**﻿﻿Módulo 5.3**

**Manejo de Fechas y horas**

**librerías varias**

**datetime**

* **datetime.datetime.**
* **datetime date**
* **datetime.time**
* **datetime.timedelta**
* **strftime**
* **strptime**

**time**

Python posee varios módulos para trabajar con fechas.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **datetime:** Este es el módulo principal para trabajar con fechas y horas. Proporciona clases como **datetime**, **date**, **time**, **timedelta**, entre otras, para realizar cálculos y operaciones relacionadas con fechas y horas.  ·  Parece que deseas información sobre la biblioteca datetime en Python y cómo realizar ejercicios prácticos con ella. La biblioteca datetime proporciona clases y funciones para trabajar con fechas y horas en Python.  La biblioteca datetime contiene varias clases útiles para trabajar con fechas y horas:   * datetime.date: Representa una fecha (año, mes y día). * datetime.time: Representa una hora del día (hora, minuto, segundo y microsegundo). * datetime.datetime: Combina una fecha y una hora para representar una fecha y hora específicas. * datetime.timedelta: Representa una duración o diferencia entre dos fechas u horas. * datetime.timezone: Representa un huso horario (disponible en Python 3.2 y versiones posteriores).  |  |  |  | | --- | --- | --- | | ·Código Python   |  | | --- | | from datetime import datetime  # Obtener la fecha y hora actual  ahora = datetime.now()  print(f"{ahora=}") |   ·Salida esperada por consola   |  | | --- | | ahora=datetime.datetime(2025, 1, 1, 12, 30, 10, 12345) |   · |   ·  Crear una fecha específica   |  |  |  | | --- | --- | --- | | ·Código Python   |  | | --- | | from datetime import datetime  fecha = datetime(2025, 12, 31)  print(f"{fecha=}") |   ·Salida esperada por consola   |  | | --- | | fecha=datetime.datetime(2025, 12, 31, 0, 0) |   · |   ·  **time:** Este módulo se utiliza para realizar operaciones relacionadas con el tiempo, como medir el tiempo de ejecución de un programa.   |  |  |  | | --- | --- | --- | | ·Código Python   |  | | --- | | import time  # Obtener el tiempo actual en segundos desde la época  ahora = time.time()  print(f"{ahora=}") |   ·Salida esperada por consola   |  | | --- | | ahora=1767236399.0000001 |   · |   ·   |  |  |  | | --- | --- | --- | | ·Código Python   |  | | --- | | import time  # Registra el tiempo antes de que ocurra el evento  inicio = time.time()  # Simula algún proceso o evento que quieres medir  print ("inicio")  for n in range (1,9999):  for m in range (1,999):  pass  print(".",end="")  print ("fin")  # Registra el tiempo después de que ocurra el evento  fin = time.time()  # Calcula la duración en segundos  duracion = fin - inicio  print(f"Calcula la duración en segundos {duracion=}") |   ·Salida esperada por consola   |  | | --- | | inicio  ...…  fin  Calcula la duración en segundos duracion=**1.3179545402526855**  Presione enter para continuar |   · |   ·  **calendar**: El módulo calendar permite trabajar con calendarios y fechas relacionadas con días de la semana.   |  |  |  | | --- | --- | --- | | ·Código Python   |  | | --- | | import calendar  # Crear un calendario para un mes y año específicos  calendario = calendar.month(2025, 12)  print(f"calendario del mes {calendario}") |   ·Salida esperada por consola   |  | | --- | | inicio  ...…  fin  Calcula la duración en segundos duracion=**1.3179545402526855**  Presione enter para continuar |   · |   ·  calendar   |  |  |  | | --- | --- | --- | | ·Código Python   |  | | --- | | import calendar  # Crear un calendario para un mes y año específicos  calendario = calendar.month(2025, 12)  print(f"calendario del mes {calendario}") |   ·Salida esperada por consola   |  | | --- | | calendario del mes December 2025  Mo Tu We Th Fr Sa Su  1 2 3 4 5 6 7  8 9 10 11 12 13 14  15 16 17 18 19 20 21  22 23 24 25 26 27 28  29 30 31 |   · |   ·  **dateutil:** Este módulo externo, llamado dateutil, proporciona funciones adicionales y flexibilidad al trabajar con fechas y horas. (usar pip)  **Parsear** fecha a partir de una cadena   |  |  |  | | --- | --- | --- | | ·Código Python   |  | | --- | | from dateutil import parser  fecha = parser.parse("2025-12-31")  print(f"{fecha=}") |   ·Salida esperada por consola   |  | | --- | | fecha=datetime.datetime(2025, 12, 31, 0, 0) |   · |   ·  **pytz**: Si necesitas manejar zonas horarias y conversiones de zona horaria, el módulo pytz es útil. (usar pip)   |  |  |  | | --- | --- | --- | | ·Código Python   |  | | --- | | from datetime import datetime  import pytz  # Obtener la zona horaria actual  zona\_horaria = pytz.timezone('America/New\_York')  fecha = datetime.now(zona\_horaria)  print(f"{fecha=}") |   ·Salida esperada por consola   |  | | --- | | fecha=datetime.datetime(2025, 12, 31, 23, 59, 59, 999999, tzinfo=<DstTzInfo 'America/New\_York' EDT-1 day, 20:00:00 DST>) |   · |   · |

