and input('More?') not in ['y', 'Y']: break

more(text, numlines)
```

chiamata della funzione <more>

Successione di Fibonacci

Questa voce o sezione sull'argomento matematica è ritenuta da controllare.
Motivo: *È stato inserito parecchio materiale mal formattato, con notazioni diverse rispetto al resto della voce, alcuni passi sono poco contestualizzati e di altri non si coglie la rilevanza*

Partecipa alla discussione e/o correggi la voce. Segui i suggerimenti del progetto di riferimento.

In matematica, la **successione di Fibonacci** (detta anche **successione aurea**) è una successione di numeri interi in cui ciascun numero è la somma dei due precedenti, eccetto i primi due che sono, per definizione, 0 e 1.[1] Questa successione, indicata con F_n o con $\mathrm{Fib}(n)$, è definita ricorsivamente: partendo dai primi due elementi, $F_0 = 0$ e $F_1 = 1$, ogni altro elemento della successione sarà dato dalla relazione:

$$F_n = F_{n-1} + F_{n-2}.$$

Gli elementi F_n sono anche detti *numeri di Fibonacci*. I primi termini della successione di Fibonacci, che prende il nome dal matematico pisano del XIII secolo Leonardo Fibonacci, sono: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, \ldots$

The while statement

```
>>> # Fibonacci series:
>>> # the sum of two elements defines the next
>>> a, b = 0, 1
```

- The first line contains a multiple assignment: the variables a and b simultaneously get the new values 0 and 1

```
>>> a
0
>>> b
1
>>> while a < 10:
...     print(a)
...     a, b = b, a+b
...
0
1
1
2
3
5
8
>>>
```

The **while** loop executes as long as the condition (here: a < 10) remains true. In Python, like in C, any non-zero integer value is true; zero is false. The condition may also be a string or list value, in fact any sequence; anything with a non-zero length is true, empty sequences are false. The test used in the example is a simple comparison. The standard comparison operators are written the same as in C: < (less than), > (greater than), == (equal to), <= (less than or equal to), >= (greater than or equal to) and != (not equal to).

Fibonacci series

```
>>> a, b = 0, 1
>>> a
0
>>> b
1
>>> a, b = b, a+b
>>> a
1
>>> b
1
>>> a, b = b, a+b
>>> a
1
>>> b
2
>>> a, b = b, a+b
>>> a
2
>>> b
3
>>>
```

Fibonacci series

```
>>> a, b = b, a+b
>>> a
3
>>> b
5
>>> a, b = b, a+b
>>> a
5
>>> b
8
>>> a, b = b, a+b
>>> a
8
>>> b
13
>>> a, b = b, a+b
>>> a
13
>>> b
21
>>>
```

```
>>> a, b = 0, 1
>>> while a < 22:
...     print(a)
...     a, b = b, a+b
...
0
1
1
2
3
5
8
13
21
>>>
```

The print() function writes the value of the argument(s) it is given.

The body of the loop is indented: indentation is Python's way of grouping statements. At the interactive prompt, **you have to type a tab for each indented line**. In practice you will prepare more complicated input for Python with a text editor; all decent text editors have an auto-indent facility. When a compound statement is entered interactively, it must be followed by a blank line to indicate completion (since the parser cannot guess when you have typed the last line). Note that each line within a basic block must be indented by the same amount.

The print() function

The **print()** function handles multiple arguments, **floating-point quantities**, and strings. Strings are printed without quotes, and a space is inserted between items, so you can format things nicely, like this:

```
>>> i = 256 ** 2
>>> i
65536
>>> print('The value of i is', i)
The value of i is 65536
>>> a, b = 0, 1
>>> while a < 1000:
...     print(a, end=',')
...     a, b = b, a+b
...
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,>>>
>>>
```

The keyword argument **end** can be used to avoid the newline after the output, or end the output with a different string:

Il programma alimenti.py

```
piero@piero-XPS-9320:~$ cd Scaricati/
piero@piero-XPS-9320:~/Scaricati$ wget
https://github.com/pierorivoira/COMPUTER-SCIENCE/raw/refs/heads/main/ALIMENTI.tar.gz
--2025-05-15 10:35:28--  https://github.com/pierorivoira/COMPUTER-SCIENCE/raw/refs/heads/main/
ALIMENTI.tar.gz
Risoluzione di github.com (github.com)... 140.82.121.4
Connessione a github.com (github.com)|140.82.121.4|:443... connesso.
Richiesta HTTP inviata, in attesa di risposta... 302 Found
Posizione: https://raw.githubusercontent.com/pierorivoira/COMPUTER-SCIENCE/refs/heads/main/
ALIMENTI.tar.gz [segue]
--2025-05-15 10:35:28--  https://raw.githubusercontent.com/pierorivoira/COMPUTER-SCIENCE/refs/
heads/main/ALIMENTI.tar.gz
Risoluzione di raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133,
185.199.111.133, 185.199.108.133, ...
Connessione a raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443...
connesso.
Richiesta HTTP inviata, in attesa di risposta... 200 OK
Lunghezza: 16456 (16K) [application/octet-stream]
Salvataggio in: ‘ALIMENTI.tar.gz’

ALIMENTI.tar.gz    100%[===============>]  16,07K  --.-KB/s    in 0,008s

2025-05-15 10:35:29 (2,05 MB/s) - ‘ALIMENTI.tar.gz’ salvato [16456/16456]

piero@piero-XPS-9320:~/Scaricati$
```

Il programma alimenti.py

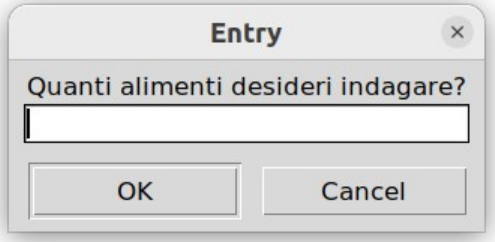
```
piero@piero-XPS-9320:~/Scaricati$ tar -xvzf ALIMENTI.tar.gz
ALIMENTI/
ALIMENTI/Leguminosa.py
ALIMENTI/Foraggio.py
ALIMENTI/Sottoprodotto.py
ALIMENTI/Farina_di_estrazione.py
ALIMENTI/Fonte_di_fibra.py
ALIMENTI/Cereale.py
ALIMENTI/importa_fieno_prato_stabile_primo_taglio.py
ALIMENTI/alimenti.py
piero@piero-XPS-9320:~/Scaricati$
```

Il programma alimenti.py

```
piero@piero-XPS-9320:~/Scaricati$ cd
piero@piero-XPS-9320:~$ pwd
/home/piero
piero@piero-XPS-9320:~$ python3 Scaricati/ALIMENTI/alimenti.py
```

Il programma alimenti.py

```
*                   *
*                   *
* * * * * * * * * * *
agrumi_pastazzo_secco
birra_lievito
calcio_saponi
cocco_pannello
frumento_tenero_germe
latte_magro_polvere
latte_siero
manioca
melasso_barbabietola
melasso_canna
patate_essiccate
proteina_patata
riso_gemma
riso_grana_verde
riso_pula_commerciale
riso_pula_vergine
riso_rottura
```



Escape Sequences Represent Special Characters

- Sono caratteri speciali che consentono di inserire in una stringa caratteri che non possono essere digitati facilmente con la tastiera

- Il carattere < \ >, ed uno o più caratteri che lo seguono nella sequenza, viene sostituito da un *singolo* carattere nell'oggetto di tipo stringa risultante

- La combinazione di *escape* specifica il numero binario corrispondente a quel determinato carattere

```
>>> s = "piero\ninsegna\tall'Istituto\v Agrario\a\\"
>>> s
"piero\ninsegna\tall'Istituto\x0b Agrario\x07\\"
>>> print(s)
piero
insegna  all'Istituto
                    Agrario\
>>>
```

Escape Sequences Represent Special Characters

`\a` bell
`\n` new line
`\t` horizontal tab
`\v` vertical tab

ESERCIZIO:

Scrivere un programma che disegni un quadrato di asterischi (10 per lato)

Escape Sequences Represent Special Characters

SOLUZIONE:

```
# script3.py
l = '*\t\t  *\n*\t\t  *\n*\t\t  *\n*\t\t  *\n*\t\t  *\n*\t\t  *\n*\t\t  *\n*\
t\t  *'
print('* ' * 10)
print(l)
print('* ' * 10)
```

Escape Sequences Represent Special Characters

SOLUZIONE:

```
# script3.py
l = '*\t\t  *\n*\t\t  *\n*\t\t  *\n*\t\t  *\n*\t\t  *\n*\t\t  *\n*\t\t  *\n*\
t\t  *'
print('* ' * 10)
print(l)
print('* ' * 10)

piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 script3.py
* * * * * * * * * *
*                 *
*                 *
*                 *
*                 *
*                 *
*                 *
*                 *
*                 *
*                 *
* * * * * * * * * *
```

ESERCIZIO:
scrivere un programma che disegni un triangolo rettangolo

Suggerimento:

```
print('* ' * 10)
# traccia la base
```

SOLUZIONE 1: triangolo pieno

```
print('*')
print('* ' * 2)
print('* ' * 3)
print('* ' * 4)
print('* ' * 5)
print('* ' * 6)
print('* ' * 7)
print('* ' * 8)
print('* ' * 9)
print('* ' * 10)
```

SOLUZIONE 1: triangolo pieno

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3 disegna_triangolo_pieno.py
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$
```

SOLUZIONE 2: triangolo vuoto

```
print('*')
print('* *')
print('*   *')
print('*     *')
print('*       *')
print('*         *')
print('*           *')
print('*             *')
print('*               *')
print('*                 *')
print('* * * * * * * * * *')
```

SOLUZIONE 2: triangolo vuoto

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3
disegna_triangolo_vuoto_1.py
*
* *
*   *
*     *
*       *
*         *
*           *
*             *
*               *
*                 *
*                   *
* * * * * * * * * * *
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$
```

SOLUZIONE 3: triangolo vuoto (due sole righe di codice)

```
f = '*\n* *\n*   *\n*     *\n*       *\n*         *\n*           *\n*             *\n*                 *\n*                     *\n* * * * * * * * * *'
print(f)
```

Files binari

Contengono stringhe di bytes (*binary data*) che possono essere creati ed estratti con il modulo **`struct`**

I dati in Python

“**Oggetti**” (*objects*), porzioni di memoria alle quali sono associati valori ed insiemi di operazioni

Il numero pi

```
>>> import math
>>> math.pi
3.141592653589793
>>>
```

Pi (π) is the **ratio of a circle's circumference (c) to its diameter (d)**:

$\pi = c/d$

- This ratio is always the same for any circle.
- Pi is an irrational number, which means it can't be expressed as a simple fraction. Therefore, pi has an infinite number of decimal places, but it can be approximated as 22/7, or 3.141.
- The pi value is given to fifteen decimal places in Python (the number of digits provided depends on the underlying C compiler); Python prints the first fifteen digits by default, and math.pi always returns a float value.

Il numero pi

```
>>> import math
>>> r = 3
>>> circumference = 2 * math.pi * r
>>> f"Circumference of a Circle = 2 * {math.pi:.4} * {r} = {circumference:.4}"
'Circumference of a Circle = 2 * 3.142 * 3 = 18.85'
>>> f"Circumference of a Circle = 2 * {math.pi:.6} * {r} = {circumference:.4}"
'Circumference of a Circle = 2 * 3.14159 * 3 = 18.85'
>>> r = 5
>>> area = math.pi * r ** 2
>>> f"Area of a Circle = {math.pi:.4} * {r}^2 = {area:.4}"
'Area of a Circle = 3.142 * 5^2 = 78.54'
>>>
```

Il numero tau

```
>>> import math
>>> math.tau
6.283185307179586
>>>
```

Il numero magico <e> (numero di Eulero)

```
>>> import math
>>> math.e
2.718281828459045
>>>
```

Il numero magico <e>

```
> 10*9*8*7*6*5*4*3*2*1
[1] 3628800
> factorial(10)
[1] 3628800
> Euler = function(n,z)
+ {
+ result = NULL
+ numbers = seq(1:n)
+ for (i in numbers)
+ {
+ result[i] = (z^i)/factorial(i)
+ }
+ return(1 + sum(result))
+ }
```

Per $z = 10$:

$e^z = 1 + z/1! + z^2/2! + z^3/3! + z^4/4! + \ldots \cong$

$e^{10} = 1 + 10/1! + 10^2/2! + 10^3/3! + 10^4/4! + \ldots \cong$

$\cong 22026.47$

Esempio:

```
> Euler(100,10)
[1] 22026.47
> exp(10)
[1] 22026.47
```

Esercizio: per quale valore di *z* si ottiene *e*?

```
> Euler(100,?)
```

Il numero magico <e>

```
> 10*9*8*7*6*5*4*3*2*1
[1] 3628800
> factorial(10)
[1] 3628800
> Euler = function(n,z)
+ {
+ result = NULL
+ numbers = seq(1:n)
+ for (i in numbers)
+ {
+ result[i] = (z^i)/factorial(i)
+ }
+ return(1 + sum(result))
+ }

> Euler(100,1)
[1] 2.718282
```

Per $z = 1$:

$e = 1/0! + 1/1! + 1/2! + 1/3! + 1/4! + \ldots \cong 2.718282$

Per $z = 10$:

$e^z = 1 + z/1! + z^2/2! + z^3/3! + z^4/4! + \ldots \cong$

$e^{10} = 1 + 10/1! + 10^2/2! + 10^3/3! + 10^4/4! + \ldots \cong$

$\cong 22026.47$

Il numero magico <e>

```
> 10*9*8*7*6*5*4*3*2*1
[1] 3628800
> factorial(10)
[1] 3628800
> Euler = function(n,z)
+ {
+ result = NULL
+ numbers = seq(1:n)
+ for (i in numbers)
+ {
+ result[i] = (z^i)/factorial(i)
+ }
+ return(1 + sum(result))
+ }

> Euler(100,1)
[1] 2.718282
```

Per $z = 1$:

$e = 1/0! + 1/1! + 1/2! + 1/3! + 1/4! + \ldots \cong 2.718282$

Per $z = 10$:

$e^z = z^0/0! + z/1! + z^2/2! + z^3/3! + z^4/4! + \ldots \cong$

$e^{10} = 1 + 10/1! + 10^2/2! + 10^3/3! + 10^4/4! + \ldots \cong$

$\cong 22026.47$

Per $z < 1$:

$e^z \cong 1 + z$

Es.

se z = 0.1

$e^{0.1} \cong 1 + 0.1 \cong 1.1$

```
> exp(0.1)
[1] 1.105171
> Euler(10,0.1)
[1] 1.105171
```

Il numero magico <e>

```
>>> import math
>>> math.factorial(10)
3628800
>>> somma = []
>>> for i in range(1, 100):
...     result = 1 / math.factorial(i)
...     somma.append(result)
...     output = 1 + sum(somma)
...
>>> output
2.7182818284590455
>>> math.e
2.718281828459045
>>>
```

Esercizio: modificare la funzione in modo da non dover aggiungere <1> al risultato

Il numero magico <e>

```
>>> import math
>>> math.factorial(10)
3628800
>>> somma = []
>>> for i in range(0, 100):
...     result = 1 / math.factorial(i)
...     somma.append(result)
...     output = sum(somma)
...
>>> output
2.7182818284590455
>>> math.e
2.718281828459045
>>>
```

Il numero magico <e>

```
>>> import math
>>> z = 10
>>> n = 100
>>> somma = []
>>> for i in range(n):
...     result = z**i / math.factorial(i)
...     somma.append(result)
...     output = sum(somma)
...
>>> output
22026.46579480671

>>> math.exp(10)
22026.465794806718
```

Quindi...

