**PYTHON**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| print(variable)  print(“texto”) | |  | | |
| print(“jonathan \n alanoca”)  print(“jonathan \t alanoca”) | | Hace un salto de línea.  Hace una tabulación en el texto | | |
| print(“jonathan”, end=” ”)  print(“jonathan”, end=”\t”) | | La siguiente línea en pantalla se escribirá a continuación .  La siguiente línea a continuación y tabulación. | | |
| type(variable)  type(1)  print(type(var\_num)) | | Devuelve el tipo de variable. | | |
| Input(“ingrese su nombre: ”).capitalize() | | Se pueden combinar de esta manera. | | |
| del variable | | Borra el valor de la variable lista etc. | | |
| apellido=’Alanoca’  print(‘Jonathan’+apellido)  print(‘jonathan’,apellido)  print(f’jonathan {apellido}’)  print(‘jonathan {}’.format(apellido))  print(‘jonathan {0} {1}’.format(apellido, ‘G’))  print(‘jonathan %s’%(apellido)’) | | Maneras de concatenar.  %s string %d entero %f decimal | | |
| Documentacion  ‘’’texto explicativo’’’  help(func)  help(class) | | Muestra por consola toda la documentación que hay dentro del la funcion class | | |
| TIPOS DE VARIABLES | | | | |
| STRINGS (metodos) | | | | |
| variable.upper() | | Texto de la variable en mayúscula. | | |
| variable.lower() | | Texto de la variable en minúscula. | | |
| variable.capitalize() | | Primera letra en mayúscula y el resto en minúscula. | | |
| variable.title() | | Primera letra de cada palabra en mayúscula. | | |
| variable.isdigit()  variable.isalum()  variable.isalpha() | | True si es únicamente números en la variable.  True si es alfanumérico (incluye el espacio).  True si hay únicamente letras en la variable. | | |
| variable.strip() | | Borra los espacios de la variable str al comienzo y al final | | |
| variable.count(“j”) | | Devuelve la cantidad de “j” en la variable. | | |
| variable.startswith(“jo”) | | Devuelve True o False si se cumple. | | |
| variable.endswith(“an”) | | Devuelve True o False si se cumple. | | |
| variable.find(“o”) | | Devuelve la posición del valor en la variable. Solo el primer valor comenzando de 0 a ++ . [-1] en caso de no ser encontrado. | | |
| variable.rfind(“o”) | | Devuelve la posición del valor pero busca desde el final. | | |
| variable.index(“o”) | | Devuelve la posición del valor en la variable. Solo el primer valor comenzando de 0 a ++.[Value Error] en caso de no ser encontrado. | | |
| variable.replace(“jony”,”jon”) | | Reemplaza jon por jony en el texto de la variable. | | |
| Len(variable) | | Devuelve la cantidad de caracteres que posee la variable. Comenzando del 1 a ++. | | |
| variable[:3]  variable[3:] | | Toma valores de las posiciones 0 a 3  Toma valores de las posiciones de 3 a ++ | | |
| var[::2]  var[::3]  var[::-1] | | Toma valores uno si uno no j\_n\_a\_t\_a\_  Cada 3 j\_ \_a\_ \_a\_  Muestra la variable al revés nahtanoj | | |
| var[2:-2] | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | j | **o** | **n** | **a** | **t** | **h** | **a** | **n** | | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | | | |
| var.split(“/”) | | Transforma la variable str en una **lista** separando cada / | | |
|  | |  | | |
| FLOAT | |  | | |
| round(var\_dec,2) | | **Redondear** los 2 decimales. | | |
| “{:.2f}”.format(var\_dec) | | **Redondear** los 2 decimales. | | |
| "%.2f"%var\_dec  "%010.2f"%var\_dec | | Formato para números. **Redondear** los 2 decimales.  Ceros a la derecha, 10 caracteres, 2 caracteres luego de la coma y redondea (0000002.35) | | |
