## **PYTHON EJEMPLOS RESUELTOS**

<https://docs.python.org/2/>

<https://docs.python.org/3/>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**01 Media aritmética**

Escriba un programa que pida dos números y que escriba su media aritmética.

Se recuerda que la media aritmética de dos números es la suma de ambos números dividida por 2.

CÁLCULO DE LA MEDIA DE DOS NÚMEROS  
Escriba un número: 100  
Escriba otro número: 5  
La media aritmética de 100.0 y 5.0 es 52.5

**#!/usr/bin/env pyhton**

**# \_\_\*\_\_ coding:utf-8 \_\_\*\_\_**

**def main(): # Función principal**

**print("CÁLCULO DE LA MEDIA DE DOS NÚMEROS")**

**# Entrada de datos**

**numero\_1 = float(input("Escriba un número: "))**

**numero\_2 = float(input("Escriba otro número: "))**

**# Cálculos**

**media = (numero\_1 + numero\_2) / 2**

**# Mostrar resultados**

**print("La media aritmética de " + str(numero\_1) +" y "+str(numero\_2) +" es "+ str(media))**

**#otra forma**

**print("La media aritmética de " , numero\_1 , " y " , numero\_2 , " es " , media)**

**#otra forma**

**print("La media aritmética de {0} y {1} es {2}".format(numero\_1,numero\_2,media))**

**#otra forma más antigua**

**print("La media aritmética de %d y %d es %d" % (numero\_1,numero\_2,media))**

**#otra forma más moderna solo válida a partir de python 3.6 f**

**#print(f"La media aritmética de {numero\_1} y {numero\_2} es {media}")**

**if \_\_name\_\_ == "\_\_main\_\_": # Condición para saber si el programa es ejecutado o importado por otro**

**main()**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**02 Índice de masa corporal**

Escriba un programa que pida el peso (en kilogramos) y la altura (en metros) de una persona y que calcule su índice de masa corporal (imc).

Se recuerda que el imc se calcula con la fórmula imc = peso / altura 2

CÁLCULO DEL ÍNDICE DE MASA CORPORAL (IMC)  
¿Cuánto pesa? 78  
¿Cuánto mide en metros? 1.73  
Su imc es 26.1  
Un ímc muy alto indica obesidad. Los valores normales de imc están entre 20 y 25,  
pero esos límites dependen de la edad, del sexo, de la constitución física, etc.

**#!/usr/bin/env pyhton**

**# \_\_\*\_\_ coding:utf-8 \_\_\*\_\_**

**def main():  
 print("CÁLCULO DEL ÍNDICE DE MASA CORPORAL (IMC)")**

**#Entrada de datos  
 peso = float(input("¿Cuánto pesa? "))  
 altura = float(input("¿Cuánto mide en metros? "))**

**#Cálculo de resultados**

**imc = peso / altura \*\* 2**

**#Mostrar resultados  
 print("Su imc es " + str(round(imc, 1)))  
 print("Un ímc muy alto indica obesidad. Los valores normales de imc están entre 20 y 25,")  
 print("pero esos límites dependen de la edad, del sexo, de la constitución física, etc.")  
  
if \_\_name\_\_ == "\_\_main\_\_": # Condición para saber si el programa es ejecutado o importado por otro**

**main()**

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**03 Pies y pulgadas**

Escriba un programa que pida una distancia en pies y pulgadas y que escriba esa distancia en centímetros.

Se recuerda que un pie son doce pulgadas y una pulgada son 2,54 cm.

CONVERTIDOR DE PIES Y PULGADAS A CENTÍMETROS  
Escriba una cantidad de pies: 4  
Escriba una cantidad de pulgadas: 9  
4 pies y 9 pulgadas son 144.78 cm

**#!/usr/bin/env pyhton**

**# \_\_\*\_\_ coding:utf-8 \_\_\*\_\_**

**def main():**

**print("CONVERTIDOR DE PIES Y PULGADAS A CENTÍMETROS")**

**#Entrada de datos**

**pies = float(input("Escriba una cantidad de pies: "))**

**pulgadas = float(input("Escriba una cantidad de pulgadas: "))**

**#Cálculo de resultados**

**centimetros = (pies \* 12 + pulgadas) \* 2.54**

**#Mostrar resultados**

**print("{0} pies y {1} pulgadas son {2} cm".format(pies,pulgadas,centimetros))**

**if \_\_name\_\_ == "\_\_main\_\_":# Condición para saber si el programa es ejecutado o importado por otro**

**main()**

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**04 Comparar números**

Escriba un programa que pida dos números y que conteste cuál es el menor y cuál el mayor o que escriba que son iguales.

COMPARADOR DE NÚMEROS  
Escriba un número: 23  
Escriba otro número: 14.5  
Menor: 14.5; Mayor: 23.0

COMPARADOR DE NÚMEROS  
Escriba un número: 5.0  
Escriba otro número: 5  
Los dos números son iguales.

**#!/usr/bin/env pyhton**

**# \_\_\*\_\_ coding:utf-8 \_\_\*\_\_**

**def main():**

**print("COMPARADOR DE NÚMEROS")**

**numero\_1 = float(input("Escriba un número: "))**

**numero\_2 = float(input("Escriba otro número: "))**

**if numero\_1 > numero\_2:**

**print("Menor: {1} Mayor: {0}".format(numero\_1,numero\_2))**

**elif numero\_1 < numero\_2:**

**print("Menor: {0} Mayor: {1}".format(numero\_1,numero\_2))**

**else:**

**print("Los dos números son iguales.")**

**if \_\_name\_\_ == "\_\_main\_\_":# Condición para saber si el programa es ejecutado o importado por otro**

**main()**

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**05 Múltiplos**

Escriba un programa que pida dos números enteros y que escriba si el mayor es múltiplo del menor.

COMPARADOR DE MÚLTIPLOS  
Escriba un número: 6  
Escriba otro número: 48  
48 es múltiplo de 6.

COMPARADOR DE MÚLTIPLOS  
Escriba un número: 6

Escriba otro número: 49  
49 no es múltiplo de 6.

#!/usr/bin/env pyhton

# \_\_\*\_\_ coding:utf-8 \_\_\*\_\_

def main():

print("COMPARADOR DE MÚLTIPLOS")

numero\_1 = int(input("Escriba un número: "))

numero\_2 = int(input("Escriba otro número: "))

if numero\_1 >= numero\_2 and numero\_1 % numero\_2 != 0:

print("{} no es múltiplo de {}.".format(numero\_1,numero\_2))

elif numero\_1 >= numero\_2 and numero\_1 % numero\_2 == 0:

print("{} es múltiplo de {}.".format(numero\_1,numero\_2))

elif numero\_1 < numero\_2 and numero\_2 % numero\_1 != 0:

print("{1} no es múltiplo de {0}.".format(numero\_1,numero\_2))

else:

print("{1} es múltiplo de {0}.".format(numero\_1,numero\_2))

if \_\_name\_\_ == "\_\_main\_\_":**# Condición para saber si el programa es ejecutado o importado por otro**

main()

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **06\_Dados**

Escriba un programa que simule un juego en el que dos jugadores (Pepe y Juan) tiran un dado. El que saque mayor puntuación, gana. Si la puntuación coincide, empatan.

**#!/usr/bin/env pyhton**

**# \_\_\*\_\_ coding:utf-8 \_\_\*\_\_**

**from** random **import** randrange  **# IMPORTACIÓN de una función de otro módulo**

def main():

print("JUEGO DE DADOS (1)")  
  
 a = randrange(1, 7)  
 b = randrange(1, 7)  
  
 print("Pepe ha sacado un {}.".format(a))  
 print("Juan ha sacado un {}.".format(b))  
  
 if a == b:  
 print("Han empatado.")  
 elif a > b:  
 print("Ha ganado Pepe.")  
 else:  
 print("Ha ganado Juan.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**07 FACTORIAL**

Calcular el factorial de un número mayor que cero.