```
>>> for i in range(1,10):
...     print(i)
...
1
2
3
4
5
6
7
8
9
>>>
```

Quindi...

```
>>> for i in range(0,10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>>
```

Quindi...

```
>>> for i in range(10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>>
```

Il numero magico <e>

```
>>> import math
>>> def eleva_a_potenza(n,z):
...     somma = []
...     for i in range(n):
...             result = z**i / math.factorial(i)
...             somma.append(result)
...             output = sum(somma)
...     return output
...
>>> eleva_a_potenza(100,10)
22026.46579480671
>>>
```

Il programma eleva_a_potenza.py

```
import math
from tkinter.simpledialog import askfloat
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare (es. 100)')
z = askfloat('Entry', "Inserisci l'esponente")
print('La somma sarà effettuata su %d termini' % n)
print('La base dei logaritmi naturali %f sarà elevata alla %f-esima potenza' %
(math.e, z))

def eleva_a_potenza(n,z):
    somma = []
    for i in range(n):
            result = z**i / math.factorial(i)
            somma.append(result)
            output = sum(somma)
    return output

output = eleva_a_potenza(n,z)
print('Il risultato è %f' % output)
```

Rappresentazione dei numeri in virgola mobile

13.25_{10} = **1101.01_2**

1*2^3 + **1***2^2 + **0***2^1 + **1***2^0 + **0***2^-1 + **1***2^-2 = 8 + 4 + 1 + 0.25 = 13.25

Mantissa

1101.01_2 = **$1.10101_2 \cdot 2^3$** Esponente, in base 10, della forma normalizzata

spostare la virgola di 3 posizioni verso sinistra significa dividere per 2^3
dividiamo e moltiplichiamo per 2^3 “normalizzando il numero”

calcoliamo l’esponente sommando 127 all’esponente della forma normalizzata:

$(3+127)_{10}$ = **130_{10}** = **10000010_2**

1*2^7 + 1*2^1 = 128 + 2 = 130

```
>>> print(bin(130))
0b10000010
>>>
```

Rappresentazione dei numeri in virgola mobile: single precision

	SEGNO	ESPONENTE	MANTISSA	
	0	**10000010**	**10101**00000000000000000	
bit	1	8	23	**32**

Il bit di segno è definito pari a:

- 0 per valori ≥ 0
- 1 per valori ≤ 0

- La mantissa è in forma normalizzata 1.xxxxx
- Essendo quell'1 sempre presente, solo la parte frazionaria del numero è memorizzata
- L'1 omesso viene detto bit nascosto (hidden bit)
- Abbiamo, così, un bit in più di precisione (23 bit memorizzati + 1 nascosto = 24 bit effettivi)

Il programma tartaglia.py

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 3
print('Il programma restituisce la riga del Triangolo di Tartaglia formata da
%d numeri' % N)
```

n=1

La riga del triangolo sarà formata da 4 numeri

```
sequenza_iniziale = [1,2,1]
unità = 1

for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova

print(sequenza_nuova)
print(len(sequenza_nuova))
```

Ogni sequenza inizia con <1>

i assume prima il valore <0> poi il valore <1>
j può assumere solo il valore <0>
sequenza_nuova = [1]
Per j=0:
Per i = 0:
sequenza_iniziale[0] = 1
sequenza_iniziale[1] = 2
sequenza_iniziale[0]+sequenza_iniziale[1]=3
sequenza_nuova=[1,3]

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3
Python 3.10.12 (main, Feb  4 2025, 14:57:36) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> n=1
>>> for j in range(n):
...     print(j)
...
0
>>>
```

Il programma tartaglia.py

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 3
print('Il programma restituisce la riga del Triangolo di Tartaglia formata da
%d numeri' % N)

sequenza_iniziale = [1,2,1]
unità = 1

for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova

print(sequenza_nuova)
print(len(sequenza_nuova))
```

n=1

i assume prima il valore <0> poi il valore <1>
j può assumere solo il valore <0>
sequenza_nuova=[1,3]
Per j=0:
Per i = 1:
sequenza_iniziale[1] = 2
sequenza_iniziale[2] = 1
sequenza_iniziale[1]+sequenza_iniziale[2]=3
sequenza_nuova=[1,3,3]

Il programma tartaglia.py

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 3
print('Il programma restituisce la riga del Triangolo di Tartaglia formata da
%d numeri' % N)

sequenza_iniziale = [1,2,1]
unità = 1

for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova

print(sequenza_nuova)
print(len(sequenza_nuova))
```

n=1

i assume prima il valore <0> poi il valore <1>
j può assumere solo il valore <0>
sequenza_iniziale=[1,3,3,1]

sequenza_nuova=[1,3,3,1]

Ogni sequenza finisce con <1>

Avendo esaurito i possibili valori di i, esco dal for annidato
Aggiungo un <1> finale alla sequenza nuova
Aggiorno la sequenza iniziale
Esco anche dal for esterno
Stampo la sequenza 1 3 3 1

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3
Python 3.10.12 (main, Feb  4 2025, 14:57:36) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> sequenza_iniziale=[1,3,3,1]
>>> len(sequenza_iniziale)
4
>>> for i in range(len(sequenza_iniziale)-1):
...     print(i)
...
0
1
2
>>>
```

Il programma tartaglia.py

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 3
print('Il programma restituisce la riga del Triangolo di Tartaglia formata da
%d numeri' % N)

sequenza_iniziale = [1,2,1]
unità = 1

for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova

print(sequenza_nuova)
print(len(sequenza_nuova))
```

n=2

i assume prima il valore <0> poi il valore <1> poi il valore <2>
j assume prima il valore <0> poi il valore <1>
sequenza_iniziale=[1,3,3,1]
Per j=1:
Per i = 0:
sequenza_iniziale[0] = 1
sequenza_iniziale[1] = 3
sequenza_iniziale[0]+sequenza_iniziale[1]=4

Ogni sequenza finisce con <1>

sequenza_nuova=[1,4]
Avendo esaurito i possibili valori di i, esco dal for annidato
Aggiungo un <1> finale alla sequenza nuova
Aggiorno la sequenza iniziale
Se n=2, j può assumere anche il valore <1>
Non posso uscire dal for esterno

Il programma tartaglia.py

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 3
print('Il programma restituisce la riga del Triangolo di Tartaglia formata da
%d numeri' % N)

sequenza_iniziale = [1,2,1]
unità = 1

for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova

print(sequenza_nuova)
print(len(sequenza_nuova))
```

n=2

i assume prima il valore <0> poi il valore <1> poi il valore <2>
sequenza_iniziale=[1,3,3,1]
Per j=1:
Per i = 1:
sequenza_iniziale[1] = 3
sequenza_iniziale[2] = 3
sequenza_iniziale[1]+sequenza_iniziale[2]=6
sequenza_nuova=[1,4,6]

Ogni sequenza finisce con <1>

Avendo esaurito i possibili valori di i, esco dal for annidato
Aggiungo un <1> finale alla sequenza nuova
Aggiorno la sequenza iniziale
Se n=2, j può assumere anche il valore <1>
Non posso uscire dal for esterno

Il programma tartaglia.py

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 3
print('Il programma restituisce la riga del Triangolo di Tartaglia formata da
%d numeri' % N)

sequenza_iniziale = [1,2,1]
unità = 1

for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova

print(sequenza_nuova)
print(len(sequenza_nuova))
```

n=2

i assume prima il valore <0> poi il valore <1> poi il valore <2>
sequenza_iniziale=[1,3,3,1]
Per j=1:
Per i = 2:
sequenza_iniziale[2] = 3
sequenza_iniziale[3] = 1
sequenza_iniziale[2]+sequenza_iniziale[3]=4
sequenza_nuova=[1,4,6,4]

Ogni sequenza finisce con <1>

Avendo esaurito i possibili valori di i, esco dal for annidato
sequenza_nuova=[1,4,6,4,1]
Aggiungo un <1> finale alla sequenza nuova
Aggiorno la sequenza iniziale
Avendo esaurito I possibili valori di j, esco dal for esterno

ESERCIZIO:

Modificare il programma <tartaglia.py> affinché stampi in output l'intero triangolo di Tartaglia

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 3
print('Il programma restituisce la riga del Triangolo di Tartaglia formata da %d
numeri' % N)

sequenza_iniziale = [1,2,1]
unità = 1


for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova

print(sequenza_nuova)
```

SOLUZIONE 1:

Modificare il programma <tartaglia.py> affinché stampi in output l'intero triangolo di Tartaglia

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 1
print('Il programma restituisce il Triangolo di Tartaglia formata da %d righe' % N)
triangolo = []
sequenza_iniziale = [1,2,1]
unità = 1

print('[1, 2, 1]')

for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova
    print(sequenza_nuova)
```

SOLUZIONE 2:

Modificare il programma <tartaglia.py> affinché stampi in output l'intero triangolo di Tartaglia

```
from tkinter.simpledialog import askinteger
n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
N = n + 1
print('Il programma restituisce il Triangolo di Tartaglia formata da %d righe' % N)
triangolo = []
sequenza_iniziale = [1,2,1]
unità = 1
for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova
    triangolo.append(sequenza_nuova)
print('\n')
print('[1, 2, 1]')
print('\n')
for k in range(len(triangolo)):
   print(triangolo[k])
   print('\n')
```

Il modulo tartaglia_triangolo_modulo.py

```
from tkinter.simpledialog import askinteger
# n = askinteger('Entry', 'Inserisci il numero di termini da sommare')
n = 97
N = n + 3
print('Il programma restituisce la riga del Triangolo di Tartaglia formata da
%d numeri' % N)
sequenza_iniziale = [1,2,1]
triangolo = []
unità = 1

for j in range(n):
    sequenza_nuova = [1]
    for i in range(len(sequenza_iniziale)-1):
        sequenza_nuova.append(sequenza_iniziale[i]+sequenza_iniziale[i+1])
    sequenza_nuova.append(unità)
    sequenza_iniziale = sequenza_nuova
    triangolo.append(sequenza_nuova)
```

Il modulo tartaglia_triangolo_modulo.py

```
>>> from tartaglia_triangolo_modulo import triangolo
Il programma restituisce la riga del Triangolo di Tartaglia formata da 100
numeri
>>> triangolo[0]
[1, 3, 3, 1]
>>> triangolo[1]
[1, 4, 6, 4, 1]
>>> triangolo[10]
[1, 13, 78, 286, 715, 1287, 1716, 1716, 1287, 715, 286, 78, 13, 1]
>>> triangolo[37]
[1, 40, 780, 9880, 91390, 658008, 3838380, 18643560, 76904685, 273438880,
847660528, 2311801440, 5586853480, 12033222880, 23206929840, 40225345056,
62852101650, 88732378800, 113380261800, 131282408400, 137846528820,
131282408400, 113380261800, 88732378800, 62852101650, 40225345056,
23206929840, 12033222880, 5586853480, 2311801440, 847660528, 273438880,
76904685, 18643560, 3838380, 658008, 91390, 9880, 780, 40, 1]
>>>
```

Usare Python all’interno di R

```
> install.packages('reticulate')
> library(reticulate)
> reticulate::py_config()
Would you like to create a default Python environment for the reticulate
package? (Sì/no/annulla) Sì
Errore: Tools for managing Python virtual environments are not installed.

