Programación con Python

POR VÍCTOR FUENTES

Análisis de texto

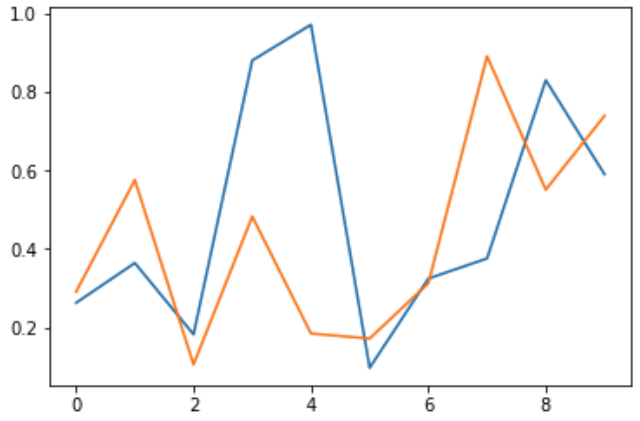
```python
texto = ('POR ÚLTIMO, LA FORMA MÁS RECOMENDABLE Y PYTHÓNICA, PERO MÁS COMPLEJA, SERÍA USAR MATPLOTLIB MEDIANTE LA INTERFAZ O

t = texto.replace('.',' ').replace(',',' ')
print(t)
print()

t = texto.lower()
print(t)
print()

for vocal in ('AEIOU'):
    print(vocal, ':', texto.count(vocal))
print()

max_veces = 0
max_letra = ""
for l in ('mnlsr') :
    print (l, ":", t.count(l))
    if t.count(l) > max_veces :
        max_veces = t.count(l)
        max_letra = l
print('consonante mas usada', max_letra, max_veces)
print()

palabras = t.split(" ")
print()


for palabra in palabras :
    if len(palabra)>6 and palabra[0] not in ('aeiou') :
        print(palabra)
print()

text = input("Entre texto: ")
```

```
POR ÚLTIMO  LA FORMA MÁS RECOMENDABLE Y PYTHÓNICA  PERO MÁS COMPLEJA  SERÍA USAR MATPLOTLIB MEDIANTE LA INTERFAZ ORIENTADA A OB
JETOS  CUANDO SE PROGRAMA CON MATPLOTLIB  NO MIENTRAS SE TRABAJA INTERACTIVAMENTE  ESTA ES LA FORMA QUE PERMITE TENER MÁS CONTR
OL SOBRE EL CÓDIGO  QUIZÁ VEAMOS ESTO EN EL FUTURO SI OS ANIMÁIS A ESCRIBIR SOBRE ELLO

por último, la forma más recomendable y pythónica, pero más compleja, sería usar matplotlib mediante la interfaz orientada a ob
jetos. cuando se programa con matplotlib, no mientras se trabaja interactivamente, esta es la forma que permite tener más contr
ol sobre el código. quizá veamos esto en el futuro si os animáis a escribir sobre ello.

A : 30
E : 33
I : 18
O : 26
U : 6

m : 17
n : 15
l : 15
s : 19
r : 22
consonante mas usada r 22


último,
recomendable
pythónica,
compleja,
matplotlib
mediante
programa
matplotlib,
mientras
trabaja
permite
control
código.

Entre texto: por ultimo
```