| LIST | |  | | |
| var\_lista=[“jony”,] | | Lista de un solo valor. | | |
| var\_lista=[1,2,3]\*3 | | [1,2,3,1,2,3,1,2,3] | | |
| var\_lista[2]  var\_lista[-2] | | Elemento de la lista en la posición 1 .  Elemento de la lista en la posición -2 | | |
| var\_lista[2,4] | | Acota la lista de 2 a 4 (2 incluye, 4 no incluye) | | |
| print (“jony” in var\_list) | | True o False si hay o no el elemento dentro de la lista. | | |
| var\_lista.append(“brazil”) | | Se agrega al final de la lista. | | |
| var\_lista.pop() | | Se borra el ultimo valor de la lista. | | |
| var\_lista.remove(“argentina”) | | Borra el valor “argentina” dentro de la lista. | | |
| var\_lista.insert(2,”bolivia”) | | Ingresa en el index 2 el valor Bolivia | | |
| var\_list.reverse()  var\_list[::-1] | | Pone en reversa la lista. No se puede combinar.  Pone en reversa la lista. Se puede combinar. | | |
| re | | Import re | | |
| var=re.findall(‘a’,variable\_o\_lista) | | Devuelve una lista con las veces que se encontró el argumento | | |
| Var=re.findall(‘Al’,var) | | Devuelve las qu | | |
| TUPLES | |  | | |
| print (“jony” in var\_tup) | | True o False si hay o no el elemento dentro de la tupla. | | |
| var\_tup.count(“jon”)  var\_tup.index(“jon”)  len(var\_tup) | | Cuantos valores jon hay dentro de la tupla.  Primera posición de jon dentro de la tupla desde la izquierda.  Cantidad de valores dentro de la tupla. | | |
| dia,mes,año=var\_tupla  (Sirve para listas) | | Dia= var\_tupla[0]  Mes= var\_tupla[1]  Año= var\_tupla[2] | | |
| variable=list(var\_tupla) | | Transformar tupla a lista. | | |
| variable=tuple(var\_lista) | | Transformar lista a tupla. | | |
| DICTIONARY | |  | | |
| var\_dicc={“bolivia”,”argentina”,1,2,1.2} | |  | | |
| var\_dicc={“peru”:”lima”,”argentina”:”bsas”}  var\_dicc={“nombre”:”jony”,“edad”:18} | |  | | |
| var\_dicc=dict(  nombre=”jony”,  edad=28,  peso=50.5  ) | | Transforma una tupla en diccionario | | |
| var\_dicc[“Brazil”]=”Sao Pablo” | | Se agrega al final de la lista. Siempre y cuando no exista esa clave. | | |
| var\_dicc[“Brazil”]=”Sao Pablo” | | Se reemplaza el valor que corresponde a la clave | | |
| del var\_dicc[“Bolivia”] | | Borra la pareja (clave y valor) del diccionario. | | |
| len(var\_dicc) | | Cantidad de parejas. | | |
| var\_dicc.key() | | Devuelve los keys del diccionario. | | |
| var\_dicc.values() | | Devuelve los valores del diccionario. | | |
| for i,u in dicc.items() | | Recorre el diccionario. i toma los keys y u toma los valores. | | |
|  | |  | | |
| BUCLE FOR | |  | | |
| for i in variable:  for i in var\_list:  for i in var\_dicc:  for i in [“azul”,”rojo”,2]: | | Recorre la carácter de la variable.  Recorre cada elemento de la lista. | | |
| For i in range(5): | | Recorre 5 veces. Valores desde 0 hasta 5 (no incluye el 5) | | |
| For i in range(5,10):  For i in range(5,100, 3):  For i in range(5,10,-1): | | Desde el 5 al 10  Desde el 5 al 100. De 3 en 3  Desde el 10 al 5 | | |
| For i in var.split(“-”): | | Separa la variable cada - (lo vuelve lista) y tom los valores en ese intervalo separado por - | | |
| BUCLE WHILE | |  | | |
| while True:  break | | Bucle infinito.  Sale del bucle forzosamente. | | |
| continue | | Cada que el bucle llega esta instrucción salta un ciclo del bucle.  (puede usarse combinado con un for y saltar un ciclo) | | |
| else: | | Se ejecuta el código luego de terminar el bucle. Pero si hay un break antes el else no se leerá. | | |