La clase datetime representa una fecha y hora específicas. Se puede crear una instancia de la clase utilizando los argumentos año, mes, día, hora, minuto, segundo y microsegundo. Por ejemplo, para crear una fecha y hora con los valores 2023-03-31 15:45:30:

datetime

date

time

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime, date, time  ahora = datetime.now()  print (f"ahora:{ahora:%Y-%m-%dT%H:%M:%S.%f}") |   ·Salida esperada por consola   |  | | --- | | ahora:2025-12-31T23:59:59.99999 |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime, date, time  now = datetime.now()  print  print ("Current date and time using str method of datetime object:");  print (str(now));  print (f"Current date and time using instance attributes:")  print (f"Current year: { now.year}")  print (f"Current month: { now.month}")  print (f"Current day: { now.day}")  print (f"Current hour: { now.hour}")  print (f"Current minute: { now.minute}")  print (f"Current second: { now.second}")  print (f"Current microsecond: { now.microsecond}")  print ("Current date and time using strftime:");  print (now.strftime('%Y-%m-%d %H:%M'));  print ("Current date and time using isoformat:");  print (now.isoformat()); |   ·Salida esperada por consola   |  | | --- | | Current date and time using str method of datetime object:  2025-12-31 23:59:59.99999  Current date and time using instance attributes:  Current year: 2025  Current month: 12  Current day: 31  Current hour: 23  Current minute: 59  Current second: 59  Current microsecond: 999999  Current date and time using strftime:  2025-12-31 23:59  Current date and time using isoformat:  2025-12-31T23:59:59.999999 |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime  # datetime(year, month, day)  paso\_fecha\_a\_string = datetime(2025, 12, 31)  print(f"datetime {paso\_fecha\_a\_string}")  # datetime(year, month, day, hour, minute, second, microsecond)  paso\_fechahora\_a\_string = datetime(2025, 12, 31, 23, 59, 59, 999999)  print(f"strftime {paso\_fechahora\_a\_string}") |   ·Salida esperada por consola   |  | | --- | | datetime datetime.datetime(2025, 12, 31, 0, 0)  strftime datetime.datetime(2025, 12, 31, 23, 59, 59, 999999) |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime  # datetime(year, month, day)  paso\_fecha\_a\_string = datetime(2025, 12, 31)  print(f"datetime {paso\_fecha\_a\_string}")  # datetime(year, month, day, hour, minute, second, microsecond)  paso\_fechahora\_a\_string = datetime(2025, 12, 31, 23, 59, 59, 999999)  print(f"strftime {paso\_fechahora\_a\_string}") |   ·Salida esperada por consola   |  | | --- | | datetime datetime.datetime(2025, 12, 31, 0, 0)  strftime datetime.datetime(2025, 12, 31, 23, 59, 59, 999999) |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime  # datetime(year, month, day)  paso\_fecha\_a\_string = datetime(2025, 12, 31)  print(f"datetime {paso\_fecha\_a\_string}")  # datetime(year, month, day, hour, minute, second, microsecond)  paso\_fechahora\_a\_string = datetime(2025, 12, 31, 23, 59, 59, 999999)  print(f"strftime {paso\_fechahora\_a\_string}") |   ·Salida esperada por consola   |  | | --- | | datetime datetime.datetime(2025, 12, 31, 0, 0)  strftime datetime.datetime(2025, 12, 31, 23, 59, 59, 999999) |   · |

·

datetime

time

··

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import time  b = time(11, 34, 56)  print("b =", b)  c = time(hour = 11, minute = 34, second = 56)  print("c =", c)  d = time(11, 34, 56, 234566)  print("d =", d)  #################################################################  mintime = time.min  print("Min Time supported", mintime)  maxtime = time.max  print("Max Time supported", maxtime) |   ·Salida esperada por consola   |  | | --- | | b = 11:34:56  c = 11:34:56  d = 11:34:56.234566  Min Time supported 00:00:00  Max Time supported 23:59:59.999999 |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import time  hora = time(23,59,59,9999)  print("Hora:", hora.hour)  print("Minutos:", hora.minute)  print("Segundos:", hora.second)  print("Microsegundos:", hora.microsecond) |   ·Salida esperada por consola   |  | | --- | | Hora: 23  Minutos: 59  Segundos: 59  Microsegundos: 9999 |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import time  hora = time(23,59,59,9999)  # convierto la hora de data a string  string = hora.isoformat()  print("formato texto iso", string)  print(f"{type(string)=}") |   ·Salida esperada por consola   |  | | --- | | formato texto iso 23:59:59.009999  type(string)=<class 'str'> |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import time  print("hora Original:", hora)  hora = hora.replace(hour = 13, second = 12)  print("New Time:", hora)  print ("ver archivo strftime y strptime")  Ftime = hora.strftime("%I:%M %p")  print("Formatted time", Ftime) |   ·Salida esperada por consola   |  | | --- | | hora Original: 23:59:59.009999  New Time: 13:59:12.009999  ver archivo strftime y strptime  Formatted time 01:59 PM |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime  hora\_inicio = time(8,00,0) # modificar  hora\_finalizacion = time(8,10,00) # modificar  print("hora\_inicio ",hora\_inicio)  print("hora\_finalizacion ",hora\_finalizacion)  while True:  actual = datetime.now()  actual = time(actual.hour, actual.minute,actual.second)  # este objeto se puede comparar sin tener en cuenta la fecha  if actual > hora\_inicio and actual < hora\_finalizacion:  print( actual )  else:  print("no")  break |   ·Salida esperada por consola   |  | | --- | | Ver ejecución cambiando parámetros |   · |

