



# INTRODUCCIÓN A LA PROGRAMACIÓN

## **Control de facturación del consumo de energía para la empresa Sencom**

### **Integrantes:**

Sara Valeria Ruiz Castillo.  
Marco Antonio Salas Fonseca.  
Osman Jatniel Cerpas Alvarado.  
Víctor Mateo Alcocer López.

### **Docente:**

MSc. Silvia Gigdalia Ticay López.

### **Fecha:**

Managua, viernes 4 de Julio de 2025



## Índice

<b>I. Introducción .....</b>	<b>4</b>
<b>II. Planteamiento del problema .....</b>	<b>5</b>
<i>Proceso a automatizar.....</i>	<i>5</i>
<b>III. Objetivos.....</b>	<b>6</b>
Objetivo General .....	6
Objetivos específicos.....	6
<b>IV. Justificación .....</b>	<b>7</b>
<b>V. Análisis del problema .....</b>	<b>8</b>
¿Qué entradas se requieren? .....	8
¿Cuál es la salida deseada?.....	8
¿Qué método produce la salida deseada? .....	8
Requisitos adicionales y restricciones.....	8
Diagrama de estructura.....	9
Requerimientos funcionales .....	10
Entradas requeridas .....	10
Salidas deseadas.....	10
<i>Método que produce la salida deseada.....</i>	<i>11</i>
Validación estricta de entradas: .....	11

Cálculos de producción energética: .....	11
Cálculo de facturación: .....	11
Emisión del reporte: .....	11
Requisitos adicionales y restricciones .....	11
Requerimientos no funcionales. ....	12
<b>VI. Fase Algoritmo .....</b>	<b>14</b>
Definiciones .....	14
Validación de nombre. ....	14
Validación de dirección.....	15
Capacidad planta. ....	15
Tiempo de funcionalidad de la planta. ....	16
Cálculos.....	17
<b>VII. Ejecución del programa. ....</b>	<b>21</b>
Documentación General del Sistema de Monitoreo Inteligente de Energía Renovable	21
1. Módulo interfaz.py .....	21
2. Módulo reporte.py .....	21
3. Módulo usuarios.py .....	22
4. Módulo principal main.py .....	22
Aspectos Técnicos Destacados .....	22
<b>VIII. Conclusión .....</b>	<b>24</b>

<b>IX. Recomendaciones .....</b>	<b>25</b>
<b>X. Anexos.....</b>	<b>26</b>
<b>XI. Referencias bibliográficas.....</b>	<b>31</b>

## **I. Introducción**

En la era digital actual, el desarrollo de software se ha consolidado como una herramienta esencial para optimizar procesos, mejorar la eficiencia y responder a las demandas cambiantes del entorno. Su aplicación se extiende a múltiples áreas, permitiendo resolver problemas concretos mediante soluciones tecnológicas personalizadas. A través de la automatización, se reducen errores, se ahorra tiempo y se facilita la toma de decisiones informadas.

En este contexto, la asignatura “Introducción a la programación” propone como actividad final el desarrollo de un caso de estudio que implique la creación de una aplicación de consola para automatizar un proceso real. El equipo ha identificado una situación que puede beneficiarse de la implementación de una solución informática, con el fin de agilizar tareas actualmente realizadas de forma manual.

La propuesta consiste en diseñar un programa en lenguaje Python que permita aplicar los conocimientos adquiridos durante el curso. Esta solución recogerá datos relevantes, los procesa adecuadamente y generará un resultado útil, demostrando la importancia de implementar el desarrollo de software en escenarios cotidianos.

Más allá de su valor académico, este proyecto representa una oportunidad para fortalecer habilidades técnicas y comprender el impacto práctico que tiene la programación en el mundo real y ofrecer una implementación de herramientas más modernas para el sistema de la empresa Sencom.

## **II. Planteamiento del problema**

En la empresa Sencom, el proceso de facturación del consumo energético se realiza de forma manual, lo que ha generado diversas dificultades. La encargada de esta tarea suele invertir una gran parte de su jornada en realizar cálculos extensos, lo que no solo representa una pérdida de tiempo, sino que también incrementa el riesgo de cometer errores. En ocasiones anteriores, se han producido facturaciones incorrectas, lo que ha generado confusión y desconfianza entre los involucrados.

### ***Proceso a automatizar***

Para resolver esta problemática, se propone automatizar el cálculo del consumo mensual de energía y su equivalente en monto monetario. El sistema desarrollado debe solicitar al usuario datos clave como nombre, dirección, cantidad de meses en funcionamiento y capacidad máxima de generación energética. Con esta información, el programa procesa los datos, validará su formato y generará un reporte final con el monto total a facturar de forma precisa y eficiente.

### **III. Objetivos.**

#### **Objetivo General**

- Diseñar un programa de consola en lenguaje de programación Python para automatizar el proceso de facturación del consumo de energía renovable de la empresa Sencom.

#### **Objetivos específicos**

- Desarrollar un algoritmo que permita mostrar la lógica del problema, para posteriormente, traducirlo a lenguaje de programación python.
- Aplicar conceptos y técnicas de programación para asegurar el funcionamiento eficiente del programa.
- Documentar el desarrollo del proyecto, incluyendo la implementación del código y las decisiones técnicas tomadas con descripciones con el fin de facilitar su comprensión y evaluación.



## **IV. Justificación**

La automatización de procesos se ha convertido en una necesidad clave para mejorar la eficiencia y precisión en las tareas administrativas dentro de cualquier organización. En este sentido, el presente proyecto busca desarrollar una solución informática que permita optimizar el proceso de facturación del consumo energético en la empresa Sencom, el cual actualmente se realiza de manera manual, consumiendo tiempo excesivo y siendo propenso a errores.

Implementar un sistema automatizado no solo facilitará el trabajo del personal encargado, sino que también garantizará cálculos más precisos, reduciendo el margen de error y mejorando la confiabilidad del servicio. Además, este proyecto representa una oportunidad de aplicar conocimientos adquiridos en la asignatura “Introducción a la programación”, mediante la creación de una herramienta útil en un contexto real.

La propuesta aporta valor tanto a nivel académico como profesional, ya que promueve el uso de la tecnología para resolver problemas cotidianos, fomenta el pensamiento lógico y refuerza competencias en el desarrollo de software. Su ejecución tiene el potencial de impactar positivamente en la productividad de la empresa y en la calidad del trabajo del equipo administrativo.