| i=0  While i<10:  i+=1 | | Ejemplo | | |
| IF | |  | | |
| If 18<var<50: | | Acotar el valor de la variable entre 18 y 100. | | |
| If var1<var2<var3<var4: | | Var4 el mayor de todos y var1 el menor de todos  Lectura derecha a izquierda compara de dos en dos(**and**) | | |
| If var1 in var\_lista:  If var1 is not var\_lista: | | Si el valor de la varible esta dentro de la lista.  Si no esta dentro de la lista | | |
| If i%2==0: | | Si es par | | |
| If variable:  If lista: | | Si la variable es True.  Si la lista existe y si tiene valores | | |
| FUNCIONES | |  | | |
| def var\_nombre():  def var\_nombre(n):  def var\_nombre(n,m):  def var\_nombre(n=””,m=””):  return  return var\_final | | Lo que va dentro de **()**  se reemplaza por los valores que le mandamos cuando llamamos a la función.  =”” Valor por default.En caso de no definirlo al llamar a la función  Termina la función  Termina la función y devuelve la variable indicada o mas de una | | |
| Decoradora | |  | | |
| def func\_exterior(func\_parametro):      def func\_interior():          print('hola')          func\_parametro()          print('adios')      return func\_interior  @func\_exterior  def suma():      print(3+4)  suma() | | Sintaxis de la funcion decoradora  Añadir decoración a la funcion  Función a decorar | | |
| def func\_exterior(func\_parametro):      def func\_interior(\*args):          print('hola')          func\_parametro(\*args)          print('adios')      return func\_interior  @func\_exterior  def suma(num1+num2):      print(num1+num2)  suma(4,5) | | \*args  Para recibir parámetros de la función  \*\*kwards  Para recibir clave valor | | |
| lambda | | | | |
| nombre\_funclambda= lambda base, altura: base\*altura | | Nombre-lmbda-parametros-return  Para funciones simples | | |
| Filter, map | | | | |
| def numeros\_pares(var):      if (var%2==0):          return True  lista[1,2,3,4,5,6,7,8,9,10]  lista2=filter(números\_pares,lista)  def numeros\_pares(var):      if (var%2==0):          return ‘Hello’  lista[1,2,3,4,5,6,7,8,9,10]  lista2=map(números\_pares,lista) | | filter(la funcion,la lista)  devuelve los valores true que cumple la función  A map podemos modificar los valores que vamos a filtrar y la podemos hacer tan compleja como queramos. | | |
|  | |  | | |
| If var<0:  raise (nom d una excepción)(“mensaje”) | | Controlar una excepción.  Nombre de excepción : ZeroDivisionError, ValueError, etc. | | |
| EXCEPCIONES | |  | | |
| try:  var=int(print(“ingrese un entero: ”)) | | Intenta esto. | | |
| except:  print(“algo salió mal”)  except ZeroDivisionError:  except ValueError:  print(“algo salió mal”) | | Una excepción general.  División por cero  Dato diferente a int. | | |
| finally: | | Finalmente haz los siguiente. (En caso que no se haya podido solucionar el error.) | | |
| except (nom d una excepción) as NomVarALlamar  print(NomVarALlamar) | | Buscara la excepción que ya declaramos y pide mostrar la descripción que le pusimos a dicha excepción previamente. | | |
| POO | | | | |
| CLASS | | | |  |
| class Mi\_class1():  var\_nombre="jony"  var\_edad=19  def información(self):  return print(f'Soy {self.var\_nombre} y tengo {self.var\_edad}')  var\_yo=Mi\_class1()  print(var\_yo.var\_nombre)  var\_yo.información() | | | | Creación de a class  Definición de las **PROPIEDADES** de la class  Creamos un **METODO** (similar a una función) self sirve para referirse a si mismo.  Creamos una **INSTANCIA**. Asociamos una variable con la class.  Llamamos a una propiedad de la class(en este caso se lo permitimos).  Llamamos a un método de la class. |
