Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

miércoles, 12 de febrero de 2014

Operaciones con fechas y horas. Calendarios

Los módulos **datetime** y **calendar** amplían las posibilidades del módulo **time** que provee funciones para manipular expresiones de tiempo.

Al comienzo de un programa tendremos que importar estos módulos para tener acceso a un conjunto amplio de clases y funciones:

```
from datetime import datetime, date, time, timedelta
import calendar
```

Mostrar fecha y hora (datetime)

```
ahora = datetime.now()  # Obtiene fecha y hora actual
print("Fecha y Hora:", ahora)  # Muestra fecha y hora
print("Fecha y Hora UTC:",ahora.utcnow())  # Muestra fecha/hora UTC
print("Día:",ahora.day)  # Muestra día
print("Mes:",ahora.month)  # Muestra mes
print("Año:",ahora.year)  # Muestra año
print("Hora:", ahora.hour)  # Muestra hora
print("Minutos:",ahora.minute)  # Muestra minuto
print("Segundos:", ahora.second)  # Muestra segundo
print("Microsegundos:",ahora.microsecond)  # Muestra microsegundo
```

Comparando fechas y horas (datetime, date)

```
print("Horas:")
hora1 = time(10, 5, 0)  # Asigna 10h 5m 0s
print("\tHora1:", hora1)
hora2 = time(23, 15, 0)  # Asigna 23h 15m 0s
print("\tHora2:", hora2)

# Compara horas
print("\tHora1 < Hora2:", hora1 < hora2)  # True

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
```

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

Aplicando formatos a fechas y horas (Máscaras)

Las siguientes claves se combinan para aplicar formatos:

%a	Nombre local abreviado de día de semana
%A	Nombre local completo de día de semana
%b	Nombre local abreviado de mes
%B	Nombre local completo de mes
%c	Representación local de fecha y hora
%d	Día de mes [01,31]
%H	Hora (horario 24 horas) [00,23]
%I	Hora (horario 12 horas) [01,12]
%j	Número de día del año [001,366]
%m	Mes [01,12]
%M	Minuto [00,59]
%p	Etiqueta AM o PM
%S	Segundo
%U	Nº semana del año. Se considera al Domingo como primer día de semana [00,53]
%w	Establece el primer día de semana [0(Domingo),1(Lunes)... 6].
%W	Nº semana del año (Se considera al Lunes como primer día de semana) [00,53]
%x	Fecha local
%X	Hora local
%y	Año en formato corto [00,99]
%Y	Año en formato largo
%Z	Nombre de Zona Horaria

Ejemplos:

```
# 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)
```

Para convertir una cadena a objeto datetime

simultáneamente varias operaciones en el mismo espacio de proceso se...

Archivo

febrero 2014 (17)

python.org

pypi.org

Sitios

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

```
objeto_datetime = datetime.strptime(cadena1, formato1)
print("strptime:", fecha1.strftime(formato1))
```

Operaciones con fechas y horas

Se utiliza la función timedelta que permite operar con: microseconds, milliseconds, seconds, minutes, hours, days y weeks

```
hoy = date.today()  # Asigna fecha actual
ayer = hoy - timedelta(days=1)  # Resta a fecha actual 1 día
mañana = hoy + timedelta(days=1)  # Suma a fecha actual 1 día
diferencia_en_dias = mañana - hoy  # Resta las dos fechas
```

Otros ejemplos de operaciones con otras unidades de tiempo

```
hoy_mas_1_millon_segundos = hoy + 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)
racion_de_5h = timedelta(hours=5)
mas_5h = ahora + racion_de_5h

print("Ayer:", ayer)
print("Hoy:", hoy)
print("Mañana:", mañana)
print("Diferencia en días entre mañana y hoy:",
      diferencia_en_dias.days)
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)
```

Diferencia entre dos fechas (datetime)

```
# Asigna datetime de la fecha actual
fecha1 = datetime.now()

# Asigna datetime específica
fecha2 = datetime(1995, 11, 5, 0, 0, 0)
diferencia = fecha1 - fecha2
print("Fecha1:", fecha1)
print("Fecha2:", fecha2)
print("Diferencia:", diferencia)
print("Entre las 2 fechas hay ",
      diferencia.days,
      "días y ",
      diferencia.seconds,
      "seg.")
```

Diferencia entre dos fechas en días (datetime y strptime)

```
formato_fecha = "%d-%m-%Y"
fecha_inicial = datetime.strptime("01-10-2013",
                                  formato_fecha)
fecha_final = datetime.strptime("25-12-2013",
                                formato_fecha)
diferencia = fecha_final - fecha_inicial
print("Fecha inicial:", fecha_inicial)
print("Fecha final:", fecha_final)
print("Diferencia:", diferencia.days, "días")
```

Diferencia de dos fechas en días, introducidas por teclado

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#  diferencia-entre-fechas-teclado.py
#
#  Diferencia de dos fechas en días introducidas por teclado,
#  con control de errores.
```

```
#

from datetime import datetime

def main():
 # Establecer formato de las fechas a introducir: dd/mm/aaaa

 formato = "%d/%m/%Y"

 # Bucle 'sin fin'

 while True:
  try:
   # Introducir fecha inicial utilizando el formato definido

   fecha_desde = input('Introducir fecha inicial (dd/mm/aaaa): ')

   # Si no se introduce ningún valor se fuerza el final del bucle

   if fecha_desde == "":
    break

   # Introducir fecha final utilizando el formato definido

   fecha_hasta = input('Introducir fecha final   (dd/mm/aaaa): ')

   # Si no se introduce ningún valor se fuerza el final del bucle

   if fecha_hasta == "":
    break

   # Se evaluan las fechas según el formato dd/mm/aaaa
   # En caso de introducirse fechas incorrectas se capturará
   # la excepción o error

   fecha_desde = datetime.strptime(fecha_desde, formato)
   fecha_hasta = datetime.strptime(fecha_hasta, formato)