·

datetime

date

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import date  # argumentos formato year, month, date yyyy,mm,dd  fecha = date(2025, 12, 31)  print(f"{fecha}")  print(f"{fecha : yyyy,mm,dd}")  '''  Attributo Descripción  min La fecha mínima representable  max La fecha máxima representable  resolution La diferencia mínima posible entre los objetos de fecha  year El rango de año debe estar entre MINYEAR y MAXYEAR  month El rango de mes debe estar entre 1 y 12  day El rango de día debe estar entre 1 y el número de días en el mes / año dado  '''  mindate = date.min  print("Min fecha suportado", mindate)  maxdate = date.max  print("Max fecha suportado", maxdate)  from datetime import date  fecha = date(2025, 12, 31)  print("Año:", fecha.year)  print("Mes:", fecha.month)  print("Día:", fecha.day) |   ·Salida esperada por consola   |  | | --- | | 2025-12-31  yyyy,mm,dd  Min fecha suportado 0001-01-01  Max fecha suportado 9999-12-31  Año: 2025  Mes: 12  Día: 31 |   · |

··

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import date  hoy = date.today()  print("hoy", hoy)  # convierto la fecha de data a string  string = date.isoformat(hoy)  print("formato texto iso", string)  print(f"{type(string)=}")  print("dia semana usando weekday():", hoy.weekday())  print("dia semana usando isoweekday():", hoy.isoweekday())  # proleptic Gregorian ordinal  print("proleptic Gregorian ordinal:", hoy.toordinal())  # date from the ordinal  print("Date desde ordinal", date.fromordinal(720523))  print("recordar. es mi cumple") |   ·Salida esperada por consola   |  | | --- | | hoy 2023-09-06  formato texto iso 2023-09-06  type(string)=<class 'str'>  dia semana usando weekday(): 2  dia semana usando isoweekday(): 3  proleptic Gregorian ordinal: 738769  Date desde ordinal 1973-09-22  recordar. es mi cumple |   · |

··

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import \*  salida= datetime(2025, 12, 31).isoformat()  print(f"{salida=}")  print(f"{type(salida)=}")  print(f"{datetime.fromisoformat('2025-12-31T12:59:59.000000')=}") |   ·Salida esperada por consola   |  | | --- | | salida='2025-12-31T00:00:00'  type(salida)=<class 'str'>  datetime.fromisoformat('2025-12-31T12:59:59.000000')=datetime.datetime(2025, 12, 31, 12, 59, 59) |   · |

··

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import timedelta, date  print (f"{(date.today() - timedelta(10))=}")  def days\_diff(start, end):  return (end - start).days  print(f"{days\_diff(date(2020, 6, 1), date(2025, 12, 31))=}") |   ·Salida esperada por consola   |  | | --- | | (date.today() - timedelta(10))=datetime.date(2023, 8, 27)  days\_diff(date(2020, 6, 1), date(2025, 12, 31))=2039 |   · |

datetime

timedelta

**timedelta** permite operar con: microseconds, milliseconds, seconds, minutes, hours, days y weeks

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime,timedelta  print("Operaciones con fechas y horas")  print("Se utiliza la función timedelta que permite operar con: microseconds, milliseconds, seconds, minutes, hours, days y weeks")  hoy = date.today() # Asigna fecha actual  dias\_menos = timedelta(days=1)  ayer = hoy - dias\_menos # Resta a fecha actual 1 día  print("Ayer", ayer)  dias\_mas = timedelta(days=1)  mañana = hoy + dias\_mas # Suma a fecha actual 1 día  print("Mañana", mañana)  diferencia\_en\_dias = hoy - ayer # Resta las dos fechas  print("Mañana", mañana) |   ·Salida esperada por consola   |  | | --- | | Operaciones con fechas y horas  Se utiliza la función timedelta que permite operar con: microseconds, milliseconds, seconds, minutes, hours, days y weeks  Ayer 2025-12-31  Mañana 2026-01-01  diferencia 1 day, 0:00:00 |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime,timedelta  hoy\_mas\_1\_millon\_segundos = date.today() + timedelta(seconds=1000000)  ahora = datetime.now()  hora\_actual = time(ahora.hour, ahora.minute, ahora.second)  mas\_5m = ahora + timedelta(seconds=300)  mas\_5m = time(mas\_5m.hour, mas\_5m.minute, mas\_5m.second)  mas\_5h = timedelta(hours=5)  mas\_5h = ahora + racion\_de\_5h  print("La fecha de hoy más 1 millón de segundos:", hoy\_mas\_1\_millon\_segundos)  print("Hora actual:", hora\_actual)  print("Hora actual + 5 minutos:", mas\_5m)  print("Hora actual + 5 horas:", mas\_5h) |   ·Salida esperada por consola   |  | | --- | | Otros ejemplos de operaciones con otras unidades de tiempo  La fecha de hoy más 1 millón de segundos: 2025-12-31  Hora actual: 23:59:59  Hora actual + 5 minutos: 01:04:59  Hora actual + 5 horas: 2026-01-01 05:00:00.00 |   · |

