# Corso Pratico Python

6. Accesso a File
6.1. StdIO + Files
6.1.1. CSV, TSV
6.2. Tipi Strutturati
6.2.1. HDF, ~~NC~~, FITS

Richiedente: Dott. Roberto Felici [CNR-ISM] • Insegnante: Dott. Scigè J. Liu [INAF-IAPS]

**StdIO (Standard Input/Output):**
operazioni di base come
•print() per l'output
•input() per l'input da tastiera.
In contesti avanzati, si interagisce direttamente
con sys.stdin e sys.stdout.

**File Handling:**
funzione integrata open() è il punto di partenza. Il pattern consigliato è l'uso del **context manager with**, che garantisce la chiusura del file anche in caso di errori

```
with open('esempio.txt', 'r') as file:
    contenuto = file.read()
```

**Parametri Utili di `read_csv`**

Spesso i file reali non sono "puliti". Ecco i parametri più comuni per gestire situazioni diverse:

- **`sep`**: Se il file usa il punto e virgola `;` invece della virgola.
  - `df = pd.read_csv('file.csv', sep=';')`
- **`usecols`**: Se vuoi caricare solo alcune colonne (ottimo per risparmiare memoria).
  - `df = pd.read_csv('file.csv', usecols=['Prodotto', 'Prezzo'])`
- **`parse_dates`**: Per convertire automaticamente le colonne temporali in oggetti Python `datetime`.
  - `df = pd.read_csv('file.csv', parse_dates=['Data'])`
- **`nrows`**: Per leggere solo le prime *n* righe di un file gigantesco.
  - `df = pd.read_csv('file.csv', nrows=100)`

```
ID,Prodotto,Prezzo,Quantita,Data
1,Laptop,1200.50,5,2023-10-01
2,Mouse,25.00,15,2023-10-02
3,Monitor,350.00,7,2023-10-02
```

```python
import pandas as pd

# Caricamento del file
df = pd.read_csv('dati_vendite.csv')

# Visualizza le prime righe
print(df.head())
```

# 6. Accesso a File
# 6.1. StdIO + Files → 6.1.1. CSV, TSV [+pandas]

**Parametri Utili di `read_csv`**

Spesso i file reali non sono "puliti". Ecco i parametri più comuni per gestire situazioni diverse:

- **`sep`**: Se il file usa il punto e virgola `;` invece della virgola.
  - `df = pd.read_csv('file.csv', sep=';')`
- **`usecols`**: Se vuoi caricare solo alcune colonne (ottimo per risparmiare memoria).
  - `df = pd.read_csv('file.csv', usecols=['Prodotto', ...])`
- **`parse_dates`**: Per convertire a... temporali in oggetti Python `date`...
  - `df = pd.read_csv(... parse_dates=['Data'])`
- **`nrows`**: Per leggere solo le prime...
  - `df = pd.read_csv(... nrows=100)`

```
ID,Prodotto,Prezzo,Quantita,Data
1,Laptop,1200.50,5,2023-10-01
2,Mouse,25.00,15,2023-10-02
3,Monitor,350.00,7,2023-10-02
```

```
import pandas as pd
```

```
-------
#csv ID,Prodotto,Prezzo,Quantita,Data
#csv 1,Laptop,1200.50,5,2023-10-01
#csv 2,Mouse,25.00,15,2023-10-02
#csv 3,Monitor,350.00,7,2023-10-02
--------
#tsv ID         Prodotto    Prezzo      Quantita    Data         Fatturato
#tsv 1          Laptop      1200.5      5           2023-10-01   6002.5
#tsv 2          Mouse       25.0        15          2023-10-02   375.0
#tsv 3          Monitor     350.0       7           2023-10-02   2450.0
--------
```

# 6. Accesso a File
## 6.1. StdIO + Files → 6.1.1. CSV, TSV [+pandas]

```
df['Fatturato'] = df['Prezzo'] * df['Quantita']

# Calcoliamo il prezzo medio per prodotto
prezzo_medio = df['Prezzo'].mean()

print("-"*8)
print(f"Prezzo medio: {prezzo_medio}€")
print(df)
print("-"*8)
```

```
#%%
import pandas as pd

# 1. Otteniamo la lista degli attributi
fx = dir(pd)

# 2. Correzione della List Comprehension
fx_read = [ff for ff in fx if ff.startswith("read")]

print(fx_read)
"""
['read_clipboard', 'read_csv', 'read_excel', 'read_feather', 'read_fwf',
 'read_hdf', 'read_html', 'read_iceberg', 'read_json', 'read_orc',
 'read_parquet', 'read_pickle', 'read_sas', 'read_spss', 'read_sql',
 'read_sql_query', 'read_sql_table', 'read_stata', 'read_table', 'read_xml']
"""
```

