

```
In [1]: import pandas as pd
import numpy as np
```

Importare dati da un file .csv

```
In [2]: clienti = pd.read_csv(filepath_or_buffer = r"C:\Users\ianto\Desktop\Corso python
    sep = ";", #separtore del file, la virgola è il default
    header = 0, #per indicare quale riga contiene l'intestazione
    #0 vuol dire che l'intestazione è nella prima riga
    #se non c'è intestazione utilizzare None
    #names = ["NumeroCliente", "IsActive", "Nome", "Cognome", "DataNascita"]
    # se non avessi l'intestazione assegnerei i nomi con l'attributo names
    usecols = ["NumeroCliente", "Nome", "DataNascita", "Regione"], #colonne
    #dtype = {"NumeroCliente": np.str}, #tipi delle colonne
    #parse_dates=["DataNascita"], #colonne da importare come date
    #dayfirst=True, #MOLTO IMPORTANTE, per indicare che nelle date il giorno
    #decimal=".", #separatore dei decimali, il default è il punto
    #index_col = "NumeroCliente", #impostare una colonna come indice
    #nrows = 10 #numero di righe da importare,
    #skiprows = 0, #per saltare alcune righe all'inizio del file
    )
```

```
In [3]: #NOTA INTERESSANTE: altro modo di comporre il percorso del file
import os
path = r"C:\Users\ianto\Desktop\Corso python\file"
os.path.join(path, "Clienti.csv")
```

```
Out[3]: 'C:\\Users\\ianto\\Desktop\\Corso python\\file\\Clienti.csv'
```

```
In [4]: clienti.head(5)
```

```
Out[4]:
```

	NumeroCliente	Nome	DataNascita	Regione
0	1	Nicoletta	01/01/2010	NaN
1	2	Giovanni	01/03/1976	Lazio
2	3	Marco	01/04/1980	Lazio
3	4	Giovanna	01/05/1977	Lazio
4	5	Alice	01/06/1969	Sicilia

```
In [5]: #Per visualizzare tutte le righe e tutte le colonne
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
In [6]: clienti.head(5)
```

```
Out[6]:
```

	NumeroCliente	Nome	DataNascita	Regione
0	1	Nicoletta	01/01/2010	NaN
1	2	Giovanni	01/03/1976	Lazio
2	3	Marco	01/04/1980	Lazio
3	4	Giovanna	01/05/1977	Lazio
4	5	Alice	01/06/1969	Sicilia

Ricavare informazioni sul dataframe

```
In [7]: type(clienti)
```

```
Out[7]: pandas.core.frame.DataFrame
```

```
In [8]: clienti.dtypes
```

```
Out[8]: NumeroCliente    int64
Nome                    object
DataNascita             object
Regione                 object
dtype: object
```

```
In [9]: clienti.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39 entries, 0 to 38
Data columns (total 4 columns):
#   Column          Non-Null Count  Dtype
---  -
0   NumeroCliente   39 non-null    int64
1   Nome            39 non-null    object
2   DataNascita     39 non-null    object
3   Regione         38 non-null    object
dtypes: int64(1), object(3)
memory usage: 1.3+ KB
```

```
In [10]: clienti.columns
```

```
Out[10]: Index(['NumeroCliente', 'Nome', 'DataNascita', 'Regione'], dtype='object')
```

```
In [11]: clienti.shape
```

```
Out[11]: (39, 4)
```

```
In [12]: clienti.index
```

```
Out[12]: RangeIndex(start=0, stop=39, step=1)
```

Perché i type sono importanti

La prossima istruzione genera un errore perché il metodo `year` può essere applicato solo a colonne di tipo `datetime`

```
In [13]: clienti["DataNascita"].dt.year
```

```
-----
AttributeError                                Traceback (most recent call last)
Cell In[13], line 1
----> 1 clienti["DataNascita"].dt.year

File ~\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\generic.py:6299, in NDFrame.__getattr__(self, name)
    6292 if (
    6293     name not in self._internal_names_set
    6294     and name not in self._metadata
    6295     and name not in self._accessors
    6296     and self._info_axis._can_hold_identifiers_and_holds_name(name)
    6297 ):
    6298     return self[name]
-> 6299 return object.__getattr__(self, name)

File ~\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\accessor.py:224, in CachedAccessor.__get__(self, obj, cls)
    221 if obj is None:
    222     # we're accessing the attribute of the class, i.e., Dataset.geo
    223     return self._accessor
--> 224 accessor_obj = self._accessor(obj)
    225 # Replace the property with the accessor object. Inspired by:
    226 # https://www.pydanny.com/cached-property.html
    227 # We need to use object.__setattr__ because we overwrite __setattr__ on
    228 # NDFrame
    229 object.__setattr__(obj, self._name, accessor_obj)

File ~\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\indexes\accessors.py:643, in CombinedDatetimelikeProperties.__new__(cls, data)
    640 elif isinstance(data.dtype, PeriodDtype):
    641     return PeriodProperties(data, orig)
--> 643 raise AttributeError("Can only use .dt accessor with datetimelike values")

AttributeError: Can only use .dt accessor with datetimelike values
```