·

|  |  |
| --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime, timedelta  formato = '%d/%m/%Y'  contador = 0  fechadesde = input('Fecha desde (dd/mm/aaaa): ')  fechahasta = input('Fecha hasta (dd/mm/aaaa): ')  if fechadesde == '' or fechahasta == '':  exit()  try:  fechadesde = datetime.strptime(fechadesde, formato)  fechahasta = datetime.strptime(fechahasta, formato)  if fechadesde > fechahasta:  print('Fecha desde debe ser menor o igual que hasta')  while fechadesde <= fechahasta:  if datetime.weekday(fechadesde) == 1:  contador +=1  fechaactual = fechadesde.strftime(formato)  print(contador, fechaactual, 'es martes')  fechadesde = fechadesde + timedelta(days=1)  except:  print('Fecha incorrecta') |   · |

·

**strftime**

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime  # Formatear una fecha en una cadena  ahora = datetime.now()  cadena\_formateada = ahora.strftime("%Y-%m-%d %H:%M:%S")  print(f"{cadena\_formateada=}") |   ·Salida esperada por consola   |  | | --- | | cadena\_formateada='2025-12-31 23:59:59' |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | print("Fechas:")  fecha1 = date.today() # Asigna fecha actual  print("\tFecha1:", fecha1)  # Suma a la fecha actual 2 días  fecha2 = date.today() + timedelta(days=2)  print("\tFecha2:", fecha2)  # Compara fechas  print("\tFecha1 > Fecha2:", fecha1 > fecha2) # False  # Asigna formato de ejemplo1  formato1 = '%a %b %d %H:%M:%S %Y'  # Asigna formato de ejemplo2  formato2 = '%d-%m-%y %I:%m %p'  hoy = datetime.today() # Asigna fecha-hora  # Muestra fecha-hora según ISO 8601  print("Fecha en formato ISO 8601:", hoy)  # Aplica formato ejemplo1  cadena1 = hoy.strftime(formato1)  # Aplica formato ejemplo2  cadena2 = hoy.strftime(formato2)  # Muestra fecha-hora según ejemplo1  print("Formato1:", cadena1)  # Muestra fecha-hora según ejemplo2  print("Formato2:", cadena2)  print("Para convertir una cadena a objeto datetime")  objeto\_datetime = datetime.strptime(cadena1, formato1)  print("strptime:", fecha1.strftime(formato1)) |   ·Salida esperada por consola   |  | | --- | | Fechas:  Fecha1: 2025-12-31  Fecha2: 2026-01-01  Fecha1 > Fecha2: False  Fecha en formato ISO 8601: 2025-12-31 23:59:59.999999  Formato1: Sun Dic 31 23:59:59 2025  Formato2: 31-12-31 23:59 PM  Para convertir una cadena a objeto datetime  strptime: Sun Dic 31 23:59:59 2025  Presione una tecla para continuar |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime, timedelta  print ("uso de libreria # datetime ")  print(f"Fecha - hora :{(datetime.now())}");  print(f"Fecha - dia :{(date.today())}");  fecha = datetime.now()  print(f"Fecha - hora :{(fecha)}")  print(f"Dia numero :{(fecha.day)}");  print(f"Mes :{(fecha.month)}");  print(f"Año :{(fecha.year)}");  print(f"La fecha es :{fecha:%d %m, %Y}")  print(f"La fecha es :{fecha:%d %B, %Y}")  print ("\nUsando strftime\n ver:https://www.guru99.com/date-time-and-datetime-classes-in-python.html")  print(f"Fecha - hora %c :{(fecha.strftime('%c'))}");  print(f"Dia numero %D :{(fecha.strftime('%D'))}");  print(f"Fecha %x :{(fecha.strftime('%x'))}");  print(f"Hora %X :{(fecha.strftime('%X'))}");  print(f"Dia de la semana %a :{(fecha.strftime('%a'))}");  print(f"Dia de la semana %A :{(fecha.strftime('%A'))}");  print(f"Dia numero %d :{(fecha.strftime('%d'))}");  print(f"Mes %b :{(fecha.strftime('%b'))}");  print(f"Mes %B :{(fecha.strftime('%B'))}");  print(f"Año %y :{(fecha.strftime('%y'))}");  print(f"Año %Y :{(fecha.strftime('%Y'))}");  print(f"La hora es %I:%M:%S: %p :{(fecha.strftime('%I:%M:%S: %p'))}");  print(f"La hora es %I:%M :{(fecha.strftime('%I:%M'))}"); |   ·Salida esperada por consola   |  | | --- | | uso de libreria # datetime  Fecha - hora :2023-09-05 17:40:09.348187  Fecha - dia :2023-09-05  Fecha - hora :2023-09-05 17:40:09.348187  Dia numero :5  Mes :9  Año :2023  La fecha es :05 09, 2023  La fecha es :05 September, 2023  Usando strftime  ver:https://www.guru99.com/date-time-and-datetime-classes-in-python.html  Fecha - hora %c :Tue Sep 5 17:40:09 2023  Dia numero %D :09/05/23  Fecha %x :09/05/23  Hora %X :17:40:09  Dia de la semana %a :Tue  Dia de la semana %A :Tuesday  Dia numero %d :05  Mes %b :Sep  Mes %B :September  Año %y :23  Año %Y :2023  La hora es %I:%M:%S: %p :05:40:09: PM  La hora es %I:%M :05:40  Presione una tecla para continuar |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime, timedelta  print ("uso de libreria # datetime ")  print(f"Fecha - hora :{(datetime.now())}");  print(f"Fecha - dia :{(date.today())}");  fecha = datetime.now()  print(f"Fecha - hora :{(fecha)}")  print(f"Dia numero :{(fecha.day)}");  print(f"Mes :{(fecha.month)}");  print(f"Año :{(fecha.year)}");  print(f"La fecha es :{fecha:%d %m, %Y}")  print(f"La fecha es :{fecha:%d %B, %Y}")  print ("\nUsando strftime\n ver:https://www.guru99.com/date-time-and-datetime-classes-in-python.html")  print(f"Fecha - hora %c :{(fecha.strftime('%c'))}");  print(f"Dia numero %D :{(fecha.strftime('%D'))}");  print(f"Fecha %x :{(fecha.strftime('%x'))}");  print(f"Hora %X :{(fecha.strftime('%X'))}");  print(f"Dia de la semana %a :{(fecha.strftime('%a'))}");  print(f"Dia de la semana %A :{(fecha.strftime('%A'))}");  print(f"Dia numero %d :{(fecha.strftime('%d'))}");  print(f"Mes %b :{(fecha.strftime('%b'))}");  print(f"Mes %B :{(fecha.strftime('%B'))}");  print(f"Año %y :{(fecha.strftime('%y'))}");  print(f"Año %Y :{(fecha.strftime('%Y'))}");  print(f"La hora es %I:%M:%S: %p :{(fecha.strftime('%I:%M:%S: %p'))}");  print(f"La hora es %I:%M :{(fecha.strftime('%I:%M'))}"); |   ·Salida esperada por consola   |  | | --- | | uso de libreria # datetime  Fecha - hora :2023-09-05 17:40:09.348187  Fecha - dia :2023-09-05  Fecha - hora :2023-09-05 17:40:09.348187  Dia numero :5  Mes :9  Año :2023  La fecha es :05 09, 2023  La fecha es :05 September, 2023  Usando strftime  ver:https://www.guru99.com/date-time-and-datetime-classes-in-python.html  Fecha - hora %c :Tue Sep 5 17:40:09 2023  Dia numero %D :09/05/23  Fecha %x :09/05/23  Hora %X :17:40:09  Dia de la semana %a :Tue  Dia de la semana %A :Tuesday  Dia numero %d :05  Mes %b :Sep  Mes %B :September  Año %y :23  Año %Y :2023  La hora es %I:%M:%S: %p :05:40:09: PM  La hora es %I:%M :05:40 |   · |