## V. Análisis del problema

### ¿Qué entradas se requieren?

- **Usuario y contraseña:** (texto) Credenciales de acceso previamente establecidas.
- **Datos del cliente:**
  - Nombre (texto)
  - Dirección (texto)
  - Capacidad de la planta (número decimal o entero)
  - Número total de meses de operación (entero positivo)

### ¿Cuál es la salida deseada?

- Código único asignado al cliente.
- Reporte de producción (en pantalla o archivo).
- Cálculo de recuperación de inversión (número o mensaje).
- Gráfico de producción mensual (visualización).
- Guía para instalar la librería matplotlib.

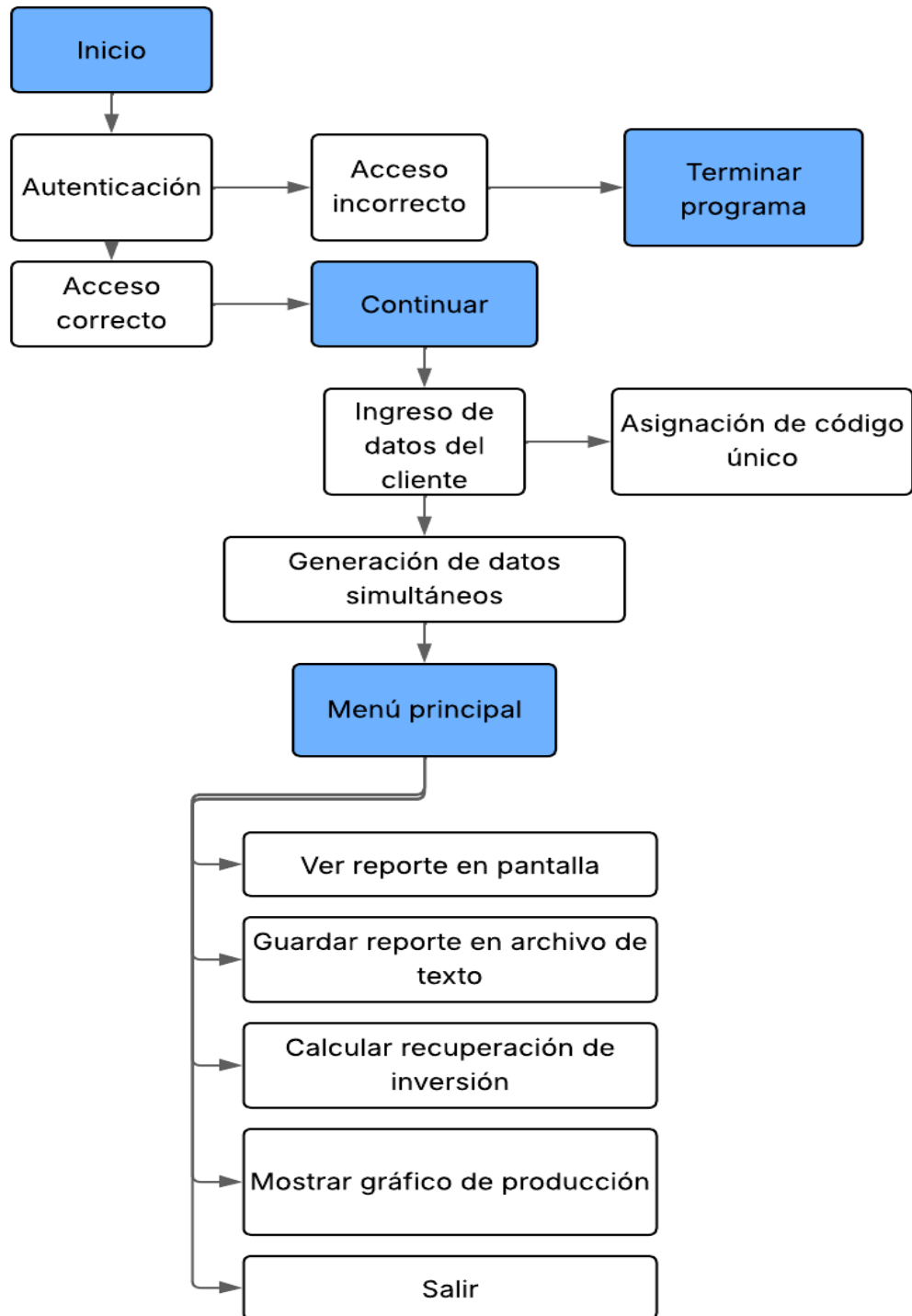
### ¿Qué método produce la salida deseada?

- **Simulación de datos:** como no hay base de datos, los datos de producción mensual se generan de forma simulada.
- **Condicionales y menús:** para seleccionar las opciones deseadas.
- **Gráficos:** uso de la librería matplotlib para generar visualizaciones.
- **Archivo de texto:** guardar resultados mediante escritura en archivos.

### Requisitos adicionales y restricciones

- El acceso está restringido mediante autenticación (usuario y contraseña).
- El menú debe ofrecer seis opciones claras.
- El gráfico solo puede visualizarse si la librería matplotlib está instalada.
- El programa debe generar datos de manera ficticia para compensar la ausencia de base de datos.
- Los reportes deben ser claros y estar correctamente formateados.

### Diagrama de estructura.



## Requerimientos funcionales

### *Entradas requeridas*

Entrada	Tipo de dato	Descripción
Nombre del cliente	Texto (String)	Validado para que solo contenga letras y espacios
Dirección del cliente	Texto (String)	Debe contener solo letras, números y espacios
Capacidad de la planta	Número→ Real	Valor numérico positivo con o sin decimales (se convierte a número)
Meses de funcionamiento	Número → Entero	Cantidad de meses que la planta ha estado operando
Consulta de acumulado (hasta cierto mes)	Número → Entero	Muestra la consulta de facturación de un mes en específico.

### *Salidas deseadas*

El algoritmo genera varios reportes mediante instrucciones, con los siguientes tipos de resultados:

Salida	Tipo de dato	Cantidad
Reporte resumido con información del proveedor y cliente	Texto	1
Producción energética mensual estimada por mes	Real (kWh)	n (número de meses)
Factura mensual actual	Real (USD)	1
Facturación acumulada (general o hasta cierto mes)	Real (USD)	1
Producción acumulada total	Real (kWh)	1
Corte de producción del mes anterior	Real (kWh)	1

### ***Método que produce la salida deseada***

El programa sigue estos pasos principales:

#### ***Validación estricta de entradas:***

Asegura que los datos ingresados sean adecuados en formato y contenido.

#### ***Cálculos de producción energética:***

- Calcula la eficiencia de la planta con un 20% de su capacidad.
- Usa promedio de 4.5 horas solares diarias y 30 días al mes para estimar producción mensual base.
- Aplica una variación aleatoria del  $\pm 5\%$  para simular fluctuaciones reales usando Aleatorio().

