**Introducción al módulo *datetime***

En esta sección, aprenderás sobre un módulo de Python llamado *datetime*.

Como puedes adivinar, proporciona **clases para trabajar con la fecha y hora**. Si crees que no necesitas profundizar en este tema, hablemos de ejemplos del uso de la fecha y la hora en la programación.

La fecha y la hora tienen innumerables usos y probablemente sea difícil encontrar una aplicación de producción que no los utilice. A continuación, se muestran algunos ejemplos:

* **Registro de eventos**: gracias al conocimiento de la fecha y la hora, podemos determinar cuándo ocurre exactamente un error crítico en nuestra aplicación. Al crear registros, puedes especificar el formato de fecha y hora.
* **Seguimiento de cambios en la base de datos**: a veces es necesario almacenar información sobre cuándo se creó o modificó un registro. El módulo *datetime* será perfecto para este caso.
* **Validación de datos**: pronto aprenderás a leer la fecha y hora actuales en Python. Conociendo la fecha y hora actuales, podrás validar varios tipos de datos, por ejemplo, si un cupón de descuento ingresado por un usuario en nuestra aplicación sigue siendo válido.
* **Almacenamiento de información importante**: ¿te imaginas las transferencias bancarias sin almacenar la información de cuándo se realizaron? La fecha y la hora de ciertas acciones deben conservarse y debemos ocuparnos de ello.

# Obtener la fecha local actual y crear objetos del tipo fecha

Una de las clases proporcionadas por el módulo datetime es una clase llamada date. Los objetos de esta clase representan una fecha que consta de año, mes y día. Mira el código en el editor para ver cómo se ve en la práctica y como obtener la fecha local actual usando el método today.

Ejecuta el código para ver qué sucede.

El método today devuelve un objeto del tipo date que representa la fecha local actual. Toma en cuenta que el objeto date tiene tres atributos: *año*, *mes* y *día*.

Ten cuidado, porque estos atributos son de solo lectura. Para crear un objeto date, debes pasar los parámetros *año*, *mes* y *día* de la siguiente manera:

from datetime import date

my\_date = date(2019, 11, 4)

print(my\_date)

Ejecuta el ejemplo para ver qué sucede.

Al crear un objeto *date*, toma en cuenta las siguientes restricciones:

|  |  |
| --- | --- |
| **Parámetro** | **Restricciones** |
| año | El parámetro *año* debe ser mayor o igual a 1 (constante MINYEAR) y menor o igual a 9999 (constante MAXYEAR). |
| mes | El parámetro *mes* debe ser mayor o igual a 1 y menor o igual a 12. |
| día | El parámetro *día* debe ser mayor o igual a 1 y menor o igual que el último día del mes y año indicados. |

**Nota:** Más adelante en este curso, aprenderás a cambiar el formato de fecha predeterminado.

from datetime import date

today = date.today()

print("Hoy:", today)

print("Año:", today.year)

print("Mes:", today.month)

print("Día:", today.day)

# Creación de un objeto de fecha a partir de una marca de tiempo

La clase date nos da la capacidad de crear un objeto del tipo *fecha* a partir de una *marca de tiempo*.

En Unix, la marca de tiempo expresa el número de segundos desde el 1 de Enero de 1970 a las 00:00:00 (UTC). Esta fecha se llama la **época Unix**, porque es cuando comenzó el conteo del tiempo en los sistemas Unix.

La marca de tiempo es en realidad la diferencia entre una fecha en particular (incluida la hora) y el 1 de enero de 1970, 00:00:00 (UTC), expresada en segundos.

Para crear un objeto de fecha a partir de una marca de tiempo, debemos pasar una marca de tiempo Unix al método fromtimestamp.

Para este propósito, podemos usar el módulo time, que proporciona funciones relacionadas con el tiempo. Uno de ellos es una función llamada time(), que devuelve el número de segundos desde el 1 de enero de 1970 hasta el momento actual en forma de número flotante. Echa un vistazo al ejemplo en el editor.

Ejecuta el código para ver el resultado.

