Manual Técnico Práctica#01 - LFP

parameters_verificator()

Menu()

menu_actions()

Compatibilidad

-Windows 10

-Linux

Lenguaje :

Esta práctica fue diseñada en su totalidad por el lenguaje de programación python, donde se implementaron matrices y otros conceptos propios del lenguaje.

Variables Globales utilizadas

```
chosen_option=""
route =""
original_text=""
course_name=""
parameters=""
shortener_text=""
lines=0
data=[]
numbers_data=[]
asc_data=[]
desc_data=[]
min_data2=[]
max_data=[]
apr_count=0
rep_count=0
avg=0
```

Librerías o importaciones utilizadas

```
from tkinter.filedialog import askopenfilename
```

Funciones Para lectura y edición del ".lfp"

En esta sección se darán a conocer las funciones utilizadas para la lectura y edición del archivo con extensión "lfp".

recieve_document()

```
# Función que recibe la ruta de un archivo.
def receive_document():
    global route
    route = askopenfilename()
```

read_document()

```
# Función que lee el archivo mediante la ruta almacenada, muestra el texto en consola.
def read_document():
    global route,original_text,chosen_option
    archive = open(route,"r")
    original_text=archive.read()
    archive.close()
    route=""
    chosen_option=""
    print("")
    print("")
    print("El texto del documento recolectado es: ")
    print("================================================")
    print(original_text)
    print("================================================")
    print("")
    print("")
    course_name_search(original_text)
    parameters_search(original_text)
    text_shortener(original_text)
```

course_name_search()

```
#Función donde buscamos el nombre de curso, que recibe como parametro el texto inicial
def course_name_search(text):
    global course_name
    for i in text:
        if i=="=":
            break
```

```
        else:
            course_name+=i
    print("")
    print("====================================")
    print("El curso a evaluar es: ")
    print(course_name)
    print("====================================")
```

parameters_search()

```
#Función que busca los parametros del texto original, y los guarda en la variable global
"parameters"
def parameters_search(text):
    text_length=len(text)
    a=text.find("}")
    global parameters
    parameters=text[a+1:text_length]
    print("")
    print("====================================")
    print("Los parámetros del análisis son: ")
    print(parameters)
    print("====================================")
```

text_shortener()

```
#Función que recibe como parámetro el texto original, y lo acorta, eliminando el nombre
#del curso y los símbolos extras:
def text_shortener(text):
    start=text.find("{")
    end=text.find("}")
    global shortener_text,lines
    shortener_text=text[start+2:end]
    not_required='"<>,""'
    for i in not_required:
        shortener_text = shortener_text.replace(i, '')
    print("")
    print("El texto sin simbolos es : ")
    print("================================================")
    print(shortener_text)
    print("================================================")
    lines=0
    students_counter(shortener_text
```

students_counter()

```
#Función que nos permite contar las lineas del texto recortado, esto por consiguiente
#indica el número de estudiantes del curso.
def students_counter(text):
    global lines
    for i in text:
        if i==("\n"):
            lines=lines+1
    print("")
    print("====================================")
    print("El numero total de estudiantes es:")
    print(lines)
    print("====================================")
```

lists_creator()

```
#Función que recibe como parámetro nuestro texto recortado y luego lo introduce en
#una lista.
def lists_creator(text):
    global data
    j=0
    interim=""
    for i in text:
        if i=="\n":
            data.append(interim.split(";"))
            data[j][1]=int(data[j][1])
            interim=""
            j+=1
        else:
            interim+=i
    menu()
```

Funciones para los parámetros

En esta sección se mostrarán las funciones utilizadas para el ordenamiento de los datos de los estudiantes, y realización de todos los parámetros solicitados.

upward()

```
#Función que muestra las notas de manera ascendente
def upward(data):
    global asc_data
    asc_data=data
```

```
    length = len(asc_data)
    for i in range(0, length):
        for j in range(0, length-i-1):
            if (asc_data[j][1] > asc_data[j + 1][1]):
                interim = data[j]
                asc_data[j]= asc_data[j + 1]
                asc_data[j + 1]= interim
    print("")

    print ("====Reporte Notas Ascendentemente====")
    for i in range(len(data)):
        print ( asc_data[i])
    print("=======================================")
```

downward()

```
#Función que muestra las notas de manera descendente
def downward(data):
    global desc_data
    desc_data=data
    length = len(desc_data)
    for i in range(0, length):
        for j in range(0, length-i-1):
            if (desc_data[j][1] < desc_data[j + 1][1]):
                interim = desc_data[j]
                desc_data[j]= desc_data[j + 1]
                desc_data[j + 1]= interim
    print ("")
    print ("====Reporte Notas Descendentemente====")
    for i in range(len(data)):
        print ( desc_data[i])
    print("=======================================")
```