#### ***Cálculo de facturación:***

- Multiplica la producción mensual por el precio unitario del kilovatio (\$0.13).
- Suma progresivamente para generar el total acumulado.

#### ***Emisión del reporte:***

Muestra los datos en consola con formato estructurado.

#### ***Requisitos adicionales y restricciones***

- **Validaciones obligatorias** en todas las entradas (no vacías, sin símbolos incorrectos, formato numérico adecuado).

- **Restricciones de tipo:**
  - La capacidad de planta no puede tener más de un punto decimal ni comenzar/terminar con uno.
  - Los meses deben ser enteros positivos.
- **Interfaz de consola:** No incluye interfaz gráfica; toda la interacción es por texto.
- **Persistencia no requerida:** No guarda los datos en archivos (a menos que se agregue en otra fase).
- **Escalabilidad limitada:** Está diseñado como prototipo básico, no multiusuario ni con almacenamiento masivo.

### **Requerimientos no funcionales.**

El sistema muestra una pantalla de bienvenida estilizada con pyfiglet y una barra de carga animada antes de iniciar.

permite registrar nuevos usuarios, almacenando su nombre de usuario y contraseña en el archivo usuarios.txt.

Tras el inicio de sesión exitoso, el sistema solicita datos del cliente: nombre, dirección, capacidad de la planta (kW) y número de meses de operación.

valida que los nombres solo contengan letras y espacios, y que los valores numéricos (enteros y decimales) sean mayores que cero.

calcula la producción energética mensual simulada usando una fórmula basada en eficiencia y horas solares, y acumula los valores mensuales.

permite al usuario ver un reporte resumido en pantalla, mostrando información clave como producción mensual, acumulada y facturación.

permite guardar el reporte en un archivo de texto (.txt), personalizado con el nombre del cliente.

Calcula y muestra el número de meses necesarios para recuperar una inversión, según la facturación mensual promedio.

Ofrece un menú para visualizar un gráfico de barras con la producción mensual usando matplotlib.

El menú principal permite al usuario salir del sistema en cualquier momento, además de repetir acciones sin reiniciar el programa.

El sistema ofrece una interfaz clara en español, con íconos y mensajes amigables para el usuario.

El código utiliza `os.name` para hacer limpieza de pantalla compatible con Windows y Unix (`cls` / `clear`).

Las contraseñas se ingresan de forma oculta utilizando la biblioteca `pwininput`, evitando que se muestren en pantalla.

La información se guarda en archivos de texto (`usuarios.txt` y `reporte_<cliente>.txt`) con codificación utf-8.

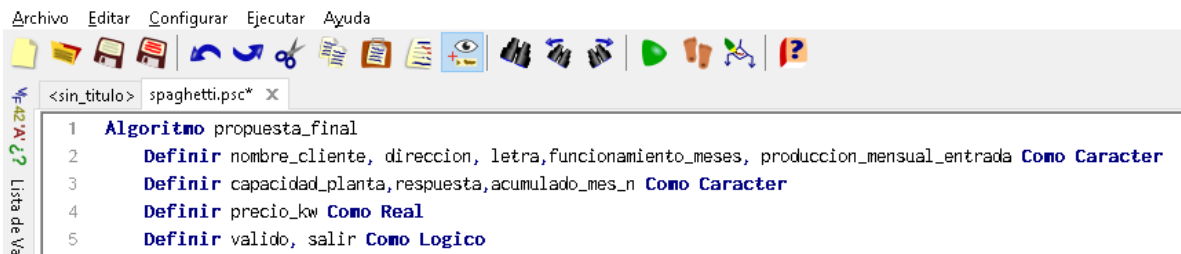
El sistema maneja errores de entrada con validaciones, mostrando mensajes claros sin detener la ejecución.

El código está modularizado en archivos separados (`interfaz.py`, `usuarios.py`, `reporte.py`, `main.py`), lo que mejora la mantenibilidad.

## VI. Fase Algoritmo

### Definiciones

Son las variables a las cuales se le estarán asignando valores a lo largo del proceso del sistema. Se definen respecto al valor que se van a trabajar en ellas como en “nombre\_cliente”, ”dirección” son definidos caracteres porque son datos que se necesitan guardar en formato de carácter porque son nombres.



```
Archivo  Editar  Configurar  Ejecutar  Ayuda
[Icons]
<sin_titulo> spaghetti.psc x
1  Algoritmo propuesta_final
2  Definir nombre_cliente, direccion, letra,funcionamiento_meses, produccion_mensual_entrada Como Caracter
3  Definir capacidad_planta,respuesta,acumulado_mes_n Como Caracter
4  Definir precio_kw Como Real
5  Definir valido, salir Como Logico
```

### Validación de nombre.

Se estarán creando bucles para la correcta digitación de las entradas.

Para “nombre\_cliente” primero se da inicio al bucle por medio de una “llave” que es un valor lógico en una variable.



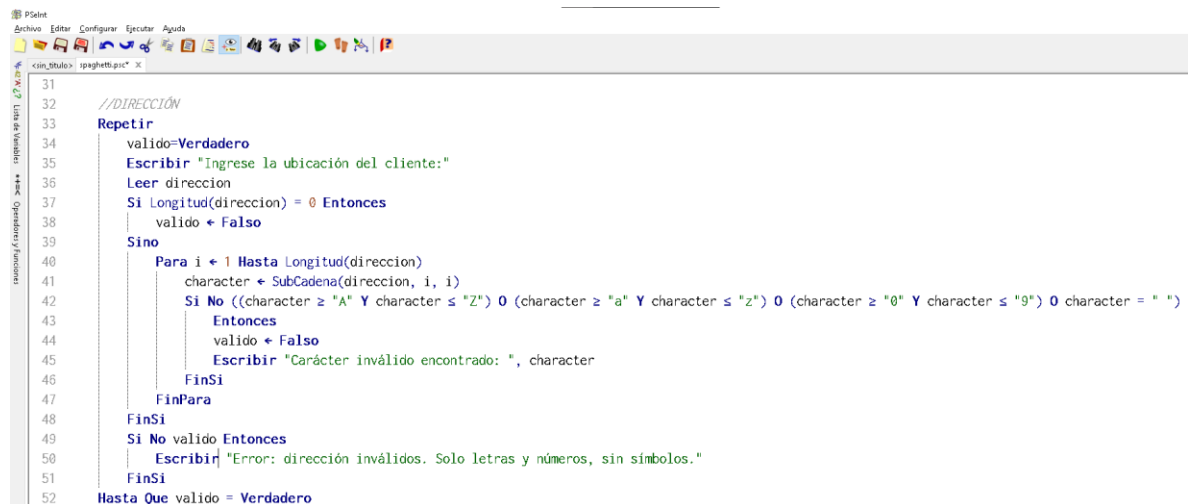
```
PSeInt
Archivo  Editar  Configurar  Ejecutar  Ayuda
[Icons]
<sin_titulo> spaghetti.psc x
10  // VALIDACIÓN DE DATOS PERSONALES
11  Repetir
12      valido ← Verdadero
13
14      Escribir "Ingrese el nombre del cliente:"
15      Leer nombre_cliente
16      Si Longitud(nombre_cliente) = 0 Entonces //Impide que la entrada esté vacía
17          valido ← Falso
18      Sino
19          Para i ← 1 Hasta Longitud(nombre_cliente) //Contador que recorre los caracteres del nombre_cliente
20              letra ← Subcadena(nombre_cliente, i, i)
21              Si No ((letra ≥ "A" Y letra ≤ "Z") O (letra ≥ "a" Y letra ≤ "z")) O letra = "á" O letra = "é" O letra = "í"
22                  valido ← Falso //Validación de caracteres
23              FinSi
24          FinPara
25      FinSi
26      Si no valido Entonces
27          Escribir "Error: Nombre mal colocado, ingrese letras"
28      FinSi
29  Hasta Que valido=Verdadero //Si no se ingresan datos válidos, el bucle se repite
30  //Repetimos las validaciones anteriores en el inciso a continuación y en los que implican solicitar datos al usuario
31
32  //DIRECCIÓN
```



