


# "TecnoSolución"



Unidad 3: Identificación de los elementos de un programa informático

Título del trabajo práctico: Modelado de un objeto cotidiano.

Creado por: Carly Díaz Gutiérrez

Fecha de realización: 17/03/24

# Índice:

1. Portada
2. Índice:
3. Introducción
4. Desarrollo:
  - a. 3.1. Implementación de la clase PC
  - b. 3.2. Simulación de la reparación de ordenadores
5. Diagramas de flujo
6. Pseudocódigo del proyecto
7. Conclusión
8. Recursos: 8.Dominio público



# Introducción:

## Simulación de un taller de reparación de ordenadores portátiles

En el corazón de la ciudad, entre calles bulliciosas y edificios históricos, se encuentra un pequeño taller de reparación de ordenadores portátiles. Un lugar donde la tecnología cobra vida de nuevo, donde las teclas que no responden vuelven a sonar y las pantallas que se apagan se iluminan con renovada energía.

En este taller, la herramienta principal no es un destornillador o un soldador, sino una poderosa clase de Python: **PC**. Esta clase, diseñada con precisión y creatividad, permite a los técnicos del taller gestionar de forma eficiente la información de los ordenadores que les traen los clientes, desde sus características técnicas hasta el estado de la reparación.

El objetivo de este trabajo práctico es **implementar la clase PC** y utilizarla para simular el proceso de reparación de ordenadores portátiles en el taller. A través de la creación de diferentes instancias de la clase y la implementación de métodos para la reparación y la gestión de información, se podrá:

- **Registrar la información de los ordenadores:** marca, modelo, procesador, memoria RAM, almacenamiento, estado, avería, fecha de entrada y fecha de salida.
- **Simular el proceso de reparación:** cambio de estado, descripción de la avería, tiempo de reparación y notificación al cliente.
- **Generar informes:** información de los ordenadores reparados, averías más comunes, tiempo de reparación promedio, etc.

El desarrollo de este trabajo práctico permitirá no solo poner en práctica los conocimientos de Python, sino también explorar la **creatividad** en la simulación del taller de reparación. Se podrán implementar diferentes funcionalidades para que el taller sea más eficiente y ofrecer un mejor servicio a los clientes.

Además de la implementación de la clase PC, se diseñarán **diagramas de flujo** para los procesos de recepción y reparación de ordenadores, mejorando la comprensión del flujo de trabajo y facilitando la identificación de posibles errores.

En definitiva, este trabajo práctico será una **experiencia de aprendizaje** completa, donde se combinarán la programación, la creatividad y la simulación de un escenario real.



Desarrollo:

3.1. Implementación de la clase PC:

```
from datetime import date
import time

class PC:

    def __init__(self, marca, modelo, procesador, memoria_ram, almacenamiento,
estado="Pendiente de revisión", averia="", fecha_entrada=None, fecha_salida=None):
        """
        Constructor de la clase PC.

        Parámetros:
            marca (str): Marca del ordenador.
            modelo (str): Modelo del ordenador.
            procesador (int): Número de núcleos del procesador.
            memoria_ram (int): Capacidad de memoria RAM en GB.
            almacenamiento (int): Capacidad de almacenamiento en GB.
            estado (str): Estado actual del ordenador ("Pendiente de revisión", "En reparación",
"Reparado").
            averia (str): Descripción de la avería (opcional).
            fecha_entrada (date): Fecha de entrada del ordenador al taller (opcional).
            fecha_salida (date): Fecha de salida del ordenador del taller (opcional).
        """

        self.marca = marca
        self.modelo = modelo
        self.procesador = procesador
        self.memoria_ram = memoria_ram
        self.almacenamiento = almacenamiento
        self.estado = estado
        self.averia = averia
        self.fecha_entrada = fecha_entrada
        self.fecha_salida = fecha_salida

    def reparar(self, averia):
        """
        Simula la reparación del ordenador.

        Parámetros:
            averia (str): Descripción de la avería.

        Retorno:
            None.
        """

        self.averia = averia
        self.estado = "En reparación"
        # Simulación del proceso de reparación (puede ser un simple delay)
        time.sleep(2)
        self.estado = "Reparado"
        self.fecha_salida = date.today()

    def mostrar_informacion(self):
        """
        Muestra la información del ordenador.
        """

        print(f"***Información del ordenador:**")
        print(f"- Marca: {self.marca}")
        print(f"- Modelo: {self.modelo}")
        print(f"- Procesador: {self.procesador} núcleos")
        print(f"- Memoria RAM: {self.memoria_ram} GB")
        print(f"- Almacenamiento: {self.almacenamiento} GB")
        print(f"- Estado: {self.estado}")
        if self.averia:
            print(f"- Avería: {self.averia}")
        if self.fecha_entrada:
            print(f"- Fecha de entrada: {self.fecha_entrada}")
        if self.fecha_salida:
            print(f"- Fecha de salida: {self.fecha_salida}")

    def get_estado(self):
        """
        Devuelve el estado del ordenador.

        Retorno:
            str: Estado del ordenador.
        """

        return self.estado

    def set_estado(self, nuevo_estado):
        """
        Cambia el estado del ordenador.

        Parámetros:
            nuevo_estado (str): Nuevo estado del ordenador.

        Retorno:
            None.
        """

        self.estado = nuevo_estado
```

```
1  from datetime import date
2  import time
3
4  class PC:
5
6      def __init__(self, marca, modelo, procesador, memoria_ram, almacenamiento, estado="Pendiente de revisión", av
7          """
8          Constructor de la clase PC.
9
10         Parámetros:
11             marca (str): Marca del ordenador.
12             modelo (str): Modelo del ordenador.
13             procesador (int): Número de núcleos del procesador.
14             memoria_ram (int): Capacidad de memoria RAM en GB.
15             almacenamiento (int): Capacidad de almacenamiento en GB.
16             estado (str): Estado actual del ordenador ("Pendiente de revisión", "En reparación", "Reparado").
17             averia (str): Descripción de la avería (opcional).
18             fecha_entrada (date): Fecha de entrada del ordenador al taller (opcional).
19             fecha_salida (date): Fecha de salida del ordenador del taller (opcional).
20         """
21         self.marca = marca
22         self.modelo = modelo
23         self.procesador = procesador
24         self.memoria_ram = memoria_ram
25         self.almacenamiento = almacenamiento
26         self.estado = estado
27         self.averia = averia
28         self.fecha_entrada = fecha_entrada
29         self.fecha_salida = fecha_salida
30
31     def reparar(self, averia):
32         """
33         Simula la reparación del ordenador.
34
35         Parámetros:
36             averia (str): Descripción de la avería.
37
38         Retorno:
39             None.
40         """
41         self.averia = averia
42         self.estado = "En reparación"
43         # Simulación del proceso de reparación (puede ser un simple delay)
44         time.sleep(2)
45         self.estado = "Reparado"
46         self.fecha_salida = date.today()
47
48     def mostrar_informacion(self):
49         """
50         Muestra la información del ordenador.
51         """
52         print(f"***Información del ordenador:**")
53         print(f"- Marca: {self.marca}")
54         print(f"- Modelo: {self.modelo}")
55         print(f"- Procesador: {self.procesador} núcleos")
56         print(f"- Memoria RAM: {self.memoria_ram} GB")
57         print(f"- Almacenamiento: {self.almacenamiento} GB")
58         print(f"- Estado: {self.estado}")
59         if self.averia:
60             print(f"- Avería: {self.averia}")
61         if self.fecha_entrada:
62             print(f"- Fecha de entrada: {self.fecha_entrada}")
63         if self.fecha_salida:
64             print(f"- Fecha de salida: {self.fecha_salida}")
65
66     def get_estado(self):
67         """
68         Devuelve el estado del ordenador.
69
70         Retorno:
71             str: Estado del ordenador.
72         """
73         return self.estado
74
75     def set_estado(self, nuevo_estado):
76         """
77         Cambia el estado del ordenador.
78
79         Parámetros:
80             nuevo_estado (str): Nuevo estado del ordenador.
81
82         Retorno:
83             None.
84         """
85         self.estado = nuevo_estado
86
87     # Creación de 5 instancias de la clase PC
88     pc1 = PC("Asus", "VivoBook 15", 8, 16, 512)
89     pc2 = PC("Lenovo", "IdeaPad 3", 6, 8, 256)
90     pc3 = PC("HP", "Pavilion 14", 4, 4, 128)
91     pc4 = PC("Acer", "Aspire 5", 8, 12, 512)
92     pc5 = PC("Dell", "Inspiron 15", 6, 8, 256)
93
94     # Simulación de la reparación de 3 ordenadores
95     pc1.reparar("Pantalla rota")
96     pc3.reparar("Batería agotada")
97     pc5.reparar("Teclado no funciona")
98
99     # Impresión de la información de los ordenadores antes y después de la reparación
100     for pc in (pc1, pc3, pc5):
101         print(f"***Información antes de la reparación:**")
102         pc.mostrar_informacion()
103
104     print()
105     pc.reparar("Simulación de reparación")
106     print(f"***Información después de la reparación:**")
107     pc.mostrar_informacion()
108     print()
109
```

Explicación del pseudocódigo:

- La clase PC define los atributos y métodos para gestionar la información de un ordenador.
- La función reparar simula la reparación del ordenador actualizando su estado y fecha de salida.
- La función mostrar\_informacion muestra los datos del ordenador.
- Las funciones get\_estado y set\_estado permiten obtener y cambiar el estado del ordenador.

