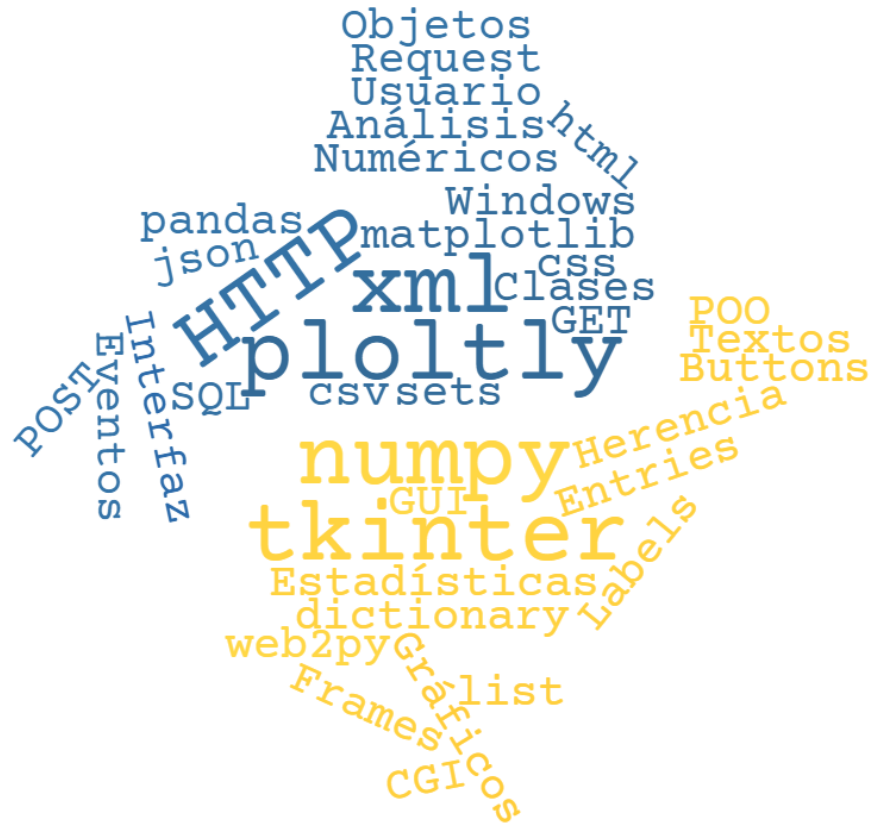


# Portfolio

## Programación en Python



30/03/2022-3/5/2022

110 HORAS

- **Análisis**

## Análisis numérico

```
In [33]: 1 print("abs=",abs(-8.7)) #8.7
2 print("round=",round(5.76543, 2)) #5.76
3 print("divmod=",divmod(8, 3)) #??
4 print("max=",max(12, -12.3, 18.9, -9.8)) #18.9
5 print("max=",min(12, -12.3, 18.9, -9.8)) #-12.3
```

```
abs= 8.7
round= 5.77
divmod= (2, 2)
max= 18.9
max= -12.3
```

## Textos Análisis

```
97 max_veces = 0
98 max_letra = ""
99 for m in ('mnlrs') :
100     print(m, ":", t.count(m))
101     if t.count(m) > max_veces :
102         max_veces = t.count(m)
103         max_letra = m
104 print ("consonante más usada", max_letra, max_veces)
105 print()
106
107 # 6. Crea una Lista con Las palabras del texto
108
109 print("Lista de palabras:")
110 print()
111
112 palabras = t.split(" ")
113 print(palabras)
114 print()
115 # 7. Imprime todas Las palabras de más de 6 letras que empiecen por consonante.
116
117 print("Palabras de mas de 6 letras con consonante:")
118 print()
119
120 #hacerlo recorriendo la coleccion de palabras:
121
122 for palabra in palabras:
123     if len(palabra)>6 and palabra[0] not in ("aeiou") :
124         print(palabra)
125 print()
126 # 8. Crea otra versión del programa que pida el texto por pantalla
127
128 print("Pedir texto por pantalla:")
129 print()
130
```

```
1 #Exemple
2 correu = input("e-mail : ")
3 if "@" not in correu :
4     print("Error: e-mail invàlid, \
5 ha de contenir '@' ")
6
7 if "." not in correu :
8     print("Error: e-mail invàlid, \
9 ha de contenir '.' ")
```

```
e-mail : .
Error: e-mail invàlid, ha de contenir '@'
```