| class Mi\_class():  def \_\_init\_\_ (self):  self.nombre="jony"  self.edad=10  def info(self):  return print(f'Soy {self.nombre} y tengo {self.edad}')  var1=Mi\_class()  var1.info() | | | | **Constructor:** Este tiene como ventaja poder definir las propiedades fuera de la class: **var1.nombre=”Jose”** |
| class Mi\_class():  def \_\_init\_\_ (self):  self.\_\_nombre="jony"  self.\_\_edad=10  def \_\_info(self):  return print(f'Soy {self.\_\_nombre} y tengo {self.\_\_edad}') | | | | **Encapsulamiento:** de esta forma no tenemos acceso a los atributos o métodos fuera de la class.  En caso de propiedad solamente podemos llamar mas no modificar.  En caso de ser un método, no podemos llamarlo fuera de la class   * \_\_var01 * set.\_\_variable1 * Def \_\_func1() |
| class Persona():  def \_\_init\_\_(self):  self.nombre="jony"  self.edad=9  class Empleado(Persona):  def \_\_init\_\_(self):  Persona.\_\_init\_\_(self)  self.id=3733777  def info(self):  print(f'Presentación: {self.nombre} {self.edad} {self.id}') | | | | **HERENCIA:** quiere decir heredar los atributos y metodos de una **class padre**  **Clase padre**  **Clase hijo**  Llamamos a el constructor de la clase padre. En caso de no tener init no hace falta llamar al init del padre.  Artributo propio de la clase hijo.  Se puede heredar de varias class   * Class Empleado(Persona, hombre)   =>De esta misma forma podemos llamar a un método del padre:  def info(self):  Perona.info01(self)  print(f'Presentación: {self.nombre} {self.edad} {self.ide}') |
| class Persona():  def \_\_init\_\_(self, nombre, edad):  self.nombre=nombre  self.edad=edad  class Empleado(Persona):  def \_\_init\_\_(self, nombre, edad, ide):  Persona.\_\_init\_\_(self, nombre, edad)  self.ide=ide  def info(self):  print(f'Presentación: {self.nombre} {self.edad} {self.ide}') | | | |
| class Moto():  def marcha(self):  print("Andando en dos ruedas.")  class Cuatri():  def marcha(self):  print("Andando en cuatro ruedas.")  class Camion():  def marcha(self):  print("Andando en seis ruedas.")    def en\_marcha (auto):  auto.marcha()    modelo=Cuatri()  en\_marcha(modelo) | | | | **POLIMIRFISMO:**  **3-Mostrará la class según la declaración de la instancia.**  1-Guardamos en una variable la class que queremos  2-Llamamos a la función pero con la variable con dato de la class.. |
| Pruebas | | | | |
| def suma(num1,num2):      '''      >>> suma(2,3)      5      '''      return num1+num2  import doctest  doctest.testmod()  Se pueden pasar varias pruebas con varios resultados esperados | | | Definimos la funcion a testear y le pasamos parámetros  Definimos el valor esperado  Importamos y llamamos al método para testear | |
| MODULOS  Usar métodos de otro archio externo al código fuente. Importando exportando e instalando | | | | |
| import saludos  saludos.saludonormal()  saludo.saludoperosonlizado(“jony”) | | | | El archivo tiene que estar en la misma ruta que nuestro código fuente.  Tomo como ejemplo este código: guardado como **saludos.py**  **def saludonormal()**  **print("Hola, buen día.")**    **def saludopersonalizado(nomb):**  **print(f'hola soy {nomb}')**  **##Cada tipo de importación ocupa mas o menos memoria** |
| from saludos import saludonormal  saludonormal() | | | |
