

Instalación de PyCharm, y creación del primer proyecto completo

Nombre del alumno: Carly Díaz Gutiérrez

Asignatura: PROGRAMACIÓN



Índice

Introducción.....	3
Objetivos.....	4
Entorno de desarrollo	5
• 3.1. Descarga e instalación de PyCharm	
• 3.2. Tutorial de instalación y configuración	
Creación de un nuevo proyecto.....	6
Análisis de los problemas.....	7
• 5.1. Problema 1: Área del triángulo	
• 5.2. Problema 2: Número par o impar	
Desarrollo de los programas.....	8
• 6.1. Diagrama de flujo y pseudocódigo	
• 6.2. Código Python	
Conclusiones.....	9

3. Introducción

En esta sección, se describe brevemente el contexto del trabajo práctico, la empresa para la que se realiza y los dos programas que se van a desarrollar.

Contexto:

El presente trabajo práctico se desarrolla en el marco de la asignatura de programación. El objetivo es poner en práctica los conocimientos adquiridos sobre la programación en Python, a través del desarrollo de dos programas para una empresa ficticia.

Empresa:

En la empresa para la que se realizan los programas se dedica a desarrollar los siguientes programas:

- **Programa 1:** Cálculo del área de un triángulo.
- **Programa 2:** Determinación si un número es par o impar.

En las próximas secciones, se detallarán los pasos necesarios para la instalación del entorno de desarrollo, la creación de los programas y el análisis de los problemas planteados.

4. Objetivos

- Los objetivos del trabajo práctico son:

Instalar, configurar y usar un entorno de desarrollo en Python.

- Se descargará e instalará PyCharm, una popular herramienta de desarrollo para Python.
- Se configurará PyCharm para un uso óptimo.
- Se aprenderá a usar las principales funcionalidades de PyCharm para crear y ejecutar proyectos en Python.

Crear un nuevo proyecto.

- Se aprenderá a crear un nuevo proyecto en PyCharm.
- Se configurará el proyecto con las opciones adecuadas.
- Se organizarán los archivos del proyecto de forma eficiente.

Declarar y usar variables, constantes y literales.

- Se aprenderán los diferentes tipos de datos en Python.
- Se declararán y usarán variables, constantes y literales de forma correcta.
- Se comprenderá la importancia de usar nombres descriptivos para las variables.

Crear comentarios de código.

- Se aprenderá a crear comentarios de código para mejorar la legibilidad y comprensión del código.
- Se usarán los comentarios para explicar el funcionamiento del código y documentar las decisiones tomadas.

Crear diagramas de flujo de los problemas planteados.

- Se aprenderá a crear diagramas de flujo para representar el flujo de la lógica de un programa.
- Se crearán diagramas de flujo para los dos problemas planteados.

5. Entorno de desarrollo

En esta sección se detallan los pasos para la descarga e instalación de PyCharm, así como el tutorial de instalación y configuración.

5.1. Descarga e instalación de PyCharm

1. Acceder a la página web oficial de PyCharm: <https://www.jetbrains.com/pycharm/download/>
2. Seleccionar la versión **Community** de PyCharm, que es gratuita para uso personal y educativo.
3. Descargar el instalador para el sistema operativo que se esté utilizando (Windows, macOS o Linux).
4. Ejecutar el instalador y seguir las instrucciones en pantalla para completar la instalación.

5.2. Tutorial de instalación y configuración

Una vez instalado PyCharm, se recomienda seguir un tutorial para familiarizarse con la interfaz y las principales funcionalidades del IDE. Se puede encontrar un tutorial oficial en la siguiente página web: https://www.jetbrains.com/guide/python/tutorials/?external_link=true

En el tutorial se aprenderá a:

- Configurar el intérprete de Python.
- Crear un nuevo proyecto.
- Editar y ejecutar código Python.
- Usar el depurador para detectar errores en el código.
- Usar la ventana de ayuda para obtener información sobre las funciones de PyCharm.