·

|  |
| --- |
| %a Locale’s abbreviated weekday name.  %A Locale’s full weekday name.  %b Locale’s abbreviated month name.  %B Locale’s full month name.  %c Locale’s appropriate date and time representation.  %d Day of the month as a decimal number [01,31]./  %f Microsecond as a decimal number [0,999999], zero-padded on the left  %H Hour (24-hour clock) as a decimal number [00,23].  %I Hour (12-hour clock) as a decimal number [01,12].  %j Day of the year as a decimal number [001,366].  %m Month as a decimal number [01,12].  %M Minute as a decimal number [00,59].  %p Locale’s equivalent of either AM or PM.  %S Second as a decimal number [00,61].  %U Week number of the year (Sunday as the first day of the week)  %w Weekday as a decimal number [0(Sunday),6].  %W Week number of the year (Monday as the first day of the week)  %x Locale’s appropriate date representation.  %X Locale’s appropriate time representation.  %y Year without century as a decimal number [00,99].  %Y Year with century as a decimal number.  %z UTC offset in the form +HHMM or -HHMM.  %Z Time zone name (empty string if the object is naive).  %% A literal '%' character. |

·

·

|  |  |  |  |
| --- | --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime, timedelta  print(f".strftime( %a ) :{(fecha.strftime('%a'))}");  print(f".strftime( %b ) :{(fecha.strftime('%b'))}");  print(f".strftime( %c ) :{(fecha.strftime('%c'))}");  print(f".strftime( %d ) :{(fecha.strftime('%d'))}");  print(f".strftime( %e ) :{(fecha.strftime('%e'))}");  print(f".strftime( %f ) :{(fecha.strftime('%f'))}");  print(f".strftime( %g ) :{(fecha.strftime('%g'))}");  print(f".strftime( %h ) :{(fecha.strftime('%h'))}");  print(f".strftime( %j ) :{(fecha.strftime('%j'))}");  print(f".strftime( %m ) :{(fecha.strftime('%m'))}");  print(f".strftime( %p ) :{(fecha.strftime('%p'))}");  print(f".strftime( %r ) :{(fecha.strftime('%r'))}");  print(f".strftime( %u ) :{(fecha.strftime('%u'))}");  print(f".strftime( %w ) :{(fecha.strftime('%w'))}");  print(f".strftime( %x ) :{(fecha.strftime('%x'))}");  print(f".strftime( %y ) :{(fecha.strftime('%y'))}");  print(f".strftime( %A ) :{(fecha.strftime('%A'))}");  print(f".strftime( %B ) :{(fecha.strftime('%B'))}");  print(f".strftime( %C ) :{(fecha.strftime('%C'))}");  print(f".strftime( %D ) :{(fecha.strftime('%D'))}");  print(f".strftime( %F ) :{(fecha.strftime('%F'))}");  print(f".strftime( %G ) :{(fecha.strftime('%G'))}");  print(f".strftime( %H ) :{(fecha.strftime('%H'))}");  print(f".strftime( %I ) :{(fecha.strftime('%I'))}");  print(f".strftime( %M ) :{(fecha.strftime('%M'))}");  print(f".strftime( %R ) :{(fecha.strftime('%R'))}");  print(f".strftime( %S ) :{(fecha.strftime('%S'))}");  print(f".strftime( %T ) :{(fecha.strftime('%T'))}");  print(f".strftime( %U ) :{(fecha.strftime('%U'))}");  print(f".strftime( %V ) :{(fecha.strftime('%V'))}");  print(f".strftime( %W ) :{(fecha.strftime('%W'))}");  print(f".strftime( %X ) :{(fecha.strftime('%X'))}");  print(f".strftime( %Y ) :{(fecha.strftime('%Y'))}"); |   ·Salida esperada por consola   |  |  | | --- | --- | | .strftime( %a ) =  .strftime( %b ) =  .strftime( %c ) =  .strftime( %d ) =  .strftime( %e ) =  .strftime( %f ) =  .strftime( %g ) =  .strftime( %h ) =  .strftime( %j ) =  .strftime( %m ) =  .strftime( %p ) =  .strftime( %r ) =  .strftime( %u )  .strftime( %w )  .strftime( %x )  .strftime( %y )  .strftime( %A )  .strftime( %B )  .strftime( %C )  .strftime( %D )  .strftime( %F )  .strftime( %G )  .strftime( %H )  .strftime( %I )  .strftime( %M )  .strftime( %R )  .strftime( %S )  .strftime( %T )  .strftime( %U )  .strftime( %V )  .strftime( %W )  .strftime( %X )  .strftime( %Y ) | Tue  Sep  Tue Sep 5 17:40:09 2023  05  5  348187  23  Sep  248  09  PM  05:40:09 PM  2  2  09/05/23  23  Tuesday  September  20  09/05/23  2023-09-05  2023  17  05  40  17:40  09  17:40:09  36  36  36  17:40:09  2023 |   · |

·

**strptime**

datetime.strptime es un método proporcionado por el módulo datetime en Python que se utiliza para analizar (parsear) cadenas de texto con fechas y horas y convertirlas en objetos datetime. Esta función es especialmente útil cuando deseas convertir una cadena de fecha y hora en un objeto datetime para realizar operaciones y cálculos de fechas y horas.

datetime.strptime(string, format):

Este método toma dos argumentos.

* El primer argumento, string, es la cadena de texto que deseas convertir en un objeto datetime.
* El segundo argumento, format, es una cadena que especifica el formato en el que se encuentra la fecha y hora en la cadena de texto.

Debes proporcionar el formato correspondiente para que Python pueda realizar la conversión correctamente.

El formato es una cadena que contiene códigos de formato que representan componentes específicos de la fecha y la hora, como año, mes, día, hora, minuto, segundo, etc.

Estos códigos de formato están precedidos por un carácter de porcentaje %.

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from datetime import datetime  # Analizar una cadena en una fecha  fecha = datetime.strptime("2025-12-31", "%Y-%m-%d")  print(f"{fecha=}") |   ·Salida esperada por consola   |  | | --- | | fecha=datetime.datetime(2025, 12, 31, 0, 0) |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | formato = '%H:%M:%S'  while True:  try:  hhmmss = input('Introducir hora (hh:mm:ss): ')  if hhmmss == "":  break  hhmmss = datetime.strptime(hhmmss, formato)  horas = hhmmss.hour  minutos = hhmmss.minute  segundos = hhmmss.second  hhmmss\_seg = (horas \* 60 \* 60) + (minutos \* 60) + segundos  resultado = float(hhmmss\_seg / 86400)  print("Resultado: ", resultado)  except:  print('Error en el formato de hora introducido.')  print('-> Formato válido: hh:mm:ss ¡Inténtalo de nuevo!') |   ·Salida esperada por consola   |  | | --- | | A partir de una hora se obtiene fracción del día  Introducir hora (hh:mm:ss): **22:10:55**  Resultado: 0.9242476851851852  Introducir hora (hh:mm:ss): **23:59:01**  Resultado: 0.9993171296296296 |   · |