| from saludo import \*  saludonormal()  saludopersonalizado(“jony”) | | | |
| from carpeta\_modulos.saludos import \*  saludonormal()  saludopersonalizado(“jony”)  from carpeta01.modulos01.saludos import \* | | | | **Paquetes:** organizar los modulos en un carpeta especial.   * Creamos una carpeta en la mima ruta de nuestro archivo fuente * Dentro de esta carpeta creamos un archivo vacio con el nombre “\_\_init\_\_.py * Dentro de esta carpeta gurdamos todos los modulos que vamos a usar en nuestro programa. |
| Dentro de setup.py  from setuptools import setup  setup(  name="saludosdejony",  version="1.0",  description="prueba de exportación de paquetes.",  author="jonathan",  author\_email="jonyalanoca2@gmail.com",  url="www.google.com",  packages=["modulos","modulos.modempaquetar"]  )  Primera carpeta  Ruta completa del modulo  from *modulos.modempaquetar.*saludos import \*  saludonomal() | | | | **DISTRIBUCIÓN DE PAQUETE:** crear e instalar modulos que serán importados encuentre donde se encuentre dentro de la pc.   1. Guardamos nuestro **modulo (saldos.py)** en una ruta similar.   \proyecto  **setup.py**  \modulos  \_\_init\_\_.py  \modempaquetar  \_\_init\_\_.py  **saludos.py**   1. Creamos un archivo llamado **setup.py** 2. En **cmd**  buscamos la ruta de setup.py   >Python setup.py sdist   1. Dentro de la carpeta del proyecto se crean dos carpetas **(dentro de \dist encontramos el “comprimido” para compartir. )** 2. Instalar un paquete: buscamos la ruta de nuetro comprimido en cmd 3. \dist>pip3 install “nombre del archivo del paquete”   C:\Users\jonyq\Desktop\prueba\dist>pip3 install saludosdejony-1.0.tar.gz   1. Desinstalar un paquete: en cualquier ruta dentro de cmd(solo nombre, sin extensión ):   >pip3 uninstall **saludosdejony** |
| INTERFACE GRAFICA | | | | |
| from tkinter import\*  root=Tk()  root.mainloop() | | | | Importamos la librería tkinter  Creamos el objeto  Mantiene abierta la ventana. |
| Métodos:  root.title(“Mi primer ventana”)  root.resizable(False, False)  root.iconbitmap(“icono.ico”)  root.geometry(“150x150+500+200”)  root.config(bg=”blue”)  root.deiconify()  root.withdraw()  root.destroy()  global segundaventana  segundaventana= Toplevel() | | | | Nombre de la ventana  True o Fase (width, height)  Solo acepta .ico  Tamaño de la ventana, posición de la ventana  Color de la ventana. También acepta hex “**#C3DE81**”  Mostrar la ventana  Ocultar la ventana  Cerra la ventana  Creamos una ventana hijo |
| btn.place() --> btn.place\_forget()  btn.grid() --> btn.grid\_forget()  btn.pack() --> btn.pack\_forget() | | | | Ocultar entry, button, label, etc. |
| marco=Frame(root)  marco=Frame(root, bg=blue, width=500, height=300)  marco.pack()  marco.pack(side=”top”, anchor”e”)  marco.pack(side=”bottom”, anchor”w”)  marco.pack(side=”right”, anchor”n”)  marco.pack(side=”left”, anchor”s”)  marco.pack(fill=”x”)  marco.pack(fill=”y”)  marco.pack(fill=”both”)  marco.pack(fill=”x”, expand=”True”)  marco.config(relief=”groove”)  marco.config(bd=20) | | | | Creamos un frame  Definimos las propiedades junto a la creación. También luego con:  marco.config(bg=”blue”)  Forma de ponerlo dentro de la ventana. PACK es auto ajuste al contenido.  Combinando se puede darle un lugar más preciso.  Se expande horizontalmente.  Verticalmente  Vertical y horizontal  Vertical y centrado  Tipo de borde  Tamaño del borde |