MATPLOTLIB, NUMPY, PLOTLY

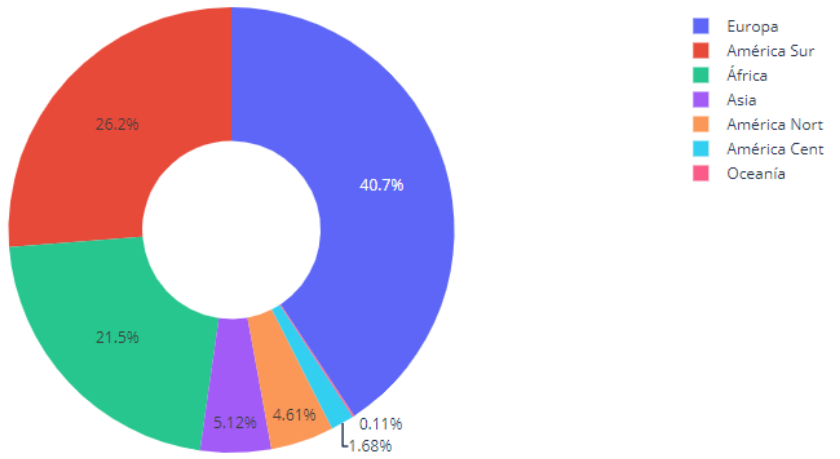
LIBRERIA MATHPLOTLIB amb NUMPY

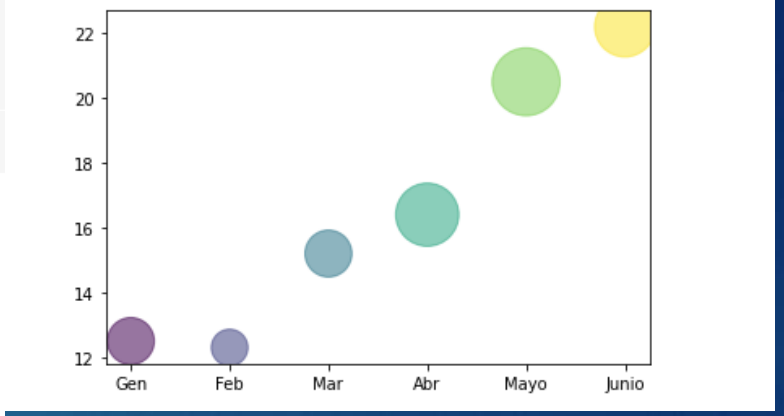
```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  plt.plot(np.random.rand(10))
4  plt.plot(np.random.rand(10))
5  plt.show()
```




```
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  meses = [ 'Gen', 'Feb', 'Mar', 'Abr', 'Mayo', 'Junio']
6  temperaturas = [12.5, 12.3, 15.2, 16.4, 20.5, 22.2]
7  plujamm = [335, 203, 334, 604, 700, 555]
8
9  s = [x*2.5 for x in plujamm]
10
11 colors = [1,2,3,4,5,6]
12 plt.scatter(meses, temperaturas, c=colors, s=s , alpha = 0.5)
13 plt.xlabel = 'meses'
14 plt.ylabel = 'temperaturas'
15 plt.show()
16
17
```

```
1  import plotly.graph_objects as go
2
3  labels = ['Europa', 'África', 'América Nort', 'América Cent', 'América Sur','Asia', 'Oceanía']
4  values = [round((x + y)/2, 2) for x, y in zip(hombres, mujeres)]
5
6  fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.4)])
7  fig.show()
8
```





Clases y herencia

```python
class Carrito:

    def __init__(self, lista = []):
        self.lista = lista

    def agregar_al_carro(self, producto):
            self.lista.append(producto)



    def eliminar_del_carro(self, referencia):
        for p in self.lista:
            if p.referencia == referencia:
                self.lista.remove(p)
                break

    def mostrar_carro(self):
        print('--------------carrito---------')
        for p in self.lista:
            print(p)


    def total_compra(self):
        total = 0
        for p in self.lista:
            total += p.pvp
        print('Total carrito', total)



carrito = Carrito([])
carrito.mostrar_carro()

carrito.agregar_al_carro(t)
carrito.mostrar_carro()
carrito.agregar_al_carro(b)
carrito.mostrar_carro()

carrito.eliminar_del_carro(t)
# carrito.mostrar_carro()

carrito.total_compra()
```

```
--------------carrito---------
--------------carrito---------
REFERENCIA        45WD
NOMBRE            Bajera
PVP               12.5
DESCRIPCIÓN       1unit
color             blanco
material                  algodon

--------------carrito---------
REFERENCIA        45WD
NOMBRE            Bajera
PVP               12.5
DESCRIPCIÓN       1unit
color             blanco
material                  algodon

REFERENCIA        56J
NOMBRE            Taladro
PVP               35.5
DESCRIPCIÓN       percutor
medida            30
peso              1.5

Total carrito 48.0
```