Occorre prima convertire la colonna nel tipo date

```
In [ ]: clienti['DataNascita'] = clienti['DataNascita'].astype("datetime64[ns]")
```

```
In [ ]: clienti.head(5)
```

```
In [ ]: clienti.dtypes
```

```
In [14]: clienti.head(10)
```

Out[14]:

	NumeroCliente	Nome	DataNascita	Regione
0	1	Nicoletta	01/01/2010	NaN
1	2	Giovanni	01/03/1976	Lazio
2	3	Marco	01/04/1980	Lazio
3	4	Giovanna	01/05/1977	Lazio
4	5	Alice	01/06/1969	Sicilia
5	6	Fabrizio	01/07/1996	Sicilia
6	7	Irene	01/08/1990	Sicilia
7	8	Maria	01/09/1999	Sicilia
8	9	Grazie	01/10/1990	Sicilia
9	10	Giovanni	01/11/1971	Toscana

In [15]: `clienti["DataNascita"].dt.year[0:5]`

AttributeError

Traceback (most recent call last)

Cell In[15], line 1

```
----> 1 clienti["DataNascita"].dt.year[0:5]
```

File ~\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\generic.py:6299, in NDFrame.__getattr__(self, name)

```
6292 if (
6293     name not in self._internal_names_set
6294     and name not in self._metadata
6295     and name not in self._accessors
6296     and self._info_axis._can_hold_identifiers_and_holds_name(name)
6297 ):
6298     return self[name]
-> 6299 return object.__getattr__(self, name)
```

File ~\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\accessor.py:224, in CachedAccessor.__get__(self, obj, cls)

```
221 if obj is None:
222     # we're accessing the attribute of the class, i.e., Dataset.geo
223     return self._accessor
--> 224 accessor_obj = self._accessor(obj)
225 # Replace the property with the accessor object. Inspired by:
226 # https://www.pydanny.com/cached-property.html
227 # We need to use object.__setattr__ because we overwrite __setattr__ on
228 # NDFrame
229 object.__setattr__(obj, self._name, accessor_obj)
```

File ~\AppData\Local\Programs\Python\Python313\Lib\site-packages\pandas\core\indexes\accessors.py:643, in CombinedDatetimelikeProperties.__new__(cls, data)

```
640 elif isinstance(data.dtype, PeriodDtype):
641     return PeriodProperties(data, orig)
--> 643 raise AttributeError("Can only use .dt accessor with datetimelike values")
```

AttributeError: Can only use .dt accessor with datetimelike values

Importare file da excel

```
In [16]: from_excel = pd.read_excel(io = r"C:\Users\ianto\Desktop\Corso python\file\Fattu
        sheet_name = 'Tabelle1',
        usecols = 'A:D,F',
        header = 0
        )
```

```
In [17]: type(from_excel)
```

```
Out[17]: pandas.core.frame.DataFrame
```

```
In [18]: from_excel.head(5)
```

```
Out[18]:
```

	IdFattura	IdProdotto	PrezzoUnitario	Quantita	Omaggio
0	1	4	28.8	28	0.0
1	1	13	38.0	24	0.0
2	2	1	7.3	18	0.0
3	2	3	14.4	4	0.0
4	2	9	12.0	43	0.0

```
In [19]: from_excel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 904 entries, 0 to 903
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   IdFattura       904 non-null   int64
1   IdProdotto      904 non-null   int64
2   PrezzoUnitario  904 non-null   float64
3   Quantita        904 non-null   int64
4   Omaggio         903 non-null   float64
dtypes: float64(2), int64(3)
memory usage: 35.4 KB
```

Importare file JSON

```
In [20]: clienti_j = pd.read_json(path_or_buf = r"C:\Users\ianto\Desktop\Corso python\fil
        clienti_j.head(5)
```