# 6. Accesso a File
## 6.1. StdIO + Files → 6.1.1. CSV, TSV [+pandas]

```
df['Fatturato'] = df['Prezzo'] * df['Quantita']

# Calcoliamo il prezzo medio per prodotto
prezzo_medio = df['Prezzo'].mean()

print("-"*8)
print(f"Prezzo medio: {prezzo_medio}€")
print(df)
print("-"*8)
```

```
42  #%%
43  import pandas as pd
44
45  # 1. Otteniamo la lista degli attributi
46  fx = dir(pd)
47
48  # 2. Correzione della List Comprehension
49  fx_read = [ff for ff in fx if ff.startswith("read")]
50
51  print(fx_read)
52  """
53  ['read_clipboard', 'read_csv', 'read_excel', 'read_feather', 'read_fwf',
54   'read_hdf', 'read_html', 'read_iceberg', 'read_json', 'read_orc',
55   'read_parquet', 'read_pickle', 'read_sas', 'read_spss', 'read_sql',
56   'read_sql_query', 'read_sql_table', 'read_stata', 'read_table', 'read_xml']
57  """
```

# 6. Accesso a File
## 6.2. Tipi Strutturati

Quando i dati diventano troppo grandi o complessi per un semplice file di testo, passiamo a formati binari strutturati. Questi formati permettono di memorizzare metadati (descrizioni dei dati) e supportano l'accesso casuale (leggere solo una parte del file senza caricarlo tutto).

- **6.2.1. HDF, NC, FITS**
- Questi formati sono i pilastri del calcolo scientifico e dei Big Data.

| Formato | Nome Esteso | Settore d'uso Principale | Libreria Python |
|---|---|---|---|
| **HDF5** | Hierarchical Data Format | Machine Learning, Dati Gerarchici | h5py, PyTables |
| **NC** | NetCDF (Network Common Data Form) | Meteorologia, Oceanografia | netCDF4, xarray |
| **FITS** | Flexible Image Transport System | Astrofisica, Imaging Astronomico | astropy |

# 6.2. Tipi Strutturati - METADATI

L'importanza dei metadati nei tipi strutturati come HDF5, NetCDF o FITS è ciò che trasforma una semplice sequenza di numeri in informazione scientifica.

Senza metadati, un array di dati è solo una griglia anonima; con i metadati, diventa "la temperatura superficiale del mare misurata dal satellite X il giorno Y".

Nei formati strutturati, i metadati sono memorizzati nell'**Header** (intestazione) o come **Attributi**.
La loro importanza risiede in tre pilastri:

- **Auto-descrizione:** Il file "spiega" se stesso.
- Contiene unità di misura (m/s, K, Pa), timestamp e coordinate spaziali.
- **Provenienza (Lineage):** Registra chi ha generato il dato, con quale software e quali parametri di elaborazione sono stati usati.
- **Accesso Selettivo:** Permettono al software di sapere esattamente dove si trova un dato nel disco senza dover leggere l'intero file (cruciale per file da decine di Gigabyte).

