Datetime in Python

In this article, we will work with you to learn how to handle dates and times in Python with specific examples to make it easier to visualize and capture functions better. Invites you to read the track.

In this article, we will work with you to learn how to handle dates and times in Python with specific examples to make it easier to visualize and capture functions better. Invites you to read the track.

Category

1. Introduce
2. What's inside datetime?
3. Datetime.date class
4. Datetime.time class
5. Datetime.datetime class
6. Class datetime.timedelta
7. Python time format
    1. strftime () - format the time into a string.
    2. strftime () - analyze a string into time
8. Display time zone in Python

Introduce

Python has the **datetime mudule** to work and handles the same date and time. Let's try out some simple programs before we dig deeper.

Example 1: Returns the current date and time

```
import datetime

datetime_object = datetime.datetime.now()
print(datetime_object)
```

When you run the program, the output will be in the form of:

```
2019-03-06 11:13:33.969330
```

In this example, we have just entered the **datetime module** using the *import datetime* statement.

The datetime class is declared in the **datetime module** , then use the *now ()* command to create a *datetime* object containing the current local date and time.

Example 2: Returns the current date

```
import datetime

date_object = datetime.date.today()
print(date_object)
```

When you run the program, the output will be in the form of:

```
2019-03-06
```

In this program, we use *today ()* command which is declared in the *date class* to get the current local date result.

What's inside datetime?

You can use the *dir ()* function to display a list of all properties of the **datetime module** .

```
import datetime

print(dir(datetime))
```

Run the program, the result is:

```
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__', '__l

 '__name__', '__package__', '__spec__', 'date', 'datetime', 'datetime_CAPI', 'sys
 'timedelta', 'timezone', 'tzinfo']
```

The classes commonly used in the **datetime module** are:

1. *Class date*
2. *Class time*
3. *Datetime class*
4. *Class timedelta*

Datetime.date class

The object of the **date class** returns the **date information** (date), not the time information. The date is transmitted as 'year, month, day'.

```
import datetime

d = datetime.date(2019, 4, 12)
print(d)
```

When you run the program, the output will be in the form of:

```
2019-04-12
```

You can also update the *date class* from the **datetime module** like this:

```
from datetime import date

a = date(2019, 4, 12)
print(a)
```

Example 3: Update the current date

Using *today () method*

```
from datetime import date

today = date.today()
print("Ngay hien tai la:", today)
```

Example 4: Call the date from the timestamp

Unix Time (Unix timestamp) is a system for expressing a point on the time axis, according to the time axis it uses the number of seconds to determine the time, with the original point from 00:00:00 on January 1 / 1970 UTC hour.

For example at 00:00:00 - 12/02/2016 the timestamp value is 1455235200; It means that from 00:00 to 1/1/1970 to 00:00:00 - 12/02/2016 is 1455235200 seconds.

And in Python, you can create date objects from a *timestamp.*

```
from datetime import date

timestamp = date.fromtimestamp(1551916800)
print("Date =", timestamp)
```

When you run the program, the output will be in the form of:

```
Date = 2019-03-07
```

Example 5: Print the current date

```
from datetime import date

# vi?t b?i TipsMake.com
today = date.today()

print("Nam hien tai:", today.year)
print("Thang hien tai:", today.month)
print("Ngay hien tai:", today.day)
```

When you run the program, the output will be in the form of:

```
Nam hien tai: 2019
Thang hien tai: 3
Ngay hien tai: 7
```

Datetime.time class

The object of *class time* returns the current time information, not including date information.

```
from datetime import time

# time(hour = 0, minute = 0, second = 0)
a = time()
print("a =", a)

# time(hour, minute and second)
b = time(11, 34, 56)
print("b =", b)

# time(hour, minute and second)
# viet boi TipsMake.com
c = time(hour = 11, minute = 34, second = 56)
print("c =", c)

# time(hour, minute, second, microsecond)
d = time(11, 34, 56, 234566)
print("d =", d)
```

The result is:

```
a = 00:00:00
b = 11:34:56
c = 11:34:56
d = 11:34:56.234566
```

Example 6: Print hours, minutes, seconds and microseconds

```
from datetime import time

a = time(11, 34, 56)

print("hour =", a.hour)
print("minute =", a.minute)
print("second =", a.second)
print("microsecond =", a.microsecond)
```

When you run the program, the output will be in the form of:

```
hour = 11
minute = 34
second = 56
microsecond = 0
```

In the above example we do not pass parameters for microseconds so the return value is in the default form of 0.

Datetime.datetime class

The object of the *datetime class* returns results that include **both time and date information** .

```
 from datetime import datetime

 #datetime ???c truy?n ? d?ng 'year, month, day'.
 a = datetime(2019, 3, 7)
 print(a)

 # datetime ???c truy?n ? d?
ng 'year, month, day, hour, minute, second, microsecond'
 b = datetime(2019, 3, 7, 23, 55, 59, 342380)
 print(b)
```

Running the program is the result:

```
 2019-03-07 00:00:00
 2019-03-07 23:55:59.342380
```

The first 3 parameters 'year, month, day' are required.

