Corso ITS: *ARTIFICIAL INTELLIGENCE SPECIALIST*

Modulo: **Programmazione ad oggetti in Python e librerie esterne**

Docente: *Andrea Ribuoli*

Giovedì **6 Marzo** 2025

09:00 - 14:00

---

integrale.py

- immaginate gli assi cartesiani **x** (*ascisse*) e **y** (*ordinate*)
- suddividete il segmento che va da 0 a 1 sull'asse delle ascisse in 100 parti uguali
- considerate i 100 rettangoli che hanno per base i 100 segmenti e
- per altezza il valore `sqrt(1 - x ** 2)`
- dove x è la posizione iniziale di ciascuna micro-base lungo l'asse *x*
- sommate le aree dei 100 rettangoli così determinati
- dividete il valore determinato per quello di **pi greco**
- generalizzate il programma in modo da rendere parametrico
- il numero di parti in cui dividere il segmento 0-1
- quanto vale il rapporto con pi greco impostando 1000 parti?
- e 10000?
- Sapete darvi una spiegazione?

In [70]:
```
1 / 100
```

Out[70]:
```
0.01
```

In [71]:
```
from math import sqrt, pi
b = 1 / 100
area = 0.0
for i in range(0, 100) :
    area += b * sqrt(1 - (b * i) ** 2)
print(area / pi)
```

```
0.25149799641972553
```

In [72]:
```
from math import sqrt, pi
def integra(parti) :
    b = 1 / parti
    area = 0.0
    for i in range(0, parti) :
        area += b * sqrt(1 - (b * i) ** 2)
    return area / pi
```

In [73]:
```
print(integra(100))
```

```
0.25149799641972553
```

In [74]:
```
print(integra(1000))
```

```
0.2501561957212206
```

In [75]:
```
print(integra(10000))
```

```
0.25001582191291544
```

In [76]:
```
print(integra(100000))
```

```
0.2500015885901169
```

area del cerchio è pari a πr^2

- con il calcolo fatto stiamo approssimando un quarto dell'area del cerchio
- ecco perchè all'aumentare del frazionamento in parti sempre più piccole
- la differenza dal valore esatto si riduce

In [77]:
```
print(integra(1000000))
```

```
0.2500001590613622
```

In [78]:
```
print(integra(10000000))
```

```
0.25000001591251536
```

- un esempio (non efficiente) per consentire all'utente di specificare una funzione:
- - questa verrà ancora integrata tra 0 e 1
- - determino `y(x)` in due tempi:
- - prima formatto la stringa costante sostituendo i riferimenti `{x}` con il valore `b*i`
- - poi sottometto la funzione `eval()` che determinerà **y**

In [79]:
```
# user_expr = input("Indica una funzione di x racchiudendo x tra parentesi g
USER_EXPR = "10 - {x} ** 2 + {x} * 25"
```

In [80]:
```
USER_EXPR.format(x = 0.23)
```

Out[80]: '10 - 0.23 ** 2 + 0.23 * 25'

In [81]:
```
10 - 0.23 ** 2 + 0.23 * 25
```

Out[81]: 15.6971

In [82]:
```
def integra(parti) :
    b = 1 / parti
    area = 0.0
    for i in range(0, parti) :
        area += b * (eval(USER_EXPR.format(x = b*i)))
    return area
```

In [83]:
```
print(integra(100000))
```

```
22.166546666650152
```

REGIONI

sito **https://it.wikipedia.org/wiki/Province_d%27Italia**

regioni.py

- Dal sito Wikipedia estrarre il legame **sigla_provincia - regione** secondo il processo visto nelle precedenti lezioni

In [84]:
```
import urllib.request
address ='https://it.wikipedia.org/wiki/Province_d%27Italia'
response = urllib.request.urlopen(address)
theBytes = response.read()
html = theBytes.decode()
print(len(html))
```

```
382684
```

In [85]:
```
from bs4 import BeautifulSoup
table1 = BeautifulSoup(html, "html.parser").table
# print(type(table1))
riga = table1.find_next("tr")
riga = riga.find_next("tr")
import pandas as pd
regioni = pd.DataFrame(columns = ["sigla", "regione"])
i = 0
while riga != None :
    tdx  = riga.find_next("td")
    tdx  =  tdx.find_next("td")
    sigla = tdx.get_text().strip()
    tdx  =  tdx.find_next("td")
```

```
    regione = tdx.get_text().strip()
    regioni.loc[i] = [sigla, regione]
    if sigla == 'VT' : break
    i += 1
    riga = riga.find_next("tr")
regioni.sort_values(by="regione").to_csv("regioni.csv", index=False)
```