Librería Pandas

```python
import pandas as pd

dades = {'Nom': ['Sònia', 'Laura', 'David', 'Rosa', 'Sam'],
         'Dept' : ['PROD', 'ADMIN', 'MANT', 'ADMIN', 'PROD'],
         'DiesV' : [32, 55, 20, 43, 30]}

df = pd.DataFrame(dades)
df = df.append({'Nom':'Sara', 'Dept':'ADMIN', 'DiesV': 3,}, ignore_index = True)
df = df.append({'Nom':'Joan', 'Dept':'MANT', 'DiesV': 5,}, ignore_index = True)
df = df.append({'Nom':'Paco', 'Dept':'PROD', 'DiesV': 7,}, ignore_index = True)
df = df.append({'Nom':'Susan', 'Dept':'VENT', 'DiesV': 9,}, ignore_index = True)
df = df.sort_values(['Dept', 'DiesV'])
print(df)
```

	Nom	Dept	DiesV
5	Sara	ADMIN	3
3	Rosa	ADMIN	43
1	Laura	ADMIN	55
6	Joan	MANT	5
2	David	MANT	20
7	Paco	PROD	7
4	Sam	PROD	30
0	Sònia	PROD	32
8	Susan	VENT	9

```python
import pandas as pd
ventas = pd.DataFrame({"A":[41,32,56,18],
                       "B":[17,54,6,78],
                       "C":[12,13,16,18] },
                      index = ["Gen", "Feb", "Mar", "Abr"])
print (ventas)
print("SUMA")
print(ventas.sum(axis=1))
print(ventas.describe())
```

```
      A   B   C
Gen  41  17  12
Feb  32  54  13
Mar  56   6  16
Abr  18  78  18
SUMA
Gen     70
Feb     99
Mar     78
Abr    114
dtype: int64
                A          B          C
count   4.000000   4.000000   4.000000
mean   36.750000  38.750000  14.750000
std    15.945219  33.260337   2.753785
min    18.000000   6.000000  12.000000
25%    28.500000  14.250000  12.750000
50%    36.500000  35.500000  14.500000
75%    44.750000  60.000000  16.500000
max    56.000000  78.000000  18.000000
```

```python
import pandas as pd
import numpy as np

df = pd.read_csv('https://raw.githubusercontent.com/asalber/manual-python/master/datos/colesterol.csv')
df.info()
print(df)

df.dropna()
df.sort_values(by='colesterol', ascending=False).dropna()
```