Recursos adicionales:

- Documentación oficial de PyCharm: <https://www.jetbrains.com/help/pycharm/>

5.3. Consideraciones adicionales

- Se recomienda instalar la última versión estable de PyCharm para tener acceso a las últimas funcionalidades y correcciones de errores.
- Si se tiene experiencia previa con otros IDEs, es posible encontrar similitudes en la interfaz y funcionalidades de PyCharm.
- Es importante personalizar la configuración de PyCharm para adaptarla a las preferencias personales de cada usuario.

Con la instalación y configuración de PyCharm completa, se está listo para comenzar a crear un nuevo proyecto y desarrollar los programas en Python.

6. Creación de un nuevo proyecto

En esta sección se explica cómo crear un nuevo proyecto en PyCharm.

Pasos para crear un nuevo proyecto:

- Abrir PyCharm.
- En la ventana de bienvenida, seleccionar New Project.
- En la ventana New Project, seleccionar la opción Python en el panel izquierdo.
- Introducir un nombre para el proyecto en el campo Name.
- Seleccionar la ubicación donde se quiere guardar el proyecto en el campo Location.
- Si se desea, se puede seleccionar un intérprete de Python diferente al predeterminado en el campo Python interpreter.
- Hacer clic en Create para crear el proyecto.

Opciones adicionales:

- Se pueden agregar diferentes frameworks o bibliotecas al proyecto durante la creación.
- Se puede crear un proyecto virtual para aislar el entorno de desarrollo del resto del sistema.
- Se puede configurar el proyecto para usar un sistema de control de versiones como Git.

Recomendaciones:

- Es recomendable usar un nombre descriptivo para el proyecto que refleje su contenido.
- Guardar el proyecto en una ubicación accesible y organizada.
- Seleccionar el intérprete de Python que se va a utilizar para el desarrollo del proyecto.
- Con la creación del proyecto completada, se tiene un espacio de trabajo para comenzar a desarrollar los programas en Python.

7. Análisis de los problemas

En esta sección se describe el análisis de los dos problemas planteados:

7.1. Problema 1: Área del triángulo

Entradas:

- **Base:** La base del triángulo (un número real).
- **Altura:** La altura del triángulo (un número real).

Salidas:

- **Área:** El área del triángulo (un número real).

Fórmula:

$$\text{Área} = (\text{Base} * \text{Altura}) / 2$$

Restricciones:

- La base y la altura del triángulo deben ser números positivos.

7.2. Problema 2: Número par o impar

Entrada:

- **Número:** Un número entero.

Salida:

- Un mensaje que indica si el número es par o impar.

Algoritmo:

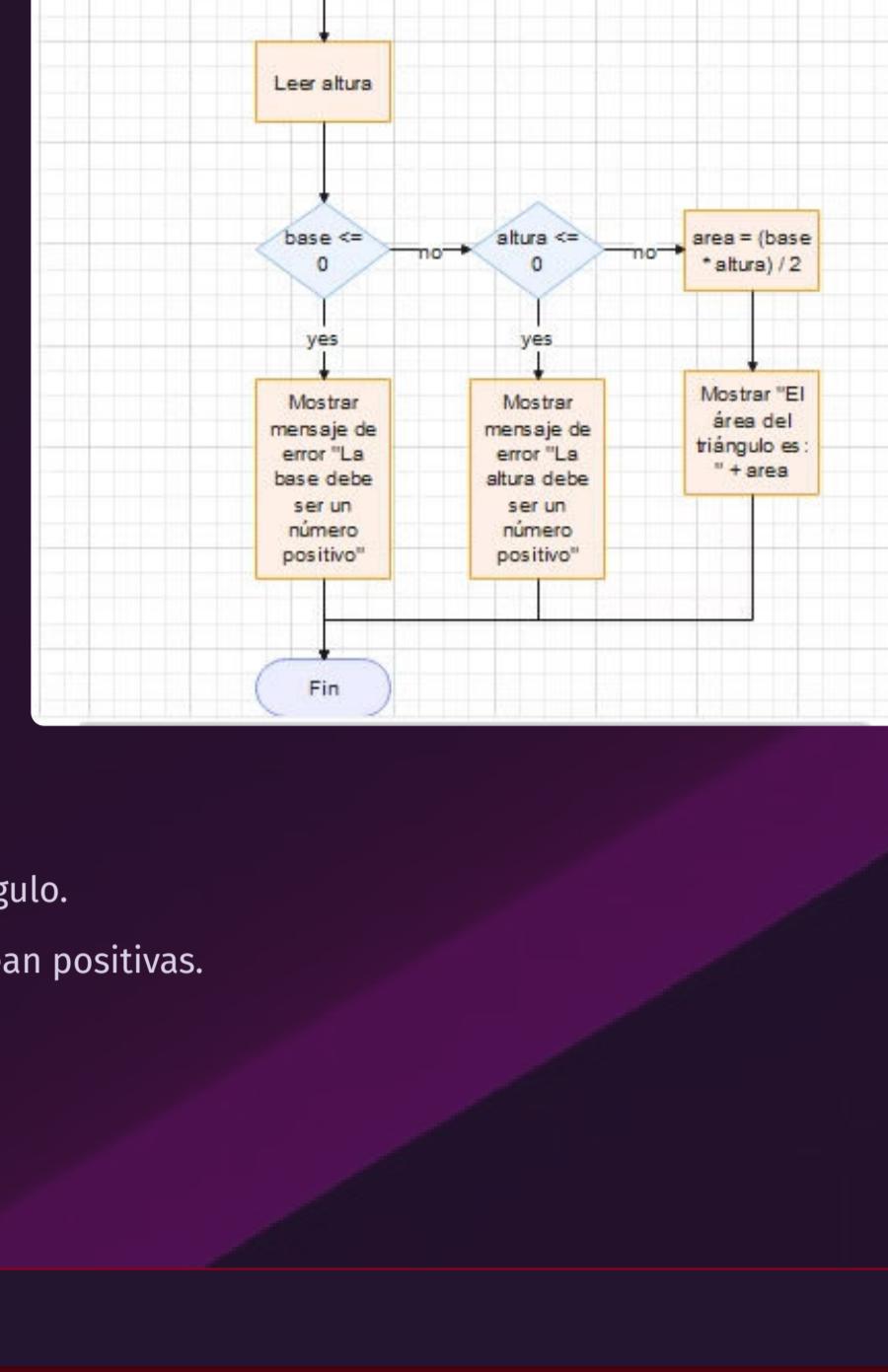
1. **Dividir** el número por 2.
2. **Obtener el resto** de la división.
3. **Si el resto es 0:**
 - Mostrar un mensaje que indica que el número es **par**.
4. **Si el resto es 1:**
 - Mostrar un mensaje que indica que el número es **ímpar**.

8. Desarrollo de los programas

En esta sección se presenta el desarrollo de los dos programas en Python, incluyendo diagramas de flujo, pseudocódigo y código Python.

8.1. Programa 1: Área del triángulo

Diagrama de flujo:



Pseudocódigo:

1. **Leer** la base y la altura del triángulo.
2. **Validar** que la base y la altura sean positivas.
3. **Calcular** el área del triángulo.
4. **Mostrar** el área del triángulo.