In [86]:
```
from csv import reader
infile = open("regioni.csv")
csvReader = reader(infile)
for riga in csvReader :
    print(riga)
```

```
['sigla', 'regione']
['CH', 'Abruzzo']
['TE', 'Abruzzo']
['AQ', 'Abruzzo']
['PE', 'Abruzzo']
['PZ', 'Basilicata']
['MT', 'Basilicata']
['VV', 'Calabria']
['CS', 'Calabria']
['CZ', 'Calabria']
['RC', 'Calabria']
['KR', 'Calabria']
['SA', 'Campania']
['AV', 'Campania']
['BN', 'Campania']
['CE', 'Campania']
['NA', 'Campania']
['PR', 'Emilia-Romagna']
['MO', 'Emilia-Romagna']
['FE', 'Emilia-Romagna']
['FC', 'Emilia-Romagna']
['BO', 'Emilia-Romagna']
['RA', 'Emilia-Romagna']
['RE', 'Emilia-Romagna']
['RN', 'Emilia-Romagna']
['PC', 'Emilia-Romagna']
['TS', 'Friuli-Venezia Giulia']
['UD', 'Friuli-Venezia Giulia']
['GO', 'Friuli-Venezia Giulia']
['PN', 'Friuli-Venezia Giulia']
['RI', 'Lazio']
['LT', 'Lazio']
['RM', 'Lazio']
['FR', 'Lazio']
['VT', 'Lazio']
['SV', 'Liguria']
['GE', 'Liguria']
['IM', 'Liguria']
['SP', 'Liguria']
['MI', 'Lombardia']
['BS', 'Lombardia']
['PV', 'Lombardia']
['SO', 'Lombardia']
['MB', 'Lombardia']
['MN', 'Lombardia']
['VA', 'Lombardia']
['LO', 'Lombardia']
['BG', 'Lombardia']
['LC', 'Lombardia']
['CO', 'Lombardia']
['CR', 'Lombardia']
['PU', 'Marche']
['MC', 'Marche']
['AN', 'Marche']
['AP', 'Marche']
['FM', 'Marche']
```

```
['CB', 'Molise']
['IS', 'Molise']
['AL', 'Piemonte']
['BI', 'Piemonte']
['NO', 'Piemonte']
['TO', 'Piemonte']
['VC', 'Piemonte']
['CN', 'Piemonte']
['AT', 'Piemonte']
['VB', 'Piemonte']
['BA', 'Puglia']
['BT', 'Puglia']
['BR', 'Puglia']
['FG', 'Puglia']
['TA', 'Puglia']
['LE', 'Puglia']
['SS', 'Sardegna']
['OR', 'Sardegna']
['NU', 'Sardegna']
['CA', 'Sardegna']
['SU', 'Sardegna']
['SR', 'Sicilia']
['TP', 'Sicilia']
['AG', 'Sicilia']
['RG', 'Sicilia']
['EN', 'Sicilia']
['ME', 'Sicilia']
['CL', 'Sicilia']
['PA', 'Sicilia']
['CT', 'Sicilia']
['GR', 'Toscana']
['LI', 'Toscana']
['LU', 'Toscana']
['MS', 'Toscana']
['AR', 'Toscana']
['PI', 'Toscana']
['SI', 'Toscana']
['FI', 'Toscana']
['PT', 'Toscana']
['PO', 'Toscana']
['TN', 'Trentino-Alto Adige']
['BZ', 'Trentino-Alto Adige']
['PG', 'Umbria']
['TR', 'Umbria']
['AO', "Valle d'Aosta"]
['BL', 'Veneto']
['VI', 'Veneto']
['TV', 'Veneto']
['VE', 'Veneto']
['RO', 'Veneto']
['VR', 'Veneto']
['PD', 'Veneto']
```

SQLAlchemy

- pip3 install SQLAlchemy
- pip3 install psycopg2

In [87]:
```
import sqlalchemy
sqlalchemy.__version__
```

Out[87]:
```
'1.4.32'
```

In [88]:
```
import psycopg2
psycopg2.__version__
```

Out[88]:
```
'2.9.1 (dt dec pq3 ext lo64)'
```

In [89]:
```
from sqlalchemy import create_engine, MetaData, Table, Column, Integer, Stri
engine = create_engine("postgresql+psycopg2://postgres:papapa@localhost/ital
                       # "mysql+mysqlconnector://root:root@localhost/crm_its"
metadata_obj = MetaData()
user_table = Table(
     "user_account",
     metadata_obj,
     Column("id", Integer, primary_key=True),
     Column("name", String(30)),
     Column("fullname", String),
 )
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 [102…
```
from sqlalchemy import create_engine, MetaData, Table, Column, Integer, Stri
engine = create_engine("postgresql+psycopg2://postgres:papapa@localhost/ital
                       # "mysql+mysqlconnector://root:root@localhost/crm_its"
metadata_obj = MetaData()
regione = Table(
     "regione",
     metadata_obj,
     Column("sigla", String(2), primary_key=True),
     Column("nome", String(30)),
 )
metadata_obj.create_all(engine)
```

In [92]:
```
!echo "\dt\q" | PGPASSWORD=papapa psql -U postgres italy
```