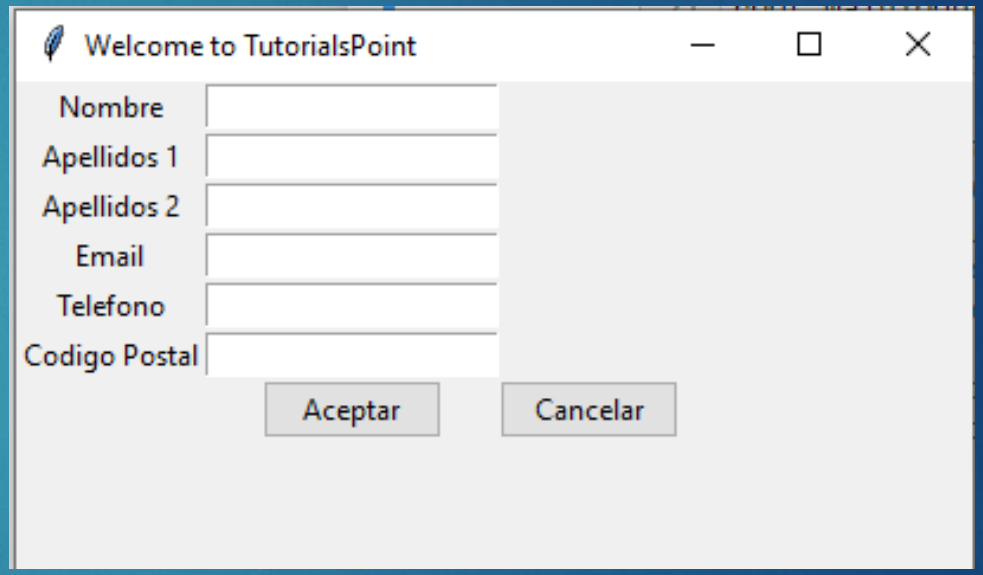
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   nombre      14 non-null     object
 1   edad        14 non-null     int64
 2   sexo        14 non-null     object
 3   peso        13 non-null     float64
 4   altura      14 non-null     float64
 5   colesterol  13 non-null     float64
dtypes: float64(3), int64(1), object(2)
memory usage: 800.0+ bytes
                          nombre  edad sexo   peso  altura  colesterol
0       José Luis Martínez Izquierdo    18    H   85.0    1.79       182.0
1                  Rosa Díaz Díaz    32    M   65.0    1.73       232.0
2             Javier García Sánchez    24    H    NaN    1.81       191.0
3               Carmen López Pinzón    35    M   65.0    1.70       200.0
4              Marisa López Collado    46    M   51.0    1.58       148.0
5                 Antonio Ruiz Cruz    68    H   66.0    1.74       249.0
6           Antonio Fernández Ocaña    51    H   62.0    1.72       276.0
7             Pilar Martín González    22    M   60.0    1.66         NaN
8              Pedro Gálvez Tenorio    35    H   90.0    1.94       241.0
9           Santiago Reillo Manzano    46    H   75.0    1.85       280.0
10            Macarena Álvarez Luna    53    M   55.0    1.62       262.0
11       José María de la Guía Sanz    58    H   78.0    1.87       198.0
12  Miguel Angel Cuadrado Gutiérrez    27    H  109.0    1.98       210.0
13            Carolina Rubio Moreno    20    M   61.0    1.77       194.0
```

Out[1]:

	nombre	edad	sexo	peso	altura	colesterol
9	Santiago Reillo Manzano	46	H	75.0	1.85	280.0
6	Antonio Fernández Ocaña	51	H	62.0	1.72	276.0
10	Macarena Álvarez Luna	53	M	55.0	1.62	262.0
5	Antonio Ruiz Cruz	68	H	66.0	1.74	249.0
8	Pedro Gálvez Tenorio	35	H	90.0	1.94	241.0
1	Rosa Díaz Díaz	32	M	65.0	1.73	232.0
12	Miguel Angel Cuadrado Gutiérrez	27	H	109.0	1.98	210.0
3	Carmen López Pinzón	35	M	65.0	1.70	200.0
11	José María de la Guía Sanz	58	H	78.0	1.87	198.0
13	Carolina Rubio Moreno	20	M	61.0	1.77	194.0
0	José Luis Martínez Izquierdo	18	H	85.0	1.79	182.0
4	Marisa López Collado	46	M	51.0	1.58	148.0

Tkinter