Out[20]:

	IdCliente	IsActive	Nome	Cognome	DataNascita	Nazione	Cap	Telefono	
0	1	True	Nicola	5CF71	2010-01-01	Francia	81622	39 320 3231	Ni
1	2	True	Giovanni	A83C2	1976-03-01	Italia	82786	328 32312	Gio
2	3	True	Marco	7929A	1980-04-01	Italia	19341	+ (39) 327 38312	Mi
3	4	True	Giovanna	270BC	1977-05-01	Italia	64791	39 320 22312	Gio
4	5	True	Alice	C5B4D	1969-06-01	Italia	99172	320 7231	A

In [21]: `clienti2_j = pd.read_json(path_or_buf = r"C:\Users\ianto\Desktop\Corso python\fi
clienti2_j.head(5)`

Out[21]:

	Cognome	DataNascita	IdCliente	IsActive	Residenza	Contatti	Nome
0	Rossi	1972-05-01	40	true	{'Nazione': 'Italia', 'Regione': 'Molise'}	[39 320 3231, 39 320 1123]	NaN
1	NaN	1980-05-01	41	true	{'Nazione': 'Italia', 'Regione': 'Lombardia'}	[39 320 3199, 39 320 8833]	Nicola

In [22]: `clienti2_j.dtypes`

Out[22]:

```
Cognome      object
DataNascita  object
IdCliente    int64
IsActive     object
Residenza    object
Contatti     object
Nome         object
dtype: object
```


In [23]: `import json

with open(path + "/Clienti2.json", "r") as f:
 data_json = json.load(f)
clienti2_j = pd.json_normalize(data_json)`

In [24]: `clienti2_j.head(5)`

Out[24]:

	Cognome	DataNascita	IdCliente	IsActive	Contatti	Residenza.Nazione	Residenza.R
0	Rossi	1972-05-01	40	true	[39 320 3231, 39 320 1123]	Italia	
1	NaN	1980-05-01	41	true	[39 320 3199, 39 320 8833]	Italia	Lon



OSSERVAZIONE: con Python potremmo anche lavorare direttamente data_json

In [25]: data_json

Out[25]:

```
[{'Cognome': 'Rossi',
  'DataNascita': '1972-05-01',
  'IdCliente': 40,
  'IsActive': 'true',
  'Residenza': {'Nazione': 'Italia', 'Regione': 'Molise'},
  'Contatti': ['39 320 3231', '39 320 1123']},
 {'Nome': 'Nicola',
  'DataNascita': '1980-05-01',
  'IdCliente': 41,
  'IsActive': 'true',
  'Residenza': {'Nazione': 'Italia', 'Regione': 'Lombardia'},
  'Contatti': ['39 320 3199', '39 320 8833']}
```

In [26]: type(data_json[0])

Out[26]: dict

Importiamo un altro file csv più complesso

In [27]:

```
fatture = pd.read_csv(filepath_or_buffer = r"C:\Users\ianto\Desktop\Corso python",
                      sep = ";", #separtore del file
                      header = None,
                      names = ["NumeroFattura", "Tipologia", "Importo", "Iva", "IdCliente", "Re",
                              "decimal = ","
                      )
```

In [28]: fatture.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   NumeroFattura         18 non-null    int64
1   Tipologia             18 non-null    object
2   Importo               18 non-null    float64
3   Iva                   17 non-null    float64
4   IdCliente             18 non-null    int64
5   ResidenzaCliente      18 non-null    object
6   DataFattura           18 non-null    object
7   NumeroFornitore       16 non-null    float64
dtypes: float64(3), int64(2), object(3)
memory usage: 1.3+ KB
```

Creiamo una nuova colonna DataFattura2 convertendo DataFattura con il metodo astype

```
In [29]: fatture['DataFattura2'] = fatture['DataFattura'].astype("datetime64[ns]")
```

La conversione è errata, la data 01/03/2017 è diventata 2017-01-03

```
In [30]: fatture.head(5)
```

```
Out[30]:
```

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	I
0	1	A	1120.0	20.0	1	Molise	01/01/2018	
1	2	V	32.0	20.0	2	Puglia	01/03/2017	
2	3	A	45.0	20.0	3	Lombardia	01/06/2017	
3	4	V	64.0	20.0	3	Lombardia	30/01/2019	
4	5	A	12.0	20.0	5	Umbria	01/01/2018	

Dobbiamo specificare il formato di partenza tramite l'argomento format del metodo di pandas to_datetime