FACTORIAL  
Escriba un número entero mayor que cero: 5  
El factorial de 5 es 120

#!/usr/bin/env pyhton

# \_\_\*\_\_ coding:utf-8 \_\_\*\_\_

def main():

print("FACTORIAL")  
 numero = int(input("Escriba un número entero mayor que cero: "))  
  
 if numero <= 0:  
 print("¡Le he pedido un número entero mayor que cero!")  
 else:  
 factorial = 1  
 for i in range(1, numero + 1):  
 factorial = factorial \* i  
 print("El factorial de {0} es {1}".format(numero,factorial))

if \_\_name\_\_ == "\_\_main\_\_":

main()

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# **PYTHON GUI EJEMPLOS TKINTER**

[**https://www.python-course.eu/python\_tkinter.php**](https://www.python-course.eu/python_tkinter.php)

[**http://effbot.org/tkinterbook/**](http://effbot.org/tkinterbook/)

[**https://www.tutorialspoint.com/python/python\_gui\_programming.htm**](https://www.tutorialspoint.com/python/python_gui_programming.htm)

[**https://wiki.python.org/moin/TkInter**](https://wiki.python.org/moin/TkInter) todos los recursos

[**https://tkdocs.com/tutorial/index.html**](https://tkdocs.com/tutorial/index.html)

[**https://www.datacamp.com/community/tutorials/gui-tkinter-python**](https://www.datacamp.com/community/tutorials/gui-tkinter-python)

[**http://www.scoberlin.de/content/media/http/informatik/tkinter/index.htm**](http://www.scoberlin.de/content/media/http/informatik/tkinter/index.htm)

[**https://docs.python.org/3/library/tk.html**](https://docs.python.org/3/library/tk.html)

[**https://docs.python.org/2.7/library/tk.html**](https://docs.python.org/2.7/library/tk.html)

<https://likegeeks.com/es/ejemplos-de-la-gui-de-python/>

<https://guia-tkinter.readthedocs.io/es/develop/index.html>

<http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html>

<http://gmendezm.blogspot.com/search/label/Todo>

<https://www.tutorialesprogramacionya.com/pythonya/index.php?inicio=60>

<https://www.programcreek.com/python/> #EXAMPLES

<https://stackabuse.com/python-gui-development-with-tkinter-part-3/>

<http://zetcode.com/>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**01\_Ventana\_Principal\_y\_Label.py** **Ventana principal y Label**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

def main():

#Crear y configurar ventana principal

**window = Tk()**

window.title("Welcome to Tkinter")

window.geometry("400x400+0+0") # ancho x alto + offset posicion x + offset posicion y

window.update\_idletasks() # actualizar valor width y height

ancho = window.winfo\_width() # conocer el ancho de la ventana

alto = window.winfo\_height() # conocer el alto de la ventana

ancho\_pantalla = window.winfo\_screenwidth() # conocer la resolución de la pantalla

alto\_pantalla = window.winfo\_screenwidth() # conocer la resolución de la pantalla

print(ancho,alto,ancho\_pantalla,alto\_pantalla ) # print es una buena opción para comprobar el valor de ciertos parámetros

#Widgets y Contenidos

**lbl = Label(window, text = "Hola mundo")**

lbl.config(font = ("Courier",44)) # sinónimo lbl.configure()

lbl.config(fg = "#0000FF") # otra forma de modificar una propiedad

# lbl[‘fg’] = “#0000FF”

# NombreWidget[‘propiedad’] = valor

# para leer una propiedad variable = NombreWidget.cget(‘propiedad’)

**lbl.pack()** #Colocar y Mostrar widget

**'''Opciones de posicionamiento:**

**lbl.grid( column = 0, row = 0) TABLA**

**lbl.pack( ) BLOQUES**

**lbl.place( x = 150,y = 200) POSICIÓN x,y**

**'''**

**window.mainloop()**  #Bucle principal de la ventana

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**02\_Button.py** **Button**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

import time # módulo time ver diferencia con " from time import \* "

def main():

#Crear y configurar ventana principal

**window = Tk()**

window.title("Botones")

window.geometry("400x400+0+0") # ancho x alto + posicion x + posicion y

#NOTA print(window.config()) el método config() o configure() sin parámetros

# devuelve una lista con todas la propiedades de un widget

#Widgets y Contenidos

**lbl = Label(window, text="PULSA")**

**lbl.pack()** #Colocar y Mostrar widget

def clicked():

if lbl.cget('text') == "PULSA": # cget('propiedad') leer el valor de una propiedad de un widget

lbl.configure(text = "PULSADO")

else:

lbl.configure(text = "PULSA")

**btn = Button**(window, text = "PULSA", cursor = 'hand1' , activebackground="#11ff11" , command = clicked)

**btn.pack()** #Colocar y Mostrar widget

def minimizando():

window.iconify() # minimizar ventana

time.sleep(2) # no hacer nada durante 2 segundos

window.deiconify() # desminimizar

**btn2 = Button**(window , text = 'Minimizar un rato' , command = minimizando)

**btn2.pack()** #Colocar y Mostrar widget

#Bucle principal de la ventana

**window.mainloop()**

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**03\_Entry.py**  **Entry**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

def main():

#Crear y configurar ventana principal

**window = Tk()**

window.title("Entry")

window.geometry('350x200')

#widgets

lbl = Label(window, text="¿Cómo te llamas?")

lbl.grid(column=0, row=0) #Colocar y Mostrar widget

txt = Entry(window,width=10) # ancho en caracteres

txt.grid(column=1, row=0) #Colocar y Mostrar widget

hola = Label(window , text = '') #Label vacío

hola.grid(column=3 , row=0) #Colocar y Mostrar widget

def clicked():

res = "Hola " + txt.get() # Leer valor introducido en Entry GET()

hola.configure(text = res) # Modificar Label hola

btn = Button(window, text="Aceptar", command=clicked)

btn.grid(column=2, row=0) #Colocar y Mostrar widget

#Bucle principal de la ventana

window.mainloop()

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**03b\_Entry\_Media\_Aritmetica.py**  **Entry**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

def main():

#Crear y configurar ventana principal

window = Tk()

window.title("Entry")

#window.geometry('500x150') # Si queremos un tamaño determinado de ventana

#widgets

titulo = Label(window, text="Programa para el cálculo de la media aritmética de dos números")

titulo.grid(row=0,column=0)

#Primer número

lbl\_1 = Label(window, text="Primer número")

lbl\_1.grid(column=0, row=1)

numero\_1 = Entry(window,width=10) # Introducir Primer número

numero\_1.insert(0,0) # Colocar un cero como valor inicial en la primera posición

numero\_1.grid(column=1, row=1)

#Segundo número

lbl\_2 = Label(window, text="Segundo número")

lbl\_2.grid(column=0, row=2)

numero\_2 = Entry(window,width=10) # Introducir Segundo número

numero\_2.insert(0,0) # Colocar un cero como valor inicial en la primera posición

numero\_2.grid(column=1, row=2)

#Resultado

label\_resultado = Label(window, text='Resultado')

label\_resultado.grid(column=0, row=3)

resultado = Label(window, text='')

resultado.grid(column=1, row=3)

# FUNCION que realiza el cálculo, es llamada por el BOTON 'calcular' con 'command=media'

def media():

try: # Comprobar que los números introducidos son válidos y no son letras

n1= float( numero\_1.get() )

n2= float( numero\_2.get() )

solucion = float( n1 + n2 ) / 2

resultado.config(text = solucion) # Modificar el contenido del label 'resultado'

except:

resultado.config(text ='Debe introducir números válidos')

#Botón que llama a la función 'media' definida arriba

calcular = Button(window,text='Calcular',command=media)

calcular.grid(column=0, row=4)

window.mainloop() #Bucle principal de la ventana

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**04\_tkinter.py Text , Button , random**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

from random import randint # MÓDULO RANDOM-ALEATORIO

def main():

def roll(): # Función que lanza el dado

mitext.delete(0.0,END) # borrar contenido de mitext Text.

numero = str(randint(1,6)) # número del 1 al 6 en formato texto

mitext.insert(END,numero) # escribir número en mitext

# Ventana principal

window=Tk()

window.title('Text , Button, Random')

window.config(background = 'pink')

# Widgets

mitext=Text(window,width=1,height=1,font = ('Arial',44))

buttonA=Button(window,text='Tirar el dado',command=roll,font = ('Arial',22))

mitext.pack(pady=20 )

buttonA.pack(fill=X, padx=20,pady=20 ,ipadx=20,ipady=10)

window.mainloop()

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**05\_tkinter.py**  **Menús**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

def main():

#Crear y configurar ventana principal

window = Tk()

window.title("Menús")

def donothing():

nuevawin = Toplevel(window) # Toplevel es un widget que crea una pop-up window

button = Button(nuevawin, text="Do nothing button") #botón ubicado en la nueva ventana emergente

button.pack() # Colocar y visualizar

destruir = Button(nuevawin , text = 'Cerrar' , command = nuevawin.destroy ) # elimina ventana nuevawin