## Diccionarios

```
1 estats={'ca':'California','ok':'Oklahoma', 'nj':'New Jersey', 'tx':'Texas'}
2
3 estat=input("Codi estat:")
4
5 if estat in estats.keys() :
6     print("El nom del estat es ", estats[estat])
7 else:
8     print("No tinc aquest codi")
9
10
```

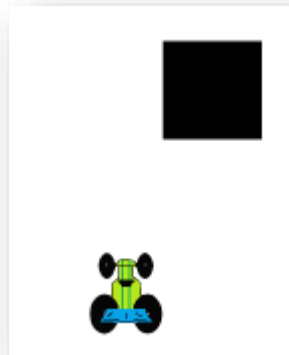
```
Codi estat:ll
No tinc aquest codi
```

- Programación orientada a objetos

## Clases

```
23 class Coche(object):
24     """ Coche, se desplaza derecha e izquierda. """
25
26     def __init__(self, juego):
27         self.ancho = 73 # Ancho del coche
28         self.x_acelera = 0 # parámetros para desplazamiento
29         self.velocidad = 0 # y velocidad
30
31         self.x = juego.ancho * 0.45 # parámetros de posición
32         self.y = juego.alto * 0.8
33         self.img = pygame.image.load('.\\img\\racecar.png')
34
35     def dibuja_y_mueve(self, juego):
36         juego.ventana.fill(WHITE) # Blanquea la pantalla
37         self.x += self.x_acelera # recalcula la coordenada x
38         juego.ventana.blit(self.img, (self.x, self.y)) # Dibuja el coche
39
40     def acelerar(self, valor):
41         self.x_acelera = valor
42
43     def calcula_colision(self, caja):
44         if self.y < (caja.y + caja.alto):
45             if (self.x > caja.x and self.x < (caja.x + caja.ancho)) or \
46                 ((self.x + self.ancho > caja.x) and (self.x + self.ancho) < (caja.x + caja.ancho)) :
47                 return True
48         return False
```

## Creación de juegos

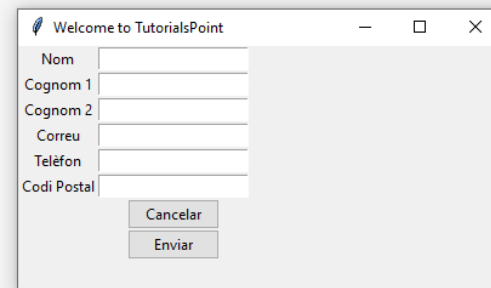


## Herencia

```
85 #-----
86 class Libro(Producto):
87
88     def __init__(self, referencia, nombre, pvp, descripcion, isbn, autor):
89         super().__init__(referencia, nombre, pvp, descripcion)
90         self.isbn = isbn
91         self.autor = autor
92
93     def __str__(self):
94         return super().__str__() + \
95             f"ISBN\t\t {self.isbn}\n" \
96             f"AUTOR\t\t {self.autor}\n"
97
98 l = Libro("k888", "El arte de ...", 25.20, "Este libro esta bien", 'ISBN7373
99
100 print(l)
101
102 #----- Definición de funciones
103
104 p1= Libro(Producto)
105 p2= Alimento(Producto)
106 p3= Textil(Producto)
107
108 carrito = Productco([p1,p2])
109 carrito.listar()
110
111 carrito.agrega_al_carro(Producto())
112 carrito.agrega_al_carro(Producto())
113 carrito.elimina_del_carro(Referencia)
114 carrito.mostrar_carro()
115 carrito.total_compra()
116
```

