# Práctica 3

**Objetivos:**

Aplicar los conceptos de clase, atributos (datos miembro) y métodos (funciones miembro).

```
class Time:
    """
    Class that represents a time with AM/PM or 24 hours format.

    Class Attributes:
        TIME_FORMATS    # ("AM", "PM", "24 HOURS")
        time_count = 0  # Counts the number of Time objects created


    Attributes:
        hours       #Stores the hours
                    #(1 to 12 for AM/PM, 0 to 23 for 24 hours)
        minutes     #Stores the minutes (0 to 59)
        seconds     #Stores the seconds (0 to 59)
        format      #Stores the time format: "AM", "PM","24 HOURS"

    """
    def __init__(self):
        """
        Method to  Initialize attributes to 0.
        """
        pass

    def __asign_format(self, pszFormat):
        """
        Checks pszFormat has correct value & assigns it to format
        Converts to upper case to avoid problems with
        capitalization.

        Args:
            pszFormat: String with time format ("AM", "PM" or "24
                       HOURS").

        Returns:
            True if the format is correct, False otherwise.
        """
        pass
```

```
    def __is_24hour_format(self):
        """
        Checks if the time format is "24 HOURS".

        Returns:
            True if the format is "24 HOURS", False otherwise.
        """
        pass

    def _is_valid_time(self):
        """
        Checks if the time is correct according to the format.

        Returns:
            True if the time is correct, False otherwise.
        """
        pass

    def set_time(self, nHoras, nMinutos, nSegundos, pszFormato):
        """
        Assigns a time to the class.

        Args:
            nHoras: Hours (1 to 12 AM/PM, 0 to 23 for 24 hours).
            nMinutos: Minutes (0 to 59).
            nSegundos: Seconds (0 to 59).
            pszFormato: Time format ("AM", "PM" or "24 HOURS").

        Returns:
            True if the time could be assigned correctly,
            False otherwise.
        """
        pass

    def get_time (self):
        """
        Returns the current time of the class.

        """
        pass

    @classmethod
    def from_string(cls, time_string):
        """
          Class method to create Time object from string.

          Args:
```

```
                time_string (str): A string representing time in the
format "HH:MM:SS FORMAT", where FORMAT is AM, PM, or 24 HOURS.

                Returns:
                Time: new Time object with the parsed time.
                If the time string is invalid, print a message.
        """
         #Import the 're' module for regular expressions
         # Define regular expression pattern to match time strings
         # Pattern should match strings like
         # "14:30:00 24 HOURS" or "02:45:30 PM"
         # Use re.match to check if the time_string matches pattern

         # Extract hours, minutes, seconds, & format from matched
         # groups

         # Create a new Time object

         # Set the time using the extracted values
         # Convert hours, minutes, and seconds to integers

         # Return the new Time object

         #Print a message if the time string format is invalid
        pass

    @staticmethod
    def is_valid_format(time_format):
        """
          Static method to check if a given time format is valid.

          Args:
          time_format (str): The time format to check.

          Returns:
          bool: True if the format is valid (AM, PM, or 24 HOURS),
                False otherwise.
        """
        pass

    @classmethod
    def get_time_count(cls):
        """
          Class method to get total number of Time objects created

          Returns:
             Practicasint: The number of Time objects created.
```

```
        """
        pass
```

Cree una función fuera de la clase para mostrar la hora en una cadena con formato.

Implemente un programa controlado por menú que permita al usuario:

1. Introducir una nueva hora
2. Visualizar hora
3. Crear una hora a partir de una cadena (formato HH:MM:SS)
4. Terminar

Use la validación de datos adecuada para garantizar que:

- El formato de hora sea válido (AM, PM o 24 HOURS)
- Los valores de hora estén dentro del rango correcto según el formato

Consideraciones adicionales:

Implemente un manejo de errores adecuado para las entradas del usuario.

- El formato de hora siempre debe almacenarse en mayúsculas, pero permita la entrada en cualquier caso.
- Para el formato de 12 horas (AM/PM), las horas deben estar entre 1 y 12.
- Para el formato de 24 HOURS, las horas deben estar entre 0 y 23.
- Los minutos y segundos siempre deben estar entre 0 y 59.

Estructura del programa

- Cree un archivo llamado *time_management.py* para la clase Time y las funciones relacionadas.
- Implemente una función main() que ejecute el menú.