# el método widget.destroy() está accesible para cualquier widget

destruir.pack() # Colocar y visualizar

# Crear Nueva Barra de Menús

menubar = Menu(window)

# Crear nueva entrada de menú 'File'

filemenu = Menu(menubar, tearoff=0)

# filemenu = Menu(menubar)

# si tearoff no es 0 el menú se puede separar desde el primer elemento del menú - - - - - -

filemenu.add\_command(label="New", command=donothing)

filemenu.add\_command(label="Open", command=donothing)

filemenu.add\_command(label="Save", command=donothing)

filemenu.add\_command(label="Save as...", command=donothing)

filemenu.add\_command(label="Close", command=donothing)

filemenu.add\_separator() # línea separadora

filemenu.add\_command(label="Exit", command=window.quit)

menubar.add\_cascade(label="File", menu=filemenu) # Nombre del Desplegable superior del menú

# Crear nueva entrada de menú 'Edit'

editmenu = Menu(menubar) # ejemplo con tearof <> 0

editmenu.add\_command(label="Undo", command=donothing)

editmenu.add\_separator() # línea separadora

editmenu.add\_command(label="Cut", command=donothing)

editmenu.add\_command(label="Copy", command=donothing)

editmenu.add\_command(label="Paste", command=donothing)

editmenu.add\_command(label="Delete", command=donothing)

editmenu.add\_command(label="Select All", command=donothing)

menubar.add\_cascade(label="Edit", menu=editmenu) # Nombre del Desplegable superior del menú

# Crear nueva entrada de menú 'Help'

helpmenu = Menu(menubar, tearoff=0)

helpmenu.add\_command(label="Help Index", command=donothing)

helpmenu.add\_command(label="About...", command=donothing)

menubar.add\_cascade(label="Help", menu=helpmenu) # Nombre del Desplegable superior del menú

window.config(menu = menubar) # Asignar a la ventana principal el menú creado

window.mainloop() #Bucle principal de la ventana

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**06\_tkinter.py**  **messagebox y Message**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

from tkinter import messagebox ###############

#from tkinter import tkMessageBox

except:

from Tkinter import \* # python 2.7

from Tkinter import messagebox ###############

#from Tkinter import tkMessageBox

def main():

#Crear y configurar ventana principal

window = Tk()

window.title("Mensajes")

#FUNCIONES

def info():

messagebox.showinfo('INFORMACIÓN', 'Probando showinfo')

def warning():

messagebox.showwarning('ADVERTENCIA', 'Probando showwarning')

def error():

messagebox.showerror('ERROR', 'Probando showerror')

def question():

respuesta = messagebox.askquestion('PREGUNTA', 'Probando askquestion')

print(respuesta)

def yesno():

respuesta = messagebox.askyesno('Si o No', 'Probando askyesno')

print(respuesta)

def yesnocancel():

respuesta = messagebox.askyesnocancel('Si , No , Cancel', 'askyesnocancel')

print(respuesta)

def okcancel():

respuesta = messagebox.askokcancel('OK , Cancel', 'askokcancel')

print(respuesta)

def retrycancel():

respuesta = messagebox.askretrycancel('Retry , Cancel', 'askretrycancel')

print(respuesta)

def mensaje():

cartel = Message( window, text = 'Hola esto es un mensaje de tipo Message', relief=RAISED ) # relief borde

cartel.pack()

# Crear Nueva Barra de Menús

menubar = Menu(window)

# Crear nueva entrada de menú 'File'

mimenu = Menu(menubar, tearoff=0)

# mimenu = Menu(menubar)

# si tearoff no es 0 el menu se puede separar desde el primer elemento del menú - - - - - -

mimenu.add\_command(label="messagebox .showinfo", command=info)

mimenu.add\_command(label="messagebox .showwarning", command=warning)

mimenu.add\_command(label="messagebox .showerror", command=error)

mimenu.add\_command(label="messagebox .askquestion", command=question)

mimenu.add\_command(label="messagebox .askyesno", command=yesno)

mimenu.add\_command(label="messagebox .askyesnocancel", command=yesnocancel)

mimenu.add\_command(label="messagebox .askokcancel", command=okcancel)

mimenu.add\_command(label="messagebox .askretrycancel", command=retrycancel)

mimenu.add\_separator() # línea separadora

mimenu.add\_command(label="Message ", command=mensaje)

mimenu.add\_separator() # línea separadora

mimenu.add\_command(label="Exit", command=window.destroy ) # metodo quit()

menubar.add\_cascade(label="Mensajes", menu=mimenu) # Nombre del Desplegable superior del menú

window.config(menu = menubar) # Asignar a la ventana principal el menu creado

window.mainloop() #Bucle principal de la ventana

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**07\_tkinter.py**  **FileDialog , Imágenes , Scrollbar**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

from tkinter import filedialog ##########

except:

from Tkinter import \* # python 2.7

from Tkinter import filedialog ##########

'''

# from PIL import Image, ImageTk # para ver cualquier tipo de imagen, no solo GIF , PNG , pgm/ppm

# necesario instalar antes pip install Pillow permite también rotar imágenes

'''

def main():

#Crear y configurar ventana principal

window = Tk()

window.title("Filedialog,Imágenes,Scrollbar")

scrollbar = Scrollbar(window) ##### SCROLLBAR

scrollbar.pack( side = RIGHT, fill = Y )

# Scrollbar se puede usar con #### Text , Entry , Canvas y Listbox ####

mytexto = Text(window, wrap=WORD) # más adelante le unimos el scrollbar

def texto():# solo hay un mytexo

file = filedialog.askopenfilename(filetypes = (("Archivo texto","\*.txt"),("all files","\*.\*")))

print(file) # en file se guarda la ruta a el archivo elegido

# file = filedialog.askopenfilename()

# files = filedialog.askopenfilenames() # para múltiples archivos

# dir = filedialog.askdirectory() # para carpetas

# from os import path # podemos indicar el directorio inicial

# file = filedialog.askopenfilename(initialdir= path.dirname(\_\_file\_\_))

objeto\_archivo = open(file)

lectura = objeto\_archivo.read()

print(lectura)

mytexto.delete(1.0 , END) # borrar todo

mytexto.insert(END,lectura) # insertar texto

# attach mytexto to scrollbar

mytexto.config(yscrollcommand=scrollbar.set)

scrollbar.config(command=mytexto.yview)

mytexto.pack()

# para que las imágenes puedan tener scrollbar habría que ponerlas dentro de un Canvas

def imagengif(): # cada vez que ejecutamos se crea otro label

picture = filedialog.askopenfilename(filetypes = (("Imagen GIF","\*.gif"),("all files","\*.\*")))

print(picture) # en picture está la ruta a el archivo

myimage = **PhotoImage**(file = picture) # Objetos PhotoImage SOLO IMAGENES **GIF , PNG , pgm/ppm**

otrolabel = Label(window , text = 'IMAGEN' , **image = myimage**)

#otrolabel['text'] = 'IMAGEN'

#otrolabel['image'] = myimage

#### IMPORTANTE ######

**otrolabel.image = myimage** # keep a reference!, the program must keep an extra reference to the image object to visualize

otrolabel.pack()

'''

def imagen(): # necesaria libreria PIL Python Imaging Library pip install Pillow

picture = filedialog.askopenfilename()

print(picture) # en picture está la ruta a el archivo

myimage = Image.open(picture)

photo = ImageTk.PhotoImage(myimage)

label = Label(image=photo)

label.image = photo # keep a reference!

label.pack()

'''

# Crear Nueva Barra de Menús

menubar = Menu(window)

# Crear nueva entrada de menú 'File'

mimenu = Menu(menubar, tearoff=0)

mimenu.add\_command(label="Abrir texto...", command = texto )

mimenu.add\_separator() # línea separadora

mimenu.add\_command(label="Abrir gif... ", command = imagengif )

mimenu.add\_separator() # línea separadora

'''

mimenu.add\_command(label="Abrir imagen... ", command = imagen )

mimenu.add\_separator() # línea separadora

'''

mimenu.add\_command(label="Exit", command=window.destroy)

menubar.add\_cascade(label="Archivos", menu=mimenu) # Nombre del Desplegable superior del menú

window.config(menu = menubar) # Asignar a la ventana principal el menú

#Bucle principal de la ventana

window.mainloop()

