


## INFORME DE LABORATORIO

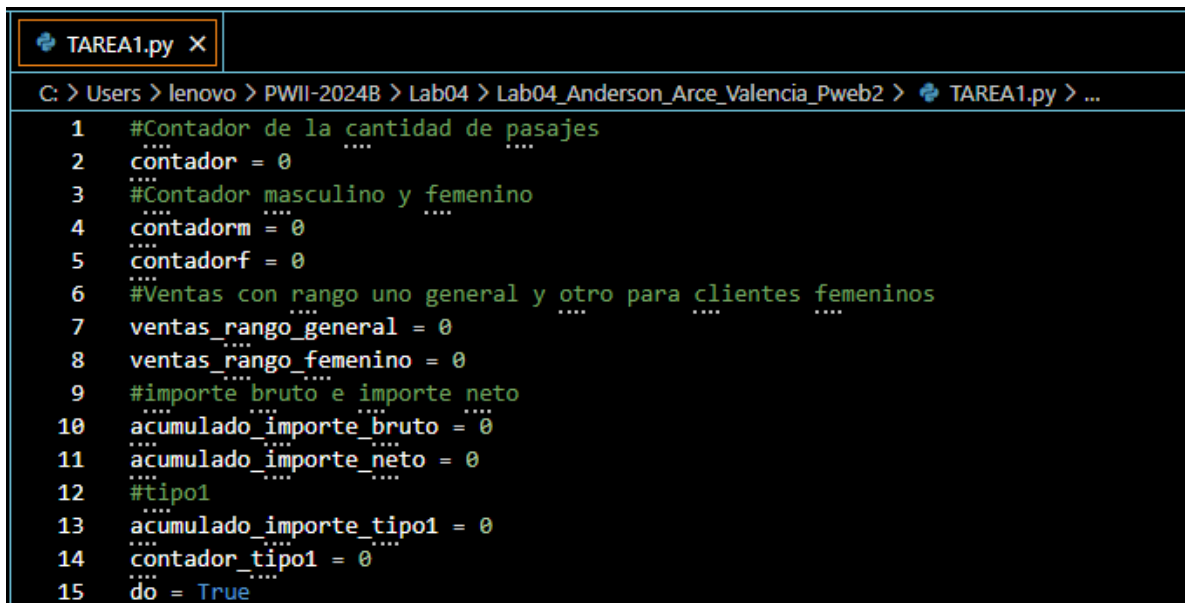
INFORMACIÓN BÁSICA					
ASIGNATURA:	Programación Web 2				
TÍTULO DE LA PRÁCTICA:	Ejercicio de Python				
NÚMERO DE PRÁCTICA:	4	AÑO LECTIVO:	2024	NRO. SEMESTRE:	4
FECHA DE PRESENTACIÓN	11/10/2024	HORA DE PRESENTACIÓN	7/00/00		
Hecho por: - Anderson Arce Valencia				NOTA (0-20)	17
DOCENTE(s): - Lino Pinto Oppe					
NOTA DE RETROALIMENTACIÓN:  AUTOEVALUACION - LAB4 DE ANDERSON ARCE VALENCIA <a href="https://docs.google.com/document/d/15O5GjYqKoS04UIYKrH0XYBG0ZR0B_5vR0NqLIQuV0Hs/edit?usp=sharing">https://docs.google.com/document/d/15O5GjYqKoS04UIYKrH0XYBG0ZR0B_5vR0NqLIQuV0Hs/edit?usp=sharing</a>					

RESULTADOS Y PRUEBAS
<p>I. PREGUNTAS PLANTEADAS:</p> <p>1. ¿Que fue lo nuevo que he utilizado para desarrollar este problema?</p> <ul style="list-style-type: none"><li>• Listas:  He utilizado listas para poder ordenar mejor las variables que he utilizado atraves de un índice, aunque fue mas por gusto personal que por funcionamiento ya que también se podia hacer uso de las variables sin más</li><li>• Match:  Este fue el nuevo tipo de estructura que el profe nos mostro para elegir una opcion lo use convenientemente junto con un while true para que siempre me salga el mismo Menu de opciones</li><li>• continue:  Para mi era necesario hacer que el bucle de verificacion vuelva al inicio si una condicional es erronea para eso utilice continue, en un principio tuve problemas con break</li><li>• while True:</li></ul>