·

time

|  |
| --- |
| **Funciones del módulo time:**  **time.altzone**  El desplazamiento de la zona horaria de DST local, en segundos al oeste de UTC, si está definido. Este valor es negativo si la zona horaria de DST local está al este de UTC (como en Europa Occidental, incluido el Reino Unido). Solo utilízalo si daylight (horario de verano) no es igual a cero.  **time.asctime([tupletime])**  Acepta una tupla de tiempo y devuelve una cadena legible de 24 caracteres, como 'Mar Dic 11 18:07:14 2008'.  **time.clock( )**  Devuelve el tiempo actual de la CPU como un número decimal de segundos. Para medir los costos computacionales de diferentes enfoques, el valor de time.clock es más útil que el de time.time().  **time.ctime([secs])**  Similar a asctime(localtime(secs)), y sin argumentos es similar a asctime().  **time.gmtime([secs])**  Acepta un instante expresado en segundos desde la época y devuelve una tupla de tiempo t con la hora UTC. Nota: t.tm\_isdst siempre es 0.  **time.localtime([secs])**  Acepta un instante expresado en segundos desde la época y devuelve una tupla de tiempo t con la hora local (t.tm\_isdst es 0 o 1, dependiendo de si se aplica el horario de verano en ese instante según las reglas locales).  **time.mktime(tupletime)**  Acepta un instante expresado como una tupla de tiempo en hora local y devuelve un valor decimal con el instante expresado en segundos desde la época.  **time.sleep(secs)**  Suspende el hilo que llama durante secs segundos.  **time.strftime(fmt[,tupletime])**  Acepta un instante expresado como una tupla de tiempo en hora local y devuelve una cadena que representa el instante según el formato especificado en la cadena fmt.  **time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')**  Analiza str de acuerdo con la cadena de formato fmt y devuelve el instante en formato de tupla de tiempo.  **time.time( )**  Devuelve el instante de tiempo actual, un número decimal de segundos desde la época.  **time.tzset()**  Restablece las reglas de conversión de tiempo utilizadas por las rutinas de la biblioteca. La variable de entorno TZ especifica cómo se realiza esto. |

·

|  |
| --- |
| **Atributos del módulo time:**  **time.timezone**  El atributo time.timezone es el desplazamiento en segundos de la zona horaria local (sin DST) desde UTC (>0 en las Américas; <=0 en la mayor parte de Europa, Asia y África).  **time.tzname**  El atributo time.tzname es un par de cadenas dependientes de la configuración regional, que son los nombres de la zona horaria local sin DST y con DST, respectivamente.  **Funciones del módulo calendar:**  calendar.calendar(year, w=2, l=1, c=6)  Devuelve una cadena multilínea con un calendario para el año especificado, formateado en tres columnas separadas por c espacios. w es el ancho en caracteres de cada fecha, y cada línea tiene una longitud de 21\*w+18+2\*c. l es el número de líneas por semana.  **calendar.firstweekday( )**  Devuelve la configuración actual para el día de la semana que inicia cada semana. De forma predeterminada, cuando se importa el módulo calendar, esto es 0, lo que significa lunes.  **calendar.isleap(year)**  Devuelve True si year es un año bisiesto; de lo contrario, devuelve False.  **calendar.leapdays(y1, y2)**  Devuelve el número total de años bisiestos en el rango de años y1 a y2.  **calendar.month(year, month, w=2, l=1)**  Devuelve una cadena multilínea con un calendario para el mes month del año year, una línea por semana más dos líneas de encabezado. w es el ancho en caracteres de cada fecha, y cada línea tiene una longitud de 7\*w+6. l es el número de líneas por semana.  **calendar.monthcalendar(year, month)**  Devuelve una lista de listas de enteros. Cada sublista denota una semana. Los días fuera del mes month del año year se establecen en 0; los días dentro del mes se establecen en su día del mes, de 1 en adelante.  **calendar.monthrange(year, month)**  Devuelve dos enteros. El primero es el código del día de la semana para el primer día del mes month en el año year, y el segundo es el número de días en el mes. Los códigos de día de la semana son de 0 (lunes) a 6 (domingo); los números de mes son de 1 a 12.  **calendar.prcal(year, w=2, l=1, c=6)**  Similar a print(calendar.calendar(year, w, l, c)).  **calendar.prmonth(year, month, w=2, l=1)**  Similar a print(calendar.month(year, month, w, l)).  **calendar.setfirstweekday(weekday)**  Establece el primer día de cada semana con el código weekday. Los códigos de día de la semana son de 0 (lunes) a 6 (domingo).  **calendar.timegm(tupletime)**  El inverso de time.gmtime: acepta un instante de tiempo en formato de tupla de tiempo y lo devuelve como un número decimal de segundos desde la época.  **calendar.weekday(year, month, day)**  Devuelve el código del día de la semana para la fecha dada  dias de la semana 0 (Lunes) a 6 (Domingo)  Mes 1 (enero) a 12 (Diciembre). |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | import time  print("sleep aguarde.")  time.sleep(2.4)  print("listo 2.4 segundos después.") |   ·Salida esperada por consola   |  | | --- | | sleep aguarde.  listo 2.4 segundos después. |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | seconds = 1767236399.000000  local\_time = time.ctime(seconds)  print(f"ctime {local\_time=}") |   ·Salida esperada por consola   |  | | --- | | ctime local\_time='Wed Dec 31 23:59:59 2025' |   · |