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**07\_B\_tkinter.py** **Iconos tipo bitmap .xbm**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

def main():

#ventana principal

root = Tk()

root.geometry("800x800+0+0") # establecer ancho , alto , offsetx y offsety de la ventana principal

# ejemplo de iconos BITMAP asignado a un widget

# iconos disponibles "error", "gray75", "gray50", "gray25", "gray12", "hourglass", "info", "questhead", "question", and "warning"

# o usar iconos bitmaps tipo XBM de esta forma "@nombre.xbm" el archivo debe estar en la misma carpeta

# GIMP puede exprotar en formato .XBM con las dimensiones que se quieran (.xbm es formato texto )

# tambien se puede incrustar el codigo .xbm en el archivo .py como BITMAP=''' .... codigo XBM ..... '''

B1 = Button(root, text ="error", relief=RAISED , bitmap="error") # relief es el borde

B2 = Button(root, text ="hourglass", relief=RAISED , bitmap="hourglass")

B3 = Button(root, text ="info", relief=RAISED , bitmap="info")

B4 = Button(root, text ="question", relief=RAISED , bitmap="question")

#B5 = Button(root, text ="question", relief=RAISED , bitmap="@bluetooth.xbm") #XBM bluetooth.xbm en la misma carpeta

B1.place(x=20 ,y=20)

B2.place(x=120 ,y=20)

B3.place(x=220 ,y=20)

B4.place(x=320 ,y=20)

#B5.place(x=420 ,y=20)

# crear un objeto BitmapImage para asignarlo a un widget

# BitmapImage se puede crear desde archivo .xbm o texto

# un archivo .xbm solo puede tener dos colores

BITMAP = """

#define im\_width 32

#define im\_height 32

static char im\_bits[] = {

0xaf,0x6d,0xeb,0xd6,0x55,0xdb,0xb6,0x2f,

0xaf,0xaa,0x6a,0x6d,0x55,0x7b,0xd7,0x1b,

0xad,0xd6,0xb5,0xae,0xad,0x55,0x6f,0x05,

0xad,0xba,0xab,0xd6,0xaa,0xd5,0x5f,0x93,

0xad,0x76,0x7d,0x67,0x5a,0xd5,0xd7,0xa3,

0xad,0xbd,0xfe,0xea,0x5a,0xab,0x69,0xb3,

0xad,0x55,0xde,0xd8,0x2e,0x2b,0xb5,0x6a,

0x69,0x4b,0x3f,0xb4,0x9e,0x92,0xb5,0xed,

0xd5,0xca,0x9c,0xb4,0x5a,0xa1,0x2a,0x6d,

0xad,0x6c,0x5f,0xda,0x2c,0x91,0xbb,0xf6,

0xad,0xaa,0x96,0xaa,0x5a,0xca,0x9d,0xfe,

0x2c,0xa5,0x2a,0xd3,0x9a,0x8a,0x4f,0xfd,

0x2c,0x25,0x4a,0x6b,0x4d,0x45,0x9f,0xba,

0x1a,0xaa,0x7a,0xb5,0xaa,0x44,0x6b,0x5b,

0x1a,0x55,0xfd,0x5e,0x4e,0xa2,0x6b,0x59,

0x9a,0xa4,0xde,0x4a,0x4a,0xd2,0xf5,0xaa

};

"""

bitmap\_1 = BitmapImage(data=BITMAP , foreground="yellow", background="black") # desde texto incrustado

#bitmap\_2 = BitmapImage(file="bitmap.xbm" , foreground="white" , background="black") # desde ruta archivo

label = Label(root , image=bitmap\_1 )

label.image = bitmap\_1 # keep a reference!

label.place(x=20 ,y=100)

# BITMAPS dentro de un CANVAS

canvas = Canvas(root,width=400, height=400,background='red')

canvas.place(x=200 ,y=200)

tecto = canvas.create\_text(200, 50, text = 'CANVAS' , fill = 'white' , font = ('Arial',22))

item\_1 = canvas.create\_bitmap(200, 100, bitmap="info", foreground="white")

item\_2 = canvas.create\_bitmap(200, 150, bitmap='question', foreground="white")

# CANVAS y objetos BitmapImage

item\_3 = canvas.create\_image(200, 200, image = bitmap\_1)

# CANVAS e imagenes .ppm .gif .png

img = PhotoImage(file="recopilation.png")

picture\_1 = canvas.create\_image(200,300, image=img)

root.mainloop() #bucle de la ventana principal

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**08\_tkinter.py**  **CANVAS ALIEN Canvas,Events**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

from time import \*

def main(): #Función principal

#Ventana principal

window=Tk()

window.title('Canvas Alien')

# LIENZO CANVAS

c=Canvas(window,width=400,height=300) # CANVAS

c.pack() # colocar y ver Canvas

body= c.create\_oval(100,150,300,250,fill='green')#coordenadas esquinas superior izquierda e inferior derecha

eye= c.create\_oval(170,70,230,130,fill='white')

eyeball= c.create\_oval(190,90,210,110,fill='black')

mouth= c.create\_oval(150,220,250,240,fill='red')

neck= c.create\_line(200,150,200,130)

hat= c.create\_polygon(180,75,220,75,200,20,fill='blue')

sleep(1)

words= c.create\_text(200,280,text='I am an Alien (a,d,w,s,arrows,mouse buttons)')

window.attributes('-topmost',True) ######## Colocar ventana delante de todas

**#FUNCIONES**

def mouth\_open():

c.itemconfig(mouth,fill='black') # modificar un item del canvas (ID o TAG)

def mouth\_close():

c.itemconfig(mouth,fill='red')

def blink():

c.itemconfig(eye,fill='green')

c.itemconfig(eyeball,state=HIDDEN)

def unblink():

c.itemconfig(eye,fill='white')

c.itemconfig(eyeball,state=NORMAL)

def steal\_hat(event):

c.itemconfig(hat,state=HIDDEN)

c.itemconfig(words,text='Give my hat back')

def thanks(event):

c.itemconfig(hat,state=NORMAL)

c.itemconfig(words,text='Thanks')

################Mouse events EVENTOS

def burp(event):

mouth\_open()

c.itemconfig(words,text='Burp!') # modificar un item del canvas

c.bind\_all('<Button-1>',burp) #enlazar boton izquierdo con la función burp

def unburp(event):

mouth\_close()

c.itemconfig(words,text=' ')

c.bind\_all('<Button-3>',unburp) #enlazar boton derecho con la función unburp

################Key events

def blink2(event):

c.itemconfig(eye,fill='green')

c.itemconfig(eyeball,state=HIDDEN)

def unblink2(event):

c.itemconfig(eye,fill='white')

c.itemconfig(eyeball,state=NORMAL)

c.bind\_all('<KeyPress-a>',blink2) # tecla 'a' pulsada

c.bind\_all('<KeyPress-d>',unblink2)

c.bind\_all('<KeyPress-w>',steal\_hat)

c.bind\_all('<KeyPress-s>',thanks)

################### mover OJO con teclas

velocidad = 0.01 ### AJUSTAR

teclas = {'Up':False,'Down':False,'Left':False,'Right':False}

def addkey(event):

key=event.keysym # devuelve símbolo de tecla pulsada

if key == "Up":

teclas['Up']=True

if key == "Down":

teclas['Down']=True

if key == "Left":

teclas['Left']=True

if key == "Right":

teclas['Right']=True

def deletekey(event):

key=event.keysym # devuelve símbolo de tecla pulsada

if key == "Up":

teclas['Up']=False

if key == "Down":

teclas['Down']=False

if key == "Left":

teclas['Left']=False

if key == "Right":

teclas['Right']=False

c.bind\_all('<KeyPress>',addkey) # cualquier tecla pulsada

c.bind\_all('<KeyRelease>',deletekey) # cualquier tecla pulsada

##########################BUCLE PRINCIPAL DEL JUEGO

while True:

if teclas["Up"]:

c.move(eyeball,0,-velocidad) # mover item .move(item , x ,y)

if teclas["Down"]:

c.move(eyeball,0,velocidad)

if teclas["Left"]:

c.move(eyeball,-velocidad,0)

if teclas["Right"]:

c.move(eyeball,velocidad,0)

window.update() ## IMPORTANTE PARA VER LA VENTANA ACTUALIZADA

###############################################################

window.mainloop()

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**08b\_tkinter.py**  **CANVAS - Espiral**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

from datetime import \* # datetime

from math import \* #math

def main():

#ventana principal

window=Tk()