```
           List of relations
 Schema |     Name     | Type  |  Owner
--------+--------------+-------+----------
 public | provincia    | table | postgres
 public | regione      | table | postgres
 public | user_account | table | postgres
(3 rows)
```

In [93]:
```
!echo "\d user_account\q" | PGPASSWORD=papapa psql -U postgres italy
```

```
                                    Table "public.user_account"
  Column  |         Type          | Collation | Nullable |                 D
efault
----------+-----------------------+-----------+----------
+----------------------------------------
 id       | integer               |           | not null | nextval('user_acc
ount_id_seq'::regclass)
 name     | character varying(30) |           |          |
 fullname | character varying     |           |          |
Indexes:
    "user_account_pkey" PRIMARY KEY, btree (id)
```

In [94]:
```
!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 [95]:
```
!echo "\d regione\q" | PGPASSWORD=papapa psql -U postgres italy
```

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

---

psycopg2 only

In [104…
```
import psycopg2
conn = psycopg2.connect(database="italy",
                        host="localhost",
                        user="postgres",
```

```
                        password="papapa",
                        port="5432")
```

In [105…
```
cursor = conn.cursor()
```

In [106…
```
cursor.execute("INSERT INTO provincia (sigla, nome) VALUES('PU', 'Pesaro e U
```

In [107…
```
cursor.execute("COMMIT TRANSACTION")
```

In [108…
```
!echo "SELECT * FROM provincia;\q" | PGPASSWORD=papapa psql -U postgres ital
```

```
 sigla |      nome       | residenti
-------+-----------------+-----------
 PU    | Pesaro e Urbino |
(1 row)
```

In [109…
```
!echo "DELETE FROM provincia;\q" | PGPASSWORD=papapa psql -U postgres italy
```

```
DELETE 1
```

In [110…
```
import psycopg2
conn = psycopg2.connect(database="italy",
                        host="localhost",
                        user="postgres",
                        password="papapa",
                        port="5432")
from csv import reader
infile = open("province.csv")
csvReader = reader(infile)
header = True
for riga in csvReader :
    if header :
        header = False
        continue
    sigla = riga[0]
    nome = riga[1].replace("'", "''")
    residenti = int(riga[2])
    cursor.execute(f"INSERT INTO provincia (sigla, nome, residenti) VALUES('
infile.close()
cursor.execute("COMMIT TRANSACTION")
```

In [ ]:
```
import psycopg2
conn = psycopg2.connect(database="italy",
                        host="localhost",
                        user="postgres",
                        password="papapa",
                        port="5432")
from csv import reader
infile = open("regioni.csv")
csvReader = reader(infile)
header = True
for riga in csvReader :
    if header :
        header = False
        continue
```

```
    sigla = riga[0]
    nome = riga[1].replace("'", "''")
    cursor.execute(f"INSERT INTO regione (sigla, nome) VALUES('{sigla}', '{n
infile.close()
cursor.execute("COMMIT TRANSACTION")
```

In [111…
```
!echo "SELECT count(*) FROM provincia;\q" | PGPASSWORD=papapa psql -U postgr
```

```
 count
-------
   110
(1 row)
```

In [112…
```
!echo "SELECT count(*) FROM regione;\q" | PGPASSWORD=papapa psql -U postgres
```

```
 count
-------
   107
(1 row)
```

In [115…
```
!echo "SELECT r.sigla FROM regione AS r WHERE r.sigla NOT IN (SELECT p.sigla
```

```
 sigla
-------
 SU
(1 row)
```

In [116…
```
!echo "SELECT p.sigla FROM provincia AS p WHERE p.sigla NOT IN (SELECT r.sig
```

```
 sigla
-------
 OT
 CI
 VS
 OG
(4 rows)
```

esempio da GitHub

- si tratta di un **Jupyter Notebook** (*.ipynb*)
- link

Suits

- diamonds (♦)
- clubs (♣)
- hearts (♥)
- spades (♠)