|  |  |  |
| --- | --- | --- |
| | GUÍA 2.4.2:Guía Ejercicios Resueltos Ciclos For y While | | |
| Sigla | Asignatura | Experiencia de Aprendizaje |
| FPY1101 | Fundamentos de Programación | EA2: Programación de aplicaciones en Python |
| Tiempo | Modalidad de Trabajo | Indicadores de logro |
| 2,5 h | Parejas | IL 2.1 al IL 2.5 |

|  |
| --- |
| **Código QR con relleno sólido**  **Antecedentes generales** |

## Esta guía tiene como objetivo conocer los aspectos generales en Python

### Ser una guía paso a paso para la construcción de programas en Python

### Todos los ejercicios tienen la solución incluida, pero antes de ver la solución, debes resolver por tu cuenta el ejercicio, de esa manera podrás reforzar y aprender. Las soluciones utilízalas para comparar con tus resultados, tomar nota o simplemente como revisión. debes ser consiente y responsable en tu autoaprendizaje.

### Debate con tu docente las respuestas obtenidas, si tienes dudas, recuerda dar aviso y argumentar, los programas no tienen soluciones únicas, teniendo derivaciones o mecanismos distintos de funcionamiento.

### Esta guía puede ser desarrollada en casa, o guiada por el docente, con el fin de seguir un paso a paso y comprender las explicaciones de la o el docente.

|  |
| --- |
| **Lista con relleno sólido Requerimientos para esta actividad** |

## Para el desarrollo de esta actividad deberás disponer de:

## Computador

## Visual Studio Code

|  |  |
| --- | --- |
| **Inteligencia artificial con relleno sólidoActividad** |  |

**EJERCICIO 1**

Realiza construcción de un programa que deba realizar lo siguiente:

Comienza con la inicialización de variables y solicita al usuario la cantidad de bultos. Luego, utiliza un bucle **FOR** para procesar cada bulto, solicitando el peso al usuario y manejando posibles errores (agregar excepciones). Dependiendo del peso ingresado, acumula valores y contadores correspondientes para bultos livianos y normales. Finalmente, imprime el total a pagar por bultos livianos y normales, así como la cantidad de bultos en cada categoría

Una empresa de transporte requiere automatizar sus procesos de cálculo para poder cobrar por la cantidad de paquetes que trae un cliente.

Para calcular el valor total a cobrar y catalogarlo para envío, requiere preguntar el peso de cada bulto y determinar el valor según lo siguiente:

Kilos

Categoría

Valor

1 - 5

Liviana

$1,000

6 - 10

Normal

$2,000

Ejemplo:

Si un cliente ingresa 3 bultos y según sus pesos estos clasifican en 1 liviano y 2 normales, el cliente debe paga $5,000

El sistema debe mostrar lo siguiente:

1 bulto liviano $1,000

2 bultos normales $4,000

Valor total a pagar: $5,000

**Solución**

tieneMasBultos = True

nroBulto = 1

valorPagarPorKilo = 0

valorPesoLiviano = 1000

valorPesoNormal = 4500

totalLiviano = 0

totalNormal = 0

contadorBultosLivianos = 0

contadorBultosNormales = 0

cantidadBultos = int(input("Ingrese cantidad de bultos: "))

for i in range(cantidadBultos):

try:

pesoBulto = int(input(f"Ingrese el peso (1 a 10kg) del bulto Nro. {nroBulto}: "))

except ValueError:

print("Peso del bulto debe estar en el rango de 1 y 10kg.")

continue

if 1 <= pesoBulto <= 5:

totalLiviano += valorPesoLiviano

contadorBultosLivianos += 1

elif 6 <= pesoBulto <= 10:

totalNormal += valorPesoNormal

contadorBultosNormales += 1

else:

print("Peso ingresado incorrecto (1 - 10kg)")

nroBulto += 1

print(f"Total a pagar por bultos livianos: {totalLiviano}")

print(f"Total a pagar por bultos normales: {totalNormal}")

print(f"Cantidad de bultos livianos: {contadorBultosLivianos}")

print(f"Cantidad de bultos normales: {contadorBultosNormales}")

**EJERCICIO 2**

Ahora, realiza la siguiente modificación al programa anterior:

* El bucle For, debe ser reemplazado por una sentencia While, que permita ejecutarse mientras la variable “nroBulto” sea menor o igual a la cantidad de bultos ingresados por el usuario.
* Recuerda incluir sentencias de validación.

**Solución**

tieneMasBultos = True

nroBulto = 1

valorPagarPorKilo = 0

valorPesoLiviano = 1000

valorPesoNormal = 4500

totalLiviano = 0

totalNormal = 0

contadorBultosLivianos = 0

contadorBultosNormales = 0

cantidadBultos = int(input("Ingrese cantidad de bultos: "))

while nroBulto <= cantidadBultos:

try:

pesoBulto = int(input(f"Ingrese el peso (1 a 10kg) del bulto Nro. {nroBulto}: "))

except ValueError:

print("Peso del bulto debe estar en el rango de 1 y 10kg.")

continue

if 1 <= pesoBulto <= 5:

totalLiviano += valorPesoLiviano

contadorBultosLivianos += 1

elif 6 <= pesoBulto <= 10:

totalNormal += valorPesoNormal

contadorBultosNormales += 1

else:

print("Peso ingresado incorrecto (1 - 10kg)")

nroBulto += 1

print(f"Total a pagar por bultos livianos: {totalLiviano}")

print(f"Total a pagar por bultos normales: {totalNormal}")

print(f"Cantidad de bultos livianos: {contadorBultosLivianos}")

print(f"Cantidad de bultos normales: {contadorBultosNormales}")

**EJERCICIO 3**

Construye un programa que tenga como objetivo el solo ser referente para la utilización de captura de errores por medio excepciones, el programa debe capturar error de valores, y división por cero.

try:

# Solicitar al usuario dos números

numerador = float(input("Ingrese el numerador (un número): "))

divisor = float(input("Ingrese el divisor (un número diferente de cero): "))

# Verificar si el divisor es cero

if divisor == 0:

raise ValueError("¡Error! El divisor no puede ser cero.")

# Realizar la división

resultado = numerador / divisor

# Mostrar el resultado

print(f"El resultado de la división es: {resultado}")

except ValueError as ve:

print(f"Error: {ve}")

except ZeroDivisionError:

print("Error: No se puede dividir por cero.")

finally:

print("Fin del programa.")