recorre la entrada por la cantidad de caracteres verificando uno por uno que sean todos letras, también tiene su añadido que le permite ingresar letras acentuadas y la “ñ”, para que no den error en el español.

En caso de no ingresar datos válidos, el bucle se repite hasta que el valor de la llave vuelva a ser verdadero para dar fin al bucle y seguir el programa.

### Validación de dirección.



```
31 //DIRECCIÓN
32
33 Repetir
34     valido=Verdadero
35     Escribir "Ingrese la ubicación del cliente:"
36     Leer direccion
37     Si Longitud(direccion) = 0 Entonces
38         valido ← Falso
39     Sino
40         Para i ← 1 Hasta Longitud(direccion)
41             character ← SubCadena(direccion, i, i)
42             Si No ((character ≥ "A" Y character ≤ "Z") O (character ≥ "a" Y character ≤ "z") O (character ≥ "0" Y character ≤ "9") O character = " "))
43                 Entonces
44                     valido ← Falso
45                     Escribir "Carácter inválido encontrado: ", character
46             FinSi
47         FinPara
48     FinSi
49     Si No valido Entonces
50         Escribir "Error: dirección inválidos. Solo letras y números, sin símbolos."
51     FinSi
52 Hasta Que valido = Verdadero
```

El mismo procedimiento sucede con el bucle para digitar correctamente dirección, con el agregado de que uno puede digitar espacios y números.

### Capacidad planta.

Para “capacidad\_planta”, el mismo concepto de la autenticación del dato de entrada es parecida con el añadido de la variable contador puntos para verificar la cantidad en decimal de los KW, evitando errores en la entrada del valor (Más de un punto decimal).

```

53 //CAPACIDAD DE LA PLANTA
54 Repetir
55     valido=Verdadero
56     contadorPuntos=0
57     Escribir "Ingrese la capacidad de producción la planta (en kw): "
58     Leer capacidad_planta
59     Si Longitud(capacidad_planta) = 0 Entonces
60         valido ← Falso
61     Sino
62         Para j ← 1 Hasta Longitud(capacidad_planta)
63             character ← SubCadena(capacidad_planta, j, j)
64
65             Si character = "." Entonces
66                 contadorPuntos ← contadorPuntos + 1
67                 Si contadorPuntos > 1 Entonces
68                     valido ← Falso
69                 FinSi
70             Sino
71                 Si character < "0" o character > "9" Entonces
72                     valido ← Falso
73                 FinSi
74             FinSi
75         FinPara
76     FinSi

```

## Tiempo de funcionalidad de la planta.

---

```

87
88 //TIEMPO DE FUNCIONALIDAD DE LA PLANTA
89 Repetir
90     valido=Verdadero
91     Escribir "Ingrese el número total de meses de operación de la planta eléctrica hasta la fecha actual:"
92     Leer funcionamiento_meses
93     Si Longitud(funcionamiento_meses) = 0 Entonces
94         valido ← Falso
95     Sino
96         Para i ← 1 Hasta Longitud(funcionamiento_meses)
97             character ← SubCadena(funcionamiento_meses, i, i)
98             Si character < "0" o character > "9" Entonces
99                 valido ← Falso
100             FinSi
101         FinPara
102     FinSi
103     Si No valido Entonces
104         Escribir "Entrada no válida. Ingrese solo números enteros positivos, sin decimales ni símbolos."
105     FinSi
106     Hasta Que valido=Verdadero
107     numero_mes=ConvertirANumero(funcionamiento_meses)
108

```

### Cálculos.

```
118  eficiencia=capacidad_planta_numero*0.2
119  horas_solares=4.5
120  produccion_dia=eficiencia*horas_solares
121  produccion_mes=produccion_dia*30
```

Se calcula la eficiencia de la planta (restándole un 80% debido a factores como pérdidas por instalación, eficiencia de paneles e inversores, entre otros).

La eficiencia obtenida se multiplica por las horas solares promedio en que un panel produce energía (4.5 horas estimadas)

Se multiplica lo obtenido para encontrar la producción solar en un mes

Se definen los límites (superiores e inferiores) para posteriormente, utilizar la Función random.

Para definir los límites, se toma en cuenta una desviación estándar del 5%, es decir, que los demás datos de producción mensual pueden variar en un 5% aproximadamente, basados en el valor central (producción en el mes, que lo tomamos como promedio de producción mensual).

Se multiplica la producción solar en un mes por  $(1-0.05)$  para límite inferior, y por  $(1+0.05)$  para el superior

El "1" representa el 100% del valor actual, y como le queremos restar 5%, aplicamos " $(1-0.05)$ ", lo mismo para el límite superior

De esta forma obtenemos valores lógicos generados por random, que mostrarán la producción estimada de cada mes, en función de la capacidad de la planta.