Creación de 5 instancias de la clase PC:

```
pc1 = PC("Asus", "VivoBook 15", 8, 16, 512)
pc2 = PC("Lenovo", "IdeaPad 3", 6, 8, 256)
pc3 = PC("HP", "Pavilion 14", 4, 4, 128)
pc4 = PC("Acer", "Aspire 5", 8, 12, 512)
pc5 = PC("Dell", "Inspiron 15", 6, 8, 256)
```



## Simulación de la reparación de ordenadores:

```
pc1.reparar("Pantalla rota")
pc3.reparar("Batería agotada")
pc5.reparar("Teclado no funciona")

# Impresión de la información de los ordenadores antes y después de la reparación
for pc in [pc1, pc3, pc5]:
    print(f"**Información antes de la reparación:**")
    pc.mostrar_informacion()
    print()
    pc.reparar("Simulación de reparación")
    print(f"**Información después de la reparación:**")
    pc.mostrar_informacion()
    print
```

### Funciones:

- `reparar(averia)`: Simula la reparación del ordenador con la descripción de la avería.
- `mostrar_informacion()`: Muestra la información del ordenador.
- `get_estado()`: Devuelve el estado del ordenador.
- `set_estado(nuevo_estado)`: Cambia el estado del ordenador.

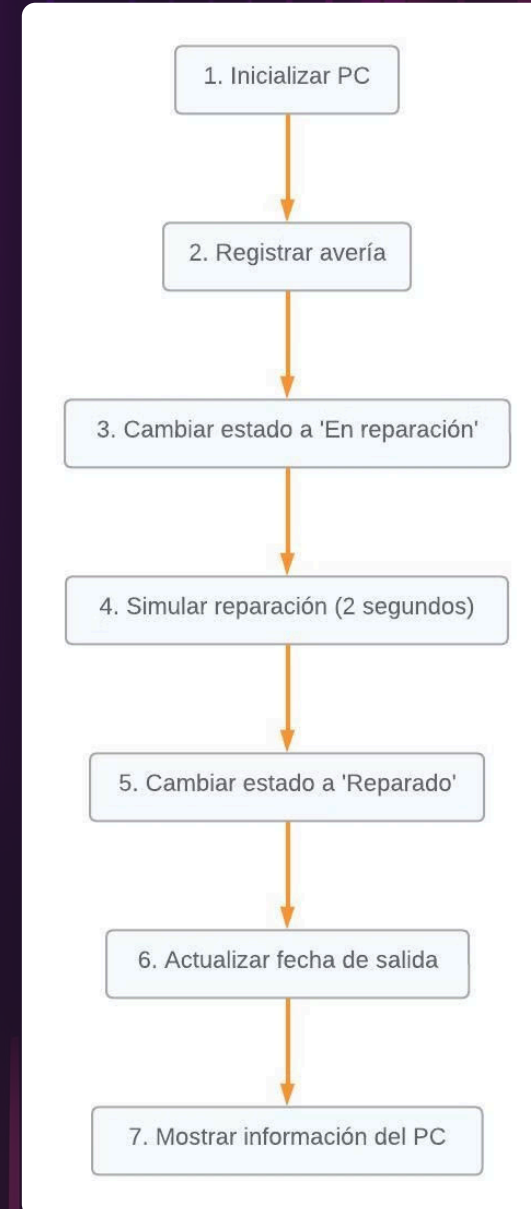
### Atributos:

- `marca`: Marca del ordenador.
- `modelo`: Modelo del ordenador.
- `procesador`: Número de núcleos del procesador.
- `memoria_ram`: Capacidad de memoria RAM en GB.
- `almacenamiento`: Capacidad de almacenamiento en GB.
- `estado`: Estado actual del ordenador ("Pendiente de revisión", "En reparación", "Reparado").
- `averia`: Descripción de la avería (opcional).
- `fecha_entrada`: Fecha de entrada del ordenador al taller (opcional).
- `fecha_salida`: Fecha de salida del ordenador del taller (opcional).

# Diagramas de flujo:

## Explicación del diagrama de flujo:

1. El proceso comienza con la creación de un objeto de la clase PC.
2. Se llama a la función **reparar** con la descripción de la avería.
3. La función **reparar** cambia el estado del ordenador a "En reparación".
4. Se simula el proceso de reparación con un retraso de 2 segundos.
5. Se cambia el estado del ordenador a "Reparado" y se actualiza la fecha de salida.
6. Actualiza la fecha de reparación del Pc.
7. Se llama a la función **mostrar\_informacion** para mostrar los datos del ordenador.