average()

```
#Función que muestra el promedio de las notas obtenidas por los estudiantes del curso
def average():
    global data,avg
    i=1
    column = [fila[i] for fila in data]
    total_sum=0
    total_count=len(column)
    for j in column:
        total_sum = total_sum+ int(j)
    avg=float(total_sum/total_count)
    print("")
```

```
    print("======Reporte Promedio De Notas======")
    print("Suma total de notas: "+str(total_sum))
    print("Número de Notas: "+str(total_count))
    print("El promedio obtenido por los estudiantes del curso "+course_name+" es:")
    print(avg)
    print("=====================================")
```

min()

```
#Función que muestra la nota mínima obtenida, muestra el nombre del alumno y
#la nota obtenida
def min(data):
    global min_data2
    min_data2=data
    length = len(min_data2)
    for i in range(0, length):
        for j in range(0, length-i-1):
            if (min_data2[j][1] > min_data2[j + 1][1]):
                interim = min_data2[j]
                min_data2[j]= min_data2[j + 1]
                min_data2[j + 1]= interim
    print ("")
    print("")
    print("=========Reporte Nota Mínima=========")
    print("La nota mínima fue obtenida por:")
    print("Estudiante  /  Nota obtenida")
    print(*min_data2[0])
    print("=====================================")
```

max()

```
#Función que muestra la nota máxima obtenida, muestra el nombre del alumno y
#la nota obtenida
def max(data):
    global max_data
    max_data=data
    length = len(max_data)
    for i in range(0, length):
        for j in range(0, length-i-1):
            if (max_data[j][1] < max_data[j + 1][1]):
                interim = max_data[j]
                max_data[j]= max_data[j + 1]
                max_data[j + 1]= interim
    print("")
    print("=========Reporte Nota Máximaa=========")
    print("Estudiante  /  Nota obtenida")
```

```
    print(*max_data[0])
    print("====================================")
```

reprobate()

```
#Función que muestra el número de estudiantes que reprobaron el curso
def reprobate():
    global data,rep_count

    promotion=61
    i=1
    column = [fila[i] for fila in data]
    for j in column:
        if not int(j)>=promotion:
            rep_count+=1
    print("")
    print("=========Reporte Alumnos Reprobados=========")
    print("Los alumnos reprobados en el curso: ")
    print(rep_count)
    print("============================================")
```

passed()

```
#Función que muestra el número de estudiantes que aprobaron el curso
def passed():
    global data,apr_count

    promotion=61
    i=1
    column = [fila[i] for fila in data]
    for j in column:
        if int(j)>=promotion:
            apr_count+=1
    print("")
    print("=========Reporte Alumnos Aprobados=========")
    print("Los alumnos aprobados en el curso: ")
    print(apr_count)
    print("============================================")
```

parameters_verificator()

```
#Función que verifica que parametros fueron recibidos para analizarlos y ejecutarlos
def parameters_verificator():
```

```
    global parameters,data
    print("los parametros a analizar son: "+parameters)
    if parameters.find("ASC")>= 0:
        upward(data)
    else:
        print("")
        print("El reporte ASC no fue solicitado")

    if parameters.find("DESC")>= 0:
        downward(data)

    else:
        print("El reporte DESC no fue solicitado")

    if parameters.find("AVG")>= 0:
        average()
    else:
        print("El reporte AVG no fue solicitado")

    if parameters.find("MIN")>= 0:
        min(data)
    else:
        print("El reporte MIN no fue solicitado")

    if parameters.find("MAX")>= 0:
        max(data)
    else:
        print("El reporte MAX no fue solicitado")

    if parameters.find("APR")>= 0:
        passed()
    else:
        print("El reporte APR no fue solicitado")

    if parameters.find("REP")>= 0:
        reprobate()
    else:
        print("El reporte REP no fue solicitado")
    menu()
```

Menu()

Ya conocidas las funciones y las variables globales dentro del algoritmo, el menú dentro del mismo es muy importante para el manejo de datos y orden entre funciones e instrucciones.

```
#Función que muestra en pantalla el menú principal, dicho menú nos dá 4 opciones a escoger.
def menu():
```

```
    print("==================Grades Control==================")
    print("|||1.Cargar Archivo                              |||")
    print("|||2.Mostrar reporte en consola                  |||")
    print("|||3.Exportar Reporte                            |||")
    print("|||4.Salir                                       |||")
    print("====================================================")
    print("")
    global chosen_option
    print("Seleccionar Una opción del menú presionando el número deseado: ")
    chosen_option=input()
    menu_actions(chosen_option)
```

menu_actions()

```
#Función donde el usuario escoge que acción desea realizar.
def menu_actions(option):
    if option =="1":
        receive_document()
        read_document()
    elif option=="2":
        parameters_verificator()
    elif option=="3":
        html_verificator()
    elif option=="4":
        print("")
        print("Fue un gusto realizar los cálculos por usted, ven a utilizarme pronto :)")
        exit()
    else:
        print("")
        print("================================")
        print("Opción invalida, intente de nuevo")
        print("================================")
        print("")
        menu()
```