# 6. Accesso a File
## 6.2. Tipi Strutturati - METADATI

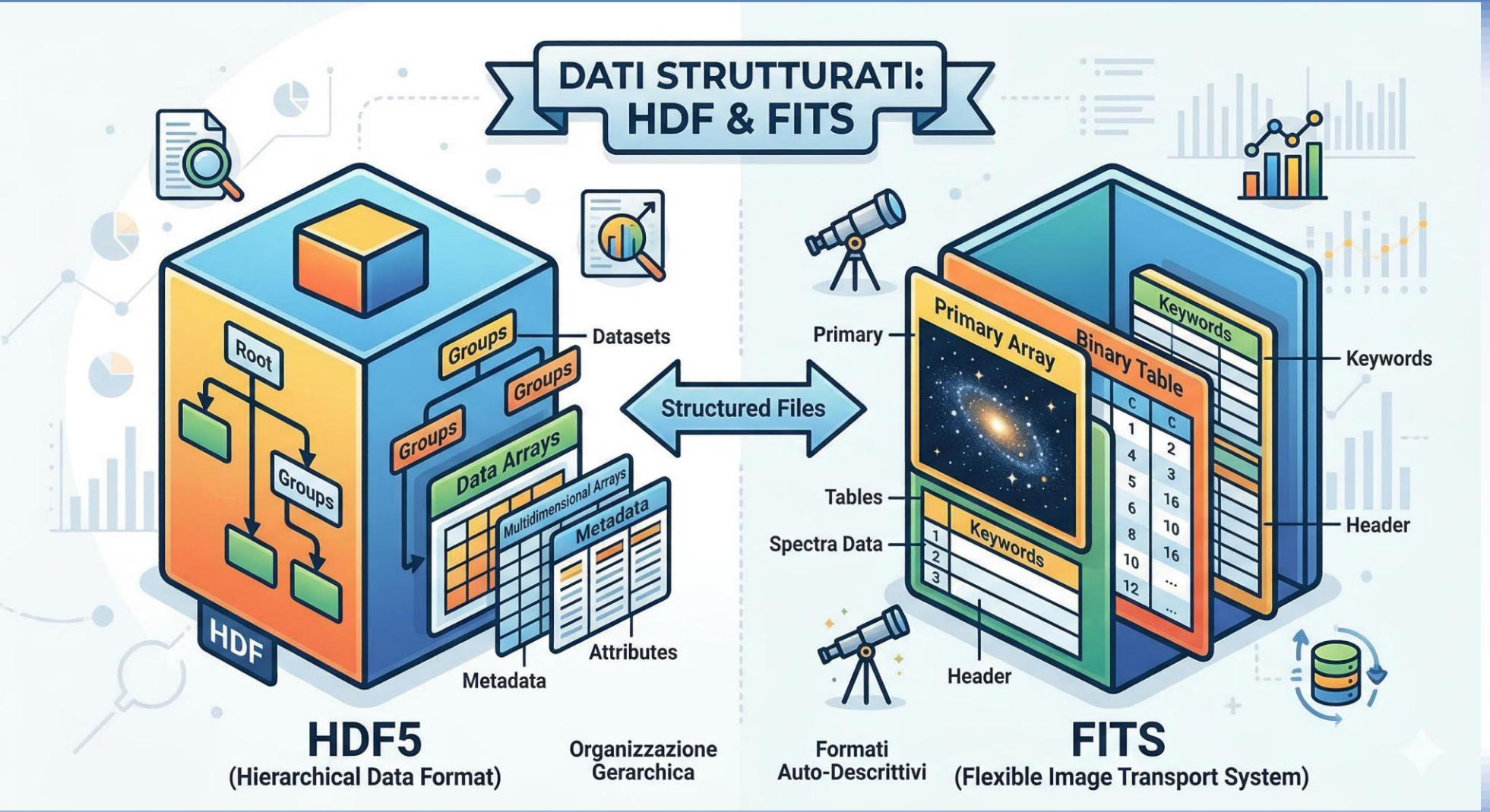
L'importanza dei metadati nei tipi strutturati come HDF5, NetCDF o FITS è ciò che trasforma una semplice sequenza di numeri in informazione scientifica.

Senza metadati, un array di dati è solo una griglia anonima; con i metadati, diventa "la temperatura superficiale del mare misurata dal satellite X il giorno Y".

Nei formati strutturati, i metadati sono memorizzati nell'**Header** (intestazione) o come **Attributi**.
La loro importanza risiede in tre pilastri:

- **Auto-descrizione:** Il file "spiega" se stesso.
- Contiene unità di misura (m/s, K, Pa), timestamp e coordinate spaziali.
- **Provenienza (Lineage):** Registra chi ha generato il dato, con quale software e quali parametri di elaborazione sono stati usati.
- **Accesso Selettivo:** Permettono al software di sapere esattamente dove si trova un dato nel disco senza dover leggere l'intero file (cruciale per file da decine di Gigabyte).