#canvas

anchoCanvas = 400

altoCanvas = 400

centroX = anchoCanvas/2

centroY = altoCanvas/2

c=Canvas(window,width=anchoCanvas,height=altoCanvas,bg = 'black') # CANVAS

c.pack()

# Variables

angulo\_tramo = 2\*pi/360

incremento = 0.1

#############################################################################################################

for i in range(6000):

#COS y SEN

coseno\_actual = cos(i \* angulo\_tramo)

seno\_actual = sin(i \* angulo\_tramo)

radio\_actual = i \* incremento

coseno\_siguiente = cos((i+1) \* angulo\_tramo)

seno\_siguiente = sin((i+1) \* angulo\_tramo)

radio\_siguiente = (i+1)\*incremento

#Coordenadas puntos INT Y EXT

x\_int = coseno\_actual \* radio\_actual + anchoCanvas/2

y\_int = seno\_actual \* radio\_actual + altoCanvas/2

x\_ext = coseno\_siguiente \* radio\_siguiente + anchoCanvas/2

y\_ext = seno\_siguiente \* radio\_siguiente + altoCanvas/2

#Dibujar línea

punto = c.create\_line(x\_int , y\_int , x\_ext , y\_ext , fill='yellow',width=4)#coordenadas dos puntos de la línea

####################################################################################################################

window.attributes('-topmost',True) # Colocar ventana delante de todas

window.mainloop() # Bucle visualización ventana principal

if \_\_name\_\_ == "\_\_main\_\_": # Averiguar si se está ejecutando o importando

main()

**08b\_tkinter.py**  **CANVAS - Espiral GIRANDO**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

from datetime import \* # datetime

from math import \* #math

def main():

#ventana principal

window=Tk()

#canvas

anchoCanvas = 400

altoCanvas = 400

centroX = anchoCanvas/2

centroY = altoCanvas/2

c=Canvas(window,width=anchoCanvas,height=altoCanvas,bg = 'black') # CANVAS

c.pack()

# Variables

angulo\_tramo = 2\*pi/360

incremento = 0.1

#############################################################################################################

f = 0

while True:

fase = angulo\_tramo \* f

for i in range(2800):

#COS y SEN

coseno\_actual = cos(i \* angulo\_tramo-f)

seno\_actual = sin(i \* angulo\_tramo-f)

radio\_actual = i \* incremento

coseno\_siguiente = cos((i+1) \* angulo\_tramo-f)

seno\_siguiente = sin((i+1) \* angulo\_tramo-f)

radio\_siguiente = (i+1)\*incremento

#Coordenadas puntos INT Y EXT

x\_int = coseno\_actual \* radio\_actual + anchoCanvas/2

y\_int = seno\_actual \* radio\_actual + altoCanvas/2

x\_ext = coseno\_siguiente \* radio\_siguiente + anchoCanvas/2

y\_ext = seno\_siguiente \* radio\_siguiente + altoCanvas/2

#Dibujar línea

punto = c.create\_line(x\_int , y\_int , x\_ext , y\_ext , fill='yellow' ,width=5)#coordenadas dos puntos de la línea

window.update()

c.delete(ALL)

f = f +1

####################################################################################################################

window.attributes('-topmost',True) # Colocar ventana delante de todas

window.mainloop() # Bucle visualización ventana principal

if \_\_name\_\_ == "\_\_main\_\_": # Averiguar si se está ejecutando o importando

main()

**08c\_tkinter.py** **CANVAS ARCOIRIS**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

from math import \* #math

from colorsys import \*  **#colorsys**

def main():

#ventana principal

window=Tk()

window.title('Canvas')

#canvas

anchoCanvas = 800

altoCanvas = 800

centroX = anchoCanvas/2

centroY = altoCanvas/2

c=Canvas(window,width=anchoCanvas,height=altoCanvas,bg = 'black') # CANVAS

c.pack()

# Variables

colorRGB=[0,0,0] # [255,255,255] colorsys usa [1,1,1]

colorHSV=[0,0,0] # [360,1,1] colorsys usa [1,1,1]

hue = 0

#Dibujar líneas horizontales cada una de un color

for i in range(360):

hue = i \* (1/360)

color\_rgb = hsv\_to\_rgb(hue,1,1) # todos los valores de 0 a 1 hsv y rgb

text\_color = '#' + str('%02x' % (int(color\_rgb[0]\*255))) + str('%02x' % (int(color\_rgb[1]\*255))) + str('%02x' % (int(color\_rgb[2]\*255)))

x1=0

y1=i\*int(altoCanvas/360)

x2=anchoCanvas

y2=y1

linea = c.create\_line(x1,y1,x2,y2,fill=text\_color, width=int(altoCanvas/360))

window.attributes('-topmost',True) # Colocar ventana delante de todas

window.mainloop() # Bucle visualización ventana principal

if \_\_name\_\_ == "\_\_main\_\_": # Averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**09\_tkinter.py**  **Juego Bubble Blaster**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

import random

import math

import time

def main():

#VENTANA

window = Tk()

window.title('Bubble Blaster')

#CANVAS

HEIGHT = 500

WIDTH = 800

c = Canvas(window, width=WIDTH, height=HEIGHT, bg='darkblue')

c.pack()

#NAVE ESPACIAL

ship\_id = c.create\_polygon(5,5,5,25,30,15,fill='red',tag='nave')

ship\_id2 = c.create\_oval(0,0,30,30,outline='red',tag='nave')

SHIP\_RADIUS = 15

MID\_X = WIDTH / 2

MID\_Y = HEIGHT / 2

c.move('nave',MID\_X,MID\_Y)

#c.move(ship\_id,MID\_X,MID\_Y) #se puede hacer referencia a un item por su ID o por algún TAG

#c.move(ship\_id2,MID\_X,MID\_Y)

SHIP\_SPEED = 2 #################### VELOCIDAD NAVE

### REGISTRO DE TECLAS PULSADAS

######################################################################

teclas\_abajo = {'a':False,'d':False,'w':False,'s':False}##############

def add\_key(event): # AÑADIR TECLA a la lista de teclas pulsadas

if event.keysym == 'a':

teclas\_abajo['a']=True

elif event.keysym == 'd':

teclas\_abajo['d']=True

elif event.keysym == 'w':

teclas\_abajo['w']=True

elif event.keysym == 's':

teclas\_abajo['s']=True

#move\_ship()

def delete\_key(event): # ELIMINAR TECLAS de la lista de teclas pulsadas

if event.keysym == 'a':

teclas\_abajo['a']=False

elif event.keysym == 'd':

teclas\_abajo['d']=False

elif event.keysym == 'w':

teclas\_abajo['w']=False

elif event.keysym == 's':

teclas\_abajo['s']=False

#move\_ship()

c.bind\_all('<KeyPress>',add\_key) # EVENTO tecla pulsada

c.bind\_all('<KeyRelease>',delete\_key) #EVENTO tecla soltada

def move\_ship(): # FUNCION MOVER NAVE ESPACIAL

if teclas\_abajo['w'] and teclas\_abajo['d']:

c.move('nave',SHIP\_SPEED,-SHIP\_SPEED)

elif teclas\_abajo['w'] and teclas\_abajo['a']:

c.move('nave',-SHIP\_SPEED,-SHIP\_SPEED)

elif teclas\_abajo['s'] and teclas\_abajo['a']:

c.move('nave',-SHIP\_SPEED,SHIP\_SPEED)

elif teclas\_abajo['s'] and teclas\_abajo['d']:

c.move('nave',SHIP\_SPEED,SHIP\_SPEED)

elif teclas\_abajo['w']:

c.move('nave',0,-SHIP\_SPEED)

elif teclas\_abajo['s']:

c.move('nave',0,SHIP\_SPEED)

elif teclas\_abajo['a']:

c.move('nave',-SHIP\_SPEED,0)

elif teclas\_abajo['d']:

c.move('nave',SHIP\_SPEED,0)

#################################################################################

# POMPAS

bubble\_id = list() #Lista vacia []

bubble\_radius = list() #Lista vacia []

bubble\_speed = list() #Lista vacia []

