Corso ITS:

PROGETTISTA E SVILUPPATORE SOFTWARE:

FULL STACK DEVELOPER E CLOUD SPECIALIST

Modulo: **Programmazione in Python**

Docente: *Andrea Ribuoli*

Martedì **6 Maggio** 2025

09:00 - 13:00

13:30 - 16:30

---

datetime.date()

In [4]:
```
import datetime
datetime.date()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [4], in <cell line: 2>()
      1 import datetime
----> 2 datetime.date()

TypeError: function missing required argument 'year' (pos 1)
```

In [5]:
```
import datetime
datetime.date(year=2025)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [5], in <cell line: 2>()
      1 import datetime
----> 2 datetime.date(year=2025)

TypeError: function missing required argument 'month' (pos 2)
```

In [6]:
```
import datetime
datetime.date(year=2025, month=5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [6], in <cell line: 2>()
      1 import datetime
----> 2 datetime.date(year=2025, month=5)

TypeError: function missing required argument 'day' (pos 3)
```

In [7]:
```
import datetime
datetime.date(year=2025, month=5, day=6)
```

Out[7]:
```
datetime.date(2025, 5, 6)
```

In [11]:
```
prev = datetime.date(2025, 4, 30)
curr = datetime.date(2025, 5, 6)
```

In [14]:
```
curr - prev
```

Out[14]:
```
datetime.timedelta(days=6)
```

---

datetime.datetime()

In [16]:
```
datetime.datetime()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [16], in <cell line: 1>()
----> 1 datetime.datetime()

TypeError: function missing required argument 'year' (pos 1)
```

In [17]:
```
datetime.datetime(year=2025)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [17], in <cell line: 1>()
----> 1 datetime.datetime(year=2025)

TypeError: function missing required argument 'month' (pos 2)
```

In [18]:
```
datetime.datetime(year=2025, month=5, day=6)
```

Out[18]:
```
datetime.datetime(2025, 5, 6, 0, 0)
```

In [19]:
```
datetime.datetime(year=2025, month=5, day=6, hour=9)
```

Out[19]:
```
datetime.datetime(2025, 5, 6, 9, 0)
```

In [20]:
```
datetime.datetime(year=2025, month=5, day=6, hour=9, minute=30)
```

Out[20]:
```
datetime.datetime(2025, 5, 6, 9, 30)
```

In [21]:
```
prev =datetime.datetime(2025, 4, 30, 16, 30)
curr =datetime.datetime(2025, 5, 6, 9, 30)
```

In [22]:
```
curr - prev
```

Out[22]:
```
datetime.timedelta(days=5, seconds=61200)
```

In [23]:
```
datetime.datetime(2025, 5, 6, 9, 0, 90)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [23], in <cell line: 1>()
----> 1 datetime.datetime(2025, 5, 6, 9, 0, 90)

ValueError: second must be in 0..59
```

In [24]:
```
datetime.datetime(2025, 5, 6, 9, 1, 30)
```

Out[24]:
```
datetime.datetime(2025, 5, 6, 9, 1, 30)
```

In [46]:
```
import urllib.request
url = "https://www.comuni-italiani.it/province.html"
response = urllib.request.urlopen(url)
theBytes = response.read()
text = theBytes.decode(encoding="iso-8859-1")
lista = []
import bs4
doc = bs4.BeautifulSoup(text)
elems = doc.find_all("table")
table = elems[3]
for tr in table.contents[2:-2]:
    if type(tr) == bs4.element.Tag :
        tds = tr.contents
        sequ = int(tds[0].get_text())
        prov = tds[1].get_text()
        resi = int(tds[2].get_text().replace(".",""))
        sigl = tds[7].get_text()
        kmq = int(tds[4].get_text().replace(".",""))
        denso = float(tds[5].get_text().replace(".","").replace(",","."))
        densc = round(resi / kmq, 1)
        lista.append([sigl, prov, resi, kmq])

# import pickle
# with open("province", "wb") as backup :
#     pickle.dump(lista, backup)
```

In [12]:
```
import pandas as pd
df = pd.DataFrame(lista)
df
```

Out[12]:

	0	1	2	3
0	AG	Agrigento	442049	3042
1	AL	Alessandria	426658	3562
2	AN	Ancona	474124	1940
3	AO	Aosta	126883	3263
4	AR	Arezzo	344374	3235
...	...	...	...	...
105	VC	Vercelli	173868	2088
106	VR	Verona	921557	3121
107	VV	Vibo Valentia	161619	1139
108	VI	Vicenza	865082	2723
109	VT	Viterbo	319008	3612