- **GUI Interfaz Usuario:**

```
15 from tkinter import *
16 from tkinter import ttk
17
18 root = Tk()
19
20 root.title("Welcome to Tutorialspoint")
21
22 root.geometry('400x200')
23 root.minsize(width=400, height=200)
24
25 frame = Frame(root)
26 Label(frame ,text = "Nom").grid(row = 0,column = 0)
27 Label(frame ,text = "Cognom 1").grid(row = 1,column = 0)
28 Label(frame ,text = "Cognom 2").grid(row = 2,column = 0)
29 Label(frame ,text = "Correu").grid(row = 3,column = 0)
30 Label(frame ,text = "Telèfon").grid(row = 4,column = 0)
31 Label(frame ,text = "Codi Postal").grid(row = 5,column = 0)
32
33 Entry(frame).grid(row = 0,column = 1)
34 Entry(frame).grid(row = 1,column = 1)
35 Entry(frame).grid(row = 2,column = 1)
36 Entry(frame).grid(row = 3,column = 1)
37 Entry(frame).grid(row = 4,column = 1)
38 Entry(frame).grid(row = 5,column = 1)
39
40 btn = ttk.Button(frame ,text="Cancelar").grid(row=6,column=1)
41 btn = ttk.Button(frame ,text="Enviar").grid(row=7,column=1)
42
43 frame.pack(anchor=NW, expand=1)
44 root.mainloop()
```



The screenshot shows a Tkinter window titled "Welcome to Tutorialspoint". Inside the window, there is a form with six labels and corresponding entry fields: "Nom", "Cognom 1", "Cognom 2", "Correu", "Telèfon", and "Codi Postal". Below these entry fields, there are two buttons: "Cancelar" and "Enviar". The window has a standard macOS-style title bar with minimize, maximize, and close buttons.

## Tkinter

- ✓ Windows,
- ✓ Frames,
- ✓ Labels,
- ✓ Entries,
- ✓ Buttons

- Creación de páginas web con web2py  
CGI Request HTTP GET POST

The screenshot shows the web2py web framework interface. On the left, there's a sidebar with the 'WEB2PY' logo and version information (Version 2.22.3-stable+timestamp.2022.02.15.15.14.38). Below this, there are sections for 'Server IP' and 'Server Port'. The 'Server IP' section lists several options: Local (IPv4) (127.0.0.1), Local (IPv6) (::1), Public (fe80::4963:12a4:7880:5072), Public (10.199.160.127), and Public (0.0.0.0). The 'Server Port' is set to 8000. There's a 'Choose Password' field with four asterisks. Below these fields are 'start server' and 'stop server' buttons. A large black arrow points from the 'start server' button to the main interface. The main interface has a dark header with 'web2py™ interfaz administrativa'. Below the header, there's a section titled 'Editar aplicación "cursos"'. This section contains a search bar, a 'contraer/expandir todo' button, and several tabs: 'modelos', 'controladores', 'vistas', 'lenguajes', 'módulos', and 'archivos privados'. The 'modelos' tab is selected, showing a list of models: 'administración base de datos', 'sql.log', and 'graficación del modelo'. Below this, there's a list of files to edit: '0.py', 'db.py', 'db\_wizard.py' (with a description 'definir tablas t\_cursos, t\_cursos\_archive, t\_inscripciones, t\_inscripciones\_archive'), 'db\_wizard\_populate.py', and 'menu.py'. A 'Crear' button is at the bottom of this list. Below the 'modelos' section is the 'Controladores' section, which has 'probar' and 'crontab' buttons. It also shows a list of files to edit: 'appadmin.py' and 'default.py'. Below this, there's a list of actions: 'expone ccache, csv, d3\_graph\_model, download, hooks, index, insert, manage, select, state, update' for 'appadmin.py' and 'expone call, cursos\_manage, download, error, hola, index, inscripciones\_manage, listado, user' for 'default.py'.

```
1.  ### ve prepend t_ to tablename and f_ to fieldnames for disambiguity
2.
3.
4.  #####
5.  db.define_table('t_cursos',
6.      Field('f_codi_curs', type='string',
7.          label=T('Codi Curs')),
8.      Field('f_nom', type='string',
9.          label=T('Nom')),
10.     Field('f_estat', type='integer',
11.         label=T('Estat')),
12.     Field('f_data_inici', type='date',
13.         label=T('Data Inici')),
14.     Field('f_data_fi', type='date',
15.         label=T('Data Fi')),
16.     Field('f_hores', type='integer',
17.         label=T('Hores')),
18.     auth.signature,
19.     format='% (f_codi_curs)s',
20.     migrate=settings.migrate)
```

```
1  #!/usr/bin/env python
2  ## coding: utf-8
3  ### required - do not delete
4  def user(): return dict(form=auth())
5  def download(): return response.download(request, db)
6  def call(): return service()
7  ### end requires
8  def index():
9      return dict()
10
11  def error():
12      return dict()
13
14  def jocs_manage():
15      form = SQLFORM.smartgrid(db.t_jocs, onupdate=auth.archive)
16      return locals()
17
18  def proveidors_manage():
19      form = SQLFORM.smartgrid(db.t_proveidors, onupdate=auth.archive)
20      return locals()
21
```