| Label(marco, text”hola”).place(x=100, y=100)  texto=Label(marco, text=”hola”)  texto.place(x=100, y=100)  texto.config(fg=”red”)  texto.config(font=”comics sans”, 18, bold)  texto.config(justify=center) | | | | Creamos un label que no se podrá reutilizar  Place nos permite dar un lugar exacto dándole pixeles en y x |
| mimagen=PhotoImage(file=”imagen.gif”)  Label(miframe,image=mimagen) | | | | Poner una imagen con tkinter+  La imagen tiene que estar en el mismo lugar que el archivo fuente. |
| txtcaja=Entry(miframe)  txtcaja.config(show=”\*”)  txtcaja.config(state=DISABLED)  txtcaja.insert(0,”hola”)  txtcaja.delete(0,END)  txtcaja.grid(row=0, column=0)  txtcaja.grid(row=0, columna=0, columnspan=4)  txtcaja.grid(row=0, column=0, sitcky=e)  txtcaja.grid(row=0, column=0, padx=20, pady=20, ipady=10 | | | | Para ocultar datos dentro del entry  Dehabilitar entrada de texto. state=NORMAL  Insertar texto a la caja. 0 seria el lugar done se inserta. No se combina con state=DISABLED  Eliminar texto, los valores representan la posición inicial y final a eliminar  Metodo para ordenar en forma de tabla.  Expansión de columna. Ocupara 4 columnas con respecto a las demás.  Si la celda es mas grande sitcky le da la posición dentro de ella n s e w (nsew)  Padx pady : son el margen hacia afuera del arco de Entry  Ipadx ipady: son el margen hacia adentro del marco del entry |
| txt\_comentario=Text(frmprincipal, width=20, height=10)  txt\_comentario.grid(row=0, column=0)  miscroll=Scrollbar(frmprincipal, command=txt\_comentario.yview)  miscroll.grid(row=0, column=1, sticky=NSEW)  txt\_comentario.config(yscrollcommand=miscroll.set) | | | | Creamos una caja de comentarios  Creamos la barra de despazamiento. En este caso en y  Lo ponemos al lado de del comentarios . StICKY para que se ajuste a la caja.  Agregarle la funcionalidad de movimiento. |
| btn\_prueba=Button(frmprincipal, text="hola")  btn\_prueba=Button(frmprincipal, text="hola", state=”disabled”)  btn\_prueba=Button(frmprincipal, text="hola", command=root.destroy)  btn\_prueba=Button(frmprincipal, text="hola"),command=nom\_funcion)  btn\_prueba=Button(frmprincipal, text="hola"),command=nom\_funcion(2))  btn\_prueba=Button(frmprincipal, text="hola", command=lambda:func) | | | | Creamos un button  Un botón deshabilitado.  Comando para cerrar la ventana  Comando llama a una función:  def nom\_funcion():  root.destroy()  llamamos a una función y le pasamos un valor.  aun no entendí esta parte pero ayuda a resolver conflicto. |
| Variables de control:  var\_entero=IntVar()  var\_decimal=DoubleVar()  var\_cadena=StringVar()  var\_booleano=BoolenVar() | var\_entero=IntVar(value=1)  var\_decimal=DoubleVar(value=0.2)  var\_cadena=StringVar(value=”jony”)  var\_booleano=BooleanVar(value=True | | | Uso de las variables:  var\_entero.set(20)  var\_entero.set(var\_entero.get()+1)  var\_entero.get()  txt\_pantalla.get() |
| Var1=StringVar(value”jony”)  Txt\_j=Entry(miframe, textvariable=var1) | | | | Varible asociada al Entry, Label |
| var1=IntVar()  rad\_01=Radiobutton(miframe, text="hola", variable= var1, value=1)  rad\_01=Radiobutton(miframe, text="hola", variable= var1, value=1, command=var\_funcion)  rad\_02=Radiobutton(miframe, text="estoy", variable= var1, value=2)  rad\_03=Radiobutton(miframe, text="adios", variable= var1, value=3)  rad\_01.grid()  rad\_01.select()  rad\_01.deselect() | | | | Aca se guarda se guarda el valor del radio seleccionado.  La variable define el grupo de radios  Además llama a una función  Método para seleccionar un radio.  Método para deseleccionar un radio. |