```
8  from tkinter import *
9  from tkinter import ttk
10
11 window = Tk()
12
13 window.title("Welcome to TutorialsPoint")
14 window.geometry('400x200')
15 window.minsize(width=400, height=200)
16
17 frame = Frame(window)
18
19 Label(frame ,text = "Nombre").grid(row = 0,column = 0)
20 Label(frame ,text = "Apellidos 1").grid(row = 1,column = 0)
21 Label(frame ,text = "Apellidos 2").grid(row = 2,column = 0)
22 Label(frame ,text = "Email").grid(row = 3,column = 0)
23 Label(frame ,text = "Telefono").grid(row = 4,column = 0)
24 Label(frame ,text = "Codigo Postal").grid(row = 5,column = 0)
25
26 Entry(frame).grid(row = 0,column = 1)
27 Entry(frame).grid(row = 1,column = 1)
28 Entry(frame).grid(row = 2,column = 1)
29 Entry(frame).grid(row = 3,column = 1)
30 Entry(frame).grid(row = 4,column = 1)
31 Entry(frame).grid(row = 5,column = 1)
32
33 btn = ttk.Button(frame ,text="Aceptar").grid(row=6,column=1)
34 frame.pack(anchor=NW, expand=1)
35
36 btn = ttk.Button(frame ,text="Cancelar").grid(row=6,column=2)
37 frame.pack(anchor=NW, expand=1)
38
39 window.mainloop()
```



JSON,XML

```xml
import xml.etree.ElementTree

response = requests.get("https://www.washingtonpost.com/arcio/news-sitemap/")

string_xml = response.content
tree = xml.etree.ElementTree.fromstring(string_xml)
xml.etree.ElementTree.dump(tree)
```

```
import requests
import json

response = requests.get('https://gorest.co.in/public/v2/users')

print(response.json())

for data in response.json():
    print(data['email'], data['name'])

```

```
[{'id': 2568, 'name': 'Rajan Joshi', 'email': 'rajan_joshi@wisoky.io', 'gender': 'male
e': 'Prathamesh Patel IV', 'email': 'patel_iv_prathamesh@haley.com', 'gender': 'female
ame': 'Shwet Mehra', 'email': 'shwet_mehra@mcclure-marvin.biz', 'gender': 'male', 'sta
'Msgr. Shwet Kaul', 'email': 'shwet_msgr_kaul@damore.biz', 'gender': 'female', 'status
Bhattacharya', 'email': 'akula_bhattacharya@wiegand-wilkinson.name', 'gender': 'female
e': 'Bhaaswar Mahajan', 'email': 'bhaaswar_mahajan@stracke.info', 'gender': 'female',
'Smriti Gill', 'email': 'gill_smriti@mosciski.biz', 'gender': 'male', 'status': 'inact
rajan_joshi@wisoky.io Rajan Joshi
patel_iv_prathamesh@haley.com Prathamesh Patel IV
shwet_mehra@mcclure-marvin.biz Shwet Mehra
shwet_msgr_kaul@damore.biz Msgr. Shwet Kaul
akula_bhattacharya@wiegand-wilkinson.name Akula Bhattacharya
bhaaswar_mahajan@stracke.info Bhaaswar Mahajan
gill_smriti@mosciski.biz Smriti Gill
nayar_darshwana@connelly.co Darshwana Nayar
iv_sitara_kaniyar@zemlak-roob.biz Sitara Kaniyar IV
johar_eekalabya@bernhard.net Eekalabya Johar
krishnadasa_panicker@koepp.co Krishnadasa Panicker
```

```
import requests
import json

response = requests.get('https://gorest.co.in/public/v2/users')
# print(response.json())

for data in response.json():
    if data['gender'] == 'female'and data['status'] == 'active':
        print(data['name'],data['email'])


```

```
Tanushri Pilla tanushri_pilla@effertz.net
Nirbhay Achari nirbhay_achari@kerluke.co
Chandraketu Reddy chandraketu_reddy@haley.net
```

SQL