```
In [31]: fatture['DataFattura2'] = pd.to_datetime(fatture['DataFattura'],
                                                format="%d/%m/%Y",
                                                )
```

```
In [32]: fatture.head(5)
```

```
Out[32]:
```

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	I
0	1	A	1120.0	20.0	1	Molise	01/01/2018	
1	2	V	32.0	20.0	2	Puglia	01/03/2017	
2	3	A	45.0	20.0	3	Lombardia	01/06/2017	
3	4	V	64.0	20.0	3	Lombardia	30/01/2019	
4	5	A	12.0	20.0	5	Umbria	01/01/2018	

ATTENZIONE! Il prossimo blocco di codice importa le date in modo errato

```
In [33]: fatture2 = pd.read_csv(filepath_or_buffer = r"C:\Users\ianto\Desktop\Corso pytho
        sep = ";", #separtore del file
        header = None,
        names = ["NumeroFattura", "Tipologia", "Importo", "Iva", "IdCliente", "Re
        decimal = ",",
        parse_dates = ["DataFattura"]
        )
        fatture2.head(5)
```

```
Out[33]:
```

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	I
0	1	A	1120.0	20.0	1	Molise	01/01/2018	
1	2	V	32.0	20.0	2	Puglia	01/03/2017	
2	3	A	45.0	20.0	3	Lombardia	01/06/2017	
3	4	V	64.0	20.0	3	Lombardia	30/01/2019	
4	5	A	12.0	20.0	5	Umbria	01/01/2018	

La prossima conversione fallisce perché non posso convertire in int64 una colonna che contiene dei null

```
In [34]: fatture['NumeroFornitore'] = fatture['NumeroFornitore'].fillna(-1).astype('int64')
```

Per acquisire i null come intero devo usare il tipo Int64

```
In [35]: fatture['NumeroFornitore'] = fatture['NumeroFornitore'].astype('Int64')
```

Oppure sostituire prima i null con un altro valore

```
In [36]: fatture['Iva'] = fatture['Iva'].fillna(0).astype(np.int64)
```

```
In [37]: fatture.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   NumeroFattura         18 non-null    int64
1   Tipologia             18 non-null    object
2   Importo               18 non-null    float64
3   Iva                   18 non-null    int64
4   IdCliente             18 non-null    int64
5   ResidenzaCliente     18 non-null    object
6   DataFattura           18 non-null    object
7   NumeroFornitore      18 non-null    Int64
8   DataFattura2          18 non-null    datetime64[ns]
dtypes: Int64(1), datetime64[ns](1), float64(1), int64(3), object(3)
memory usage: 1.4+ KB
```

```
In [38]: fatture.head(5)
```

Out[38]:

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	N
0	1	A	1120.0	20	1	Molise	01/01/2018	
1	2	V	32.0	20	2	Puglia	01/03/2017	
2	3	A	45.0	20	3	Lombardia	01/06/2017	
3	4	V	64.0	20	3	Lombardia	30/01/2019	
4	5	A	12.0	20	5	Umbria	01/01/2018	

Attenzione alla conversione in stringa

In [39]: `fatture2 = fatture`

In [40]: `fatture2.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   NumeroFattura         18 non-null    int64
1   Tipologia             18 non-null    object
2   Importo               18 non-null    float64
3   Iva                   18 non-null    int64
4   IdCliente             18 non-null    int64
5   ResidenzaCliente      18 non-null    object
6   DataFattura           18 non-null    object
7   NumeroFornitore       18 non-null    Int64
8   DataFattura2          18 non-null    datetime64[ns]
dtypes: Int64(1), datetime64[ns](1), float64(1), int64(3), object(3)
memory usage: 1.4+ KB
```

In [41]: `fatture2.head(5)`

Out[41]:

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	N
0	1	A	1120.0	20	1	Molise	01/01/2018	
1	2	V	32.0	20	2	Puglia	01/03/2017	
2	3	A	45.0	20	3	Lombardia	01/06/2017	
3	4	V	64.0	20	3	Lombardia	30/01/2019	
4	5	A	12.0	20	5	Umbria	01/01/2018	

In [42]: `fatture2['NumeroFornitore'] = fatture2['NumeroFornitore'].astype('str')`