Install venv with:
$ sudo apt-get install python3-venv python3-pip python3-dev

> q()
piero@piero-XPS-9320:~$ sudo apt-get install python3-venv python3-pip python3-
dev
>
```

Usare Python all’interno di R

```
> library(reticulate)
> reticulate::py_config()
Would you like to create a default Python environment for the reticulate
package? (Sì/no/annulla) Sì
...
Installing collected packages: numpy
Successfully installed numpy-2.2.3
Virtual environment 'r-reticulate' successfully created.
python:         /home/piero/.virtualenvs/r-reticulate/bin/python
libpython:
/usr/lib/python3.10/config-3.10-x86_64-linux-gnu/libpython3.10.so
pythonhome:
/home/piero/.virtualenvs/r-reticulate:/home/piero/.virtualenvs/r-reticulate
version:        3.10.12 (main, Jan 17 2025, 14:35:34) [GCC 11.4.0]
numpy:          /home/piero/.virtualenvs/r-reticulate/lib/python3.10/site-
packages/numpy
numpy_version:  2.2.3
> use_python('/bin/python3.10', required = TRUE)
>
```

Estrarre un numero (pseudo)casuale

```
>>> import random
>>> random.random()
0.8287089642821494
>>>
```

Oggetti “built-in”: numeri

```
>>> for i in range(1,11):
...     print(i)
...
1
2
3
4
5
6
7
8
9
10
>>>
```

Oggetti “built-in”: numeri

```
>>> import random
>>> random.random()
0.8287089642821494
>>> for i in range(1,11):
...     print(random.random())
...
0.8594926661744832
0.06936841835168606
0.5885420854284539
0.965105117036638
0.803728988120997
0.7559884522057374
0.17251377438513837
0.5684117024306461
0.39295735916522334
0.7685184418536651
>>>
```

Oggetti “built-in”: numeri

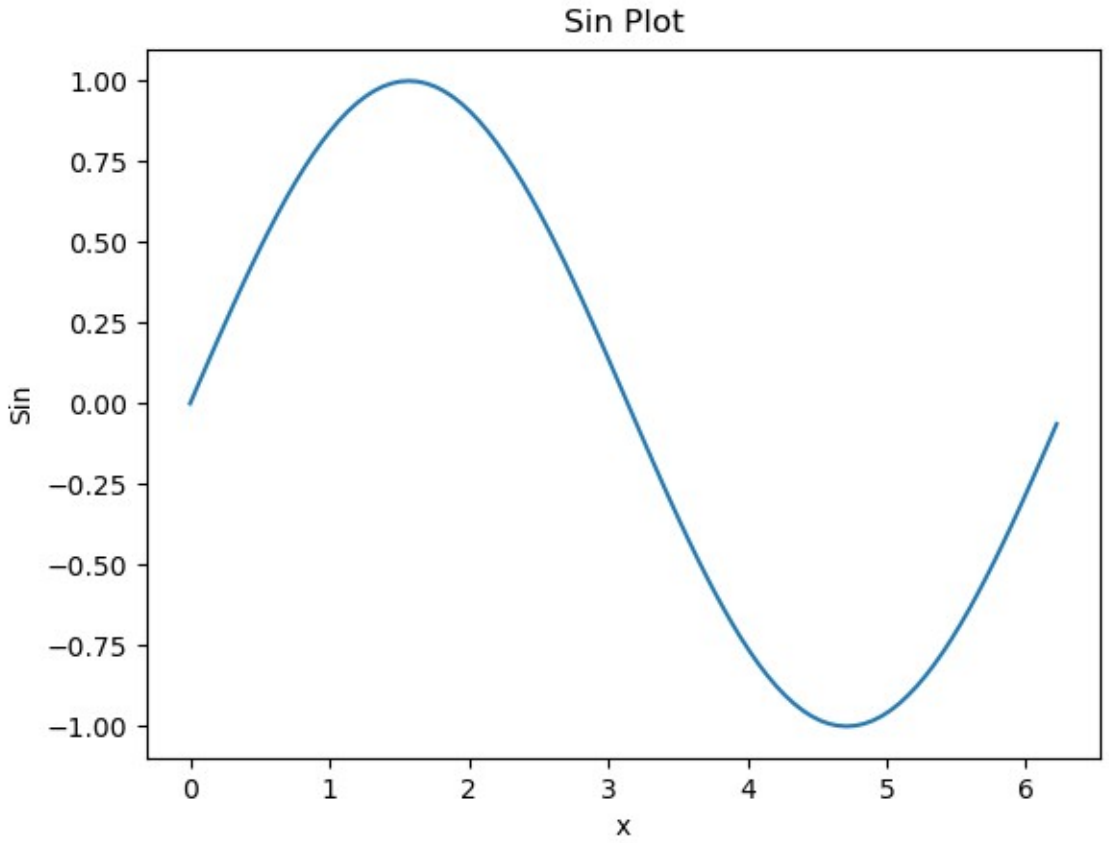
```
>>> import random
>>> randomlist = []
>>> for i in range(1,11):
...     n = random.random()
...     randomlist.append(n)
...
>>> randomlist
[0.31984567728569424, 0.5487040462281298, 0.6358083181180593,
0.9734906064546093, 0.6468907310587074, 0.21312048593018285,
0.8358032488432969, 0.18960849552618586, 0.6494585694906477,
0.5224444657617 85]
>>>
```

Costruire il grafico di una funzione

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> x = [i for i in range(100)]
>>> x
[0, 1, 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, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78,
79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99]
>>> x = np.array(x)
>>> x
array([ 0,  1,  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, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
>>> x = 2 * np.pi * x * 0.01
>>> np.pi
3.141592653589793
```

Costruire il grafico di una funzione

```
>>> y = np.sin(x)
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0x75a000587490>]
>>> plt.title('Sin Plot')
Text(0.5, 1.0, 'Sin Plot')
>>> plt.xlabel('x')
Text(0.5, 0, 'x')
>>> plt.ylabel('Sin')
Text(0, 0.5, 'Sin')
>>> plt.show()
>>>
```



Oggetti “built-in”: dictionaries

Python **dictionaries**, also known as **mappings**, are collections of other objects which are **stored by key** instead of by relative position. Mappings don’t maintain any reliable left-to-right order; they simply map keys to associated values and are also mutable: like lists, they may be changed in place and can grow and shrink on demand. Also like lists, they are a flexible tool for representing collections, but their more mnemonic keys are better suited when a collection’s items are named or labeled – fields of a database record, for example.

>>>

Dictionaries: importa_fieno_prato_stabile_primo_taglio.py

```
import os
import numpy as np
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
from Foraggio import fieno_prato_stabile_I_taglio
print('<fieno_prato_stabile_I_taglio> importato con successo dal modulo <Foraggio>!')
FIEPRASTA1TA = list(np.float_(fieno_prato_stabile_I_taglio))
FIEPRASTA1TA_dict = {}
FIEPRASTA1TA_dict['DM'] = FIEPRASTA1TA[0]
FIEPRASTA1TA_dict['PG'] = FIEPRASTA1TA[1]
FIEPRASTA1TA_dict['RDP'] = FIEPRASTA1TA[2]
FIEPRASTA1TA_dict['RUP'] = FIEPRASTA1TA[3]
FIEPRASTA1TA_dict['FG'] = FIEPRASTA1TA[4]
FIEPRASTA1TA_dict['NDF'] = FIEPRASTA1TA[5]
FIEPRASTA1TA_dict['ADF'] = FIEPRASTA1TA[6]
FIEPRASTA1TA_dict['FAT'] = FIEPRASTA1TA[7]
FIEPRASTA1TA_dict['ASH'] = FIEPRASTA1TA[10]
FIEPRASTA1TA_dict['Ca'] = FIEPRASTA1TA[11]
FIEPRASTA1TA_dict['P'] = FIEPRASTA1TA[12]
FIEPRASTA1TA_dict['Mg'] = FIEPRASTA1TA[13]
FIEPRASTA1TA_dict['K'] = FIEPRASTA1TA[14]
FIEPRASTA1TA_dict['S'] = FIEPRASTA1TA[15]
FIEPRASTA1TA_dict['Na'] = FIEPRASTA1TA[16]
FIEPRASTA1TA_dict['UFC'] = FIEPRASTA1TA[17]
FIEPRASTA1TA_dict['UFL'] = FIEPRASTA1TA[18]
FIEPRASTA1TA_dict['kcal'] = FIEPRASTA1TA[19]
FIEPRASTA1TA_dict['Mj'] = FIEPRASTA1TA[20]
print(FIEPRASTA1TA dict)
```

A list of dictionaries

```
>>> import os
>>> user = os.getlogin()
>>> user
'piero'
>>> os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
>>> from importa_fieno_prato_stabile_primo_taglio import FIEPRASTA1TA_dict
<fieno_prato_stabile_I_taglio> importato con successo dal modulo <Foraggio>!
{'DM': 87.0, 'PG': 9.5, 'RDP': 6.0, 'RUP': 3.5, 'FG': 30.0, 'NDF': 46.5, 'ADF': 34.8, 'FAT': 2.6,
'ASH': 7.7, 'Ca': 0.42, 'P': 0.23, 'Mg': 0.25, 'K': 1.3, 'S': 0.15, 'Na': 0.009, 'UFC': 0.41,
'UFL': 0.61, 'kcal': 1055.3, 'Mj': 4.42}
>>> from importa_fieno_prato_stabile_secondo_taglio import FIEPRASTA2TA_dict
<fieno_prato_stabile_II_taglio> importato con successo dal modulo <Foraggio>!
{'DM': 87.0, 'PG': 10.0, 'RDP': 6.3, 'RUP': 3.7, 'FG': 26.0, 'NDF': 45.0, 'ADF': 29.7, 'FAT': 3.2,
'ASH': 7.0, 'Ca': 0.6, 'P': 0.3, 'Mg': 0.25, 'K': 1.3, 'S': 0.15, 'Na': 0.009, 'UFC': 0.54, 'UFL':
0.65, 'kcal': 1124.5, 'Mj': 4.71}
>>> fieni = [FIEPRASTA1TA_dict, FIEPRASTA2TA_dict]
>>> fieni
[{'DM': 87.0, 'PG': 9.5, 'RDP': 6.0, 'RUP': 3.5, 'FG': 30.0, 'NDF': 46.5, 'ADF': 34.8, 'FAT': 2.6,
'ASH': 7.7, 'Ca': 0.42, 'P': 0.23, 'Mg': 0.25, 'K': 1.3, 'S': 0.15, 'Na': 0.009, 'UFC': 0.41,
'UFL': 0.61, 'kcal': 1055.3, 'Mj': 4.42}, {'DM': 87.0, 'PG': 10.0, 'RDP': 6.3, 'RUP': 3.7, 'FG':
26.0, 'NDF': 45.0, 'ADF': 29.7, 'FAT': 3.2, 'ASH': 7.0, 'Ca': 0.6, 'P': 0.3, 'Mg': 0.25, 'K': 1.3,
'S': 0.15, 'Na': 0.009, 'UFC': 0.54, 'UFL': 0.65, 'kcal': 1124.5, 'Mj': 4.71}]
>>>
```

A list of dictionaries

```
>>> fieni[1]
{'DM': 87.0, 'PG': 10.0, 'RDP': 6.3, 'RUP': 3.7, 'FG': 26.0, 'NDF': 45.0, 'ADF': 29.7, 'FAT': 3.2,
'ASH': 7.0, 'Ca': 0.6, 'P': 0.3, 'Mg': 0.25, 'K': 1.3, 'S': 0.15, 'Na': 0.009, 'UFC': 0.54, 'UFL':
0.65, 'kcal': 1124.5, 'Mj': 4.71}
>>> fieni[1][0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 0
>>> fieni[1]['DM']
87.0
>>>
```

Non è possibile accedere all'ingresso di un dizionario in base alla sua posizione!

Il programma systeminfo.py

```
import json, os
from tkinter.simpledialog import askstring
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
print("Puoi scegliere fra <sysname>, <nodename>, <release>, <version> e <machine>")
question = askstring('Entry', "Inserisci l'informazione desiderata")
info = os.uname()
info = "{0}".format(info)
with open("uname.txt", "w") as text_file:
    text_file.write("%s" % info)
with open('uname.txt') as f:
    newtext = f.read().replace("posix.uname_result(", "{'")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("=", "':")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(", ", ", '")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(")", "}")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("'", '"')
with open('uname.txt', 'w') as f:
    f.write(newtext)
```

Il programma systeminfo.py

```
>>> import os
>>> info = os.uname()
>>> info
posix.uname_result(sysname='Linux', nodename='piero-XPS-9320', release='6.8.0-
58-generic', version='#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', machine='x86_64')
>>> type(info)
<class 'posix.uname_result'>
>>> info = "{0}".format(info)
# convertiamo <info> in una stringa di caratteri
>>> type(info)
<class 'str'>
>>> info
"posix.uname_result(sysname='Linux', nodename='piero-XPS-9320',
release='6.8.0-58-generic', version='#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC
Fri Mar 28 16:09:21 UTC 2', machine='x86_64')"
>>>
```

Il programma systeminfo.py

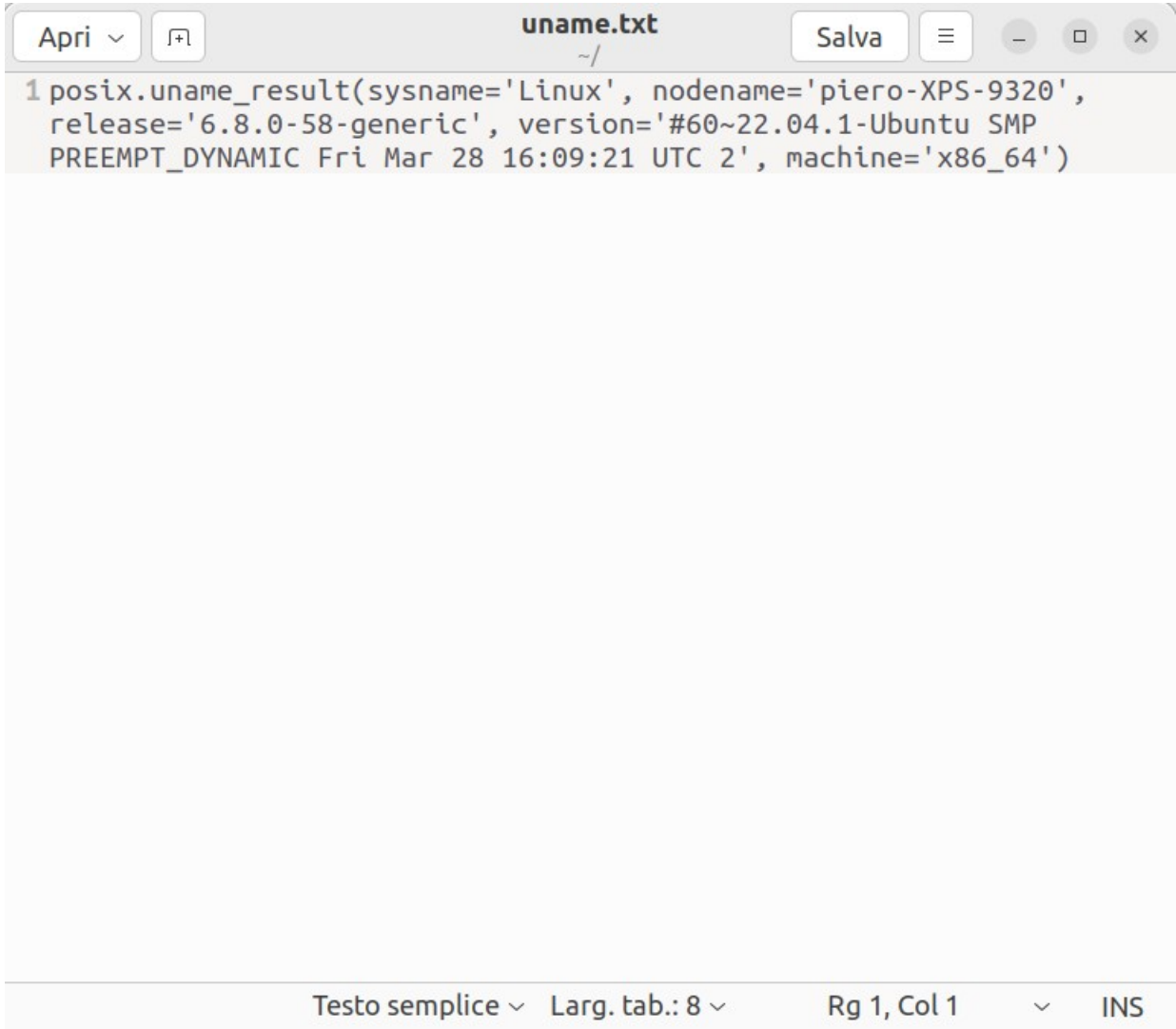
```
import json, os
from tkinter.simpledialog import askstring
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
print("Puoi scegliere fra <sysname>, <nodename>, <release>, <version> e <machine>")
question = askstring('Entry', "Inserisci l'informazione desiderata")
info = os.uname()
info = "{0}".format(info)
with open("uname.txt", "w") as text_file:
    text_file.write("%s" % info)
with open('uname.txt') as f:
    newtext = f.read().replace("posix.uname_result(", "{'")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("=", "':")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(", ", ", '")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(")", "}")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("'", '"')
with open('uname.txt', 'w') as f:
    f.write(newtext)
```