Si ejecutas el código de muestra varias veces, podrás ver cómo se incrementa la marca de tiempo. Vale la pena agregar que el resultado de la función time depende de la plataforma, porque **en los sistemas Unix y Windows, los segundos intercalares no se cuentan**.

**Nota:** En esta parte del curso también hablaremos sobre el módulo *time*.

from datetime import date

import time

timestamp = time.time()

print("Marca de tiempo:", timestamp)

d = date.fromtimestamp(timestamp)

print("Fecha:", d)

# Creando un objeto de fecha usando el formato ISO

El módulo datetime proporciona varios métodos para crear un objeto date. Uno de ellos es el método fromisoformat, que toma una fecha en el formato **AAAA-MM-DD** compatible con el estándar ISO 8601.

El estándar ISO 8601 define cómo se representan la fecha y la hora. Se usa a menudo, por lo que vale la pena tomarse un momento para familiarizarse con él. Echa un vistazo a la imagen que describe los valores requeridos por el formato:



Ahora observa el código en el editor y ejecútalo.

En nuestro ejemplo, AAAA es 2019, MM es 11 (noviembre) y DD es 04 (cuarto de noviembre).

Cuando sustituyas la fecha, asegúrate de agregar 0 antes de un mes o de un día expresado por un número menor que 10.

**Nota:** El método fromisoformat ha estado disponible en Python desde la versión 3.7.

from datetime import date

d = date.fromisoformat('2019-11-04')

print(d)

# El método replace()

A veces, es posible que debas reemplazar el año, el mes o el día con un valor diferente. No puedes hacer esto con los atributos de año, mes y día porque son de solo lectura. En este caso, puedes utilizar el método llamado replace.

Ejecuta el código en el editor.

Resultado:

1991-02-05

1992-01-16

**salida**

Los parámetros *year*, *month* y *day* son opcionales. Puedes pasar solo un parámetro al método replace, por ejemplo, *año*, o los tres como en el ejemplo.

El método replace devuelve un objeto *date* modificado, por lo que debes recordar asignarlo a alguna variable.

from datetime import date

d = date(1991, 2, 5)

print(d)

d = d.replace(year=1992, month=1, day=16)

print(d)

# ¿Que dia de la semana es?

Uno de los métodos más útiles que facilita el trabajo con fechas es el método llamado weekday. Devuelve el día de la semana como un número entero, donde 0 es el Lunes y 6 es el Domingo. Ejecuta el código en el editor.

Resultado:

0

**salida**



La clase date tiene un método similar llamado isoweekday, que también devuelve el día de la semana como un número entero, pero 1 es Lunes y 7 es Domingo:

from datetime import date

d = date(2019, 11, 4)

print(d.isoweekday())

Resultado:

1

**salida**

Como puedes ver, para la misma fecha obtenemos un número entero diferente, pero expresando el mismo día de la semana. El entero devuelto por el método isodayweek sigue la especificación ISO 85601.

from datetime import date

d = date(2022, 11, 17)

print(d.weekday())

# El método replace()

A veces, es posible que debas reemplazar el año, el mes o el día con un valor diferente. No puedes hacer esto con los atributos de año, mes y día porque son de solo lectura. En este caso, puedes utilizar el método llamado replace.

Ejecuta el código en el editor.

Resultado:

1991-02-05

1992-01-16

**salida**

Los parámetros *year*, *month* y *day* son opcionales. Puedes pasar solo un parámetro al método replace, por ejemplo, *año*, o los tres como en el ejemplo.

El método replace devuelve un objeto *date* modificado, por lo que debes recordar asignarlo a alguna variable.

# ¿Que dia de la semana es?

Uno de los métodos más útiles que facilita el trabajo con fechas es el método llamado weekday. Devuelve el día de la semana como un número entero, donde 0 es el Lunes y 6 es el Domingo. Ejecuta el código en el editor.