MIN\_BUBBLE\_RADIUS = 10

MAX\_BUBBLE\_RADIUS = 30

MAX\_BUBBLE\_SPEED = 10

GAP = 100 # margen a la derecha fuera de la pantalla

def create\_bubble(): # CREAR POMPAS

x = WIDTH + GAP

y = random.randint(0, HEIGHT)

r = random.randint(MIN\_BUBBLE\_RADIUS, MAX\_BUBBLE\_RADIUS)

id1 = c.create\_oval(x-r, y-r, x+r, y+r, outline='white')

bubble\_id.append(id1)

bubble\_radius.append(r)

bubble\_speed.append(random.randint(1,MAX\_BUBBLE\_SPEED))

def move\_bubbles(): # MOVER POMPAS

for i in range(len(bubble\_id)):

c.move(bubble\_id[i], -1\*bubble\_speed[i], 0)

def get\_coords(id): # Obtener situacion

pos = c.coords(id) # x1,y1,x2,y2 0,1,2,3

x = (pos[0] + pos[2]) / 2

y = (pos[1] + pos[3]) / 2

return x, y

def del\_bubble(id): # ELIMINAR POMPAS colisionadas

del bubble\_radius[id] # del borrar de la lista

del bubble\_speed[id]

c.delete(bubble\_id[id]) # eliminar del CANVAS

del bubble\_id[id]

def clean\_up\_bubbles(): # ELIMINAR POMPAS por la izquierda

for id in range(len(bubble\_id)-1,-1,-1):

x,y = get\_coords(bubble\_id[id])

if x < -GAP: # si se sale por la izquierda de la pantalla

del\_bubble(id)

def distance(id1,id2): # DISTANCIA

x1, y1 = get\_coords(id1)

x2, y2 = get\_coords(id2)

return math.sqrt((x2-x1)\*\*2+(y2-y1)\*\*2)

def collision(): # COLISIONES

points = 0

for id in range(len(bubble\_id)-1,-1,-1):

d = distance(ship\_id2, bubble\_id[id])

b = SHIP\_RADIUS + bubble\_radius[id]

if d < b:

points += bubble\_radius[id] + bubble\_speed[id]

del\_bubble(id)

return points

# INFORMACION EN PANTALLA

c.create\_text(50,30,text='TIME',fill='white')

c.create\_text(150,30,text='SCORE',fill='white')

time\_text = c.create\_text(50,50,fill='white')

score\_text = c.create\_text(150,50,fill='white')

def show\_score(score): # Mostrar marcador

c.itemconfig(score\_text,text=str(score))

def show\_time(time\_left): # Mostrar tiempo

c.itemconfig(time\_text, text=str(time\_left))

# VARIABLES

BUBBLE\_CHANCE = 10

TIME\_LIMIT = 8 # cada 1000 puntos añadimos tiempo

BONUS\_SCORE = 1000

score = 0

bonus = 0

end = time.time() + TIME\_LIMIT

#################################### MAIN GAME LOOP ##################

# BUCLE PRINCIPAL DEL JUEGO

while time.time() < end:

if random.randint(1, BUBBLE\_CHANCE) == 1:

create\_bubble()

move\_bubbles()

clean\_up\_bubbles()

score += collision() # Actualizar puntos

if (int(score / BONUS\_SCORE)) > bonus:

bonus += 1

end += TIME\_LIMIT

show\_score(score) # Mostrar puntos

show\_time(int(end - time.time())) # Mostrar TIEMPO

move\_ship() ####### MOVER NAVE solución para darle fluidez al movimiento de la nave

window.update() ##### IMPORTANTE PARA VER LA VENTANA en cada fotograma

#####################################################################

# FIN JUEGO

c.create\_text(MID\_X, MID\_Y, text='GAME OVER', fill='white',

font=('Helvetica', 30))

c.create\_text(MID\_X, MID\_Y + 30, text='Score: ' + str(score),

fill='white')

c.create\_text(MID\_X, MID\_Y + 45, text='Bonus Time: ' +

str(bonus\*TIME\_LIMIT), fill='white')

window.mainloop() # Ventana principal

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**10\_tkinter.py**  **Minicalculadora**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

from tkinter import messagebox ###############

#from tkinter import messagebox

except:

from Tkinter import \* # python 2.7

from Tkinter import messagebox ###############

#from Tkinter import messagebox

def main():

#Metodo para calcular la suma

def Suma():

n1=float(caja1.get())

n2=float(caja2.get())

suma=n1+n2

messagebox.showinfo("Mensaje","El resultado es: %.2f"%suma)

caja1.delete(0,20)

caja2.delete(0,20)

#Metodo para calcular la resta

def Resta():

n1=float(caja1.get())

n2=float(caja2.get())

suma=n1-n2

messagebox.showinfo("Mensaje","El resultado es: %.2f"%suma)

caja1.delete(0,20)

caja2.delete(0,20)

#Metodo para calcular la multiplicacion

def Multiplicacion():

n1=float(caja1.get())

n2=float(caja2.get())

suma=n1\*n2

messagebox.showinfo("Mensaje","El resultado es: %.2f"%suma)

#caja1.delete(0,20)

#caja2.delete(0,20)

#Metodo para calcular la division

def Division():

n1=float(caja1.get())

n2=float(caja2.get())

suma=n1/n2

messagebox.showinfo("Mensaje","El resultado es: %.2f"%suma)

caja1.delete(0,20)

caja2.delete(0,20)

#Creacion del GUI

gui = Tk()

#Titulo del GUI

gui.title("MiniCalculadora")

#Dimensiones (ancho,alto,posicion x,posicion y)

gui.geometry("400x250+400+200")

#Creacion de una etiqueta

var1 = StringVar()

var1.set("Escribe el primer numero:")

label1 = Label(gui,textvariable=var1,height = 2)

label1.pack()

#Creacion de una caja de texto para el primer numero

numero1=StringVar()

caja1=Entry(gui,bd=4,textvariable=numero1)

caja1.pack()

#Creacion de otra etiqueta

var2 = StringVar()

var2.set("Escribe el segundo numero:")

label2 = Label(gui,textvariable=var2,height = 2)

label2.pack()

#Creacion de otra caja de texto para el segundo numero

numero2=StringVar()

caja2=Entry(gui,bd=4,textvariable=numero2)

caja2.pack()

#Boton para la suma

boton1 = Button(gui, text = "Suma", command = Suma,width=15)

boton1.pack()

#Boton para la resta

boton2 = Button(gui, text = "Resta", command = Resta,width=15)

boton2.pack()

#Boton para multiplicacion

boton3 = Button(gui, text = "Multiplicacion", command = Multiplicacion,width=15)

boton3.pack()

#Boton para la division

boton4 = Button(gui, text = "Division", command = Division,width=15)

boton4.pack()

#Cargar el GUI

gui.mainloop()

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**11\_tkinter.py**  **CALCULADORA NORMAL**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

from math import \*

def main():

ventana=Tk()

ventana.title("CALCULADORA")

ventana.geometry("392x600")

ventana.configure(background="SkyBlue4")

color\_boton=("gray77")

def btnClik(num):

global operador

operador=operador+str(num)

input\_text.set(operador) #ESTA PARTE SIRVE PARA VISUALIZAR LA OPERACIÓN EN LA PANTALLA

def clear():

global operador

operador=("")

input\_text.set("0")

def operacion():

global operador

try:

opera=str(eval(operador))#SIRVE PARA REALIZAR LA OPERACIÓN PREVIAMENTE VISUALIZADA EN PANTALLA

except:

clear()

opera=("ERROR")

input\_text.set(opera)#MUESTRA EL RESULTADO

ancho\_boton=11

alto\_boton=3

input\_text=StringVar() ### Class variable de tkinter

# se puede asociar a una propiedad de un tkinter widget

# en este caso se asociará a la propiedad 'textvariable' del 'Entry' llamado 'Salida' creado más abajo

operador=""

clear() #MUESTRA "0" AL INICIAR LA CALCULADORA

Boton0=Button(ventana,text="0",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(0)).place(x=17,y=180)

Boton1=Button(ventana,text="1",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(1)).place(x=107,y=180)

Boton2=Button(ventana,text="2",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(2)).place(x=197,y=180)

Boton3=Button(ventana,text="3",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(3)).place(x=287,y=180)

Boton4=Button(ventana,text="4",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(4)).place(x=17,y=240)

Boton5=Button(ventana,text="5",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(5)).place(x=107,y=240)

Boton6=Button(ventana,text="6",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(6)).place(x=197,y=240)

Boton7=Button(ventana,text="7",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(7)).place(x=287,y=240)

Boton8=Button(ventana,text="8",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(8)).place(x=17,y=300)

Boton9=Button(ventana,text="9",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(9)).place(x=107,y=300)