Creamos un arreglo para guardar la producción mensual.

```
124     produccion_acumulada_total=0
125     para i=1 hasta numero_mes
126         produccion_mensual=Aleatorio((produccion_mes*0.95),(produccion_mes*1.05))
127         mes[i]=produccion_mensual
128         produccion_acumulada_total=produccion_acumulada_total+produccion_mensual
129
130     FinPara
131     Fac_acumulada_total=produccion_acumulada_total*precio_kw//la suma de toda la produccion por el precio
132
133     mes_calculado = ((numero_mes - 1) Mod 12) + 1 // Esto ajusta el mes para que esté dentro del rango 1-12
134
```

Luego convertimos las entradas de números asignándoles valor como mes. Y en el segundo bloque calcula cual fue el mes anterior al solicitado.

```
Segun mes_calculado Hacer
1: nombreMes = "Enero"
2: nombreMes = "Febrero"
3: nombreMes = "Marzo"
4: nombreMes = "Abril"
5: nombreMes = "Mayo"
6: nombreMes = "Junio"
7: nombreMes = "Julio"
8: nombreMes = "Agosto"
9: nombreMes = "Septiembre"
10: nombreMes = "Octubre"
11: nombreMes = "Noviembre"
12: nombreMes = "Diciembre"
Fin Segun

mes_anterior ← ((numero_mes - 2 + 12) Mod 12) + 1 //para mostrar el anterior
Segun mes_anterior Hacer
1: nombre = "Enero"
2: nombre = "Febrero"
3: nombre = "Marzo"
4: nombre = "Abril"
5: nombre = "Mayo"
6: nombre = "Junio"
7: nombre = "Julio"
8: nombre = "Agosto"
9: nombre = "Septiembre"
10: nombre = "Octubre"
11: nombre = "Noviembre"
12: nombre = "Diciembre"
Fin Segun
```

El programa pregunta si el usuario quiere ver la facturación acumulada hasta un mes específico.

- Si responde “Sí”, se valida el número del mes ingresado (debe ser un número positivo entre 1 y el mes actual).

- Si la entrada es válida, se calcula la facturación acumulada sumando la producción de cada mes hasta el mes elegido, y se muestra el nombre de ese mes.

- También se genera un reporte con fecha actual, detalles del proveedor, datos del cliente, producción de energía, y facturación correspondiente.

Si el usuario responde “No”, se genera un reporte estándar sin calcular facturación acumulada, solo la producción y cobro del mes actual.

Si la respuesta es inválida o está vacía, se muestra un error y se repite la pregunta.

Y se muestra luego las salidas dependiendo de la decisión de quien trabaja el algoritmo.

```
FinSegun
//Para calcular la fecha actual
fa ← FechaActual() // retorna un solo nro entero en formato AAAAMMDD
anio ← trunc(fa/10000)
mes_fecha ← trunc(fa/100)%100
dia ← fa%100
// MOSTRAR FACTURA CON CONSULTA
Escribir "=====
Escribir "          REPORTE RESUMIDO DE PRODUCCIÓN DE PLANTA SOLAR"
Escribir "=====
Escribir "-----
Escribir "Fecha de emisión del reporte: ", dia, "/", mes_fecha, "/", anio
Escribir "-----
Escribir "-----
Escribir "          DATOS DEL PROVEEDOR          "
Escribir "-----
Escribir "Nombre del proveedor: Ing. Milton Ruiz"
Escribir "Departamento: Ventas"
Escribir "Teléfono: +(505) 22512800"
Escribir "Celular: +(505) 8631 7616"
Escribir "Email: sclientes@sencomca.com"
Escribir "Dirección: km 6 Carretera Norte"
Escribir "-----
Escribir "          DATOS DEL CLIENTE          "
Escribir "-----
Escribir " Cliente: ", nombre_cliente
Escribir " Ubicación: ", direccion
Escribir " Capacidad de la planta: ", capacidad_planta, " kW"
Escribir "-----
Escribir "          DETALLES DE PRODUCCIÓN          "
Escribir "-----
Escribir " Producción acumulada hasta ", nombreMes, ": ", produccion_acumulada_total, " kWh"
Escribir " Corte del mes anterior (", nombre, "): ", produccion_acumulada_total - mes[numero_mes], " kWh"
Escribir " Producción del mes actual (", nombreMes, "): ", mes[numero_mes], " kWh"
Escribir " A facturar: ", mes[numero_mes], " * $0.13          : $", mes[numero_mes] * 0.13
Escribir " Facturación acumulada hasta ", nombreMes, " : $", Fac_acumulada_total
Escribir " Facturación acumulada hasta ", nombreMes, " : $", accumulated_bill * precio_kw
Escribir " * Este valor no incluye IVA *"
Escribir "=====
Escribir ""
```

Sino

```
Si respuesta = "NO" Entonces
    //Para calcular la fecha actual
    fa ← FechaActual() // retorna un solo nro entero en formato AAAAMMDD
    anio ← trunc(fa/10000)
    mes_fecha ← trunc(fa/100)%100
    dia ← fa%100
    // FACTURA NORMAL SIN CONSULTA
    Escribir "=====
    Escribir "          REPORTE RESUMIDO DE PRODUCCIÓN DE PLANTA SOLAR
    Escribir "=====
    Escribir "-----
    Escribir "Fecha de emisión del reporte: ", dia, "/", mes_fecha, "/", anio
    Escribir "-----
    Escribir "-----
    Escribir "          DATOS DEL PROVEEDOR          "
    Escribir "-----
    Escribir "Nombre del proveedor: Ing. Milton Ruiz"
    Escribir "Departamento: Ventas"
    Escribir "Teléfono: +(505) 22512800"
    Escribir "Celular: +(505) 8631 7616"
    Escribir "Email: sclientes@sencomca.com"
    Escribir "Dirección: km 6 Carretera Norte"
    Escribir "-----
    Escribir "          DATOS DEL CLIENTE          "
    Escribir "-----
    Escribir " Cliente: ", nombre_cliente
    Escribir " Ubicación: ", direccion
    Escribir " Capacidad de la planta: ", capacidad_planta, " kW"
    Escribir "-----
    Escribir "          DETALLES DE PRODUCCIÓN          "
    Escribir "-----
    Escribir " Producción acumulada hasta ", nombreMes, ": ", produccion_acumulada_total, " kWh"
    Escribir " Corte del mes anterior (", nombreMes, "): ", produccion_acumulada_total - mes[numero_mes], " kWh"
    Escribir " Producción del mes actual (", nombreMes, "): ", mes[numero_mes], " kWh"
    Escribir " A facturar: ", mes[numero_mes], " * $0.13 : $", mes[numero_mes] * 0.13
    Escribir " Facturación acumulada hasta ", nombreMes, " : $", Fac_acumulada_total
    Escribir " * Este valor no incluye IVA *"
    Escribir "=====
    Escribir ""
FinSi
FinSi
```

La diferencia en las salidas está en la facturación acumulada en cierto mes (especificado por el usuario).