In [43]: `fatture2.info()`

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   NumeroFattura         18 non-null    int64
1   Tipologia             18 non-null    object
2   Importo               18 non-null    float64
3   Iva                  18 non-null    int64
4   IdCliente            18 non-null    int64
5   ResidenzaCliente     18 non-null    object
6   DataFattura          18 non-null    object
7   NumeroFornitore      18 non-null    object
8   DataFattura2         18 non-null    datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(3), object(4)
memory usage: 1.4+ KB

```

Il null è diventato una stringa

```
In [44]: fatture2.head(5)
```

```
Out[44]:
```

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	N
0	1	A	1120.0	20	1	Molise	01/01/2018	
1	2	V	32.0	20	2	Puglia	01/03/2017	
2	3	A	45.0	20	3	Lombardia	01/06/2017	
3	4	V	64.0	20	3	Lombardia	30/01/2019	
4	5	A	12.0	20	5	Umbria	01/01/2018	

Riconverto la stringa in null. Facciamo comunque sempre dei test perché su Pandas potrei varie tipologie di null

```
In [45]: fatture2['NumeroFornitore'] = fatture2['NumeroFornitore'].astype('str').replace('0', '0')
```

```
In [46]: fatture2.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   NumeroFattura         18 non-null    int64
1   Tipologia             18 non-null    object
2   Importo               18 non-null    float64
3   Iva                  18 non-null    int64
4   IdCliente            18 non-null    int64
5   ResidenzaCliente     18 non-null    object
6   DataFattura          18 non-null    object
7   NumeroFornitore      18 non-null    object
8   DataFattura2         18 non-null    datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(3), object(4)
memory usage: 1.4+ KB

```

```
In [47]: fatture2.head(5)
```

Out[47]:

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	N
0	1	A	1120.0	20	1	Molise	01/01/2018	
1	2	V	32.0	20	2	Puglia	01/03/2017	
2	3	A	45.0	20	3	Lombardia	01/06/2017	
3	4	V	64.0	20	3	Lombardia	30/01/2019	
4	5	A	12.0	20	5	Umbria	01/01/2018	

Attenzione a non dimenticare .copy()

In [48]: `fatture_new = fatture`
`fatture_new["NumeroFattura"] = 2`

In [49]: `fatture_new.head(5)`

Out[49]:

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	N
0	2	A	1120.0	20	1	Molise	01/01/2018	
1	2	V	32.0	20	2	Puglia	01/03/2017	
2	2	A	45.0	20	3	Lombardia	01/06/2017	
3	2	V	64.0	20	3	Lombardia	30/01/2019	
4	2	A	12.0	20	5	Umbria	01/01/2018	

Anche la colonna NumeroFattura del dataframe fatture è stata modificata!

In [50]: `fatture.head(5)`

Out[50]:

	NumeroFattura	Tipologia	Importo	Iva	IdCliente	ResidenzaCliente	DataFattura	N
0	2	A	1120.0	20	1	Molise	01/01/2018	
1	2	V	32.0	20	2	Puglia	01/03/2017	
2	2	A	45.0	20	3	Lombardia	01/06/2017	
3	2	V	64.0	20	3	Lombardia	30/01/2019	
4	2	A	12.0	20	5	Umbria	01/01/2018	

Esercizio: importiamo un file excel contenente più fogli

valorizziamo l'argomento `sheet_name` con `None`

```
In [51]: data = pd.read_excel(r"C:\Users\ianto\Desktop\Corso python\file\Dati_su_piu_fogl  
sheet_name=["Fatture","Clienti"])
```

```
In [52]: type(data)
```

```
Out[52]: dict
```

```
In [53]: data.keys()
```

```
Out[53]: dict_keys(['Fatture', 'Clienti'])
```

```
In [54]: data["Fatture"].head(5)
```

```
Out[54]:
```

	IdFattura	Tipologia	Importo	Iva	IdCliente	IdFornitore
0	1	A	40	20.0	1	1.0
1	2	V	32	20.0	2	1.0
2	3	A	45	20.0	3	1.0
3	4	V	64	20.0	3	1.0
4	5	A	12	20.0	5	1.0

Un dizionario è un insieme di coppie chiave-valore. Visualizziamo l'elenco delle chiavi

Visualizziamo il valore della chiave Fatture