Il programma systeminfo.py

```
>>> with open("uname.txt", "w") as text_file:
...     text_file.write("%s" % info)
...
184
>>>
```

scriviamo la stringa <info> nel file di testo <uname.txt>

```
piero@piero-XPS-9320:~$ gedit uname.txt
```



Il programma systeminfo.py

```
with open('uname.txt') as f:
    newtext = f.read().replace("posix.uname_result(", "{'")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("=", "':")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(", ", ", '")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(")", "}")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("'", '"')
with open('uname.txt', 'w') as f:
    f.write(newtext)
```

Il programma systeminfo.py

```
>>> info
"posix.uname_result(sysname='Linux', nodename='piero-XPS-9320',
release='6.8.0-58-generic', version='#60~22.04.1-Ubuntu SMP
PREEMPT_DYNAMIC Fri Mar 28 16:09:21 UTC 2', machine='x86_64')"
>>> with open("uname.txt", "w") as text_file:
...     text_file.write("%s" % info)
...
184
>>> with open('uname.txt') as f:
...     newtext = f.read().replace("posix.uname_result(", "{'")
...
>>> newtext
"{'sysname='Linux', nodename='piero-XPS-9320', release='6.8.0-58-generic',
version='#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28 16:09:21 UTC
2', machine='x86_64')"
>>>
```

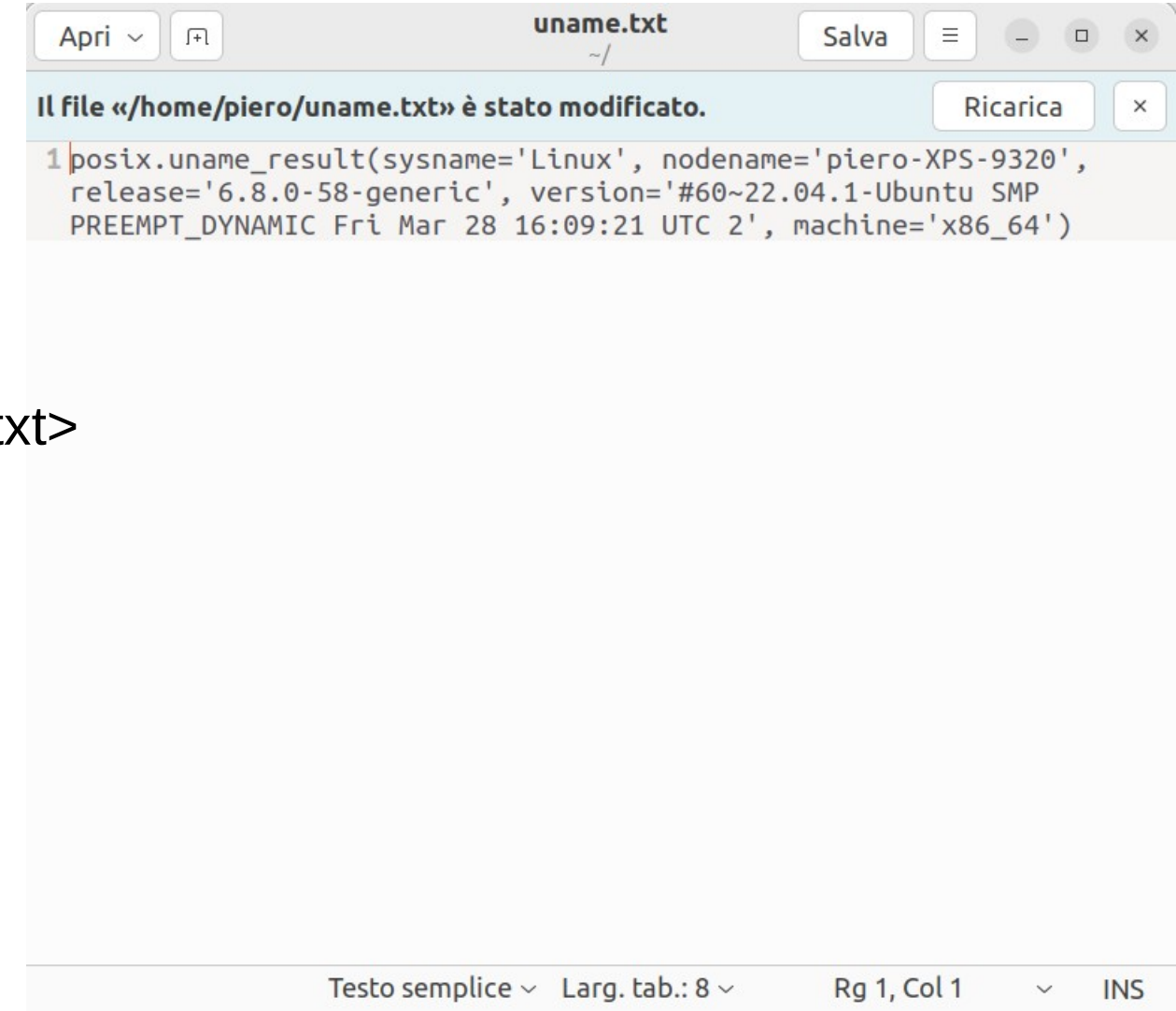
Il programma systeminfo.py

```
with open('uname.txt') as f:
    newtext = f.read().replace("posix.uname_result(", "{'")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("=", "':")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(", ", ", '")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(")", "}")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("'", '"')
with open('uname.txt', 'w') as f:
    f.write(newtext)
```

Il programma systeminfo.py

```
>>> newtext
"{'sysname='Linux', nodename='piero-XPS-9320', release='6.8.0-58-
generic', version='#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', machine='x86_64')"

>>> with open('uname.txt', 'w') as f:
...     f.write(newtext)
...
167
>>>
```

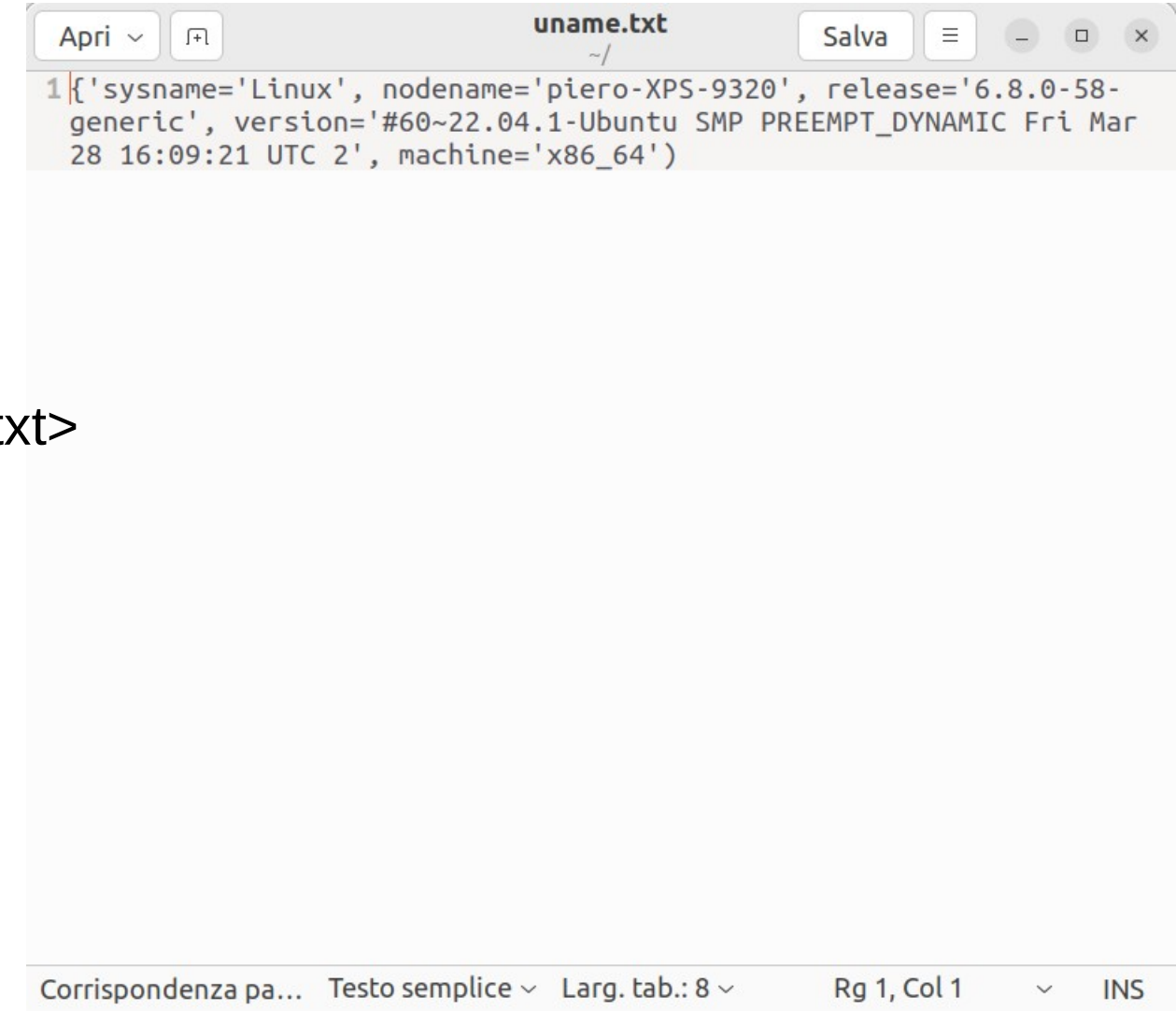
scriviamo la stringa <newtext> nel file <uname.txt>



Il programma systeminfo.py

```
>>> newtext
"{'sysname='Linux', nodename='piero-XPS-9320', release='6.8.0-58-
generic', version='#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', machine='x86_64')"

>>> with open('uname.txt', 'w') as f:
...     f.write(newtext)
...
167
>>>
```

scriviamo la stringa <newtext> nel file <uname.txt>



Il programma systeminfo.py

```
with open('uname.txt') as f:
    newtext = f.read().replace("posix.uname_result(", "{'")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("=", "':")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(", ", ", '")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(")", "}")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("'", '"')
with open('uname.txt', 'w') as f:
    f.write(newtext)
```

Il programma systeminfo.py

```
>>> newtext
"{'sysname='Linux', nodename='piero-XPS-9320', release='6.8.0-58-generic',
version='#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28 16:09:21 UTC
2', machine='x86_64')"
>>> with open('uname.txt', 'w') as f:
...     f.write(newtext)
...
167
>>> with open('uname.txt') as f:
...     newtext = f.read().replace("=", "':")
...
>>> newtext
"{'sysname':'Linux', nodename':'piero-XPS-9320', release':'6.8.0-58-
generic', version':'#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', machine':'x86_64')"
>>> with open('uname.txt', 'w') as f:
...     f.write(newtext)
...
172
>>>
```

salviamo le modifiche

Il programma systeminfo.py

```
with open('uname.txt') as f:
    newtext = f.read().replace("posix.uname_result(", "{'")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("=", "':")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(", ", ", '")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(")", "}")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("'", '"')
with open('uname.txt', 'w') as f:
    f.write(newtext)
```

Il programma systeminfo.py

```
>>> newtext
"{'sysname':'Linux', nodename':'piero-XPS-9320', release':'6.8.0-58-generic',
version':'#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28 16:09:21 UTC 2',
machine':'x86_64')"
>>> with open('uname.txt') as f:
...     newtext = f.read().replace(", ", ", '")
...
>>> newtext
"{'sysname':'Linux', 'nodename':'piero-XPS-9320', 'release':'6.8.0-58-
generic', 'version':'#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', 'machine':'x86_64')"

>>> with open('uname.txt', 'w') as f:
...     f.write(newtext)
...
176
>>>
```

salviamo le modifiche

Il programma systeminfo.py

```
with open('uname.txt') as f:
    newtext = f.read().replace("posix.uname_result(", "{'")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("=", "':")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(", ", ", '")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(")", "}")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("'", '"')
with open('uname.txt', 'w') as f:
    f.write(newtext)
```

Il programma systeminfo.py

```
>>> newtext
"{'sysname':'Linux', 'nodename':'piero-XPS-9320', 'release':'6.8.0-58-
generic', 'version':'#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', 'machine':'x86_64')"
>>> with open('uname.txt') as f:
...     newtext = f.read().replace(")", "}")
...
>>> newtext
"{'sysname':'Linux', 'nodename':'piero-XPS-9320', 'release':'6.8.0-58-
generic', 'version':'#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', 'machine':'x86_64'}"
>>> with open('uname.txt', 'w') as f:
...     f.write(newtext)
...
176
>>>
```

salviamo le modifiche

Il programma systeminfo.py

```
with open('uname.txt') as f:
    newtext = f.read().replace("posix.uname_result(", "{'")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("=", "':")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(", ", ", '")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace(")", "}")
with open('uname.txt', 'w') as f:
    f.write(newtext)
with open('uname.txt') as f:
    newtext = f.read().replace("'", '"')
with open('uname.txt', 'w') as f:
    f.write(newtext)
```