Código Python:

```
def area_triangulo(base, altura):
    """
    Calcula el área de un triángulo.

    Parámetros:
    base: La base del triángulo (un número real).
    altura: La altura del triángulo (un número real).

    Retorno:
    El área del triángulo (un número real).
    """

    if base <= 0:
        raise ValueError("La base debe ser un número positivo")
    if altura <= 0:
        raise ValueError("La altura debe ser un número positivo")

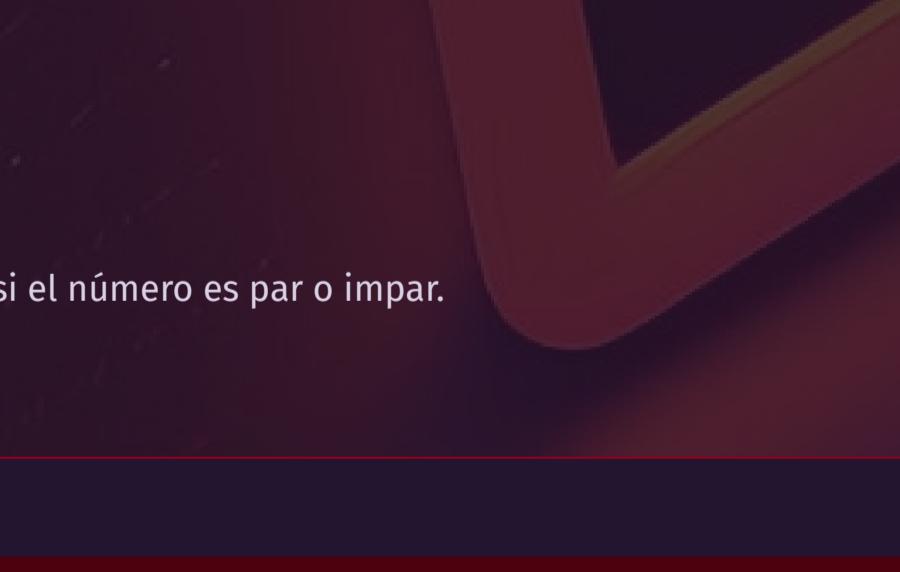
    area = (base * altura) / 2
    return area

# Leer la base y la altura del triángulo
base = float(input("Introduzca la base del triángulo: "))
altura = float(input("Introduzca la altura del triángulo: "))

# Calcular el área del triángulo
try:
    area = area_triangulo(base, altura)
except ValueError as error:
    print(error)
else:
    print(f"El área del triángulo es: {area}")
```

8.2. Programa 2: Número par o impar

Diagrama de flujo:



Pseudocódigo:

1. **Leer** un número.
2. **Dividir** el número por 2.
3. **Comprobar** si el resto es 0.
4. **Mostrar** un mensaje que indica si el número es par o impar.

Código Python:

```
def es_par(numero):
    """
    Determina si un número es par o impar.

    Parámetros:
    numero: Un número entero.

    Retorno:
    True si el número es par, False si es impar.
    """

    resto = numero % 2
    return resto == 0
```

```
# Leer un número
numero = int(input("Introduzca un número: "))
```

```
# Determinar si el número es par o impar
if es_par(numero):
    print("El número es par")
else:
    print("El número es impar")
```

9. Conclusiones

En este trabajo práctico se ha desarrollado dos programas en Python para resolver dos problemas planteados: calcular el área de un triángulo y determinar si un número es par o impar.

Se han logrado los siguientes objetivos:

- **Instalación y configuración de PyCharm:** Se ha instalado y configurado PyCharm, un entorno de desarrollo para Python.
- **Creación de un nuevo proyecto:** Se ha creado un nuevo proyecto en PyCharm para cada programa.
- **Análisis de los problemas:** Se ha realizado un análisis de los dos problemas planteados, incluyendo la identificación de las entradas, salidas, restricciones y algoritmos para resolverlos.
- **Desarrollo de los programas:** Se ha desarrollado el código Python para cada programa, incluyendo diagramas de flujo, pseudocódigo y código Python.

A través de este trabajo práctico se ha puesto en práctica los conocimientos adquiridos sobre la programación en Python, incluyendo:

- Declaración y uso de variables, constantes y literales.
- Creación de comentarios de código.
- Uso de estructuras condicionales.
- Funciones y módulos.

En general, el trabajo práctico ha sido una experiencia enriquecedora que ha permitido:

- Desarrollar las habilidades básicas para la programación en Python.
- Aplicar los conocimientos teóricos a la resolución de problemas prácticos.
- Mejorar la capacidad de análisis y resolución de problemas.
- Familiarizarse con el uso de un entorno de desarrollo como PyCharm.

Se considera que los objetivos del trabajo práctico se han cumplido satisfactoriamente.

Recomendaciones para futuros trabajos:

- Implementar pruebas unitarias para los programas desarrollados.
- Desarrollar programas más complejos que aborden problemas más desafiantes.
- Explorar otras funcionalidades de Python como la programación orientada a objetos.
- Investigar y utilizar bibliotecas de Python para tareas específicas.

Este trabajo práctico ha sentado las bases para un mayor aprendizaje y desarrollo en el campo de la programación en Python.

Anexo:

Este proyecto ha sido subido a la plataforma [git](#) para el dominio público.