```
In [55]: data["Clienti"].head(5)
```

```
Out[55]:
```

	NumeroCliente	IsActive	Nome	Cognome	DataNascita	Nazione	Regione
0	1	True	Nicoletta	5CF71	2010-01-01	Francia	NaN
1	2	True	Giovanni	A83C2	1976-03-01	Italia	Lazio
2	3	True	Marco	7929A	1980-04-01	Italia	Lazio
3	4	True	Giovanna	270BC	1977-05-01	Italia	Lazio
4	5	True	Alice	C5B4D	1969-06-01	Italia	Sicilia

posso fare un ciclo sulle chiavi

```
In [56]: for chiave in data.keys():  
numero_righe = len(data[chiave])  
print(f'il foglio {chiave} ha {numero_righe} righe')
```

il foglio Fatture ha 18 righe

il foglio Clienti ha 39 righe

Posso rimuovere chiavi da un dizionario

Api: application programming interface

```
In [57]: import requests
```

Creiamo una variabile con l'url per la chiamata api

```
In [58]: url = "https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst"
```

Utilizziamo il metodo get della libreria requests

```
In [59]: res = requests.get("https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst")
```

```
Out[59]: <Response [200]>
```

Controlliamo che lo status_code sia 200

```
In [60]: res.status_code
```

```
Out[60]: 200
```

Visualizziamo il contenuto della risposta: i dati sono difficilmente lavorabili

```
In [61]: res.content[0:500]
```

```
Out[61]: b'{"id":451192837,"node_id":"R_kgDOGuSoBQ","name":"PortfolioDataAnalyst","full_name":"iantomasinicola/PortfolioDataAnalyst","private":false,"owner":{"login":"iantomasinicola","id":59792312,"node_id":"MDQ6VXNlcjU5NzkyMzEy","avatar_url":"https://avatars.githubusercontent.com/u/59792312?v=4","gravatar_id":"","url":"https://api.github.com/users/iantomasinicola","html_url":"https://github.com/iantomasinicola","followers_url":"https://api.github.com/users/iantomasinicola/followers","following_url":"http"
```

Anche in formato testo la situazione è difficilmente gestibile

```
In [62]: res.text[0:500]
```

```
Out[62]: '{"id":451192837,"node_id":"R_kgDOGuSoBQ","name":"PortfolioDataAnalyst","full_name":"iantomasinicola/PortfolioDataAnalyst","private":false,"owner":{"login":"iantomasinicola","id":59792312,"node_id":"MDQ6VXNlcjU5NzkyMzEy","avatar_url":"https://avatars.githubusercontent.com/u/59792312?v=4","gravatar_id":"","url":"https://api.github.com/users/iantomasinicola","html_url":"https://github.com/iantomasinicola","followers_url":"https://api.github.com/users/iantomasinicola/followers","following_url":"http"
```

Invece convertendo i dati in json, posso utilizzare le funzionalità dei dizionari di Python!

```
In [63]: res.json()
```

```

Out[63]: {'id': 451192837,
          'node_id': 'R_kgDOGuSoBQ',
          'name': 'PortfolioDataAnalyst',
          'full_name': 'iantomasinicola/PortfolioDataAnalyst',
          'private': False,
          'owner': {'login': 'iantomasinicola',
                    'id': 59792312,
                    'node_id': 'MDQ6VXNlcjU5NzkyMzEy',
                    'avatar_url': 'https://avatars.githubusercontent.com/u/59792312?v=4',
                    'gravatar_id': '',
                    'url': 'https://api.github.com/users/iantomasinicola',
                    'html_url': 'https://github.com/iantomasinicola',
                    'followers_url': 'https://api.github.com/users/iantomasinicola/followers',
                    'following_url': 'https://api.github.com/users/iantomasinicola/following{other_user}',
                    'gists_url': 'https://api.github.com/users/iantomasinicola/gists{/gist_id}',
                    'starred_url': 'https://api.github.com/users/iantomasinicola/starred{/owner}{/repo}',
                    'subscriptions_url': 'https://api.github.com/users/iantomasinicola/subscriptions',
                    'organizations_url': 'https://api.github.com/users/iantomasinicola/orgs',
                    'repos_url': 'https://api.github.com/users/iantomasinicola/repos',
                    'events_url': 'https://api.github.com/users/iantomasinicola/events{/privacy}',
                    'received_events_url': 'https://api.github.com/users/iantomasinicola/received_events',
                    'type': 'User',
                    'user_view_type': 'public',
                    'site_admin': False},
          'html_url': 'https://github.com/iantomasinicola/PortfolioDataAnalyst',
          'description': 'Progetto di Data analysis con Python, Microsoft Sql Server e Excel',
          'fork': False,
          'url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst',
          'forks_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/forks',
          'keys_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/keys{/key_id}',
          'collaborators_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/collaborators{/collaborator}',
          'teams_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/teams',
          'hooks_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/hooks',
          'issue_events_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/issues/events{/number}',
          'events_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/events',
          'assignees_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/assignees{/user}',
          'branches_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/branches{/branch}',
          'tags_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/tags',
          'blobs_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/git/blobs{/sha}',
          'git_tags_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/git/tags{/sha}',
          'git_refs_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalyst/git/refs{/sha}',

```