BotonC=Button(ventana,text="π",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("pi")).place(x=197,y=300)

BotonComa=Button(ventana,text=",",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(".")).place(x=287,y=300)

BotonSuma=Button(ventana,text="+",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("+")).place(x=17,y=360)

BotonResta=Button(ventana,text="-",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("-")).place(x=107,y=360)

BotonMulti=Button(ventana,text="\*",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("\*")).place(x=197,y=360)

BotonDiv=Button(ventana,text="/",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("/")).place(x=287,y=360)

BotonSqrt=Button(ventana,text="√",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("sqrt")).place(x=17,y=420)

BotonParen1=Button(ventana,text="(",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("(")).place(x=17,y=480)

BotonParen2=Button(ventana,text=")",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik(")")).place(x=107,y=480)

BotonResto=Button(ventana,text="%",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("%")).place(x=197,y=480)

Botonln=Button(ventana,text="ln",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("log")).place(x=287,y=480)

BotonC=Button(ventana,text="C",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=clear).place(x=107,y=420)

BotonExp=Button(ventana,text="EXP",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=lambda:btnClik("\*\*")).place(x=197,y=420)

BotonResul=Button(ventana,text="=",bg=color\_boton,width=ancho\_boton,height=alto\_boton,command=operacion).place(x=287,y=420)

Salida=Entry(ventana,font=('arial',20,'bold'),width=22,textvariable=input\_text,bd=20,insertwidth=4,bg="powder blue",justify="right").place(x=10,y=60)

ventana.mainloop()

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**12\_tkinter.py**  **TRES EN RAYA**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

v0=Tk()

v0.title("Gato")

v0.resizable(0,0)

color,nl,matriz,ganador=["red"],[],[0,0,0,0,0,0,0,0,0],[0]

def imprimir(t):

print (t)

def doit(f):

v0.after(100,f) # Evento por tiempo

# Crea un gato

def gato():

ind=0

c1,c2=0,0

while ind < 9:

nl.append(Button(v0,text="",width=10,height=5,bg="white"))

nl[ind].grid(row=c2,column=c1)

ind+=1

if c1==2: c1,c2=0,c2+1

else: c1+=1

nl[0].config(command=lambda: jugar(0)), nl[1].config(command=lambda: jugar(1))

nl[2].config(command=lambda: jugar(2)), nl[3].config(command=lambda: jugar(3))

nl[4].config(command=lambda: jugar(4)), nl[5].config(command=lambda: jugar(5))

nl[6].config(command=lambda: jugar(6)), nl[7].config(command=lambda: jugar(7))

nl[8].config(command=lambda: jugar(8))

def juego\_finalizado(): # Revisa si el gato está lleno. Si lo está devuelve True

ind,largo=0,len(matriz)

while ind < largo:

if matriz[ind] == 0:

return False

break

ind+=1

return True

def tres\_linea(lista):

if matriz[lista[0]] != 0 and matriz[lista[1]] != 0 and matriz[lista[2]] != 0:

if matriz[lista[0]] == matriz[lista[1]] == matriz[lista[2]]:

ganador[0]=matriz[lista[0]]

return True

return False

def gana():

if tres\_linea([0,1,2]) or tres\_linea([3,4,5]) or tres\_linea([6,7,8]) or tres\_linea([0,3,6]) or tres\_linea([1,4,7]) or tres\_linea([2,5,8]) or tres\_linea([0,4,8]) or tres\_linea([2,4,6]):

return True

def limpiar\_botones():

color="white"

ind,largo=0,len(nl)

while ind < largo:

nl[ind].config(bg=color)

ind+=1

matriz[0],matriz[1],matriz[2],matriz[3],matriz[4],matriz[5],matriz[6],matriz[7],matriz[8]=0,0,0,0,0,0,0,0,0

def declarar\_ganador():

v0.after(200,declarar\_ganador)

if gana():

limpiar\_botones()

v0.withdraw()

v1=Toplevel(v0)

if ganador[0] == 1:

l1=Label(v1,text="Ganador: Jugador 1 (Azul)",font=(16))

color[0]="red"

if ganador[0] == 2:

l1=Label(v1,text="Ganador: Jugador 2 (Rojo)",font=(16))

color[0]="blue"

l1.pack()

b1=Button(v1,text="OK",command=lambda: v1.withdraw() or v0.deiconify(),font=(16))

b1.pack()

v1.focus\_force()

def reiniciar\_juego():

v0.after(200,reiniciar\_juego)

if juego\_finalizado():

v0.withdraw()

v1=Toplevel(v0)

v1.title("Finalizado")

v1.resizable(0,0)

l1=Label(v1,text="Nadie Ganó.",font=(16))

l1.pack()

b1=Button(v1,text="OK",command=lambda: v1.withdraw() or v0.deiconify(),font=(16))

b1.pack()

doit(limpiar\_botones())

def jugar(posicion):

if matriz[posicion] != 0:

print( "Ya se jugó en esta posición")

else:

if color[0]=="red":

matriz[posicion]=1

color[0]="blue"

nl[posicion].config(bg=color)

elif color[0]=="blue":

matriz[posicion]=2

color[0]="red"

nl[posicion].config(bg=color)

print( "ESTADO DE LA MATRIZ:",matriz)

gato()

declarar\_ganador()

reiniciar\_juego()

v0.mainloop()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**13\_tkinter.py**  **RELOJ ANALÓGICO \_ DIGITAL**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

from datetime import \* # datetime

from math import \* #math

def main():

#ventana principal

window=Tk()

window.title('Canvas')

#canvas

anchoCanvas = 400

altoCanvas = 400

centroX = anchoCanvas/2

centroY = altoCanvas/2

c=Canvas(window,width=anchoCanvas,height=altoCanvas,bg = 'black') # CANVAS

c.pack()

# Variables

angulo\_hora = 2\*pi/12

angulo\_segundo = 2\*pi/60

angulo\_minuto = 2\*pi/60

radio\_punto = 10

radio\_reloj = 100

long\_segundero = 95

long\_minutero = 95

long\_horero = 60

# Dibujar 12 puntos horas

for h in range(12):

coseno = cos(h \* angulo\_hora)

seno = sin(h \* angulo\_hora)

x\_base = coseno \* radio\_reloj + anchoCanvas/2

y\_base = seno \* radio\_reloj + altoCanvas/2

punto = c.create\_oval(x\_base - radio\_punto , y\_base - radio\_punto, x\_base + radio\_punto , y\_base + radio\_punto , fill='white')#coordenadas esquinas superior izquierda e inferior derecha

#Crear manecillas todas mirando hacia arriba

coordenadas\_segundero = [centroX , centroY , centroX , centroY - long\_segundero]

segundero = c.create\_line( coordenadas\_segundero , fill ='white' , width = 2 , **tag =**'**manecilla**')

#un item puede ser accedido por su ID (segundero) o por algún tag asignado(‘manecilla’) (puede tener más de un tag)

coordenadas\_minutero = [centroX , centroY , centroX , centroY - long\_minutero]

minutero = c.create\_line( coordenadas\_minutero , fill ='yellow' , width = 5)

coordenadas\_horero = [centroX , centroY , centroX , centroY - long\_horero]

horero = c.create\_line( coordenadas\_horero , fill ='red' , width = 8)

hora\_digital = c.create\_text(anchoCanvas/2 , 50 ,text = '00:00:00' , font = ( 'Time' , 30), fill = 'white')

# Bucle infinito para mover manecillas y actualizar reloj

while True:

ahora = datetime.now()

segundos = ahora.second

minutos = ahora.minute

horas = ahora.hour

#hora digital

if horas<10:

horas\_text = '0'+ str(horas)

else:

horas\_text = str(horas)

if minutos<10:

minutos\_text = '0'+ str(minutos)

else:

minutos\_text = str(minutos)

if segundos<10:

segundos\_text = '0'+ str(segundos)

else:

segundos\_text = str(segundos)

c.itemconfigure(hora\_digital,text = str(horas\_text)+':'+str(minutos\_text)+':'+str(segundos\_text)) #itemconfigure

#segundero

coseno = cos(segundos \* angulo\_segundo)

seno = sin(segundos \* angulo\_segundo)

######### MODIFICAR COORDENADAS DE UN ITEM por su ID

c.coords(segundero ,centroX , centroY , centroX + long\_segundero \* seno , centroY - long\_segundero \* coseno )

# o por TAG  **c.coords(‘manecilla’ ,centroX , centroY , centroX + long\_segundero \* seno , centroY - long\_segundero \* coseno**  )