Il programma systeminfo.py

```
>>> newtext
"{'sysname':'Linux', 'nodename':'piero-XPS-9320', 'release':'6.8.0-58-
generic', 'version':'#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', 'machine':'x86_64'}"
>>> with open('uname.txt') as f:
...     newtext = f.read().replace("'", '"')
...
```

sostituiamo l'apostrofo con le virgolette

```
>>> newtext
'{"sysname":"Linux", "nodename":"piero-XPS-9320", "release":"6.8.0-58-
generic", "version":"#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2", "machine":"x86_64"}'
>>> with open('uname.txt', 'w') as f:
...     f.write(newtext)
...
176
>>>
```

salviamo le modifiche

Il programma systeminfo.py

```
with open('uname.txt', 'r') as file:
    info = file.read()
info = json.loads(info)
if question == 'sysname':
    output = info['sysname']
    print("Il sistema operativo è <%s>" % output)
elif question == 'nodename':
    output = info['nodename']
    print("Il nome del nodo è <%s>" % output)
elif question == 'release':
    output = info['release']
    print("Il nome della release è <%s>" % output)
elif question == 'version':
    output = info['version']
    print("Il nome della versione è <%s>" % output)
elif question == 'machine':
    output = info['machine']
    print("Il nome della macchina è <%s>" % output)
```

Il programma systeminfo.py

```
>>> with open('uname.txt', 'r') as file:
...     info = file.read()
...
>>> info
'{"sysname":"Linux", "nodename":"piero-XPS-9320", "release":"6.8.0-58-
generic", "version":"#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2", "machine":"x86_64"}'
>>>
```

Il programma systeminfo.py

```
with open('uname.txt', 'r') as file:
    info = file.read()
info = json.loads(info)
if question == 'sysname':
    output = info['sysname']
    print("Il sistema operativo è <%s>" % output)
elif question == 'nodename':
    output = info['nodename']
    print("Il nome del nodo è <%s>" % output)
elif question == 'release':
    output = info['release']
    print("Il nome della release è <%s>" % output)
elif question == 'version':
    output = info['version']
    print("Il nome della versione è <%s>" % output)
elif question == 'machine':
    output = info['machine']
    print("Il nome della macchina è <%s>" % output)
```

Il programma systeminfo.py

```
>>> info
'{"sysname":"Linux", "nodename":"piero-XPS-9320", "release":"6.8.0-58-generic",
"version":"#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28 16:09:21 UTC 2",
"machine":"x86_64"}'
>>> type(info)
<class 'str'>
>>> import json
>>> info = json.loads(info)
```

trasformiamo <info> in un dizionario, ossia in un elenco di elementi ciascuno dei quali è identificato da una chiave

```
>>> type(info)
<class 'dict'>
>>> info
{'sysname': 'Linux', 'nodename': 'piero-XPS-9320', 'release': '6.8.0-58-
generic', 'version': '#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 28
16:09:21 UTC 2', 'machine': 'x86_64'}
>>> info['sysname']
'Linux'
>>>
```

ESERCIZIO:

modificare il programma **systeminfo.py** in modo tale che visualizzi in output **tutte le informazioni sul sistema contemporaneamente**

ESERCIZIO:

```
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$ python3
systeminfo_all.py
Il programma visualizza le seguenti informazioni: <sysname>,
<nodename>, <release>, <version> e <machine>
Il sistema operativo è <Linux>
Il nome del nodo è <piero-XPS-9320>
Il nome della release è <6.8.0-58-generic>
Il nome della versione è <#60~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC
Fri Mar 28 16:09:21 UTC 2>
Il nome della macchina è <x86_64>
piero@piero-XPS-9320:~/COMPUTER_SCIENCE/AI$
```

The range() function: email.py

The **range()** function

```
>>> import os
>>> from tkinter.simpledialog import askstring, askinteger
>>> user = os.getlogin()
>>> INDIRIZZO = askstring('Entry', 'Inserisci il tuo indirizzo e-mail senza il
nome di battesimo')
>>> email = user + INDIRIZZO
>>> NUMERO = askinteger('Entry', 'Scegli un numero intero')
>>> for i in range(NUMERO):
...     print(email)
...
piero.rivoira@yahoo.it
piero.rivoira@yahoo.it
piero.rivoira@yahoo.it
>>>
```

The if statement

```
>>> the_world_is_flat = True
>>> if the_world_is_flat:
...     print("Be careful not to fall off!")
...
Be careful not to fall off!
>>>
```

The if else statement: inserisci_gruppo.py

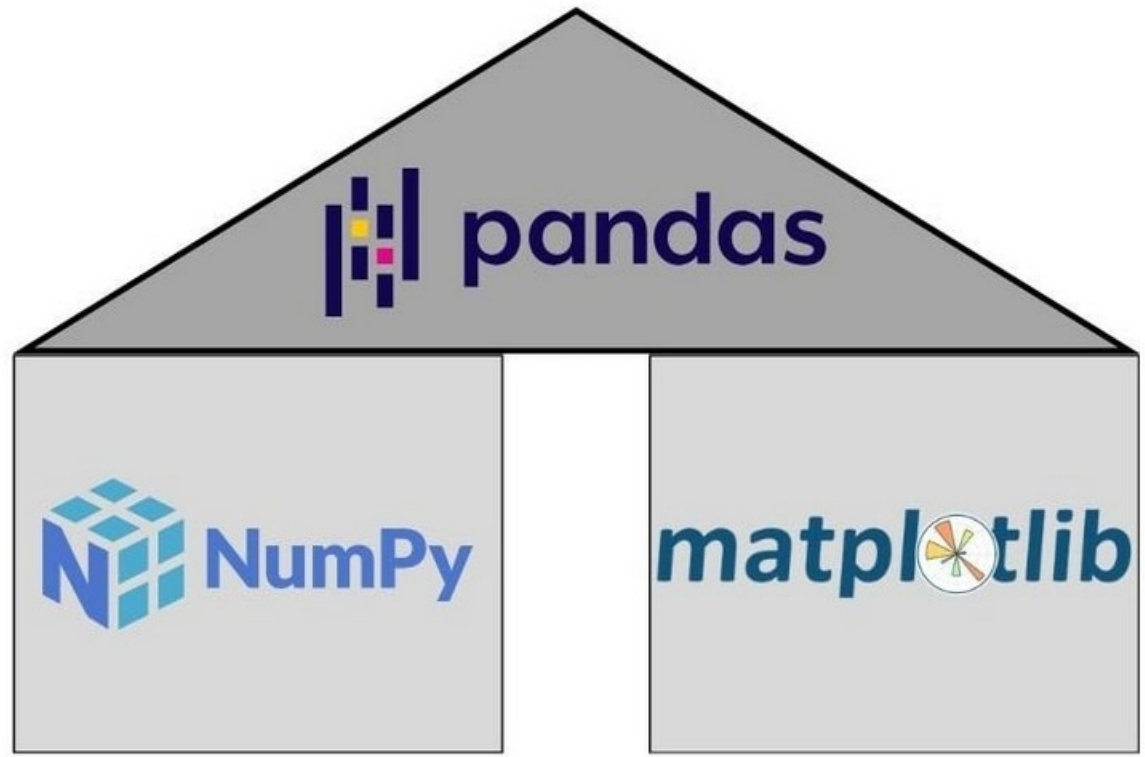
```
from tkinter.simpledialog import askstring
#global gruppo
gruppo = askstring('Entry', 'Scegli il GRUPPO di bovine (<F> per freschissime, <f> per
fresche, <s> per stanche)')
if gruppo == 'F' or gruppo == 'f' or gruppo == 's':
   print('GRUPPO di bovine <%s> inserito con SUCCESSO!' % gruppo)
   print('')
else:
   print('ERRORE: prova a reinserire il gruppo!')
if gruppo == 'F':
   print('La RAZIONE sarà calcolata per il GRUPPO <freschissime>!')
   print('(1^-9^ settimana di lattazione)')
   print('')
   print('<SETTIMANA DI GRAVIDANZA> impostata a 0 (zero)')
elif gruppo == 'f':
   print('La RAZIONE sarà calcolata per il GRUPPO <fresche>!')
   print('(10^-20^ settimana di lattazione)')
   print('')
elif gruppo == 's':
   print('La RAZIONE sarà calcolata per il GRUPPO <stanche>!')
   print('(dalla 21^ settimana di lattazione)')
   print('')
```

Funzioni: trova_massimo.py

```
from tkinter.simpledialog import askinteger, askfloat
DIMENSIONE = askinteger('Entry', 'Inserisci il numero delle osservazioni')
print(DIMENSIONE)
DATI = []
for i in range(DIMENSIONE):
   dato = askfloat('Entry', 'Inserisci i dati uno alla volta')
   print(dato)
   DATI.append(dato)
print(DATI)
def maxarray(lista):
   M = lista[0]
   for i in lista:
      if i > M:
         M = i
   return M
t = maxarray(DATI)
print('Il valore massimo è %f' % t)
```

[28.61, 127.73, 459.26, 35.74, 49.52]
M = 28.61
Per i = uno dei valori della list
Se i > M
M = i, es. M = 127.73

pandas is built on NumPy and Matplotlib



Rectangular data

Name	Breed	Color	Height (cm)	Weight (kg)	Date of Birth
Bella	Labrador	Brown	56	25	2013-07-01
Charlie	Poodle	Black	43	23	2016-09-16
Lucy	Chow Chow	Brown	46	22	2014-08-25
Cooper	Schnauzer	Gray	49	17	2011-12-11
Max	Labrador	Black	59	29	2017-01-20
Stella	Chihuahua	Tan	18	2	2015-04-20
Bernie	St. Bernard	White	77	74	2018-02-27

Leggere dati in Python

```
>>> import os
>>> import pandas as pd
>>> dogs = pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/dogs.csv')
>>> dogs
      Name         Breed  Color  Height (cm)  Weight (kg) Date of Birth
0    Bella      Labrador  Brown           56           25    2013-07-01
1  Charlie        Poodle  Black           43           23    2016-09-16
2     Lucy     Chow Chow  Brown           46           22    2014-08-25
3   Cooper     Schnauzer   Gray           49           17    2011-12-11
4      Max      Labrador  Black           59           29    2017-01-20
5   Stella     Chihuahua    Tan           18            2    2015-04-20
6   Bernie  St. Bernard  White           77           74    2018-02-27
```

Pandas DataFrames

```
>>> print(dogs)
      Name         Breed  Color  Height (cm)  Weight (kg) Date of Birth
0    Bella      Labrador  Brown           56           25    2013-07-01
1  Charlie        Poodle  Black           43           23    2016-09-16
2     Lucy     Chow Chow  Brown           46           22    2014-08-25
3   Cooper     Schnauzer   Gray           49           17    2011-12-11
4      Max      Labrador  Black           59           29    2017-01-20
5   Stella     Chihuahua    Tan           18            2    2015-04-20
6   Bernie  St. Bernard  White           77           74    2018-02-27
>>>
```

Exploring a DataFrame: .head()

```
>>> print(dogs.head())
      Name       Breed  Color  Height (cm)  Weight (kg) Date of Birth
0    Bella    Labrador  Brown           56           25    2013-07-01
1  Charlie      Poodle  Black           43           23    2016-09-16
2     Lucy   Chow Chow  Brown           46           22    2014-08-25
3   Cooper   Schnauzer   Gray           49           17    2011-12-11
4      Max    Labrador  Black           59           29    2017-01-20
>>>
```

Exploring a DataFrame: .info()

```
>>> print(dogs.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Name            7 non-null      object
 1   Breed           7 non-null      object
 2   Color           7 non-null      object
 3   Height (cm)     7 non-null      int64
 4   Weight (kg)     7 non-null      int64
 5   Date of Birth   7 non-null      object
dtypes: int64(2), object(4)
memory usage: 464.0+ bytes
None
>>>
```

Exploring a DataFrame: .shape

```
>>> print(dogs.shape)
(7, 6)
>>>
```

Exploring a DataFrame: .describe()

```
>>> print(dogs.describe())
       Height (cm)  Weight (kg)
count     7.000000     7.000000
mean     49.714286    27.428571
std      17.960274    22.292429
min      18.000000     2.000000
25%      44.500000    19.500000
50%      49.000000    23.000000
75%      57.500000    27.000000
max      77.000000    74.000000
>>>
```

Components of a DataFrame: .values

```
>>> print(dogs.values)
[['Bella' 'Labrador' 'Brown' 56 25 '2013-07-01']
 ['Charlie' 'Poodle' 'Black' 43 23 '2016-09-16']
 ['Lucy' 'Chow Chow' 'Brown' 46 22 '2014-08-25']
 ['Cooper' 'Schnauzer' 'Gray' 49 17 '2011-12-11']
 ['Max' 'Labrador' 'Black' 59 29 '2017-01-20']
 ['Stella' 'Chihuahua' 'Tan' 18 2 '2015-04-20']
 ['Bernie' 'St. Bernard' 'White' 77 74 '2018-02-27']]
>>>
```

Components of a DataFrame: .values

```
>>> dogs_observations = dogs.values
>>> dogs_observations
array([['Bella', 'Labrador', 'Brown', 56, 25, '2013-07-01'],
       ['Charlie', 'Poodle', 'Black', 43, 23, '2016-09-16'],
       ['Lucy', 'Chow Chow', 'Brown', 46, 22, '2014-08-25'],
       ['Cooper', 'Schnauzer', 'Gray', 49, 17, '2011-12-11'],
       ['Max', 'Labrador', 'Black', 59, 29, '2017-01-20'],
       ['Stella', 'Chihuahua', 'Tan', 18, 2, '2015-04-20'],
       ['Bernie', 'St. Bernard', 'White', 77, 74, '2018-02-27']],
      dtype=object)
>>> dogs_observations[0]
array(['Bella', 'Labrador', 'Brown', 56, 25, '2013-07-01'], dtype=object)
>>> dogs_observations[0:3]
array([['Bella', 'Labrador', 'Brown', 56, 25, '2013-07-01'],
       ['Charlie', 'Poodle', 'Black', 43, 23, '2016-09-16'],
       ['Lucy', 'Chow Chow', 'Brown', 46, 22, '2014-08-25']], dtype=object)
>>>
```