```
'trees_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalys
t/git/trees{/sha}',
'statuses_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAna
lyst/statuses/{sha}',
'languages_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAn
alyst/languages',
'stargazers_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataA
nalyst/stargazers',
'contributors_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDat
aAnalyst/contributors',
'subscribers_url': 'https://api.github.com/repos/iantomasinicola/PortfolioData
Analyst/subscribers',
'subscription_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDat
aAnalyst/subscription',
'commits_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnal
yst/commits{/sha}',
'git_commits_url': 'https://api.github.com/repos/iantomasinicola/PortfolioData
Analyst/git/commits{/sha}',
'comments_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAna
lyst/comments{/number}',
'issue_comment_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDa
taAnalyst/issues/comments{/number}',
'contents_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAna
lyst/contents/{+path}',
'compare_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnal
yst/compare/{base}...{head}',
'merges_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnaly
st/merges',
'archive_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnal
yst/{archive_format}/{ref}',
'downloads_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAn
alyst/downloads',
'issues_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnaly
st/issues{/number}',
'pulls_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnalys
t/pulls{/number}',
'milestones_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataA
nalyst/milestones{/number}',
'notifications_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDa
taAnalyst/notifications{?since,all,participating}',
'labels_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAnaly
st/labels{/name}',
'releases_url': 'https://api.github.com/repos/iantomasinicola/PortfolioDataAna
lyst/releases{/id}',
'deployments_url': 'https://api.github.com/repos/iantomasinicola/PortfolioData
Analyst/deployments',
'created_at': '2022-01-23T18:22:31Z',
'updated_at': '2024-09-09T22:28:16Z',
'pushed_at': '2022-03-29T07:47:12Z',
'git_url': 'git://github.com/iantomasinicola/PortfolioDataAnalyst.git',
'ssh_url': 'git@github.com:iantomasinicola/PortfolioDataAnalyst.git',
'clone_url': 'https://github.com/iantomasinicola/PortfolioDataAnalyst.git',
'svn_url': 'https://github.com/iantomasinicola/PortfolioDataAnalyst',
'homepage': 'https://www.yimp.it/portfolio-data-analyst/',
'size': 161,
'stargazers_count': 2,
'watchers_count': 2,
'language': 'TSQL',
'has_issues': True,
'has_projects': True,
```



```
'has_downloads': True,
'has_wiki': True,
'has_pages': False,
'has_discussions': False,
'forks_count': 0,
'mirror_url': None,
'archived': False,
'disabled': False,
'open_issues_count': 0,
'license': None,
'allow_forking': True,
'is_template': False,
'web_commit_signoff_required': False,
'topics': ['data-analysis', 'excel', 'python', 'sql'],
'visibility': 'public',
'forks': 0,
'open_issues': 0,
'watchers': 2,
'default_branch': 'main',
'temp_clone_token': None,
'network_count': 0,
'subscribers_count': 1}
```

```
In [64]: json_res = res.json()
```

```
In [65]: type(json_res)
```

```
Out[65]: dict
```

Accedo a dei valori specifici del dizionario

```
In [66]: owner = json_res["owner"]["avatar_url"]
owner
```

```
Out[66]: 'https://avatars.githubusercontent.com/u/59792312?v=4'
```

```
In [67]: url_home = json_res["owner"]["html_url"]
url_home
```

```
Out[67]: 'https://github.com/iantomasinicola'
```

```
In [68]: topics = json_res["topics"]
topics
```

```
Out[68]: ['data-analysis', 'excel', 'python', 'sql']
```

```
In [69]: type(topics)
```

```
Out[69]: list
```

```
In [70]: print(json_res["description"])
```

Progetto di Data analysis con Python, Microsoft Sql Server e Excel

```
In [71]: lista = [[json_res["name"], json_res["owner"]["html_url"], json_res["topics"]]]
```

