Gestión de fechas y horas

Abordaremos a continuación cómo tratar la gestión de fechas y horas en Python: creación de fechas, horas, lecturas de fechas y horas por teclado, y algunas operaciones básicas con fechas y horas. Todos los elementos que utilizaremos están disponibles en el módulo `datetime` de Python, por lo que deberemos importarlo al inicio del programa:

```
import datetime
```

1. Obtención de fechas y horas

Para obtener fechas y horas tenemos distintas alternativas:

- Usar `datetime.datetime.now()` para obtener la fecha y hora actuales
- Usar `datetime.date.today()` para obtener sólo la fecha actual (sin la hora)
- Usar `datetime.datetime(año, mes, dia)` o bien `datetime.datetime(año, mes, dia, hora, minuto, segundo)` para establecer una fecha (o fecha y hora) determinada.

Aquí vemos algunos ejemplos:

```
import datetime

# Fecha y hora actuales
ahora = datetime.datetime.now()
ahora2 = datetime.date.today()

# 15 de julio de 2000, a las 16:14:32
fecha_pasada = datetime.datetime(2000, 7, 15, 16, 14, 32)
```

1.1. Acceder a las partes de la fecha

Una vez hemos construido una fecha con los elementos anteriores, podemos acceder a sus diferentes partes (hora, minuto, mes, etc) usando las distintas propiedades del objeto que hemos creado: `day`, `month`, `year`, `hour`, `minute`, `second`:

```
print(ahora.day)
print(ahora2.month)
print(fecha_pasada.minute)
```

1.2. Formato de fechas y horas

El formato de la fecha/hora que utilicemos es importante, tanto para obtenerla de alguna fuente (por ejemplo, desde el teclado) como para mostrarlo por pantalla. Aquí podemos consultar las opciones del formato disponibles, que podemos emplear tanto para lectura como para escritura.

Para **leer una fecha de teclado** (o entrada de texto) usamos `datetime.strptime`, indicando el texto del que leer la fecha, y el formato de fecha:

```
# Fecha que diga el usuario en formado dia/mes/año
texto = input("Escribe una fecha (d/m/a): ")
fecha_usuario = datetime.datetime.strptime(texto, '%d/%m/%Y')
```

Para **escribir una fecha con un formato determinado**, usamos `datetime.strftime` sobre el objeto fecha, indicando el formato de salida:

```
# Fecha que diga el usuario en formado dia/mes/año
ahora = datetime.datetime.now()
print("Ahora: ", ahora.strftime('%d/%m/%Y %H:%M:%S'))
```

1.3. Usando el formato local de fechas

Por defecto, los resultados que obtenemos con el método `strftime` son en inglés. Si queremos mostrar los resultados en español tenemos que especificarlo con la librería `locale`. El siguiente ejemplo muestra cómo hacerlo:

```
import locale

locale.setlocale(locale.LC_TIME, "es_ES")
```

Una vez completado el paso anterior los resultados ya se muestran en español:

```
x = datetime(2015, 10, 24, 23, 30)
x = x.strftime("%A, %d de %B de %Y a las %I:%M %p")
print(x) #sábado, 24 de octubre de 2015 a las 11:30 PM
```

2. Operaciones básicas con fechas

Algunas de las operaciones básicas que podemos hacer con fechas son:

- Calcular la diferencia entre dos fechas: simplemente las restamos y accedemos a la propiedad que nos interese de esa diferencia. Por ejemplo, podemos acceder a los segundos de diferencia con `total_seconds()`, y luego convertirlos a la unidad que queramos:

```
fecha1 = ... # Creamos una fecha y hora
fecha2 = ... # Creamos otra fecha y hora
diferencia = fecha2 - fecha1
print("Horas de diferencia:", diferencia.total_seconds() // 3600)
```

- Obtener una fecha desde otra, por ejemplo, sumando o restando días/meses. Utilizamos el método `datetime.timedelta` indicando la unidad a sumar/restar (días o unidades inferiores).

```
ahora = datetime.datetime.now()
# Dentro de 10 días
futuro = ahora + datetime.timedelta(days = 10)
```

> **Ejercicio 1:**
>
> Escribe un programa que le pida al usuario su fecha de nacimiento y le diga:
>
> - Cuántos años tiene
> - Cuándo será la fecha de su próximo cumpleaños
> - Cuántos días faltan para su próximo cumpleaños
>
> **AYUDA:** vídeo con la solución del ejercicio