·

|  |  |  |  |
| --- | --- | --- | --- |
| y=time.struct\_time(tm\_year=1970, tm\_mon=1, tm\_mday=1, tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday=3, tm\_yday=1, tm\_isdst=0)  x=time.struct\_time(tm\_year=2025, tm\_mon=12, tm\_mday=31, tm\_hour=23, tm\_min=59, tm\_sec=59, tm\_wday=3, tm\_yday=361, tm\_isdst=0)   |  | | --- | | Index Attribute Values  0 tm\_year 0000, ...., 2018, ..., 9999  1 tm\_mon 1, 2, ... , 12  2 tm\_mday 1, 2, ... , 31  3 tm\_hour 0, 1, ... , 23  4 tm\_min 0, 1, ... , 59  5 tm\_sec 0, 1, ... , 61  6 tm\_wday 0, 1, ... , 6; Monday is 0  7 tm\_yday 1, 2, ... , 366  8 tm\_isdst 0, 1 or -1 |   ·Código Python   |  | | --- | | import time  from time import struct\_time  print(f"{time.gmtime(0)}")  print(f"{time.gmtime(1767236399)}") |   ·Salida esperada por consola   |  | | --- | | time.struct\_time(tm\_year=2023, tm\_mon=8, tm\_mday=30, tm\_hour=22, tm\_min=12, tm\_sec=27, tm\_wday=2, tm\_yday=242, tm\_isdst=0)  time.struct\_time(tm\_year=2026, tm\_mon=1, tm\_mday=1, tm\_hour=2, tm\_min=59, tm\_sec=59, tm\_wday=3, tm\_yday=1, tm\_isdst=0) |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | import time  result = time.localtime(1767236399)  print(f"{result=}")  print(f"localtime {result.tm\_year=}")  print(f"localtime {result.tm\_hour=}") |   ·Salida esperada por consola   |  | | --- | | result=time.struct\_time(tm\_year=2025, tm\_mon=12, tm\_mday=31, tm\_hour=23, tm\_min=59, tm\_sec=59, tm\_wday=2, tm\_yday=365, tm\_isdst=0)  localtime result.tm\_year=2025  localtime result.tm\_hour=23 |   · |

·

|  |  |  |
| --- | --- | --- |
| **Gmt +3**  ·Código Python   |  | | --- | | import time  result = time.gmtime(1545925769)  print(f"{result=}")  print(f"gmtime {result.tm\_year=}")  print(f"gmtime {result.tm\_hour=}") |   ·Salida esperada por consola   |  | | --- | | result=time.struct\_time(tm\_year=2026, tm\_mon=1, tm\_mday=1, tm\_hour=2, tm\_min=59, tm\_sec=59, tm\_wday=3, tm\_yday=1, tm\_isdst=0)  gmtime result.tm\_year=2026  gmtime result.tm\_hour=2 |   · |

·

|  |  |  |
| --- | --- | --- |
| Código Python   |  | | --- | | import time  t = (2025, 12, 31, 23, 59, 59, 0, 0, 0)  local\_time = time.mktime(t)  print(f"mktime {local\_time=}") |   ·Salida esperada por consola   |  | | --- | | mktime local\_time=1767236399.0 |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | import time  t=time.struct\_time(tm\_year=2025, tm\_mon=12, tm\_mday=31, tm\_hour=23, tm\_min=59, tm\_sec=59, tm\_wday=2, tm\_yday=365, tm\_isdst=0)  s=1767236399.0  localtime seconds=1767236399 |   ·Salida esperada por consola   |  | | --- | | t=time.struct\_time(tm\_year=2025, tm\_mon=12, tm\_mday=31, tm\_hour=23, tm\_min=59, tm\_sec=59, tm\_wday=2, tm\_yday=365, tm\_isdst=0)  s=1767236399.0  localtime seconds=1767236399 |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | import time  t = (2025, 12, 31, 23, 59, 59, 0, 0, 0)  result = time.asctime(t)  print(f"asctime {result=}") |   ·Salida esperada por consola   |  | | --- | | **asctime result='Mon Dec 31 23:59:59 2025'** |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | import time  named\_tuple = time.localtime() # get struct\_time  time\_string = time.strftime("%m/%d/%Y, %H:%M:%S", named\_tuple)  print(f"strftime {time\_string=}") |   ·Salida esperada por consola   |  | | --- | | strftime time\_string='12/31/2025, 23:59:59' |   · |

·

|  |  |  |
| --- | --- | --- |
| Código Python   |  | | --- | | import time  time\_string = "31 December, 2025"  result = time.strptime(time\_string, "%d %B, %Y")  print(f"strptime {result=}") |   ·Salida esperada por consola   |  | | --- | | **strptime result=time.struct\_time(tm\_year=2025, tm\_mon=12, tm\_mday=31, tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday=2, tm\_yday=365, tm\_isdst=-1)** |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from time import time, ctime  t = time()  print(f"ctime {ctime(t)=}") |   ·Salida esperada por consola   |  | | --- | | ctime ctime(t)='Sun Dic 31 23:59:59 2025' |   · |

·

|  |  |  |
| --- | --- | --- |
| ·Código Python   |  | | --- | | from time import struct\_time  time\_tuple = (2025, 12, 31, 23, 59, 59, 1, 57, 0)  time\_obj = struct\_time(time\_tuple)  print(f"struct\_time {time\_obj=}") |   ·Salida esperada por consola   |  | | --- | | **struct\_time time\_obj=time.struct\_time(tm\_year=2025, tm\_mon=12, tm\_mday=31, tm\_hour=23, tm\_min=59, tm\_sec=59, tm\_wday=1, tm\_yday=57, tm\_isdst=0)** |   · |

·