Resultado:

0

**salida**



La clase date tiene un método similar llamado isoweekday, que también devuelve el día de la semana como un número entero, pero 1 es Lunes y 7 es Domingo:

from datetime import date

d = date(2019, 11, 4)

print(d.isoweekday())

Resultado:

1

**salida**

Como puedes ver, para la misma fecha obtenemos un número entero diferente, pero expresando el mismo día de la semana. El entero devuelto por el método isodayweek sigue la especificación ISO 85601.

from datetime import date

d = date(2019, 11, 4)

print(d.weekday())

# Creando objetos time

Ya sabes cómo presentar una fecha utilizando el objeto date. El módulo datetime también tiene una clase que te permite presentar la hora. ¿Puedes adivinar su nombre? Sí, se llama time:

time(hour, minute, second, microsecond, *tzinfo, fold*)

El constructor de la clase time acepta los siguientes parámetros opcionales:

|  |  |
| --- | --- |
| **Parámetro** | **Restricciones** |
| hour | El párametro *hour* debe ser mayor o igual que 0 y menor que 23. |
| minute | El párametro *minute* debe ser mayor o igual que 0 y menor que 59. |
| second | El párametro *second* debe ser mayor o igual que 0 y menor que 59. |
| microsecond | El párametro *microsecond* debe ser mayor o igual que 0 y menor que 1000000. |
| tzinfo | El párametro *tzinfo* debe ser un objeto de la subclase tzinfo o None (por defecto). |
| fold | El párametro *fold* debe ser 0 o 1 (predeterminadamente 0). |

El párametro *tzinfo* está asociado con las zonas horarias, mientras que *fold* está asociado con el tiempo de pared. No los usaremos durante este curso, pero te recomendamos que te familiarices con ellos.

Veamos cómo crear un objeto de tiempo en la práctica. Ejecuta el código en el editor.

Resultado:

Tiempo: 14:53:20.000001

Hora: 14

Minuto: 53

Segundo: 20

Microsegundo: 1

**salida**

En el ejemplo, pasamos cuatro parámetros al constructor de la clase: *hour*, *minute*, *second*, and *microsecond*. Se puede acceder a cada uno de ellos utilizando los atributos de clase.

**Nota:** Pronto te diremos cómo puedes cambiar el formato de hora predeterminado.

from datetime import time

t = time(14, 53, 20, 1)

print("Tiempo:", t)

print("Hora:", t.hour)

print("Minuto:", t.minute)

print("Segundo:", t.second)

print("Microsegundo:", t.microsecond)

# El módulo time

Además de la clase time, la biblioteca estándar de Python ofrece un módulo llamado time, que proporciona una función relacionada con el tiempo. Ya se tuvo la oportunidad de aprender la función llamada time cuando se habló de la clase date. Ahora veremos otra función útil disponible en este módulo.

Debes pasar muchas horas frente a una computadora mientras realiza este curso. A veces puedes sentir la necesidad de tomar una siesta. ¿Por qué no? Escribamos un programa que simule la siesta corta de un estudiante. Echa un vistazo al código en el editor.

Resultado:

Estoy muy cansado. Tengo que echarme una siesta. Hasta luego.

¡Dormí bien! ¡Me siento genial!

**salida**

La parte más importante del código de muestra es el uso de la función sleep (sí, puedes recordarla de una de las prácticas de laboratorio anteriores en el curso), que suspende la ejecución del programa por el determinado número de segundos.

En nuestro ejemplo, son 5 segundos. Tienes razón, es una siesta muy corta.

Extiende el sueño del estudiante cambiando la cantidad de segundos. Toma en cuenta que la función sleep acepta solo un número entero o de punto flotante.

import time

class Student:

def take\_nap(self, seconds):

print("Estoy muy cansado. Tengo que echarme una siesta. Hasta luego.")

time.sleep(seconds)

print("¡Dormí bien! ¡Me siento genial!")

student = Student()

student.take\_nap(5)

# La función ctime()

El módulo time proporciona una función llamada ctime, que **convierte el tiempo en segundos desde el 1 de enero de 1970 (época Unix) en una cadena**.