Ya que en python no hay do while para poder reemplazarlo por asi decirlo era utilizando un while infinito y si una condicional es erronea entonces que el true se vuelva False para parar el while, obviamente hay que declarar una variable con valor True para luego modificarlo

## 2. Codigo Fuente(Desarrollo)

### 2.1 Variables Utilizados



```
TAREA1.py X
C: > Users > lenovo > PWII-2024B > Lab04 > Lab04_Anderson_Arce_Valencia_Pweb2 > TAREA1.py > ...
1  #Contador de la cantidad de pasajes
2  contador = 0
3  #Contador masculino y femenino
4  contadorm = 0
5  contadorf = 0
6  #Ventas con rango uno general y otro para clientes femeninos
7  ventas_rango_general = 0
8  ventas_rango_femenino = 0
9  #importe bruto e importe neto
10 acumulado_importe_bruto = 0
11 acumulado_importe_netto = 0
12 #tipo1
13 acumulado_importe_tipo1 = 0
14 contador_tipo1 = 0
15 do = True
```

Al principio de mi codigo solo habia declarado el contador general y el contador de clientes masculinos y femeninos, posteriormente para la opcion 2 tuve que agregar unos 6 mas para cada caso de la opción 2. Ademas habia declarado do = true para mi WHILE infinito

## 2.2 Mi WHILE DO Y MATCH

```
while do:
    print("Empresa de Transporte aéreo")
    print("[1]Registrar venta de pasaje")
    print("[2]Reportar ventas")
    print("[3]Salir")
    opcion = int(input("¿Qué opción vas a elegir?"))
    match opcion:
        case 1:
            while True:
                tipoc = int(input("¿Cuál tipo de cliente eres? 1 o 2: "))
                cantidadp = int(input("¿Cantidad de pasajes? "))
                Genero = input("¿Eres Masculino(m) o Femenino(f)? ").lower()
                serviciot = int(input("¿Tipo de servicio? ([1]-Económica / [2]-Ejecutiva / [3]-Primera clase): "))
                list1 = [tipoc, cantidadp, Genero, serviciot]
                print(list1)
```

Como se observa he utilizado el While “do” para englobar todo el match para que esta se pueda repetir infinitamente hasta que el cliente elija la opcion 3

```
case 3:
    do = False
```

Y se cierra el ciclo

Ademas como se puede observar para el caso 1 utilice otro while true para que pueda verificar los datos ingresados , hasta que todos esten correctos

```

case 1:
while True:
    tipoc = int(input("¿Cuál tipo de cliente eres? 1 o 2: "))
    cantidadp = int(input("¿Cantidad de pasajes? "))
    Genero = input("¿Eres Masculino(m) o Femenino(f)? ").lower()
    serviciot = int(input("¿Tipo de servicio? ([1]-Económica / [2]-Ejecutiva / [3]-Primera clase): "))
    list1 = [tipoc, cantidadp, Genero, serviciot]
    print(list1)

    if list1[1] <= 0:
        print("La cantidad de pasajes no puede ser 0.")
        continue

    if list1[2] != "m" and list1[2] != "f":
        print("Solo puede ser Masculino (m) o Femenino (f).")
        continue

    if list1[3] != 1 and list1[3] != 2 and list1[3] != 3:
        print("Solo puede elegir entre las tres opciones del [1, 3].")
        continue
    else:
        break

```

Si cumple con todas las verificaciones el while se interrumpe con break

## 2.3 LOS SERVICIOS(IMPORTE BRUTO) Y EL CONTADOR DE GENERO

```

    if list1[2] == "m":
        contadorm += 1
    elif list1[2] == "f":
        contadorf += 1

    contador += cantidadp

    if list1[3] == 1:
        economico = 70
        importe_Bruto = cantidadp * economico
    elif list1[3] == 2:
        ejecutivo = 140
        importe_Bruto = cantidadp * ejecutivo
    elif list1[3] == 3:
        primera_Clase = 280
        importe_Bruto = cantidadp * primera_Clase

    acumulado_importe_bruto += importe_Bruto