## **VII. Ejecución del programa.**

### **Documentación General del Sistema de Monitoreo Inteligente de Energía Renovable**

El presente sistema fue diseñado con el objetivo de brindar una solución sencilla e interactiva para el monitoreo y reporte de la producción energética en plantas solares. El sistema está implementado en Python y está organizado en varios módulos funcionales: interfaz, reporte, usuarios y un módulo principal main. A continuación, se describe cada uno de ellos, incluyendo su funcionalidad esencial y los aspectos técnicos más relevantes.

#### **1. Módulo interfaz.py**

Este módulo está encargado de la interacción visual inicial con el usuario. Utiliza la librería `pyfiglet` para mostrar una pantalla de bienvenida estilizada, centrando el texto según el tamaño de la terminal del usuario. Posteriormente, presenta una animación de carga simulada a través de una barra progresiva con caracteres, utilizando `time.sleep()` y escritura directa con `sys.stdout`.

Ambas funciones —`bienvenida()` y `barra_de_carga()`— ayudan a brindar una experiencia visual más atractiva y profesional al usuario desde el inicio del programa. Este módulo no interactúa directamente con los datos, sino que establece el tono visual del sistema.

#### **2. Módulo reporte.py**

Este es el núcleo funcional del sistema y contiene tanto las rutinas de entrada y validación de datos como la generación de reportes y análisis de inversión. Al ejecutarse, solicita al usuario el nombre, dirección, capacidad de la planta solar y el número de meses en operación. A partir de estos datos, se simula la producción mensual usando valores aleatorios en un rango del 95% al 105% de la producción teórica estimada.

Dentro del módulo se encuentran funciones que permiten:

- Validar entradas del usuario (nombres, valores numéricos).
- Calcular la producción mensual, acumulada y la facturación correspondiente (con un precio fijo de \$0.15 por kWh).
- Mostrar el reporte en pantalla con información detallada del cliente, producción y facturación.
- Guardar el reporte en un archivo `.txt` con nombre personalizado según el cliente.

- Calcular el tiempo estimado para recuperar la inversión según el promedio de facturación mensual.
- Graficar la producción energética mensual mediante la librería matplotlib.

También incluye un menú interactivo (menu()) que guía al usuario por las distintas opciones disponibles. Si la librería matplotlib no está instalada, el sistema ofrece una guía detallada sobre cómo instalarla.

### **3. Módulo usuarios.py**

Este módulo gestiona la seguridad del sistema mediante autenticación de usuarios. Permite registrar nuevos usuarios, guardar credenciales en un archivo usuarios.txt y luego validar accesos mediante una rutina de inicio de sesión. Usa la librería pwininput para ocultar las contraseñas ingresadas.

El sistema admite hasta tres intentos de autenticación por usuario. Si las credenciales son correctas, se permite el acceso al módulo de reportes. En caso de no existir ningún usuario registrado, el sistema informa y sugiere realizar un registro primero. Este módulo es fundamental para restringir el acceso a los datos e informes generados.

### **4. Módulo principal main.py**

Este archivo es el punto de entrada del programa. Inicializa el entorno visual con las funciones del módulo interfaz, y luego invoca el inicio de sesión a través de usuarios.inicio(). Si la autenticación es exitosa, el sistema continúa con la recopilación de datos e interacción con las funciones del módulo reporte.

También se asegura de limpiar la consola al inicio para brindar una vista más limpia y profesional. Toda la lógica condicional para el flujo del programa se encuentra encapsulada bajo el bloque if `__name__ == "__main__":`, evitando así ejecuciones no deseadas si se importa desde otro script.

### ***Aspectos Técnicos Destacados***

- **Seguridad básica:** Implementación de autenticación mediante archivo local con contraseña oculta.
- **Experiencia de usuario:** Uso de gráficos ASCII y barra de carga para mejorar la interacción visual.



- **Portabilidad:** Funciona tanto en sistemas Windows como Unix/Linux (uso de `os.name`).
- **Persistencia:** Guardado de reportes en formato `.txt` con codificación UTF-8.
- **Manejo de errores:** Validaciones robustas para entradas de texto, enteros y decimales.
- **Visualización de datos:** Generación de gráficas de barras con `matplotlib`.

Este sistema puede ser utilizado como base para aplicaciones más complejas en el monitoreo de energías renovables, incluyendo mejoras como conexión a bases de datos, interfaces gráficas o exportación en formatos como Excel o PDF. Su diseño modular permite que cada componente se desarrolle y mantenga de forma independiente.

## **VIII. Conclusión**

El desarrollo de esta aplicación en Python representa una solución efectiva a la problemática de facturación manual en la empresa Sencom, permitiendo automatizar cálculos, reducir errores y optimizar el tiempo de trabajo. A través de este proyecto, se logró aplicar de forma práctica los conocimientos adquiridos en la asignatura “Introducción a la programación”, demostrando cómo el desarrollo de software puede responder a necesidades reales del entorno laboral. Además de su valor técnico, esta experiencia fortaleció habilidades en la lógica, validación de datos y diseño funcional de soluciones informáticas. La propuesta final no solo mejora la eficiencia operativa, sino que también reafirma el potencial de la programación como herramienta clave en la mejora de procesos empresariales.

## **IX. Recomendaciones**

Para aprovechar al máximo el funcionamiento del programa desarrollado, se recomienda:

**1. Implementar una interfaz gráfica web:**

Se sugiere desarrollar una versión del sistema con una interfaz gráfica a través de una página web, lo cual facilita su uso por parte del personal y permitiría el acceso desde diferentes dispositivos con conexión a internet.

**2. Utilizar bases de datos para el almacenamiento:**

Para mejorar el manejo y conservación de la información a largo plazo, se recomienda integrar una base de datos que permita almacenar los registros de facturación y consumo energético de manera estructurada y segura.

**3. Generar un archivo ejecutable (.exe):**

Convertir el programa a un archivo ejecutable permitiría su instalación y uso en equipos sin necesidad de tener Python configurado, facilitando su distribución y ejecución por parte de usuarios no técnicos.

**4. Agregar generación automática de reportes en PDF o Excel:**

Incorporar la opción de exportar los resultados a formatos como PDF o Excel facilitaría la documentación y el envío de los datos de facturación.

**5. Diseñar una función de respaldo automático de datos:**

Implementar copias de seguridad automáticas garantizaría la protección de los datos en caso de fallos o pérdidas accidentales.