#minutero

coseno = cos(minutos \* angulo\_minuto)

seno = sin(minutos \* angulo\_minuto)

######### MODIFICAR COORDENADAS DE UN ITEM por su ID

c.coords(minutero ,centroX , centroY , centroX + long\_minutero \* seno , centroY - long\_minutero \* coseno )

#horero

coseno = cos(horas \* angulo\_hora)

seno = sin(horas \* angulo\_hora)

######### MODIFICAR COORDENADAS DE UN ITEM por su ID

c.coords(horero ,centroX , centroY , centroX + long\_horero \* seno , centroY - long\_horero \* coseno )

window.update() #actualizar ventana para ver los cambios

window.attributes('-topmost',True) # Colocar ventana delante de todas

window.mainloop() # Bucle visualización ventana principal

if \_\_name\_\_ == "\_\_main\_\_": # Averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**14\_tkinter.py**  **MINI PAINT**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

def main():

root = Tk()

root.geometry("800x800+0+0") **# establecer ancho , alto , offsetx y offsety de la ventana principal**

h = Scrollbar(root, orient=HORIZONTAL)

v = Scrollbar(root, orient=VERTICAL)

canvas = Canvas(root, scrollregion=(0, 0, 1000, 1000), yscrollcommand=v.set, xscrollcommand=h.set)

h['command'] = canvas.xview #ajustar el ancho del canvas

v['command'] = canvas.yview

canvas.grid(column=0, row=0, sticky=(N,W,E,S)) ## sticky indica la alineación dentro de la celda

h.grid(column=0, row=1, sticky=(W,E))

v.grid(column=1, row=0, sticky=(N,S))

root.grid\_columnconfigure(0, weight=1) #controlar anchura de la columna

root.grid\_rowconfigure(0, weight=1) #controlar altura de la fila

lastx, lasty = 0, 0

def xy(event):

global lastx, lasty

lastx, lasty = canvas.canvasx(event.x), canvas.canvasy(event.y) # x,y del ratón

def setColor(newcolor):

global color

color = newcolor

canvas.dtag('all', 'paletteSelected') # eliminar etiquetas TAGS

canvas.itemconfigure('palette', outline='white') # borde BLANCO - modificar elementos con tag = palette (rectangles)

canvas.addtag('paletteSelected', 'withtag', 'palette%s' % color) #añadir etiquetas TAGS

canvas.itemconfigure('paletteSelected', outline='#999999') # borde GRIS - modificar elementos con tag = paletteSelected

def addLine(event):

global lastx, lasty

x, y = canvas.canvasx(event.x), canvas.canvasy(event.y)

canvas.create\_line((lastx, lasty, x, y), fill=color, width=5, tags='currentline') # mientras dibujamos la linea tiene grosor 5

lastx, lasty = x, y

def doneStroke(event):

canvas.itemconfigure('currentline', width=1) # dibujar la linea con grosor 1

canvas.bind("<Button-1>", xy) # boton izq asociado a funcion def xy()

canvas.bind("<B1-Motion>", addLine) # movimiento del raton con boton izq pulsado asociado a def addLine()

canvas.bind("<B1-ButtonRelease>", doneStroke) # al soltar el boton izq se dibuja la linea con grosor 1

# cuadraditos de colores para elegir color

id = canvas.create\_rectangle((10, 10, 30, 30), fill="red", tags=('palette', 'palettered'))

canvas.tag\_bind(id, "<Button-1>", lambda x: setColor("red")) # asociar al cuadradito el cambio de color

id = canvas.create\_rectangle((10, 35, 30, 55), fill="blue", tags=('palette', 'paletteblue'))

canvas.tag\_bind(id, "<Button-1>", lambda x: setColor("blue")) # asociar al cuadradito el cambio de color

id = canvas.create\_rectangle((10, 60, 30, 80), fill="black", tags=('palette', 'paletteblack', 'paletteSelected'))

canvas.tag\_bind(id, "<Button-1>", lambda x: setColor("black")) # asociar al cuadradito el cambio de color

setColor('black')

canvas.itemconfigure('palette', width=5) # asignar a todos los item con TAG=palette ancho trazo = 5 (cuadraditos)

root.mainloop() #bucle principal de la ventana

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# **PYTHON GUI OOP TKINTER**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Versión Hola mundo tkinter OOP**

**#!/usr/bin/env python**

**# -\*- coding: utf-8 -\*-**

**try**:

**from** tkinter **import** \* # python 3.

**except:**

**from** Tkinter **import** \* # python 2.7

**class** **Application**(Frame):

**def** \_\_init\_\_(self, master=**None**):

super().\_\_init\_\_(master)

self.master = master

self.pack()

self.create\_widgets()

**def** create\_widgets(self):

self.hi\_there = Button(self)

self.hi\_there["text"] = "Hello World**\n**(click me)"

self.hi\_there["command"] = self.say\_hi

self.hi\_there.pack(side="top")

self.quit = Button(self, text="QUIT", fg="red", command = self.master.destroy)

self.quit.pack(side="bottom")

**def** say\_hi(self):

print("hi there, everyone!")

**def main():**

root = Tk()

app = Application(master=root)

app.mainloop()

**if**  \_\_name\_\_ == **"\_\_main\_\_"**: # averiguar si se está ejecutando o importando

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*4

**001\_TKINTER\_OOP\_PADDLE\_BALL.py**

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

try:

from tkinter import \* # python 3.

except:

from Tkinter import \* # python 2.7

import random

import time

class Ball():

def \_\_init\_\_(self, canvas, paddle, color):

self.canvas = canvas

self.paddle = paddle

self.id = canvas.create\_oval(10, 10, 25, 25, fill=color)

starts = [-3, -2, -1, 1, 2, 3]

random.shuffle(starts)

self.x = starts[0]

self.y = -3

self.canvas\_height = canvas.winfo\_height()

self.canvas\_width = canvas.winfo\_width()

self.is\_hitting\_bottom = False

canvas.move(self.id, 245, 100)

def draw(self):

self.canvas.move(self.id, self.x, self.y)

pos = self.canvas.coords(self.id)

if pos[1] <= 0:

self.y = 1

if pos[3] >= self.canvas\_height:

# self.y = -1

self.is\_hitting\_bottom = True

if self.hit\_top\_paddle(pos) == True:

self.y = -3

if self.hit\_bottom\_paddle(pos) == True:

self.y = 1

if pos[0] <= 0:

self.x = 3

if pos[2] >= self.canvas\_width:

self.x = -3

def hit\_top\_paddle(self, pos):

paddle\_pos = self.canvas.coords(self.paddle.id)

if pos[2] >= paddle\_pos[0] and pos[0] <= paddle\_pos[2]:

if pos[3] >= paddle\_pos[1] and pos[3] <= paddle\_pos[3]:

return True

return False

def hit\_bottom\_paddle(self, pos):

paddle\_pos = self.canvas.coords(self.paddle.id)

if pos[2] >= paddle\_pos[0] and pos[0] <= paddle\_pos[2]:

if pos[1] >= paddle\_pos[1] and pos[1] <= paddle\_pos[3]:

return True

return False

class Paddle:

def \_\_init\_\_(self, canvas, color):

self.canvas = canvas

self.id = canvas.create\_rectangle(0, 0, 100, 10, fill=color)

self.x = 0

self.canvas\_width = canvas.winfo\_width()

canvas.move(self.id, 200, 300)

canvas.bind\_all('<KeyPress-Left>', self.move\_left)

canvas.bind\_all('<KeyPress-Right>', self.move\_right)

def draw(self):

self.canvas.move(self.id, self.x, 0)

pos = self.canvas.coords(self.id)

if pos[0] <= 0:

self.x = 0

if pos[2] >= self.canvas\_width:

self.x = 0

def move\_left(self, event):

self.x = -2

def move\_right(self, event):

self.x = 2

def main():

tk = Tk()

tk.title('Game')

canvas = Canvas(tk, width=550, height=400, bd=0, highlightthickness=0)

canvas.pack()

tk.update()

paddle = Paddle(canvas, 'blue')

ball = Ball(canvas, paddle, 'red')

while 1:

if ball.is\_hitting\_bottom == False:

ball.draw()

paddle.draw()

tk.update\_idletasks()

tk.update()

time.sleep(0.01)

if \_\_name\_\_ == "\_\_main\_\_": # averiguar si se está ejecutando o importando

main()