# 6. Accesso a File
## 6.2. Tipi Strutturati → 6.2.1. HDF, ~~NC~~, FITS



# 6. Accesso a File
# 6.2. Tipi Strutturati → 6.2.1. HDF, ~~NC~~, FITS

```python
import h5py

with h5py.File('dati_scientifici.h5', 'r') as f:
    # Esplora i metadati del file globale
    print("Attributi Globali:", list(f.attrs.items()))

    # Accedi a un dataset specifico
    dataset = f['/esperimento1/sensore_A']
    print("Unità di misura:", dataset.attrs['units'])
```

```python
# Apriamo il file in modalità scrittura ('w').
# L'uso di 'with' garantisce la chiusura corretta del file.
with h5py.File(nome_file, 'w') as f:

    # --- 1. METADATI GLOBALI (Attributi del file Root '/') ---
    f.attrs['descrizione'] = 'Esempio di dati scientifici strutturati'
    f.attrs['data_creazione'] = datetime.now().isoformat()
    f.attrs['autore'] = 'Corso Pratico Python'

    # --- 2. CREAZIONE DI GRUPPI (come cartelle) ---
    # Creiamo un gruppo per un ipotetico esperimento
    gruppo_exp1 = f.create_group('esperimento_alfa')

    # Aggiungiamo metadati al gruppo
    gruppo_exp1.attrs['luogo'] = 'Laboratorio Centrale'
    gruppo_exp1.attrs['responsabile'] = 'Dr. Rossi'
```

```
[Gruppo] esperimento_alfa/meteo_ausiliario

  [Dataset] esperimento_alfa/meteo_ausiliario/umidita_relativa
    -> Metadati:
       - unita: Percentuale
    -> Shape: (4,), Tipo: int64
    -> Dati: [60 62 65 58]

 [Dataset] esperimento_alfa/temperature
   -> Metadati:
      - descrizione: Letture orarie da 3 sensori diversi
      - frequenza_campionamento: 1 ora
      - unita: Gradi Celsius
   -> Shape: (4, 3), Tipo: float32
   -> Dati: [[22.5 22.7 22.4]
[23.  23.2 22.9]
[23.5 23.8 23.6]
[22.8 23.  22.7]]
```

# 6. Accesso a File
# 6.2. Tipi Strutturati → 6.2.1. HDF, ~~NC~~, FITS

G:\Drive condivisi\Gruppo-ScigeMarty\2026-99-99--...ezioni AF2026\Esercitazione1\Esercitazione_063.py

Esercitazione_060.py | Esercitazione_061.py | Esercitazione_062.py | Esercitazione_063.py

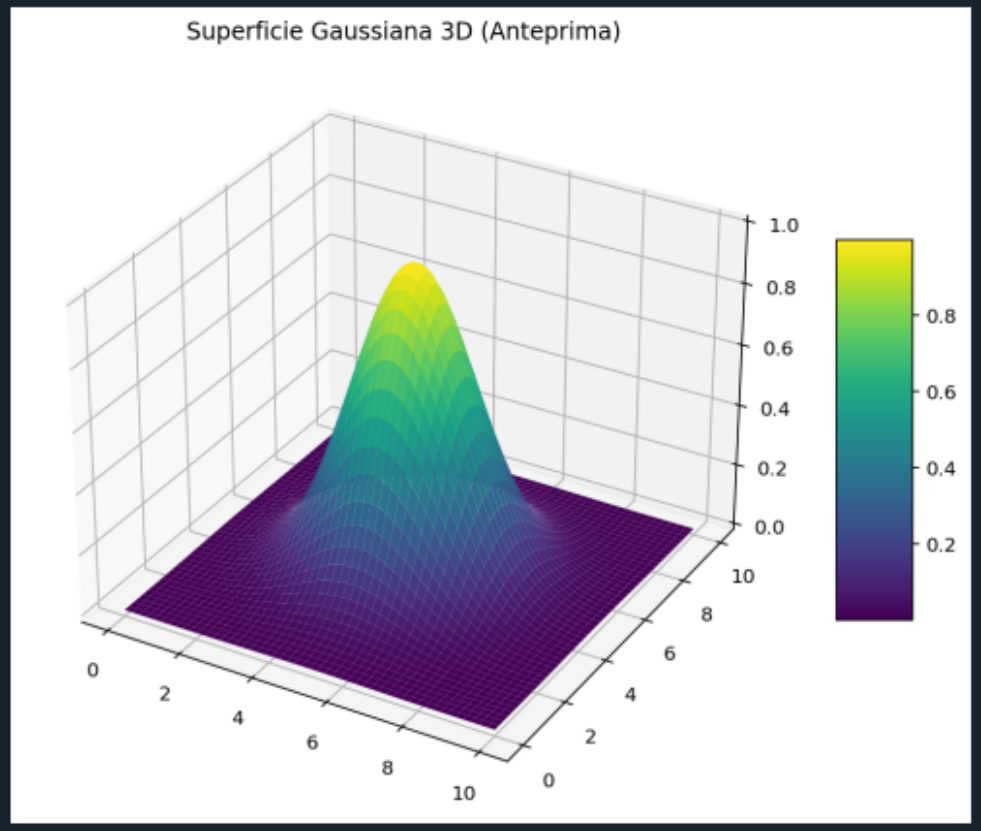
```python
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 19 00:44:41 2026

@author: scige
"""
#%%
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
from mpl_toolkits.mplot3d import Axes3D

# --- GENERAZIONE GAUSSIANA ---
N = 50  # Risoluzione griglia
x, y = np.meshgrid(np.linspace(0, 10, N), np.linspace(0, 10, N))

# Centrata in x=5, y=5 con deviazione standard sigma=1.5
x0, y0, sigma = 5, 5, 1.5
z = np.exp(-((x - x0)**2 + (y - y0)**2) / (2 * sigma**2))

# --- PLOT 3D PRIMA DEL SALVATAGGIO ---
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(x, y, z, cmap='viridis', edgecolor='none')
ax.set_title("Superficie Gaussiana 3D (Anteprima)")
plt.colorbar(surf, ax=ax, shrink=0.5, aspect=5)
plt.show()
#%%
```