## X. Anexos.

### Módulo interfaz.py.

```
interfaz.py > ...
1  import os
2  from pyfiglet import Figlet
3  import time
4  import sys
5
6  def bienvenida():
7      fuente = Figlet(font="slant") # Podés probar otros estilos: "standard", "big", "banner"
8      texto = fuente.renderText('BIEENVID@S')
9
10     # Centrar cada línea
11     columnas = os.get_terminal_size().columns
12     for linea in texto.splitlines():
13         print(linea.center(columnas))
14     print("⚡ Monitoreo inteligente de energía renovable ⚡".center(columnas))
15
16     time.sleep(3)
17
18     os.system("cls" if os.name == "nt" else "clear")
19
20 def barra_de_carga(tiempo_total=3, pasos=30):
21     print("🔄 Cargando...", end="", flush=True)
22     for i in range(pasos):
23         time.sleep(tiempo_total / pasos)
24         sys.stdout.write("▣")
25         sys.stdout.flush()
26     print(" ✅ Listo\n")
27     os.system("cls" if os.name == "nt" else "clear")
28
```

### Módulo main.py.

```
main.py
1  import usuarios
2  import reporte
3  import interfaz
4  import os
5  os.system("cls" if os.name == "nt" else "clear")
6  if __name__ == "__main__":
7      interfaz.bienvenida()
8      interfaz.barra_de_carga()
9      if usuarios.inicio():
10         reporte.iniciar_reporte()
11     else:
12         print("👋 Saliendo del programa.")
```

## Módulo usuarios.py.

```
usuarios.py > ...
1  import os
2  import pwininput
3
4  def cargar_usuarios():
5      usuarios = {}
6      if os.path.exists("usuarios.txt"):
7          with open("usuarios.txt", "r", encoding="utf-8") as archivo:
8              for linea in archivo:
9                  partes = linea.strip().split(",")
10                 if len(partes) == 2:
11                     usuario, clave = partes
12                     usuarios[usuario] = clave
13     return usuarios
14
15 def registrar_usuarios():
16     usuarios = cargar_usuarios()
17     while True:
18         nombre_usuario = input("Ingrese el nombre de usuario: ")
19         if nombre_usuario in usuarios:
20             print("❌ Ese nombre de usuario ya existe. Intente con otro.")
21             continue
22         clave_usuario = pwininput.pwininput("Ingrese la contraseña: ", mask="*")
23         confirmar = input("¿Desea guardar los cambios? (si/no): ").strip().lower()
24         if confirmar == "si":
25             with open("usuarios.txt", "a", encoding="utf-8") as archivo:
26                 archivo.write(f"{nombre_usuario},{clave_usuario}\n")
27             print("✅ Usuario registrado con éxito.\n")
28             break
29         else:
30             print("Registro cancelado.\n")
31             break
32
33
34 def inicio_sesion():
35     usuarios = cargar_usuarios()
36     if not usuarios:
37         print("⚠ No hay usuarios registrados. Por favor, registre uno primero.")
38         return False
39
40     usuario = input("Usuario: ")
41     intentos = 0
42     while intentos < 3:
43         clave = pwininput.pwininput("Contraseña: ", mask="*")
44         if usuario in usuarios and usuarios[usuario] == clave:
45             print("✅ Acceso permitido.\n")
46             print(f"Bienvenid@ {usuario}")
47             return True
48         else:
49             print("❌ Usuario o contraseña incorrectos.\n")
50             intentos += 1
51     print("🚫 Acceso denegado tras 3 intentos fallidos.\n")
52     return False
```

## Módulo reporte.py.

```
1  from datetime import datetime
2  import matplotlib.pyplot as plt
3  import random
4
5  # ===== FUNCIONES DE VALIDACIÓN =====
6  def entrada_valida_letras(mensaje):
7      while True:
8          nombre = input(mensaje).strip()
9          if all(part.isalpha() for part in nombre.split()) and 3 <= len(nombre.replace(" ", "")) <= 30:
10             return nombre
11             print("❌ Entrada inválida. Solo letras y espacios permitidos (4 a 30 caracteres sin contar espacios).")
12
13  def entrada_valida_direccion():
14      direccion = input("Ingrese la dirección del cliente: ").strip()
15      return direccion
16
17  def entrada_valida_decimal(mensaje):
18      while True:
19          try:
20              valor = float(input(mensaje))
21              if valor > 0:
22                  return valor
23              else:
24                  print("❌ El valor debe ser mayor que 0.")
25          except ValueError:
26              print("❌ Entrada inválida. Ingrese un número decimal.")
27
28  def entrada_valida_entero(mensaje):
29      while True:
30          try:
31              valor = int(input(mensaje))
32              if valor > 0:
33                  return valor
34              else:
35                  print("❌ El número debe ser mayor que 0.")
36          except ValueError:
37              print("❌ Entrada inválida. Ingrese un número entero.")
38
39  def seleccionar_mes(numero):
40      meses = ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
41              "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"]
42      return meses[(numero - 1) % 12]
43
44  def explicar_instalacion_libreria():
45      print("\n===== INSTALACIÓN DE LA LIBRERÍA matplotlib =====")
46      print("Para que el gráfico funcione correctamente, debe instalar la librería matplotlib.")
47      print("📁 Pasos para instalar matplotlib:")
48      print("1. Abra la terminal o símbolo del sistema (cmd).")
49      print("2. Escriba el siguiente comando y presione Enter:")
50      print("    pip install matplotlib")
51      print("⚠️ Si el comando 'pip' no funciona, asegúrese de que Python esté agregado al PATH.")
52      print("    También puede intentar con:")
53      print("    python -m pip install matplotlib")
54      print("✅ Una vez instalado, podrá visualizar los gráficos sin problemas.")
55
```

```

56 # ===== ENTRADA DE DATOS INICIALES =====
57 def iniciar_reporte():
58     global nombre_cliente, direccion, capacidad_planta, numero_mes
59     global produccion_energetica, produccion_acumulada_total, fac_acumulada_total
60     global nombre_mes_actual, nombre_mes_anterior
61     global PRECIO_KW
62
63     # ===== ENTRADA DE DATOS INICIALES =====
64     nombre_cliente = entrada_valida_letras("Ingrese el nombre del cliente: ")
65     direccion = entrada_valida_direccion()
66     capacidad_planta = entrada_valida_decimal("Ingrese la capacidad de producción de la planta (en kW): ")
67     numero_mes = entrada_valida_entero("Ingrese el número total de meses de operación de la planta: ")
68
69     # ===== CÁLCULOS =====
70     PRECIO_KW = 0.15
71     eficiencia = capacidad_planta * 0.2
72     horas_solares = 4.5
73     produccion_dia = eficiencia * horas_solares
74     produccion_mes = produccion_dia * 30
75
76     produccion_energetica = []
77     produccion_acumulada_total = 0
78
79     for _ in range(numero_mes):
80         valor = random.uniform(produccion_mes * 0.95, produccion_mes * 1.05)
81         produccion_energetica.append(valor)
82         produccion_acumulada_total += valor
83         (variable) produccion_acumulada_total: float | Literal[0]
84     fac_acumulada_total = produccion_acumulada_total * PRECIO_KW
85     nombre_mes_actual = seleccionar_mes(numero_mes)
86     nombre_mes_anterior = seleccionar_mes(numero_mes - 1)
87
88     # ===== MENÚ PRINCIPAL =====
89     menu()