Pseudocódigo del proyecto:

Clase PC:

Atributos

```
❶  marca: str

    modelo: str

    procesador: int

    memoria_ram: int

    almacenamiento: int

    estado: str ("Pendiente de revisión", "En reparación", "Reparado")

    averia: str (opcional)

    fecha_entrada: date (opcional)

    fecha_salida: date (opcional)
```

Métodos

```
❶  def __init__(self, marca, modelo, procesador, memoria_ram, almacenamiento,
    estado="Pendiente de revisión", averia="", fecha_entrada=None, fecha_salida=None):

    Inicializar atributos

    self.marca = marca

    self.modelo = modelo

    self.procesador = procesador

    self.memoria_ram = memoria_ram

    self.almacenamiento = almacenamiento

    self.estado = estado

    self.averia = averia

    self.fecha_entrada = fecha_entrada

    self.fecha_salida = fecha_salida

    def reparar(self, averia):

        Simular la reparación

        self.averia = averia

        self.estado = "En reparación"

        Esperar 2 segundos (simulación)

        time.sleep(2)

        self.estado = "Reparado"

        self.fecha_salida = date.today()

        def mostrar_informacion(self):

            Mostrar información del ordenador

            print(f"***Información del ordenador:**")

            print(f"- Marca: {self.marca}")

            print(f"- Modelo: {self.modelo}")

            print(f"- Procesador: {self.procesador} núcleos")

            print(f"- Memoria RAM: {self.memoria_ram} GB")

            print(f"- Almacenamiento: {self.almacenamiento} GB")

            print(f"- Estado: {self.estado}")

            if self.averia:

                print(f"- Avería: {self.averia}")

            if self.fecha_entrada:

                print(f"- Fecha de entrada: {self.fecha_entrada}")

            if self.fecha_salida:

                print(f"- Fecha de salida: {self.fecha_salida}")

            def get_estado(self):

                Devolver el estado del ordenador

                return self.estado

            def set_estado(self, nuevo_estado):

                Cambiar el estado del ordenador

                self.estado = nuevo_estado
```

Funciones principales:

```
❶  Crear 5 instancias de la clase PC

    • pc1 = PC("Asus", "VivoBook 15", 8, 16, 512)

    • pc2 = PC("Lenovo", "IdeaPad 3", 6, 8, 256)

    • pc3 = PC("HP", "Pavilion 14", 4, 4, 128)

    • pc4 = PC("Acer", "Aspire 5", 8, 12, 512)

    • pc5 = PC("Dell", "Inspiron 15", 6, 8, 256)

    Simular la reparación de 3 ordenadores

    • pc1.reparar("Pantalla rota")

    • pc3.reparar("Batería agotada")

    • pc5.reparar("Teclado no funciona")

    Imprimir la información de los ordenadores antes y después de la reparación

    for pc in [pc1, pc3, pc5]:

        • print(f"***Información antes de la reparación:**")

        pc.mostrar_informacion()

        print()

        • pc.reparar("Simulación de reparación")

        • print(f"***Información después de la reparación:**")

        pc.mostrar_informacion()

        print()
```

Explicación del pseudocódigo:

- Se define la clase PC con sus atributos y métodos.
- Se crean 5 instancias de la clase PC.
- Se simula la reparación de 3 ordenadores.
- Se imprime la información de los ordenadores antes y después de la reparación.



# Conclusión:

En este documento se ha presentado el diseño e implementación de una clase Python para gestionar la información de un ordenador en un taller de reparación.

## La clase PC incluye:

- Atributos para almacenar la marca, modelo, procesador, memoria RAM, almacenamiento, estado, avería, fecha de entrada y fecha de salida del ordenador.
- Métodos para reparar el ordenador, mostrar su información, obtener y cambiar su estado.

## Se ha desarrollado un diagrama de flujo y pseudocódigo para:

- Facilitar la comprensión del funcionamiento del programa.
- Detectar y corregir errores en el algoritmo.
- Permitir modificaciones al programa de forma más sencilla.
- Se ha mejorado el diagrama de flujo utilizando nombres más descriptivos para cada paso.

## Próximos pasos:

- Implementar una interfaz gráfica de usuario para facilitar la interacción con el programa.
- Ampliar la funcionalidad del programa para incluir la gestión de clientes, facturas y pagos.
- Desarrollar una versión web del programa para que pueda ser utilizado desde cualquier dispositivo.



# Recursos:

- Documentación oficial de [Python](#)
- Tutorial de [Python](#)
- [DateTime](#) Estructura
- [Diagramas](#) de [flujo](#)

# Dominio público

Este documento, al igual que el código fuente del proyecto, se encuentra en dominio público. Esto significa que cualquiera puede utilizarlo, modificarlo y distribuirlo libremente, sin necesidad de pedir permiso o dar crédito.

El proyecto se ha subido a GitHub para facilitar su acceso y colaboración. Puedes encontrar el repositorio en el siguiente enlace:



¡Esperamos que este proyecto te sea útil!