# 6. Accesso a File
# 6.2. Tipi Strutturati → 6.2.1. HDF, ~~NC~~, FITS

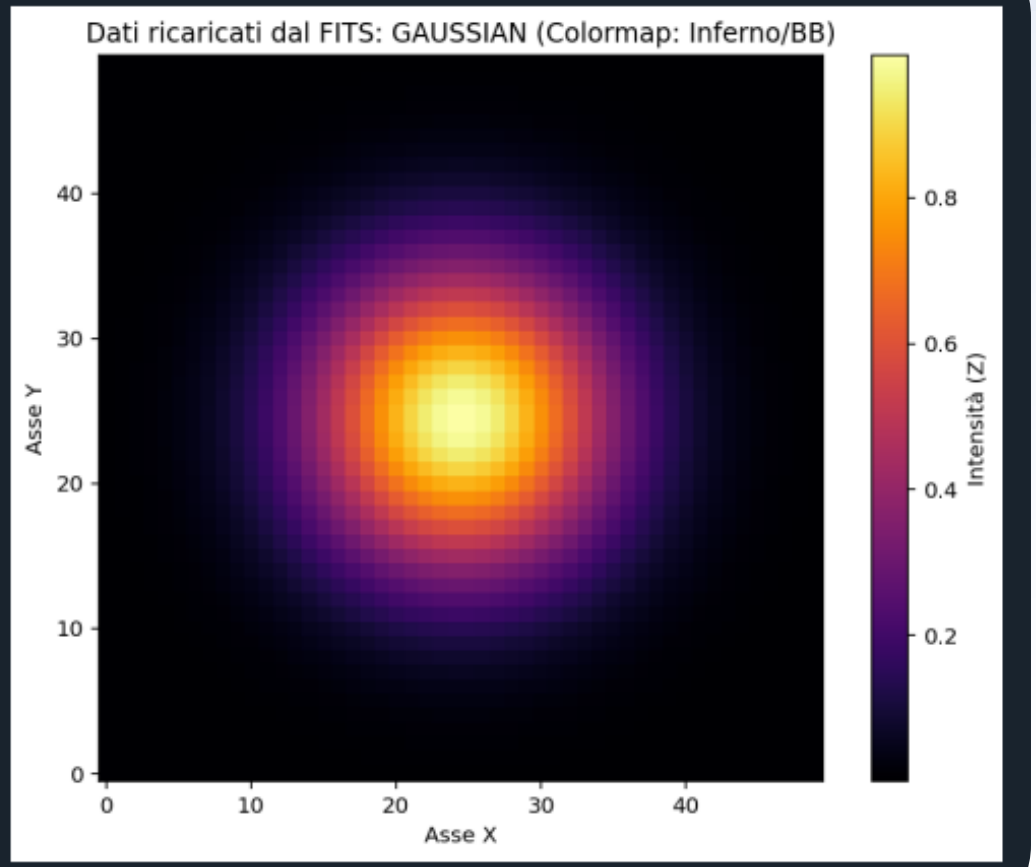
Esercitazione_060.py | Esercitazione_061.py | Esercitazione_062.py | Esercitazione_063.py*

```
hdu.writeto(nome_file, overwrite=True)
print(f"File {nome_file} salvato correttamente.")
#%%

# --- RICARICAMENTO ---
with fits.open(nome_file) as hdul:
    data_fits = hdul[0].data
    header_fits = hdul[0].header

# --- PLOT 2D IN SCALA DI COLORI (BB-like) ---
plt.figure(figsize=(8, 6))
# 'inferno' è la scala che meglio approssima il calore di un corpo nero
img = plt.imshow(data_fits, cmap='inferno', origin='lower')
plt.colorbar(img, label='Intensità (Z)')

# Recupero metadati per il titolo
obj_name = header_fits['OBJECT']
plt.title(f"Dati ricaricati dal FITS: {obj_name} (Colormap: Inferno/BB)")
plt.xlabel("Asse X")
plt.ylabel("Asse Y")
plt.show()
#%%
```