# • Análisis de datos

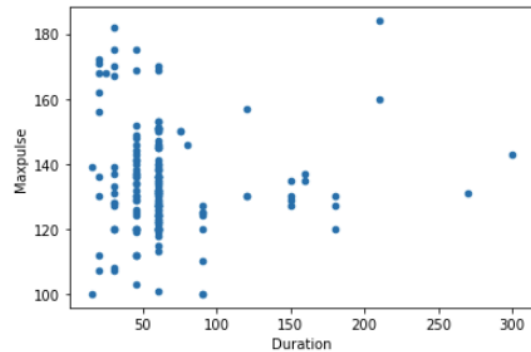
```
5 import numpy as np
6 import pandas as pd
```

```
In [2]: 1 # importando el dataset a un Dataframe de Pandas
2 ONG_data = pd.read_csv('./dat/LEARNING2.csv', header=0, sep=";")
3 ONG_data
4
```

```
Out[2]:
```

	ODATEDW	OSOURCE	TCODE	STATE	ZIP	DOB	NOEXCH	DOMAIN	CLUSTER	AGE	AGEFLAG	HOMEOWNR	NUM
0	8901	GRI	0	IL	61081	3712	0	T2	36.0	60.0			
1	9401	NWN	0	LA	70611	0	0	T2	39.0	NaN			U
2	9401	MSD	1	TN	37127-	3211	0	T1	35.0	65.0	I		
3	8901	ENQ	0	MN	56475	2603	0	R3	51.0	72.0			H
4	9201	HCC	1	LA	70791	0	0	T2	40.0	NaN			
5	9301	USB	1	UT	84720	2709	0	T1	35.0	70.0	E		H
6	9401	FRC	1	CA	90056	0	0	U1	2.0	NaN			H

```
7 import pandas as pd
8
9 import matplotlib.pyplot as plt
10
11 df = pd.read_csv('./dat/dataw3.csv')
12
13 df.plot(kind = 'scatter', x = 'Duration', y = 'Maxpulse')
14
15 #plt.show() es necesario comentarlo para guardar el archivo
16
17 plt.savefig('./dat/actividad3.jpg')
18
```

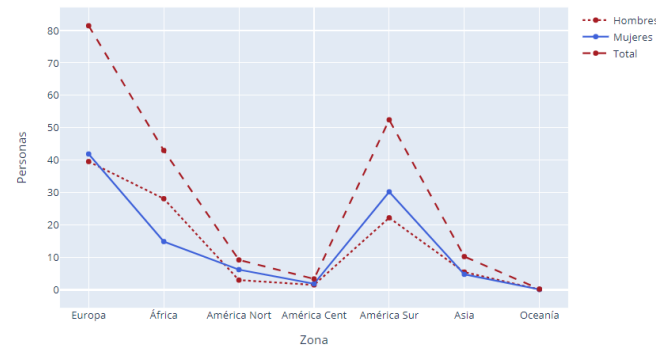


## Pandas

- numpy
- matplotlib

## Plotly

Residentes por Zona origen datos 2001



# SQL