| var1=IntVar()  che01=Checkbutton(miframe, text="hola", variable=var1, onvalue=1, offvalue=2)  che02=Checkbutton(miframe, text="bye", variable=var1, onvalue=3, offvalue=4)  che01.grid() | | | |  |
| barra=Menu(root)  root.config(menu=barra)  espacio\_archivo=Menu(barra, tearoff=False)  barra.add\_cascade(label="Archivo", menu=espacio\_archivo)  espacio\_archivo.add\_command(label="Guardar como…")  espacio\_archivo.add\_command(label="Cerrar", command=root.destroy) | | | | Creamos la barra  Definimos en el root  Creamos el espacio para la pestaña. Tearoff: borrar línea de separación  Definimos la cascada, damos nombre y enlazamos el la barra con archivo  Podemos asignar comandos. |
| from tkinter import messagebox  messagebox.showinfo(title="hola", message="esta")  messagebox.showwarning(title="hola", message="esta")  messagebox.showerror(title="hola", message="esta")  messagebox.askquestion(title="hola", message="esta")  messagebox.askyesno(title="hola", message="esta" , default="no")  messagebox.askokcancel(title="hola", message="esta")  messagebox.askyesnocancel(title="hola", message="esta")  messagebox.askretrycancel(title="hola", message="esta") | | | | Para ver los valores print(messagebox.show…..) o asignar a una variable  Poner en default una opción. |
| from tkinter import filedialog  filedialog.asksaveasfilename()  filedialog.asksaveasfile()  filedialog.askopenfilename((title="Abrir", initialdir= "/", filetypes = (("python files","\*.py"),("all files","\*.\*"))))  filedialog.askopenfile()  filedialog.askdirectory()  filedialog.askopenfilenames()  filedialog.askopenfiles() | | | |  |
| MYSQL | | | | |
| >pip install mysql-connector-python  from mysql.connector import connect  conectar=connect(      host="localhost",      user="root",      password="",      port="3306",      database=""  )  conectar.close() | | | | Instalar mysql para Python. En cmd..  Importar librería.  Datos de la conexión guardados en una variable.  Opcional. El puesto de mysql  Base de dato en donde vamos a trabajar. Opcional depende el caso.  Cerramos la conexión. |
| cursor=conectar.cursor() | | | | Crear un cursor |
| cursor.execute(“..”)  cursor.execute(“SHOW DATABASES”)  for i in cursos()  print(i)  cursor.execute(“CREATE DATABASE NOMBREDELABASE”)  cursor.execute (“CREATE TABLE TABLA1(ID INT(2), NOMBRE VARCHAR(30))”)  var1=” CREATE TABLE TABLA1(ID INT(2), NOMBRE VARCHAR(30))”  cursor.execute (var1) | | | | Dentro van los comandos SQL. Los datos se guardan dentro de cursor  Nos muestra los valores de cursos, en este caso las bases de datos. |
| * cursor.execute(“INSERT INTO TABLA1 VALUES (‘JONY’, 28)”) * var=“INSERT INTO TABLA1 VALUES (‘JONY’, 28)”   cursor.execute(var)   * valores=(‘jonathan’, )   cursor.execute(“INSERT INTO TABLA1 VALUES (NULL, %s )”, valores)   * var=“INSERT INTO TABLA1 VALUES (%s, %s)”   valores=(‘JOSE’, 19)  cursor.execute(var, valores)   * var="INSERT INTO TABLA1 (NOMBRE, EDAD) VALUES(%s, %s)"   valores=[      ('MARCELO', 66),      ('PEDRO', 19),      ('MAXI', 88)  ]  cursor.executemany(var, valores)  conectar.commit() | | | | Insertar valores:  Para pasar una tupla de un solo valor le ponemos la coma al final.  Ponemos NULL en caso de haber una clave primaria con auto\_increment  executemany: lo usamos cuando usamos una LIST,  Con este comando aprobamos los cambios que realizamos en mysql |