   # Se comprueba que fecha_hasta sea mayor o igual que fecha_desde

   if fecha_hasta >= fecha_desde:

    # Se cálcula diferencia en día y se muestra el resultado

    diferencia = fecha_hasta - fecha_desde
    print("Diferencia:", diferencia.days, "días")

   else:
    print("La fecha fecha final debe ser mayor o igual que la inicial")

  except:
   print('Error en la/s fecha/s. ¡Inténtalo de nuevo!')

 return 0

if __name__ == '__main__':
 main()
```

A partir de una hora se obtiene fracción del día

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# A partir de una hora introducida por teclado se obtiene
# fracción del día, teniendo en cuenta que 24 horas = 86400 seg
# Formato de entrada: hh:mm:ss
# Valores Hora...: 0 a 23
# Valores Minuto.: 0 a 59
# Valores Segundo: 0 a 59

from datetime import datetime
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!')
```

Diferencia de dos fechas (date)

```
hoy = date.today()
navidad_año_proximo = date(2024, 12, 25)
faltan = navidad_año_proximo - hoy
print ("Hoy:", hoy)
print ("La navidad del 2024", navidad_año_proximo)
print ("Faltan", faltan.days, "días")
```

Expresar una fecha en formato largo

```
print("Hoy es...", datetime.ctime(fecha1))
```

A partir de una fecha se obtiene tupla con año, nº semana y día de semana

```
print("Fecha", fecha1,
      "Año, nº sem., día sem.:",
      datetime.isocalendar(fecha1))
```

A partir de una fecha se obtiene tupla con año, nº semana y día de semana

También, se muestra el día de la semana en letras (abreviado).

```
print("Desglose de la fecha", fecha2,":")
tupla_mensajes = ("Año", "Núm. semana", "Núm. día de semana")
tupla_valores = datetime.isocalendar(fecha2)
tupla_diassem = ("Lun", "Mar", "Mié", "Jue","Vie", "Sáb","Dom")
for mensaje,valor in zip(tupla_mensajes, tupla_valores):
 print(mensaje,"-->", valor)

print("Día de semana-->", tupla_diassem[tupla_valores[2]-1])
```

Obtener día de la semana por su número

La función **weekday()** devuelve el número de día de la semana a que corresponda la fecha indicada, según los siguientes valores por día: 0-Lunes, 1-Martes, 2-Miércoles, 3-Jueves, 4-Viernes , 5-Sábado y 6-Domingo

```
dia_semana = datetime.weekday(fecha1)
print(fecha1, "->", dia_semana,"->",
      tupla_diassem[dia_semana])
```

Obtener y contar los días que sean martes entre dos fechas

```
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')
```

Obtener día de la semana por su número

La función **isoweekday()** devuelve el número de día de la semana a que corresponda la fecha indicada, según los siguientes valores por día: 1-Lunes, 2-Martes, 3-Miércoles, 4-Jueves, 5-Viernes, 6-Sábado y 7-Domingo.

```
dia_semana = datetime.isoweekday(fecha1)
print(fecha1, "->", dia_semana,"->",
      tupla_diassem[dia_semana-1])
```

Dado el ordinal se obtiene la fecha correspondiente

```
print("Si la fecha 01-01-0001 tiene el ordinal 1 entonces...")
for num in range(1,7):
    print("El día", 10 ** num ,
          "se corresponde con",
          date.fromordinal(10 ** num))
```

Dada una fecha se obtiene un ordinal (01-01-0001 -> 1)

```
fecha3 = datetime(1, 1, 1, 0, 0, 0)
print("La fecha", fecha3,
      "tiene el ordinal",
      date.toordinal(fecha3))
print("La fecha", fecha1,
      "tiene el ordinal",
      date.toordinal(fecha1))
```

Obtener una tupla a partir de fecha-hora (datetime)

```
tupla_fechahora = fecha1.timetuple()
for elemento in tupla_fechahora:
 print(elemento)
```

Convertir un ordinal en fecha-hora (fromtimestamp)

El ordinal 0 se corresponde con la fecha -> 1-1-1970 01:00:00

```
print("Ordinal 0 -> 1-1-1970 01:00:00")
ordinales_tiempo = (0, 1, 2, 60, 3600)
for elemento in ordinales_tiempo:
    print(elemento, "->" , datetime.fromtimestamp(elemento))
```

Obtener calendario del mes actual (calendar.month)

```
año = date.today().year
mes = date.today().month
calendario_mes = calendar.month(año, mes)
print(calendario_mes)
```

Obtener calendario del mes actual (calendar.TextCalendar)

Se establece el lunes como primer día de la semana

```
calendario = calendar.TextCalendar(calendar.MONDAY)
calendario.prmonth(año, mes)
```

Obtener matriz con calendario de mes actual: (Calendar monthdayscalendar)

```
calendario = calendar.Calendar()
for elemento in calendario.monthdayscalendar(año, mes):
    print(elemento)
```

Obtener matriz de tuplas con calendario: (Calendar monthdays2calendar)

El primer valor de cada par es el número de día del mes y el segundo valor se corresponde con el número de día de la semana: 0:lunes, 1:martes, 2:miércoles, etc.

```
calendario = calendario.monthdays2calendar(año, mes)
for elemento in calendario:
    print(elemento)
```

Calendario completo del año 2018

Día de comienzo: Lunes

```
print(calendar.TextCalendar(calendar.MONDAY).formatyear(2018,
                                                        2, 1, 1, 2))
```

Relacionado:

- El módulo time

Ir al índice del tutorial de Python

Publicado por Pherkad en 11:49

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua