



MANUAL TECNICO SIMPLEQL

LFPSS2020 SECCION A

Selvin Orlando Hernández

201700603

CLASES EMPLEADAS EN LA PRACTICA

CLASE *main*

```
import Menu

if __name__ == '__main__':

    Menu.verificarOpcion()
```

En ella se encuentra la invocación del método **verificarOpcion** que se encuentra en la clase menú y que como su nombre lo indica analiza lo que el usuario quiere realizar

Clase *Menu*

En esta clase se pondrá encontrar los métodos que verifican y realizan las tareas elegidas por el usuario como cargar, seleccionar, sumar, etc.

```
import json
import os
import GenerarHTML
loadfile=False
lista = []

def inicio():...

def cargar_datos(ruta):...

def imprimirDatos():...
#funcion para calcular los Maximos
def Maximos(atribMax):
    if atribMax.lower() == "edad":...
    elif atribMax.lower() == "promedio":...

#Funcion para calcular los minimos
def Minimos(atribMax):...

#funcion para contar registros
def contar():...

#funcion SUMADORA
def sumar(atribMax):...

def verificarOpcion():...

while True:...
```

```
#funcion para contar registros
def contar():...

#funcion SUMADORA
def sumar(atribMax):...

def imprimirDatos2(selex):...

# INICIA BLOQUE DE OPCIONES
def verificarOpcion():...

while True:...
```

CLASES EMPLEADAS EN LA PRACTICA

CLASE GenerarHTML

- Esta es la encargada de crear el reporte en HTML y almacenarla en la ubicación de la clase principal

```
ain.py x GenerarHTML.py x Menu.py x
#generando html
def html_create(lista,tope):

    if int(tope)>0 and int(tope) <= len(lista):
        template = open("HTML/template.html", "r")
        output = open("us.html", "w")

        i = 0
        while i < int(tope):

            text = template.read().format(get_num="0", get_nombre="", get_edad="", get_activo="", get_promedio="")
            output.write(text)

            output.writelines("<tr>")
            output.writelines("<th scope = 'row'>%s</th >" % (i + 1))

            output.writelines("<td> %s </td>" % lista[i].get("nombre"))
            output.writelines("<td> %s </td>" % lista[i].get("edad"))
            output.writelines("<td> %s </td>" % lista[i].get("activo"))
            output.writelines("<td> %s </td>" % lista[i].get("promedio"))
            output.writelines("</tr>")

            i += 1
        output.writelines("</tbody>")
        output.writelines("</table>")
    html_create()
```

PRINCIPALES METODOS DE LA CLASE

MENU

Cargar_datos Mediante la librería json podemos importa archivos en un diccionario y mantenerlo en memoria mientras se ejecuta el script

```

12 def cargar_datos(ruta):
13     with open(ruta) as cont:
14         archivo = json.load(cont)
15         # print("*****-----DATOS EN JSON-----*****")
16         for archivo in archivo:
17             # print("Nombre: "+archivo.get('nombre')+ " Edad: "+str(archivo.get('edad'))))
18             global lista
19             lista.append(archivo)
20         # print("La clase de la estructura es: ")
21         # print(type(archivo))
22         # print("*****----- FIN DE DATOS EN JSON-----*****")
23

```

Luego de que se haya cargado en memoria se puede hacer uso de los diferentes métodos como lo son seleccionar, sumar, contar, etc.