Components of a DataFrame: .columns and .index

```
>>> print(dogs.columns)
Index(['Name', 'Breed', 'Color', 'Height (cm)', 'Weight (kg)',
       'Date of Birth'],
      dtype='object')
>>> dogs.index
RangeIndex(start=0, stop=7, step=1)
>>>
```

pandas Philosophy

> There should be one -- and preferably only one -- obvious way to do it.
>
> - *The Zen of Python* by Tim Peters, Item 13

Leggere dati in Python

BRITISH ECOLOGICAL SOCIETY Journal of Animal Ecology

Volume 79, Issue 3
May 2010
Pages 610-619

Free Access

Scaling of basal metabolic rate with body mass and temperature in mammals

Andrew Clarke, Peter Rothery, Nick J. B. Isaac

First published: 22 March 2010 | https://doi.org/10.1111/j.1365-2656.2010.01672.x |
Citations: 173

Summary

Leggere dati in Python

```
>>> import os
>>> import pandas as pd
>>> basal_metabolic_rate =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
>>> basal_metabolic_rate.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            30 non-null     object
 1   species               634 non-null    object
 2   BasalMetabolicRate_W  635 non-null    float64
 3   BodyMass_g            635 non-null    float64
 4   Body_temp             635 non-null    object
 5   Ambient_temp          635 non-null    object
dtypes: float64(2), object(4)
memory usage: 31.4+ KB
>>>
```

Leggere dati in Python

```
>>> print(basal_metabolic_rate.shape)
(667, 6)

>>> basal_metabolic_rate.describe()
       BasalMetabolicRate_W     BodyMass_g
count            635.000000     635.000000
mean            4553.009559    6354.377039
std            21124.257254   31503.606291
min                0.040000       2.400000
25%              143.000000      29.800000
50%              378.000000     106.800000
75%             1354.500000     862.500000
max           286847.000000  407000.000000
>>> basal_metabolic_rate.columns
Index(['Unnamed: 0', 'species', 'BasalMetabolicRate_W', 'BodyMass_g',
       'Body_temp', 'Ambient_temp'],
      dtype='object')
```

Leggere dati in Python

```
>>> print(basal_metabolic_rate.columns)
Index(['Unnamed: 0', 'species', 'BasalMetabolicRate_W', 'BodyMass_g',
       'Body_temp', 'Ambient_temp'],
      dtype='object')
>>>
```

Leggere dati in Python

```
>>> basal_metabolic_rate['species']
0                            NaN
1                            NaN
2                            NaN
3      Tachyglossus aculeatus
4           Zaglossus bruijni
                ...
662     Philantomba monticola
663     Raphicerus campestris
664          Taurotragus oryx
665                          NaN
666                          NaN
Name: species, Length: 667, dtype: object
>>>
```

Leggere dati in Python

```
>>> basal_metabolic_rate['BasalMetabolicRate_W']
0           NaN
1           NaN
2           NaN
3       2404.00
4       6778.00
         ...
662    10075.00
663    20619.00
664      200.83
665         NaN
666      634.00
Name: BasalMetabolicRate_W, Length: 667, dtype: float64
>>> basal_metabolic_rate['BasalMetabolicRate_W'].count()
635
>>>
```

Leggere dati in Python

```
>>> basal_metabolic_rate['BodyMass_g']
0           NaN
1           NaN
2           NaN
3        2725.0
4       10300.0
         ...
662      4200.0
663      9600.0
664    150000.0
665         NaN
666       634.0
Name: BodyMass_g, Length: 667, dtype: float64
>>>
```

Leggere dati in Python

```
>>> import os
>>> import pandas as pd
>>> basal_metabolic_rate =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
>>> basal_metabolic_rate.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            30 non-null     object
 1   species               634 non-null    object
 2   BasalMetabolicRate_W  635 non-null    float64
 3   BodyMass_g            635 non-null    float64
 4   Body_temp             635 non-null    object
 5   Ambient_temp          635 non-null    object
dtypes: float64(2), object(4)
memory usage: 31.4+ KB
>>> BodyMass = basal_metabolic_rate['BodyMass_g']
>>> type(BodyMass)
<class 'pandas.core.series.Series'>
>>> BodyMass_df = BodyMass.to_frame()
>>> type(BodyMass_df)
<class 'pandas.core.frame.DataFrame'>
>>>
```

Leggere dati in Python

```
>>> BodyMass_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 1 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   BodyMass_g  635 non-null    float64
dtypes: float64(1)
memory usage: 5.3 KB
>>> BodyMass_df
     BodyMass_g
0           NaN
1           NaN
2           NaN
3        2725.0
4       10300.0
..          ...
662      4200.0
663      9600.0
664    150000.0
665          NaN
666        634.0

[667 rows x 1 columns]
>>>
```

Leggere dati in Python

```
>>> BodyMass_df = BodyMass_df.rename(columns={'BodyMass_g': 'Body Mass (g)'})
>>> BodyMass_df
     Body Mass (g)
0              NaN
1              NaN
2              NaN
3           2725.0
4          10300.0
..             ...
662         4200.0
663         9600.0
664       150000.0
665            NaN
666          634.0

[667 rows x 1 columns]
>>> BodyMass_df.count()
BodyMass_g    635
dtype: int64
>>>
```

Leggere dati in Python

```
>>> import dill, os, pickle
>>> import pandas as pd
>>> basal_metabolic_rate =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
>>> user = os.getlogin()
>>> os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
>>> with open("metabolic.pkl", "wb") as file:
...     dill.dump(basal_metabolic_rate, file)
...
>>> exit()
```

Leggere dati in Python

```
piero@piero-XPS-9320:~$ python3
>>> import dill, os, pickle
>>> import pandas as pd
>>> user = os.getlogin()
>>> os.chdir('/home/%s/COMPUTER_SCIENCE/AI' % user)
>>> with open("metabolic.pkl", "rb") as file:
...     basal_metabolic_rate = pickle.load(file)
...
>>> basal_metabolic_rate
                    Unnamed: 0  ... Ambient_temp
0                          NaN  ...          NaN
1    PROTOTHERIA: MONOTREMATA   ...          NaN
2          Order: Monotremata   ...          NaN
3                          NaN  ...         19.7
4                          NaN  ...         16.2
..                         ...  ...          ...
662                        NaN  ...            *
663                        NaN  ...         19.1
664                        NaN  ...         21.5
665                        NaN  ...          NaN
666                        NaN  ...          576

[667 rows x 6 columns]
>>>
```

Leggere dati in Python

```
>>> basal_metabolic_rate.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            30 non-null     object
 1   species               634 non-null    object
 2   BasalMetabolicRate_W  635 non-null    float64
 3   BodyMass_g            635 non-null    float64
 4   Body_temp             635 non-null    object
 5   Ambient_temp          635 non-null    object
dtypes: float64(2), object(4)
memory usage: 31.4+ KB
>>>
```

Leggere dati in Python

```
>>> type(basal_metabolic_rate)
<class 'pandas.core.frame.DataFrame'>
>>> BodyMass = basal_metabolic_rate['BodyMass_g']
>>> type(BodyMass)
<class 'pandas.core.series.Series'>
>>> BodyMass_df = BodyMass.to_frame()
>>> type(BodyMass_df)
<class 'pandas.core.frame.DataFrame'>
>>> BodyMass_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 1 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   BodyMass_g   635 non-null    float64
dtypes: float64(1)
memory usage: 5.3 KB
>>>
```

Leggere dati in Python

```
>>> BodyMass_df = BodyMass_df.rename(columns={'BodyMass_g': 'Body Mass (g)'})
>>> BodyMass_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 1 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Body Mass (g)   635 non-null    float64
dtypes: float64(1)
memory usage: 5.3 KB
>>> BodyMass_df.count()
BodyMass_g    635
dtype: int64
```

Leggere dati in Python

```
>>> BodyMass_df
     Body Mass (g)
0              NaN
1              NaN
2              NaN
3           2725.0
4          10300.0
..             ...
662         4200.0
663         9600.0
664       150000.0
665            NaN
666          634.0

[667 rows x 1 columns]
>>>
```

NaN = Not a Number

Leggere dati in Python

```
>>> BodyMass_df_clean = BodyMass_df.dropna(how='all')
>>> BodyMass_df_clean
     Body Mass (g)
3           2725.0
4          10300.0
5            693.0
8            329.0
9            935.0
..             ...
661        69125.0
662         4200.0
663         9600.0
664       150000.0
666          634.0

[635 rows x 1 columns]
>>>
```

Leggere dati in Python

```
>>> basal_metabolic_rate.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            30 non-null     object
 1   species               634 non-null    object
 2   BasalMetabolicRate_W  635 non-null    float64
 3   BodyMass_g            635 non-null    float64
 4   Body_temp             635 non-null    object
 5   Ambient_temp          635 non-null    object
dtypes: float64(2), object(4)
memory usage: 31.4+ KB
>>>
```

Leggere dati in Python

```
>>> BasalMetabolicRate = basal_metabolic_rate['BasalMetabolicRate_W']
>>> BasalMetabolicRate
0            NaN
1            NaN
2            NaN
3         2404.00
4         6778.00
           ...
662      10075.00
663      20619.00
664        200.83
665           NaN
666        634.00
Name: BasalMetabolicRate_W, Length: 667, dtype: float64
>>> BasalMetabolicRate_df = BasalMetabolicRate.to_frame()
```

Oppure:

```
>>> BasalMetabolicRate_df =
basal_metabolic_rate['BasalMetabolicRate_W'].to_frame()
```

Leggere dati in Python

```
>>> BasalMetabolicRate_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 1 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   BasalMetabolicRate_W  635 non-null    float64
dtypes: float64(1)
memory usage: 5.3 KB
>>>
```

Leggere dati in Python

```
>>> BasalMetabolicRate_df =
BasalMetabolicRate_df.rename(columns={'BasalMetabolicRate_W': 'Basal Metabolic
Rate (W)'})
>>> BasalMetabolicRate_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 1 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Basal Metabolic Rate (W)  635 non-null    float64
dtypes: float64(1)
memory usage: 5.3 KB
>>>
```

Oppure:

```
>>> BasalMetabolicRate_df =
basal_metabolic_rate['BasalMetabolicRate_W'].to_frame().rename(columns={'Basal
MetabolicRate_W': 'Basal Metabolic Rate (W)'})
```

Leggere dati in Python

Oppure:

```
>>> import pandas as pd

>>> BasalMetabolicRate_df =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
['BasalMetabolicRate_W'].to_frame().rename(columns={'BasalMetabolicRate_W':
'Basal Metabolic Rate (W)'})
```

Oppure:

```
>>> BasalMetabolicRate_df_clean =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
['BasalMetabolicRate_W'].to_frame().rename(columns={'BasalMetabolicRate_W':
'Basal Metabolic Rate (W)'}).dropna(how='all')
```

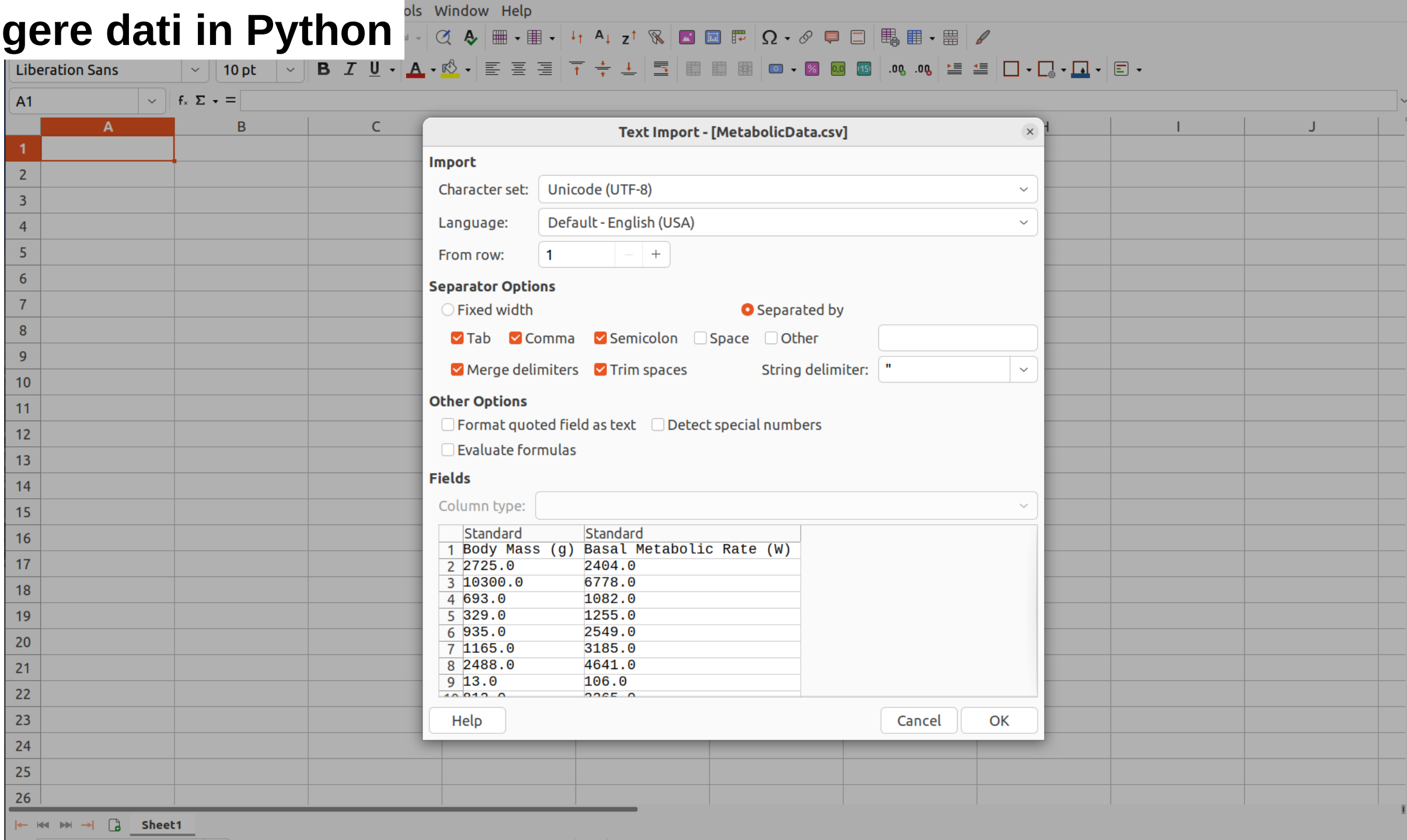
Leggere dati in Python

```
>>> BasalMetabolicRate_df_clean
     Basal Metabolic Rate (W)
3                     2404.00
4                     6778.00
5                     1082.00
8                     1255.00
9                     2549.00
..                        ...
661                 106663.00
662                  10075.00
663                  20619.00
664                    200.83
666                    634.00

[635 rows x 1 columns]
>>>
```

Leggere dati in Python

```
>>> MetabolicData = pd.concat([BodyMass_df_clean, BasalMetabolicRate_df_clean], axis=1)
>>> MetabolicData
     Body Mass (g)  Basal Metabolic Rate (W)
3           2725.0                   2404.00
4          10300.0                   6778.00
5            693.0                   1082.00
8            329.0                   1255.00
9            935.0                   2549.00
..             ...                       ...
661        69125.0                 106663.00
662         4200.0                  10075.00
663         9600.0                  20619.00
664       150000.0                    200.83
666          634.0                    634.00

[635 rows x 2 columns]
>>> os.getcwd()
'/home/piero/COMPUTER_SCIENCE/AI'
>>> os.chdir('DATA')
>>> os.getcwd()
'/home/piero/COMPUTER_SCIENCE/AI/DATA'
>>> MetabolicData.to_csv('MetabolicData.csv', index=False)
>>>
```

Leggere dati in Python



Leggere dati in Python

ols Window Help

Liberation Sans | 10 pt

A26 | = 982

	A	B	C	D	E	F	G	H	I
1	Body Mass (g)	Basal Metabolic Rate (W)							
2	2725	2404							
3	10300	6778							
4	693	1082							
5	329	1255							
6	935	2549							
7	1165	3185							
8	2488	4641							
9	13	106							
10	812	2265							
11	122	547							
12	336	1144							
13	75.5	318							
14	104	335							
15	751	1886							
16	34	119							
17	25.8	141							
18	46.5	252							
19	28.2	189							
20	66.9	351							
21	101	285							
22	118.2	462							
23	1354	3172							
24	558	1356							
25	1782	3281							
26	982	2.21							

MetabolicData

Leggere dati in Python

```
>>> BodyMass_df_clean.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 635 entries, 3 to 666
Data columns (total 1 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Body Mass (g)   635 non-null    float64
dtypes: float64(1)
memory usage: 9.9 KB
>>> BodyMass_df_clean.count()
Body Mass (g)    635
dtype: int64
>>> os.getcwd()
'/home/piero/COMPUTER_SCIENCE/AI'
>>> os.chdir('DATA')
>>> os.getcwd()
'/home/piero/COMPUTER_SCIENCE/AI/DATA'
>>> BodyMass_df_clean.to_csv('BodyMass.csv', index=False)
>>>
```

Leggere dati in Python

```
>>> import dill, os, pickle
>>> import pandas as pd
>>> basal_metabolic_rate =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
>>> BodyMass_df = basal_metabolic_rate['BodyMass_g'].to_frame()
>>> BodyMass_df.count()
BodyMass_g    635
dtype: int64
>>>
>>> BodyMass_df.count().item()
635
```

Leggere dati in Python

```
>>> BodyMass_df_clean.count().item()
635
>>> range(BodyMass_df_clean.count().item())
range(0, 635)
>>> for i in range(BodyMass_df_clean.count().item()):
...     print(i)
...
0
1
2
3
...
632
633
634
>>>
```

Leggere dati in Python

```
>>> import dill, os, pickle
>>> import pandas as pd
>>> basal_metabolic_rate =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
>>> BodyMass_df_clean =
basal_metabolic_rate['BodyMass_g'].to_frame().rename(columns={'BodyMass_g': 'Body Mass
(g)'}).dropna(how='all')
>>> BodyMass_ls_clean = BodyMass_df_clean['Body Mass (g)'].to_list()
>>> type(BodyMass_ls_clean)
<class 'list'>
```

Oppure:

```
>>> BodyMass_ls_clean =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
['BodyMass_g'].to_frame().rename(columns={'BodyMass_g': 'Body Mass
(g)'}).dropna(how='all')['Body Mass (g)'].to_list()
```

Calcolare la media dei valori contenuti nella list <BodyMass_ls_clean>

Calcolare la media dei valori contenuti nella list <BodyMass_ls_clean> Qualche suggerimento

```
>>> len(BodyMass_ls_clean)
635

>>> somma = 0
```
inizializziamo una variabile di tipo int che dovrà contenere la somma dei valori della list

```
>>> range(len(BodyMass_ls_clean))
range(0, 635)
```
usiamo la funzione <**range**> per creare l'elenco dei valori che una variabile indicatrice i dovrà assumere per "scorrere" l'intera list

usiamo un ciclo <**for**> per sommare alla variabile <somma> (il cui valore iniziale è zero) ogni successivo valore della list

<**i**> va da zero a 634

all'interno del ciclo <**for**> inserire un'espressione che, ad ogni reiterazione del ciclo stesso, "aggiorni" il valore della variabile <**somma**>

a questo punto non resterà che dividere <**somma**> per la dimensione del vettore

Calcolare la media dei valori contenuti nella list <BodyMass_ls_clean> Soluzione

```
>>> somma = 0
>>> for i in range(len(BodyMass_ls_clean)):
...     somma = somma + BodyMass_ls_clean[i]
...
>>> somma
4035029.42
>>> media = somma / len(BodyMass_ls_clean)
>>> media
6354.377039370079
>>>
```

Leggere dati in Python

```
>>> BodyMass_series_clean =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
['BodyMass_g'].to_frame().rename(columns={'BodyMass_g': 'Body Mass
(g)'}).dropna(how='all')['Body Mass (g)']

>>> type(BodyMass_series_clean)
<class 'pandas.core.series.Series'>

>>> BodyMass_df_clean = BodyMass_series_clean.to_frame()

>>> type(BodyMass_df_clean)
<class 'pandas.core.frame.DataFrame'>
>>>
```

Leggere dati in Python

```
>>> BodyMass_series_clean =
pd.read_csv('~/COMPUTER_SCIENCE/AI/DATA/metabolismo_basale.csv')
['BodyMass_g'].to_frame().rename(columns={'BodyMass_g': 'Body Mass
(g)'}).dropna(how='all')['Body Mass (g)']
>>> type(BodyMass_series_clean)
<class 'pandas.core.series.Series'>
>>>

>>> BodyMass_df_clean.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 635 entries, 3 to 666
Data columns (total 1 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Body Mass (g)  635 non-null    float64
dtypes: float64(1)
memory usage: 9.9 KB
>>> BodyMass_df_clean['Body Mass (g)'].mean()
6354.377039370079
>>>
```

Il programma statistica.py

```
import dill, os, pickle
import pandas as pd
from tkinter.simpledialog import askstring
user = os.getlogin()
os.chdir('/home/%s/COMPUTER_SCIENCE/AI/DATA' % user)
file_name = askstring('Entry', 'Inserisci il nome del file csv con i dati (senza
estensione)')
#print('Your dataframe will be named <df>')
df = pd.read_csv('%s.csv' % file_name)
print('Questa è la struttura dei tuoi dati:')
print(df.info())
column_name = askstring('Entry', 'Inserisci il nome della colonna che ti interessa')
DATA = df['%s' % column_name]
def maxarray(dataframe):
    M = dataframe[0]
    for i in dataframe:
        if i > M:
            M = i
    return M
MAX = maxarray(DATA)
print('Il valore massimo della variabile <%s> è %f' % (column_name, MAX))
```

Il programma statistica.py

```
def minarray(dataframe):
    m = dataframe[0]
    for i in dataframe:
        if i < m:
            m = i
    return m
MIN = minarray(DATA)
print('Il valore minimo della variabile <%s> è %f' % (column_name, MIN))

DATA_ls = DATA.to_list()
SUM = 0
for i in range(len(DATA_ls)):
    SUM = SUM + DATA_ls[i]
AVERAGE = SUM / len(DATA_ls)
print('La media della variabile <%s> è %f' % (column_name, AVERAGE))
```

Lo shell script statistica

```
#!/bin/bash
cd COMPUTER_SCIENCE/AI
python3 statistica.py
```

```
piero@piero-XPS-9320:~$ chmod +x statistica
```

rendiamo eseguibile il file come programma

```
piero@piero-XPS-9320:~$ ./statistica
piero@piero-XPS-9320:~$ statistica
statistica: comando non trovato
piero@piero-XPS-9320:~$
```

Impostare la variabile ambientale PATH

```
piero@piero-XPS-9320:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
local/games:/snap/bin:/snap/bin
piero@piero-XPS-9320:~$ PATH=$PATH:/home/piero
```

#oppure

```
piero@piero-XPS-9320:~$ export PATH="$PATH:/home/piero"
piero@piero-XPS-9320:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/
local/games:/snap/bin:/snap/bin:/home/piero
```

#ATTENZIONE: il cambiamento vale solo per la sessione corrente!

Impostare la variabile ambientale PATH

```
piero@piero-XPS-9320:~$ statistica
Questa è la struttura dei tuoi dati:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 635 entries, 0 to 634
Data columns (total 2 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Body Mass (g)              635 non-null    float64
 1   Basal Metabolic Rate (W)   635 non-null    float64
dtypes: float64(2)
memory usage: 10.0 KB
None
Il valore massimo della variabile <Body Mass (g)> è 407000.000000
Il valore minimo della variabile <Body Mass (g)> è 2.400000
La media della variabile <Body Mass (g)> è 6354.377039
piero@piero-XPS-9320:~$
```

What are the Shell Environment Variables?

They are...

Il programma statistica.py

```
import dill, os, pickle
import pandas as pd
from tkinter.simpledialog import askstring
```
importiamo i necessari moduli
```
user = os.getlogin()
```
usiamo la funzione `getlogin()` del modulo `os` per ottenere il nome utente e lo salviamo
come stringa user
```
os.chdir('/home/%s/COMPUTER_SCIENCE/AI/DATA' % user)
```
impostiamo come cartella di lavoro quella in cui è stato salvato il file dei dati
usiamo l'operatore `%` per sostituire ad esso la stringa denominata `user`
```
file_name = askstring('Entry', 'Inserisci il nome del file csv con i dati
(senza estensione)')
```
usiamo la funzione `askstring` affinché l'utente possa inserire il nome del file che contiene i dati, che viene salvato nella stringa `file_name`
```
df = pd.read_csv('%s.csv' % file_name)
```
usiamo la funzione del `read_csv` modulo pandas per leggere il file (il cui nome era stato salvato nella stringa `file_name`) e salvarlo come dataframe `df`
usiamo l'operatore `%` per sostituire ad esso la stringa denominata `file_name`

Il programma statistica.py

```
print(df.info())
```

usiamo l'attributo `info()` del dataframe `df` per inviare all'output standard (lo schermo) alcune informazioni sulla struttura del dataframe (numero ed intestazione delle colonne) per l'utente

```
column_name = askstring('Entry', 'Inserisci il nome della colonna che ti interessa')
```

usiamo la funzione `askstring` affinché l'utente possa scegliere la variabile che gli interessa (ogni colonna contiene i valori di una variabile diversa) e salviamo il nome della variabile (intestazione della colonna corrispondente) nella stringa `column_name`

```
DATA = df['%s' % column_name]
```

creiamo il dataframe `DATA` per immagazzinare i valori della variabile scelta

usiamo l'operatore `%` per sostituire ad esso la stringa denominata `column_name`

```
def maxarray(dataframe):
    M = dataframe[0]
    for i in dataframe:
        if i > M:
            M = i
    return M
```

creiamo la funzione `maxarray` che ha come argomento il dataframe `dataframe`

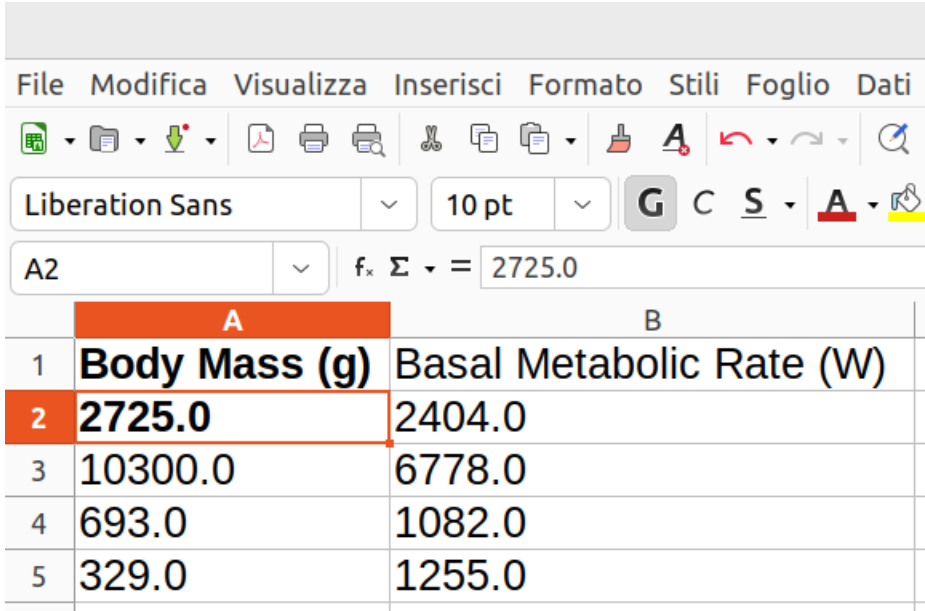
Il programma statistica.py

```
def maxarray(dataframe):
    M = dataframe[0]
    for i in dataframe:
        if i > M:
            M = i
    return M
```

creiamo la funzione `maxarray` che ha come argomento il dataframe `dataframe`

```
    M = dataframe[0]
```

inizializziamo la variabile M facendole assumere il primo valore della variabile prescelta

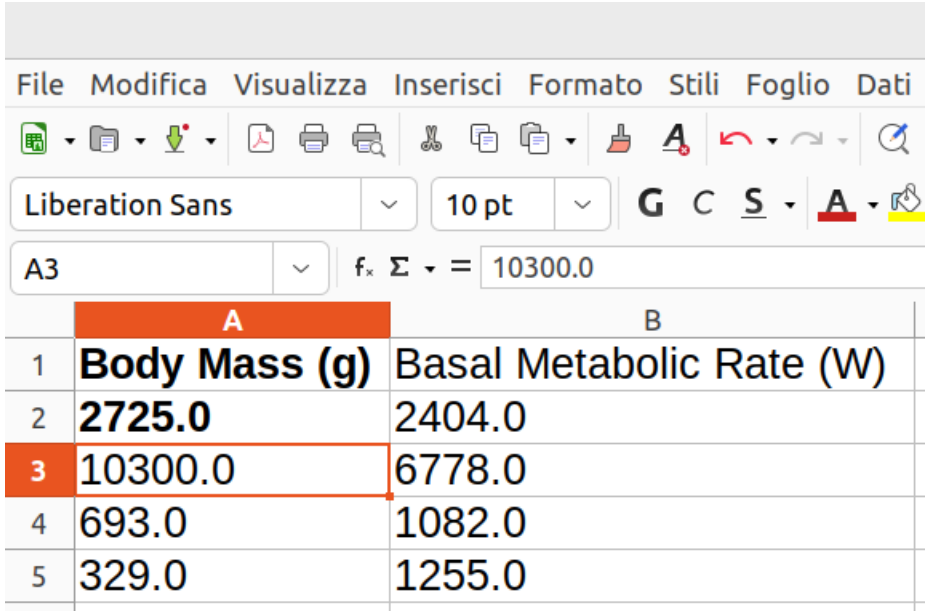


	A	B
1	**Body Mass (g)**	Basal Metabolic Rate (W)
2	**2725.0**	2404.0
3	10300.0	6778.0
4	693.0	1082.0
5	329.0	1255.0

Il programma statistica.py

```
def maxarray(dataframe):
    M = dataframe[0]
    for i in dataframe:
        if i > M:
            M = i
    return M
```

usiamo il ciclo `for` e la variabile indicatrice `i`
ad ogni ripetizione del ciclo, i assume un valore del
dataframe
all'inizio i = M = 2725.0
nel ciclo successivo i = 10300.0, quindi i > M
M viene posta uguale ad i, ossia M = i = 10300.0
quindi il ciclo ricomincia: i diventa uguale a 693.0
poiché i < M, M rimane uguale a 10300.0
e così via



	A	B
1	**Body Mass (g)**	Basal Metabolic Rate (W)
2	**2725.0**	2404.0
3	10300.0	6778.0
4	693.0	1082.0
5	329.0	1255.0

Il programma statistica.py

```
MAX = maxarray(DATA)
```
passiamo alla funzione `maxarray` l'argomento `DATA` (il dataframe che contiene i dati)
e salviamo il risultato della stessa funzione nell'oggetto `MAX`

```
print('Il valore massimo della variabile <%s> è %f' % (column_name, MAX))
```

usiamo gli operatori di formattazione `%s` e `%f` per sostituirli, rispettivamente, con i valori della stringa `column_name` (che contiene il nome della variabile) e dell'oggetto di tipo float (numero a virgola mobile) `MAX`, che rappresenta il risultato della funzione `maxarray`

```
def minarray(dataframe):
    m = dataframe[0]
    for i in dataframe:
        if i < m:
            m = i
    return m
MIN = minarray(DATA)
print('Il valore minimo della variabile <%s> è %f' % (column_name, MIN))
```
stessa cosa per calcolare il minimo

Il programma statistica.py

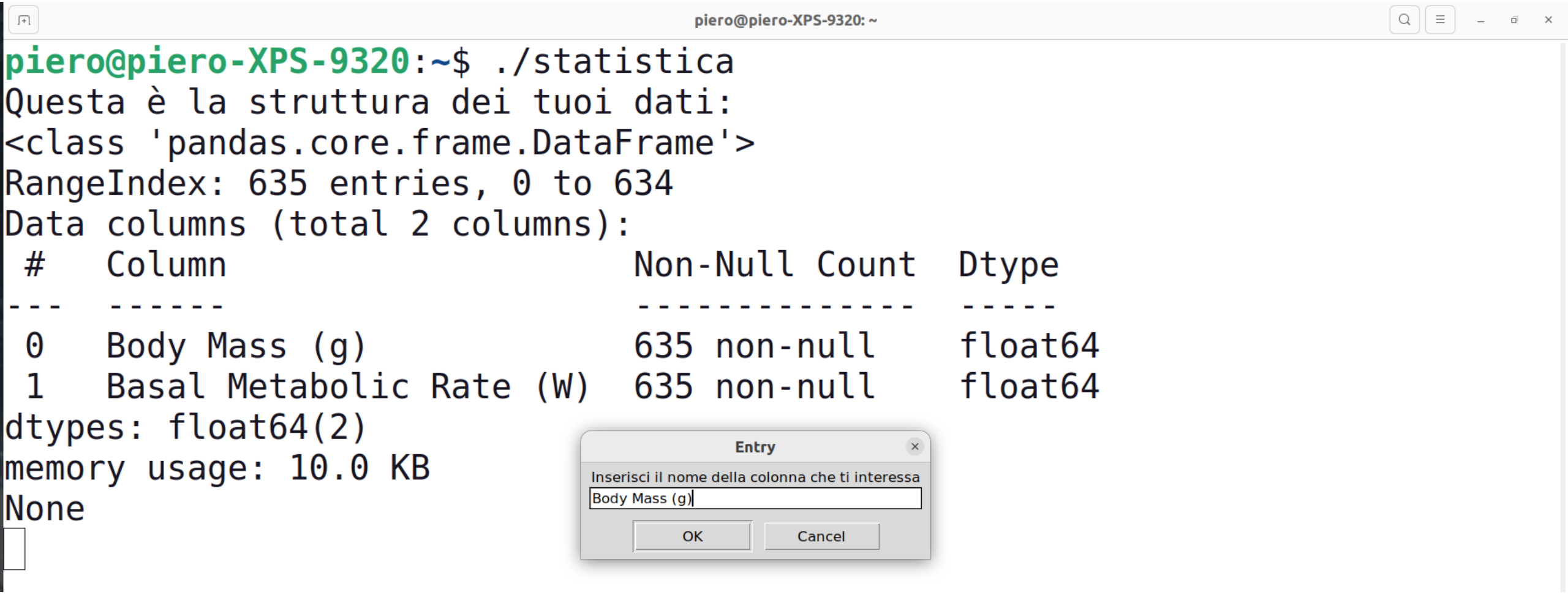
```
DATA_ls = DATA.to_list()
```
il dataframe `DATA` viene convertito nella list `DATA_ls`
```
SUM = 0
```
la variabile `SUM` viene inizializzata a zero
```
for i in range(len(DATA_ls)):
```
per i che va da 0 a 634:
```
    SUM = SUM + DATA_ls[i]
```
il valore di `SUM` viene aggiornato:
```
AVERAGE = SUM / len(DATA_ls)
```
si applica la formula della media
```
print('La media della variabile <%s> è %f' % (column_name, AVERAGE))
```
usiamo gli operatori di formattazione `%s` e `%f` per sostituirli, rispettivamente, con i valori della stringa `column_name` (che contiene il nome della variabile) e dell'oggetto di tipo float (numero a virgola mobile) `AVERAGE`, che rappresenta la media

Lo shell script <statistica>

```
#!/bin/bash
cd COMPUTER_SCIENCE/AI
python3 statistica.py
```

Il programma statistica.py

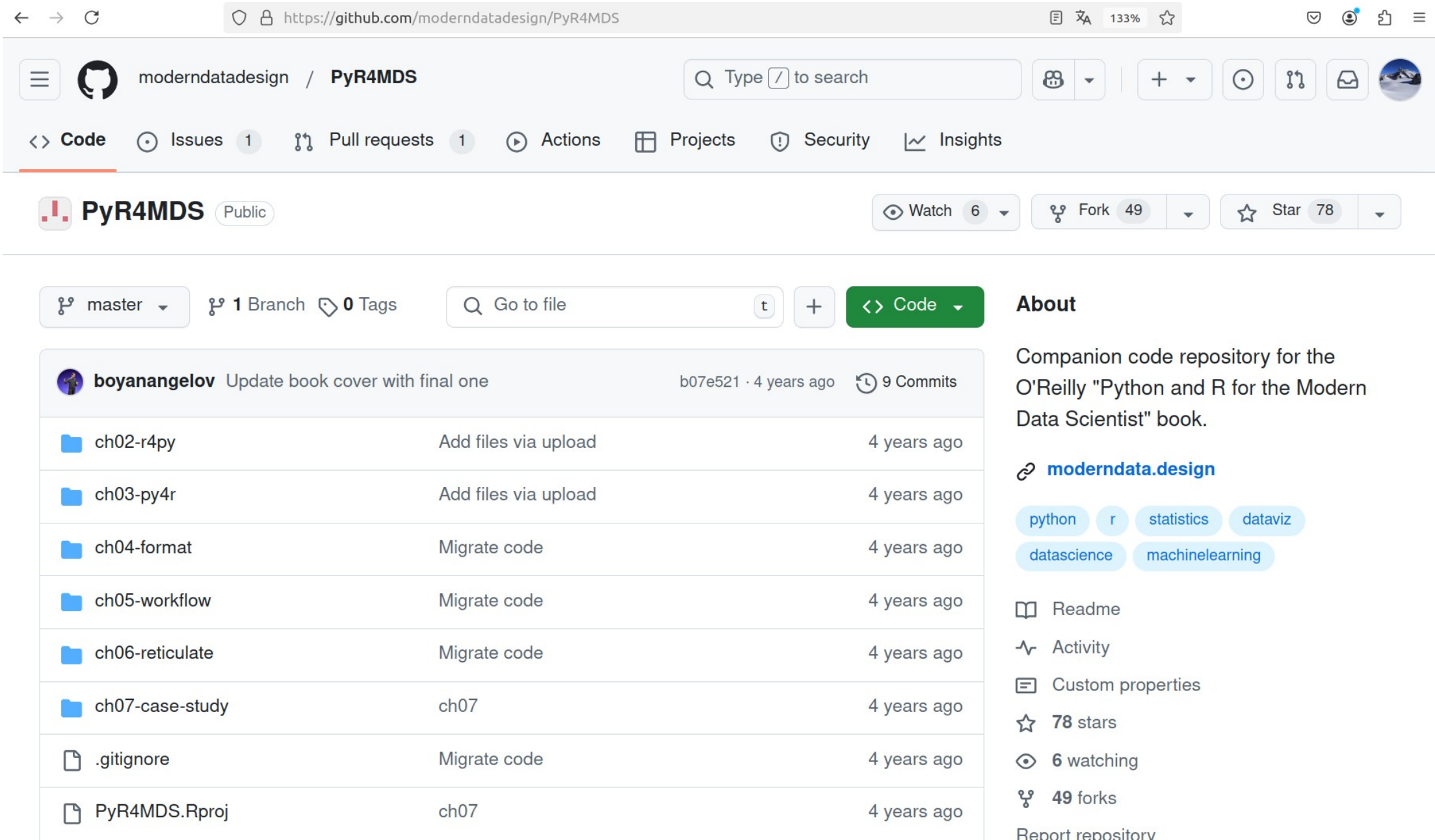
Il programma statistica.py



Il programma statistica.py

```
piero@piero-XPS-9320:~$ ./statistica
Questa è la struttura dei tuoi dati:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 635 entries, 0 to 634
Data columns (total 2 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Body Mass (g)              635 non-null    float64
 1   Basal Metabolic Rate (W)   635 non-null    float64
dtypes: float64(2)
memory usage: 10.0 KB
None
Il valore massimo della variabile <Body Mass (g)> è 407000.000000
Il valore minimo della variabile <Body Mass (g)> è 2.400000
La media della variabile <Body Mass (g)> è 6354.377039
piero@piero-XPS-9320:~$
```

Python and R (https://github.com/moderndatadesign/PyR4MDS)



Python and R: prepariamo i dati

```
piero@piero-XPS-9320:~$ R

R version 4.4.2 (2024-10-31) -- "Pile of Leaves"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R è un software libero ed è rilasciato SENZA ALCUNA GARANZIA.
Siamo ben lieti se potrai redistribuirlo, ma sotto certe condizioni.
Scrivi 'license()' o 'licence()' per maggiori dettagli.

R è un progetto collaborativo con molti contributi esterni.
Scrivi 'contributors()' per maggiori informazioni e 'citation()'
per sapere come citare R o i pacchetti nelle pubblicazioni.

Scrivi 'demo()' per una dimostrazione, 'help()' per la guida
oppure 'help.start()' per la guida nel browser HTML.
Scrivi 'q()' per uscire da R.

[Caricato workspace precedentemente salvato]

>
```

Python and R: prepariamo i dati

```
> library(readxl)
> BasalMetabolicRate = read_excel('~/Scaricati/metabolismo_basale.xlsx')
> str(BasalMetabolicRate)
tibble [667 × 6] (S3: tbl_df/tbl/data.frame)
 $ ...1                : chr [1:667] NA "PROTOTHERIA: MONOTREMATA" "Order: Monotremata" I
 $ species             : chr [1:667] NA NA NA "Tachyglossus aculeatus" ...
 $ BasalMetabolicRate_W: num [1:667] NA NA NA 2.4 6.78 ...
 $ BodyMass_g          : num [1:667] NA NA NA 2725 10300 ...
 $ Body_temp           : chr [1:667] NA NA NA "30.7" ...
 $ Ambient_temp        : chr [1:667] NA NA NA "19.7" ...
> tasso_metabolico_basale = BasalMetabolicRate$BasalMetabolicRate_W[!
is.na(BasalMetabolicRate$BasalMetabolicRate_W)]
> str(tasso_metabolico_basale)
 num [1:635] 2.4 6.78 1.08 1.25 2.55 ...
> massa_corporea = BasalMetabolicRate$BodyMass_g[!is.na(BasalMetabolicRate$BodyMass_g)]
> str(massa_corporea)
 num [1:635] 2725 10300 693 329 935 ...
> df = data.frame(massa_corporea, tasso_metabolico_basale)
> str(df)
'data.frame':635 obs. of  2 variables:
 $ massa_corporea        : num  2725 10300 693 329 935 ...
 $ tasso_metabolico_basale: num  2.4 6.78 1.08 1.25 2.55 ...
```

Python and R: prepariamo i dati

```
> max(massa_corporea)
[1] 407000
> max(tasso_metabolico_basale)
[1] 634
> write.csv(df, "~/COMPUTER_SCIENCE/AI/DATA/data.csv", row.names=FALSE)
>
```