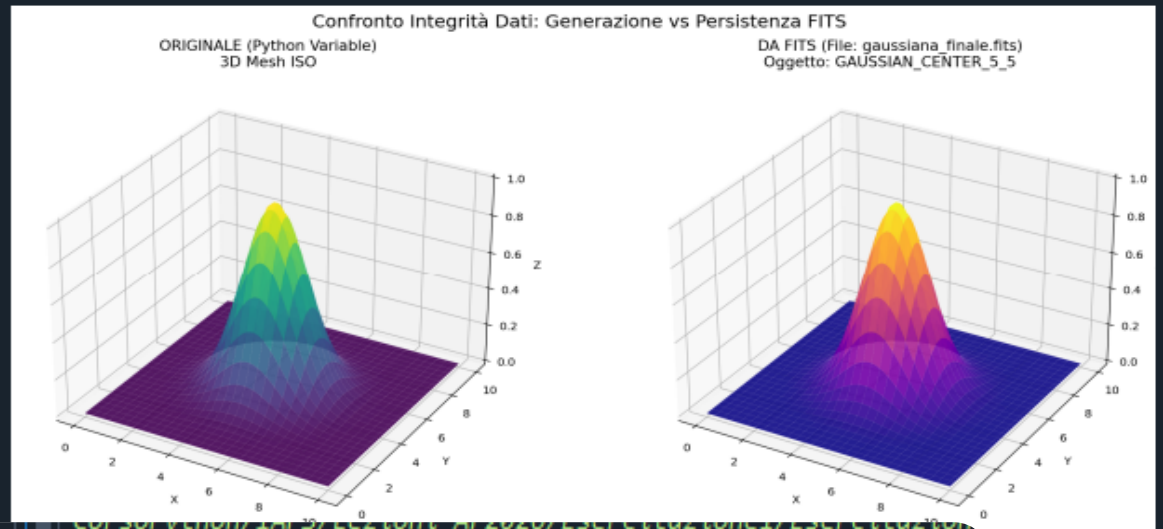
# 6. Accesso a File
# 6.2. Tipi Strutturati → 6.2.1. HDF, ~~NC~~, FITS

```python
fig = plt.figure(figsize=(15, 7))

# Sottoplot SX: Dato Originario
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
surf1 = ax1.plot_surface(X, Y, Z_orig, cmap='viridis', edgecolor='none', alpha=0.9)
ax1.set_title('ORIGINALE (Python Variable)\n3D Mesh ISO', fontsize=13)
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')

# Sottoplot DX: Dato caricato dal FITS
ax2 = fig.add_subplot(1, 2, 2, projection='3d')
# Usiamo una colormap diversa (es. plasma) per distinguerli visivamente
surf2 = ax2.plot_surface(X, Y, Z_fits, cmap='plasma', edgecolor='none', alpha=0.9)
ax2.set_title(f'DA FITS (File: {filename})\nOggetto: {info_obj}', fontsize=13)
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')

plt.suptitle("Confronto Integrità Dati: Generazione vs Persistenza FITS", fontsize=16
plt.tight_layout()
plt.show()
```




```python
#%%
from astropy.io import fits

# Apriamo il file creato in precedenza
with fits.open('gaussiana_finale.fits') as hdul:
    header = hdul[0].header

    print("--- Contenuto dell'Header FITS ---")
    print(header)
```

```
--- Contenuto dell'Header FITS ---
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                  -64 / array data type
NAXIS   =                    2 / number of array dimensions
NAXIS1  =                   60
NAXIS2  =                   60
EXTEND  =                    T
OBJECT  = 'GAUSSIAN_CENTER_5_5'
SIGMA   =                  1.2

In [67]:
```

# 6. Accesso a File – packages utili per gli esercizi

```
"""
python.exe -m pip install --upgrade pip
pip install spyder
spyder

pip install matplotlib pandas scipy numpy
pip install numba pyopencl
pip install h5py astropy numpy
"""
```