```python
import sqlite3

def conectar_usuarios():
    """ abre la conexion con la base de datos """

    db_name = "../dat/practica2.db"
    conexion = None

    try :
        conexion = sqlite3.connect(db_name)
    except Exception as e:
        print("Error en la base de datos: ", e)
    return (conexion)

def crear_tabla_usuarios():
    """ si no existe, crea la tabla de usuarios """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    # Ahora crearemos una tabla de usuarios con nombres, edades y emails
    sql = "CREATE TABLE IF NOT EXISTS usuarios " \
          "(codigo CHAR(6) PRIMARY KEY, nombre VARCHAR(100), activo CHAR(1), email VARCHAR(100), depto CHAR(3))"
    cursor.execute(sql)
    conexion.commit()               # Guardar cambios
    conexion.close()

def crear_usuarios(lista_usuarios):
    """ inserta un array de tuplas de usuarios en la base de datos """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    # Ahora utilizamos el método executemany() para insertar varios
    cursor.executemany("INSERT INTO usuarios VALUES (?,?,?,?,?)", lista_usuarios)
    conexion.commit()               # Guardar cambios
    conexion.close()
```

```python
def consultar_usuarios() :
    """ retorna una lista de tuplas de usuarios """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    # Recuperamos los registros de la tabla de usuarios
    cursor.execute("SELECT * FROM usuarios")

    # Recorremos todos los registros con fetchall
    # y los volcamos en una lista de usuarios
    personal = cursor.fetchall()
    conexion.close()
    return (personal)

def consultar_usuario(codigo) :
    """ retorna una lista de 1 tupla con el usuario encontrado """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    # Recuperamos los registros de la tabla de usuarios
    cursor.execute(f"SELECT * FROM usuarios WHERE codigo = '{codigo}'")

    # Recuperamos todos los registros y los volcamos en una lista de usuarios
    personal = cursor.fetchall()
    conexion.close()
    return (personal)

if __name__ == '__main__':
    crear_tabla_usuarios()
    eliminar_todo()
      # Crea datos de prueba

    crear_usuario (('0','Pruebas', 'S/N',  'pruebas@ejemplo.com', 'DEPTO'))
#      # Creamos una lista con varios usuarios (codigo, nombre, edad, mail)
#      # ========================================================================
    usuarios = [('1946', 'Mario Benet',   'S', 'mario@ejemplo.com', 'MANT'), \
                ('4622', 'Marta Artigas', 'N', '5589@telefonica.es', 'ADMIN'),\
                ('3717', 'Joan Piqué',    'S', 'joan@dosnoms.cat', 'ADMIN')]
```

```python
def crear_usuario(usuario):
    """ inserta una tupla de usuario en la base de datos """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    sql = "INSERT INTO usuarios(codigo, nombre, activo, email, depto) VALUES (?,?,?,?,?);"
    print(sql, usuario)
    cursor.execute (sql, usuario)
    conexion.commit()               # Guardar cambios
    conexion.close()

def eliminar_usuario(codigo):
    """ elimina un usuario """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    sql = f"DELETE FROM usuarios WHERE codigo = '{codigo}';"
    cursor.execute (sql)
    conexion.commit()               # Guardar cambios
    conexion.close()

def eliminar_todo():
    """ elimina todos los usuarios """

    conexion = conectar_usuarios()
    cursor = conexion.cursor()

    cursor.execute ("DELETE FROM usuarios")
    conexion.commit()               # Guardar cambios
    conexion.close()
```

```python
#      # ========================================================================
    crear_usuarios (usuarios)
    eliminar_usuario('0')

    for usuario in consultar_usuarios():
        print(usuario)
```

```
INSERT INTO usuarios(codigo, nombre, activo, email, depto) VALUES (?,?,?,?,?); ('0', 'Pruebas', 'S/N', 'pruebas@ejemplo.com',
'DEPTO')
('1946', 'Mario Benet', 'S', 'mario@ejemplo.com', 'MANT')
('4622', 'Marta Artigas', 'N', '5589@telefonica.es', 'ADMIN')
('3717', 'Joan Piqué', 'S', 'joan@dosnoms.cat', 'ADMIN')
```