```

```

91 # Evita ejecución automática al importar
92 if __name__ == "__main__":
93     iniciar_reporte()
94
95 def encabezado():
96     print("=" * 63)
97     print("      REPORTE RESUMIDO DE PRODUCCIÓN DE PLANTA SOLAR")
98     print("=" * 63)
99     print("Nombre del proveedor: Ing. Milton Ruiz")
100    print("Departamento: Ventas")
101    print("Teléfono: +(505) 22512800")
102    print("Celular: +(505) 8631 7616")
103    print("Email: sclientes@sencomca.com")
104    print("Dirección: km 6 Carretera Norte")
105    print("=" * 63)
106
107 # ===== FUNCIONES DE REPORTE =====
108 def mostrar_reporte_en_pantalla():
109     fecha_actual = datetime.now()
110     dia, mes_fecha, anio = fecha_actual.day, fecha_actual.month, fecha_actual.year
111     encabezado()
112     print("Fecha de emisión del reporte: {}/{}/{}".format(dia, mes_fecha, anio))
113     print("Cliente: {}".format(nombre_cliente))
114     print("Ubicación: {}".format(direccion))
115     print("Capacidad de la planta: {:.2f} kW".format(capacidad_planta))
116     print("Producción acumulada hasta {}: {:.2f} kWh".format(nombre_mes_actual, produccion_acumulada_total))
117     print("Corte del mes anterior ({}): {:.2f} kWh".format(nombre_mes_anterior, produccion_acumulada_total - produccion_energetica[-1]))
118     print("Producción del mes actual ({}): {:.2f} kWh".format(nombre_mes_actual, produccion_energetica[-1]))
119     print("A facturar: ${:.2f}".format(produccion_energetica[-1] * PRECIO_KW))
120     print("Facturación acumulada hasta {}: ${:.2f}".format(nombre_mes_actual, fac_acumulada_total))
121     print("* Este valor no incluye IVA *")
122     print("=" * 63)
123

```

```

124 def guardar_reporte_en_txt():
125     with open("reporte_{}.txt".format(nombre_cliente.replace(" ", "_")), "w", encoding='utf-8') as archivo:
126         fecha_actual = datetime.now()
127         dia, mes_fecha, anio = fecha_actual.day, fecha_actual.month, fecha_actual.year
128         archivo.write("=" * 63 + "\n")
129         archivo.write("                REPORTE RESUMIDO DE PRODUCCIÓN DE PLANTA SOLAR\n")
130         archivo.write("=" * 63 + "\n")
131         archivo.write("Fecha de emisión del reporte: {}/{}/{}\n".format(dia, mes_fecha, anio))
132         archivo.write("Cliente: {}\n".format(nombre_cliente))
133         archivo.write("Ubicación: {}\n".format(direccion))
134         archivo.write("Capacidad de la planta: {:.2f} kW\n".format(capacidad_planta))
135         archivo.write("Producción acumulada hasta {}: {:.2f} kWh\n".format(nombre_mes_actual, produccion_acumulada_total))
136         archivo.write("Producción del mes actual ({}): {:.2f} kWh\n".format(nombre_mes_actual, produccion_energetica[-1]))
137         archivo.write("Facturación acumulada: ${:.2f}\n".format(fac_acumulada_total))
138         archivo.write("* Este valor no incluye IVA *\n")
139     print("✅ Reporte guardado como archivo de texto.")
140
141 def calcular_recuperacion_inversion():
142     inversion = entrada_valida_decimal("Ingrese el monto de la inversión (en dólares): ")
143     facturacion_mensual_prom = fac_acumulada_total / numero_mes
144     if facturacion_mensual_prom == 0:
145         print("❌ La facturación mensual promedio es 0, no se puede calcular la recuperación.")
146         return
147     meses_recuperacion = inversion / facturacion_mensual_prom
148     print(f"\n✅ La inversión de ${inversion:.2f} se recuperaría en aproximadamente {meses_recuperacion:.1f} meses.")
149
150 def graficar_produccion():
151     nombres_meses = [seleccionar_mes(i + 1) for i in range(numero_mes)]
152     plt.figure(figsize=(10, 5))
153     plt.bar(nombres_meses, produccion_energetica, color='skyblue')
154     plt.title("Producción energética mensual")
155     plt.xlabel("Mes")
156     plt.ylabel("Producción (kWh)")
157     plt.xticks(rotation=45)
158     plt.tight_layout()
159     plt.grid(axis='y', linestyle='--', alpha=0.7)
160     plt.show()
161

```

```

162 # ===== MENÚ =====
163 def menu():
164     while True:
165         print("\n===== MENÚ PRINCIPAL =====")
166         print("1. Ver reporte en pantalla")
167         print("2. Guardar reporte en archivo de texto")
168         print("3. Calcular recuperación de inversión")
169         print("4. Ver gráfico de producción mensual (para ver el grafico recuerde tener instalado la libreria matplotlib)")
170         print("5. ¿como instalar la libreria matplotlib?")
171         print("6. salir")
172
173
174         opcion = input("Seleccione una opción (1-6): ")
175
176         if opcion == "1":
177             mostrar_reporte_en_pantalla()
178         elif opcion == "2":
179             guardar_reporte_en_txt()
180         elif opcion == "3":
181             calcular_recuperacion_inversion()
182         elif opcion == "6":
183             print("\n👋 Gracias por usar el sistema. Hasta pronto.")
184             break
185         elif opcion == "5":
186             explicar_instalacion_libreria()
187         elif opcion == "4":
188             graficar_produccion()
189         else:
190             print("❌ Opcion no válida. Intente nuevamente.")

```



## **XI. Referencias bibliográficas.**

[https://www.netacad.com/launch?id=da0847b7-e6fc-4597-bc31-](https://www.netacad.com/launch?id=da0847b7-e6fc-4597-bc31-38ddd6b07a2e&tab=curriculum&view=173610eb-7ec1-5972-ac9a-4e5443ccd67f)

[38ddd6b07a2e&tab=curriculum&view=173610eb-7ec1-5972-ac9a-4e5443ccd67f](https://www.netacad.com/launch?id=da0847b7-e6fc-4597-bc31-38ddd6b07a2e&tab=curriculum&view=173610eb-7ec1-5972-ac9a-4e5443ccd67f)

Python Crash Course A Hands On, Project Based Introduction to Programming - Eric Matthes –  
2nd Edition