Sumar: este método suma los valores de edad o promedio que se encuentren cargados en memoria

```

#funcion SUMADORA
def sumar(atribMax):
    if atribMax.lower() == "edad":
        print("La suma de ", atribMax + " es: ")
        i = 0
        edades = []
        while i < len(lista):
            edad = lista[i].get("edad")
            edades.append(edad)

            i += 1
        print(sum(edades))
    elif atribMax.lower() == "promedio":
        print("La suma de ", atribMax + " es: ")
        i = 0
        proms = []
        while i < len(lista):
            prom = lista[i].get("promedio")
            proms.append(prom)

            i += 1
        print(sum(proms))

```

Contar: este método devuelve la cantidad de registros cargados en memoria, el método únicamente utiliza la función len que retorna dicho valor

verificarOpcion:

Este es el encargado de realizar las distintas operaciones que permite el programa y básicamente es una serie de splits que comparan que comando y que valores se está ingresando

```
# INICIA BLOQUE DE OPCIONES
def verificarOpcion():

    while True:
        inicio()
        cadena = input("$ ")
        global loadfile
        verificar = cadena.split(sep=' ')

        #MENU cargar archivos
        try:
            if verificar[0].lower() == "cargar":
                # print("Esta en la funcion cargar")
                # print(verificar)
                i = 1
                while i < len(verificar):
                    cargar_datos(verificar[i].replace(", ", "") + ".json")
                    # print(verificar[i].replace(" ", ""))
                    i += 1
                print("Archivo(s) cargado(s)")
            elif loadfile==False:
                loadfile = not loadfile
        except:
            verificarOpcion()
```

```

in.py GenerarHTML.py Menu.py
    elif loadfile==False:
        loadfile = not loadfile
    except:
        print("ERROR: Archivo Inexistente, compruebe la escritura del nombre")

#MENU SELECT
    try:

        if verificar[0].lower()=="seleccionar":
            if loadfile==True:
                # print("estas en la func. seleccionar")
                verificar.pop(0)
                eleccion= " ".join(verificar)
                imprimirDatos2(eleccion)
            else:
                print("No se ha cargado ningun archivo")

        except:
            print("ERROR: comando invalido o dato inexisten, vuelva a intentarlo")

#Menu maximos

    try:

        if verificar[0].lower()=="maximo":
            if loadfile==True:

verificarOpcion()

        if verificar[0].lower()=="maximo":
            if loadfile==True:
                print("")
                verificar.pop(0)
                eleccion= " ".join(verificar)
                Maximos(eleccion)
            else:
                print("No se ha cargado ningun archivo")

        except:
            print("ERROR: comando invalido o dato inexisten, vuelva a intentarlo")

#menu minimos
    try:

        if verificar[0].lower()=="minimo":
            if loadfile==True:
                print("")
                verificar.pop(0)
                eleccion= " ".join(verificar)
                Maximos(eleccion)
            else:
                print("No se ha cargado ningun archivo")

        except:
            print("ERROR: comando invalido o dato inexisten, vuelva a intentarlo")

```

```
#menu SUMAR
```

```
try:
```

```
    if verificar[0].lower()=="suma":
```

```
        if loadfile==True:
```

```
            print("")
```

```
            verificar.pop(0)
```

```
            eleccion= " ".join(verificar)
```

```
            sumar(eleccion)
```

```
        else:
```

```
            print("No se ha cargado ningun archivo")
```

```
except:
```

```
    print("ERROR: comando invalido o dato inexisten, vuelva a intentarlo")
```

```
#menu CONTAR
```

```
try:
```

```
    if verificar[0].lower()=="cuenta":
```

```
        if loadfile==True:
```

```
            print("")
```

```
            contar()
```

```
        else:
```

```
            print("No se ha cargado ningun archivo")
```

```
except:
```

```
    print("ERROR: comando invalido o dato inexisten, vuelva a intentarlo")
```

```
#menu REPORTAR
```

```
try:
```

```
    if verificar[0].lower()=="reportar":
```

```
        if loadfile==True:
```

```
            print("reportando")
```

```
            verificar.pop(0)
```

```
            eleccion = " ".join(verificar)
```

```
            print(eleccion)
```

```
            GenerarHTML.html_create(lista,eleccion)
```

```
        else:
```

```
            print("No se ha cargado ningun archivo")
```

```
except:
```

```
    print("ERROR: comando invalido o dato inexisten, vuelva a intentarlo")
```