```
In [72]: lista
```

```
Out[72]: [['PortfolioDataAnalyst',  
          'https://github.com/iantomasinicola',  
          ['data-analysis', 'excel', 'python', 'sql']]]
```

```
In [73]: pd.DataFrame(columns=["name", "owner", "topics"], data= lista)
```

```
Out[73]:
```

	name	owner	topics
0	PortfolioDataAnalyst	https://github.com/iantomasinicola	[data-analysis, excel, python, sql]

Vediamo come possiamo creare un DataFrame

```
In [74]: nuovo_dict = {"name": json_res["name"],  
                      "stato": json_res["owner"]["html_url"],  
                      "topics": json_res["topics"]}
```

```
In [75]: nuovo_dict
```

```
Out[75]: {'name': 'PortfolioDataAnalyst',  
          'stato': 'https://github.com/iantomasinicola',  
          'topics': ['data-analysis', 'excel', 'python', 'sql']}
```

```
In [76]: pd.DataFrame(nuovo_dict)
```

```
Out[76]:
```

	name	stato	topics
0	PortfolioDataAnalyst	https://github.com/iantomasinicola	data-analysis
1	PortfolioDataAnalyst	https://github.com/iantomasinicola	excel
2	PortfolioDataAnalyst	https://github.com/iantomasinicola	python
3	PortfolioDataAnalyst	https://github.com/iantomasinicola	sql

Esercizio 2: fare degli esperimenti con l'url per ottenere una lista di repository

```
In [77]: url = "https://api.github.com/users/iantomasinicola/repos"
```

```
In [78]: progetti = requests.get(url).json()
```

```
In [79]: lista = []  
for el in progetti:  
    url = "https://api.github.com/repos/iantomasinicola/" + el["name"]  
    lista.append([ el["name"], requests.get(url).json()["topics"] ])  
lista
```

```
Out[79]: [['corso-performance-sql-server', ['sqlserver', 'tsql']],
          ['CorsoPython', []],
          ['DatabaseYimp', ['creare-database', 'database', 'sql']],
          ['eBook', ['database', 'mysql', 'mysql-database', 'sql', 'sql-server']],
          ['Esercizi', []],
          ['Esercizi_excel', []],
          ['fantacalcio-e-intelligenza-artificiale',
           ['fantacalcio', 'intelligenza-artificiale', 'python']],
          ['iantomasinicola', []],
          ['indici-sql-server', ['database', 'indexdb', 'sql', 'sql-server']],
          ['lavorare-sui-dataframe-di-pandas',
           ['data-analyst', 'data-science', 'python']],
          ['LezioniPython', ['machine-learning', 'machine-learning-python', 'python']],
          ['Machine-Learning-con-SQL', []],
          ['PortfolioDataAnalyst', ['data-analysis', 'excel', 'python', 'sql']],
          ['progetto-sql', ['sql']]]
```

```
In [80]: lista = []
         for el in progetti:
             lista.append([ el["name"], requests.get(url).json()["topics"] ])
         lista
```

```
Out[80]: [['corso-performance-sql-server', ['sql']],
          ['CorsoPython', ['sql']],
          ['DatabaseYimp', ['sql']],
          ['eBook', ['sql']],
          ['Esercizi', ['sql']],
          ['Esercizi_excel', ['sql']],
          ['fantacalcio-e-intelligenza-artificiale', ['sql']],
          ['iantomasinicola', ['sql']],
          ['indici-sql-server', ['sql']],
          ['lavorare-sui-dataframe-di-pandas', ['sql']],
          ['LezioniPython', ['sql']],
          ['Machine-Learning-con-SQL', ['sql']],
          ['PortfolioDataAnalyst', ['sql']],
          ['progetto-sql', ['sql']]]
```

```
In [81]: pd.DataFrame(columns=["NomeProgetto", "Argomento"], data=lista)
```

Out[81]:

	NomeProgetto	Argomento
0	corso-performance-sql-server	[sql]
1	CorsoPython	[sql]
2	DatabaseYimp	[sql]
3	eBook	[sql]
4	Esercizi	[sql]
5	Esercizi_excel	[sql]
6	fantacalcio-e-intelligenza-artificiale	[sql]
7	iantomasinicola	[sql]
8	indici-sql-server	[sql]
9	lavorare-sui-dataframe-di-pandas	[sql]
10	LezioniPython	[sql]
11	Machine-Learning-con-SQL	[sql]
12	PortfolioDataAnalyst	[sql]
13	progetto-sql	[sql]