```

Como se puede ver en la imagen si en el indice 2 de la lista en el que esta los generos es igual a "m" o a "f" se le agrega a dos contadores diferentes, igualmente por si acaso he recogido la cantidad de boletos que se han comprado

Finalmente los if y elif del final evalúan que tipo de servicio quieren si es ejecutivo, economico o primera clase segun eso se obtiene un importe bruto y finalmente se le agrega a la variable de acumulado de importe bruto.

## 2.4 LOS SERVICIOS(DESCUENTO E IMPORTE NETO)

```
if 2 <= list1[1] <= 5:
    desc = importe_Bruto * 0.05
elif 6 <= list1[1] <= 10:
    desc = importe_Bruto * 0.12
elif list1[1] >= 11:
    desc = importe_Bruto * 0.15
else:
    desc = 0

importe_neto = importe_Bruto - desc
acumulado_importe_neto += importe_neto
```

Igualmente según la cantidad de boletos del indice 1 se obtiene un descuento y con ello se obtiene el importe neto. Finalmente se le agrega al acumulado del importe neto.

## 2.5 LAS MEDICIONES FINALES

```
if 70 <= importe_neto <= 500:
    ventas_rango_general += 1
if list1[2] == "f" and 140 <= importe_neto <= 1000:
    ventas_rango_femenino += 1

if list1[0] == 1:
    acumulado_importe_tipo1 += importe_neto
    contador_tipo1 += 1

print("El importe bruto es: " + str(importe_Bruto))
print("El monto de descuento es: " + str(desc))
print("El importe neto es: " + str(importe_neto))
```

Aqui agregado mas condicionales para poder obtener las ventas según un rango pedido y género, de igualmente obtener el importe neto de los clientes tipo1.

## 2.6 LA OPCION 2


```
case 2:
    # Opción 2
    print("Ventas registradas:")
    print("- Cantidad de clientes masculinos: " + str(contadorm))
    print("- Ventas cuyo Importe Neto sea >=70 y <=500: " + str(ventas_rango_general) )
    print("- Ventas de clientes femeninos cuyo Importe Neto sea >=140 y <=1000: " + str(ventas_rango_femenino))
    print("- Acumulado del Importe de Ventas (Bruto): " + str(acumulado_importe_bruto))
    print("- Acumulado del Importe Neto de clientes de tipo 1: " + str(acumulado_importe_tipo1) )
    promedio_tipo1 = acumulado_importe_tipo1 / contador_tipo1
    print("- Promedio de Importe Neto de clientes de tipo 1:" + str(promedio_tipo1))
case 3:
    do = False
```


Aqui solo imprimi en consola todo lo que pedia con las variables que hemos obtenido de las operaciones, por otro lado hay una pequeña operacion para sacar el promedio de importe Neto de los clientes de Tipo1.

## LOS COMMITS REALIZADOS:

### Commits


History for PWII-2024B / Lab04 / Lab04\_Anderson\_Arce\_Valencia\_Pweb2 on `master1`

 All users

 All time

 Commits on Oct 11, 2024

#### He ordenado mejor las variables

 AndersonLinoArceValencia committed 40 minutes ago

e9d6daf   <>

tarea finalizada, lo nuevo que he utilizado en este proyecto fue el atributo continue, el match y la utilizacion de varios contadores para obtener los resultados que pide la opcion 2

 AndersonLinoArceValencia committed 44 minutes ago

bc1c43c   <>

#### Carpeta del lab 4

 AndersonLinoArceValencia committed 1 hour ago

2a1da54   <>

 End of commit history for this file

## REFERENCIAS Y BIBLIOGRAFÍA

*Colocare las referencias utilizadas para el desarrollo de la práctica en formato IEEE*

### RÚBRICA PARA EL CONTENIDO DEL INFORME Y DEMOSTRACIÓN

El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumplió con el ítem correspondiente.

Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.

El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	x	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2		2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
TOTAL		20	7	17	