¿Recuerdas el resultado de la función time? Eso es lo que necesitas pasar a ctime. Echa un vistazo al ejemplo en el editor.

Resultado:

Mon Nov 4 14:53:00 2019

**salida**

La función ctime devuelve una cadena para la marca de tiempo pasada. En nuestro ejemplo, la marca de tiempo expresa el 4 de noviembre de 2019 a las 14:53:00.

También es posible llamar a la función ctime sin especificar el tiempo en segundos. En este caso, se devolverá la hora actual:

import time

print(time.ctime

import time

timestamp = 1572879180

st = time.gmtime(timestamp)

print(time.asctime(st))

print(time.mktime((2019, 11, 4, 14, 53, 0, 0, 308, 0)))

# Creación de objetos datetime

En el módulo datetime, la fecha y la hora se pueden representar como objetos separados o como un solo objeto. La clase que combina fecha y hora se llama datetime.

datetime(year, month, day, hour, minute, second, microsecond, *tzinfo, fold*)

Su constructor acepta los siguientes parámetros:

|  |  |
| --- | --- |
| **Parámetro** | **Restricciones** |
| year | El parámetro *year* debe ser mayor o igual a 1 (constante MINYEAR) y menor o igual a 9999 (constante MAXYEAR). |
| month | El parámetro *month* debe ser mayor o igual a 1 y menor o igual a 12. |
| day | El parámetro *day* debe ser mayor o igual a 1 y menor o igual al último día del mes y año indicados. |
| hour | El parámetro *hour* debe ser mayor o igual que 0 y menor que 23. |
| minute | El parámetro *minute* debe ser mayor o igual que 0 y menor que 59. |
| second | El parámetro *second* debe ser mayor o igual que 0 y menor que 59. |
| microsecond | El parámetro *microsecond* debe ser mayor o igual que 0 y menor que 1000000. |
| tzinfo | El parámetro *tzinfo* debe ser una subclase del objeto tzinfo o None (de manera predeterminada). |
| fold | El parámetro *fold* debe ser 0 o 1 (predeterminadamente 0). |

Ahora echemos un vistazo al código en el editor para ver cómo creamos un objeto del tipo datetime.

Resultado:

Fecha y Hora: 2019-11-04 14:53:00

Fecha: 2019-11-04

Hora: 14:53:00

**salida**

El ejemplo crea un objeto datetime que representa el 4 de noviembre de 2019 a las 14:53:00. Todos los parámetros pasados al constructor van a atributos de clase de solo lectura. Son *year*, *month*, *day*, *hour*, *minute*, *second*, *microsecond*, *tzinfo*, y *fold*.

El ejemplo muestra dos métodos que devuelven dos objetos diferentes. El método llamado date devuelve el objeto *date* con el año, mes y día dados, mientras que el método llamado time devuelve el objeto *time* con la hora y minuto dados.

from datetime import datetime

dt = datetime(2019, 11, 4, 14, 53)

print("Fecha y Hora:", dt)

print("Fecha:", dt.date())

print("Hora:", dt.time())

**Métodos que devuelven la fecha y hora actuales**

La clase datetime tiene varios métodos que devuelven la fecha y hora actuales. Estos métodos son:

* today(): devuelve la fecha y hora local actual con el atributo *tzinfo* establecido a *None*.
* now(): devuelve la fecha y hora local actual igual que el método *today*, a menos que le pasemos el argumento opcional *tz*. El argumento de este método debe ser un objeto de la subclase *tzinfo*.
* utcnow(): devuelve la fecha y hora UTC actual con el atributo *tzinfo* establecido a *None*.

Ejecuta el código en el editor para verlos todos en la práctica. ¿Qué puedes decir sobre la salida?

Como puedes ver, el resultado de los tres métodos es el mismo. Las pequeñas diferencias se deben al tiempo transcurrido entre llamadas posteriores.

**Nota:** Puedes leer más sobre los objetos *tzinfo* en la documentación.

from datetime import datetime

print("hoy:", datetime.today())

print("ahora:", datetime.now())

print("utc\_ahora:", datetime.utcnow())