110 rows × 4 columns

In [47]:
```
df = pd.DataFrame(lista, columns=['sigla', 'provincia', 'residenti', 'kmq'])
df
```

Out[47]:

	sigla	provincia	residenti	kmq
0	AG	Agrigento	442049	3042
1	AL	Alessandria	426658	3562
2	AN	Ancona	474124	1940
3	AO	Aosta	126883	3263
4	AR	Arezzo	344374	3235
...	...	...	...	...
105	VC	Vercelli	173868	2088
106	VR	Verona	921557	3121
107	VV	Vibo Valentia	161619	1139
108	VI	Vicenza	865082	2723
109	VT	Viterbo	319008	3612

110 rows × 4 columns

In [48]:
```
df.to_csv("province.csv")
```

In [15]:
```
df.to_csv("province.csv", index=False)
```

In [ ]:
```
url = "'https://it.wikipedia.org/wiki/Province_d%27Italia'"
```

In [ ]:
```
import urllib.request
url = "https://it.wikipedia.org/wiki/Province_d%27Italia"
response = urllib.request.urlopen(url)
theBytes = response.read()
text = theBytes.decode()
import bs4
doc = bs4.BeautifulSoup(text)
elems = doc.find_all("tbody")
tbody = elems[0]
for tr in tbody.contents[1:-1]:
    if type(tr) == bs4.element.Tag :
        tds = tr.contents
        prov = tds[1].contents[1].get_text()
        print(prov)
```

In [ ]:
```
dir(df)
```

In [ ]:
```
!pip install SQLAlchemy
!pip install mysql-connector
```

esempio di messaggio di errore cercando il server *PostgreSQL*

```
OperationalError: (psycopg2.OperationalError) could not
connect to server: Connection refused
        Is the server running on host "localhost" (127.0.0.1)
and accepting
        TCP/IP connections on port 5432?
```

- un **DBMS**, *DataBase Management System*, è un servente
- nella maggior parte dei casi sarà in ascolto su una **porta** TCP/IP nota (`5432` per Postgres)
- se la sua installazione è **locale** vuol dire che esegue sul nostro stesso computer
- in tal caso l'**indirizzo IP** è `127.0.0.1`
- questo indirizzo è raggiungibile con il nome logico: **localhost**

In [37]:
```
from sqlalchemy import create_engine, MetaData, Table, Column, Integer, Stri
engine = create_engine("postgresql+psycopg2://postgres:papapa@localhost/ital
metadata_obj = MetaData()
province = Table(
     "provincia",
     metadata_obj,
     Column("sigla", String(2), primary_key=True),
     Column("nome", String(30)),
     Column("residenti", Integer),
```

```
    )
metadata_obj.create_all(engine)
```

In [38]:
```
!echo "\dt\q" | PGPASSWORD=papapa psql -U postgres italy
```

```
          List of relations
 Schema |   Name    | Type  |  Owner
--------+-----------+-------+----------
 public | provincia | table | postgres
(1 row)
```

In [39]:
```
!echo "\d provincia\q" | PGPASSWORD=papapa psql -U postgres italy
```

```
                    Table "public.provincia"
  Column   |         Type          | Collation | Nullable | Default
-----------+-----------------------+-----------+----------+---------
 sigla     | character varying(2)  |           | not null |
 nome      | character varying(30) |           |          |
 residenti | integer               |           |          |
Indexes:
    "provincia_pkey" PRIMARY KEY, btree (sigla)
```

In [40]:
```
!echo "SELECT * FROM provincia;\q" | PGPASSWORD=papapa psql -U postgres ital
```

```
 sigla | nome | residenti
-------+------+-----------
(0 rows)
```

In [41]:
```
!echo "DROP TABLE provincia;\q" | PGPASSWORD=papapa psql -U postgres italy
```

```
DROP TABLE
```

In [ ]:
```
from sqlalchemy import create_engine, MetaData, Table, Column, Integer, Stri
from sqlalchemy import create_engine
engine = create_engine("mysql+mysqlconnector://root:root@localhost/crm_its")

metadata_obj = MetaData()
province = Table(
     "provincia",
     metadata_obj,
     Column("sigla", String(2), primary_key=True),
     Column("nome", String(30)),
     Column("residenti", Integer),
 )
metadata_obj.create_all(engine)
```