Example 7: Print the year, month, hour, minute, timestamp

```
 from datetime import datetime

 a = datetime(2019, 3, 7, 23, 55, 59, 342380)
 print("year =", a.year)
 print("month =", a.month)
 print("hour =", a.hour)
 print("minute =", a.minute)
 print("timestamp =", a.timestamp())
```

The result is:

```
 year = 2019
 month = 3
 hour = 23
 minute = 55
 timestamp = 1551977759.34238
```

Class datetime.timedelta

Timedelta is a **time period that describes the difference between two timelines** .

```
 from datetime import datetime, date

 # kho?ng th?i gian chênh l?ch gi?a 2 ngày tháng
 t1 = date(year = 2018, month = 7, day = 12)
 t2 = date(year = 2017, month = 12, day = 23)
 t3 = t1 - t2
 print("t3 =", t3)


t4 = datetime(year = 2018, month = 7, day = 12, hour = 7, minute = 9, second = 3
```

```
t5 = datetime(year = 2019, month = 6, day = 10, hour = 5, minute = 55, second = 1

 t6 = t4 - t5
 print("t6 =", t6)
```

The result is:

```
 t3 = 201 days, 0:00:00
 t6 = -333 days, 1:14:20
```

t3 and t6 here are *timedelta* objects

Example 8: Time difference between two timedelta objects

```
 from datetime import timedelta

 t1 = timedelta(weeks = 2, days = 5, hours = 1, seconds = 33)
 t2 = timedelta(days = 4, hours = 11, minutes = 4, seconds = 54)
 t3 = t1 - t2

 print("t3 =", t3)
```

Running the program, we get the following output:

```
 t3 = 14 days, 13:55:39
```

Example 9: Handling timedelta with negative value

```
 from datetime import timedelta

 t1 = timedelta(seconds = 33)
 t2 = timedelta(seconds = 54)
 t3 = t1 - t2

 print("t3 =", t3)
 print("t3 =", abs(t3))
```

The results are:

```
 t3 = -1 day, 23:59:39
 t3 = 0:00:21
```

Example 10: Convert the timedelta difference time to the total number of seconds

You can convert the result to the total number of seconds using *Total_seconds () method.*

```
 from datetime import timedelta

 t = timedelta(days = 5, hours = 1, seconds = 33, microseconds = 233423)
 print("tong so giay =", t.total_seconds())
```

We get the result:

```
Tong so giay = 435633.233423
```

Operators support timedelta:

1. ***t1 = t2 + t3***
   t2 = (hours = 8, seconds = 12)
   t3 = (hours = 2, minutes = 3)
   >>> t1 = (hours = 10, minutes = 3, seconds = 12)
2. ***t1 = t2 - t3***
   t2 = (hours = 12, seconds = 2)
   t3 = (hours = 1, minutes = 4)
   >>> t1 = (hours = 11, minutes = 56, seconds = 2)
3. ***t1 = t2 * i***
   t1 = i * t2
   t2 = (hours = 10, seconds = 2)
   i = 3
   >>> t1 = (days = 1, hours = 6, seconds = 6)
4. ***t1 = t2***
   t2 = (hours = 25, seconds = 2)
   >>> t1 = (days: 1, hours: 1, seconds: 2)
5. ***+ t1***
   Returns t1
6. ***-t1***
   t1 = (hours = 10, seconds = 2)
   >>> -t1 = (days = -1, hours = 13, minutes = 59, seconds = 58)
7. ***abs (t) :*** Absolute value, equivalent to + t when t.days> = 0, and is -t when t.days 0.
   t = (hours = -25, minutes = 3)
   >>> t = (days = -2, hours = 23, minutes = 3)
   >>> abs (t) = (days = 1, hours = 0, minutes = 57)
8. ***str (t) :*** Returns the string according to the form [D day [s],] [H] H: MM: SS [.UUUUUU], D can receive negative values.
9. ***repr (t) :*** Returns the string according to the form datetime.timedelta (D [, S [, U]]), D can receive negative values

Python time format

Dates and times are used in different forms, the US uses *mm / dd / yyyy* format, while *dd / mm / yyyy is* more common in the UK.

Python has *strftime ()* and *strptime ()* methods to handle this.

strftime () - format the time into a string.

Example 11: Date format using strftime ()

```
from datetime import datetime

# ngày gi? hi?n t?i
```

```
now = datetime.now()

t = now.strftime("%H:%M:%S")
print("time:", t)

s1 = now.strftime("%m/%d/%Y, %H:%M:%S")
# ??nh d?ng mm/dd/YY H:M:S
print("s1:", s1)

s2 = now.strftime("%d/%m/%Y, %H:%M:%S")
# ??nh d?ng dd/mm/YY H:M:S
print("s2:", s2)
```

The program returns the result:

```
time: 04:34:52
s1: 03/07/2019, 04:34:52
s2: 07/03/2019, 04:34:52
```

Here *% Y,% m,% d,% H* . are format parameters. The *strftime ()* method is used to return a string formatted based on those parameters.

Parameter scope:

1. % Y - year [0001, ., 2018, 2019, ., 9999]
2. % m - month [01, 02, ., 11, 12]
3. % d - day [01, 02, ., 30, 31]
4. % H - hour [00, 01, ., 22, 23
5. % M - month [00, 01, ., 58, 59]
6. % S - second [00, 01, ., 58, 59]

strftime () - analyze a string into time

```
from datetime import datetime

date_string = "7 March, 2019"
print("date_string =", date_string)

date_object = datetime.strptime(date_string, "%d %B, %Y")
print("date_object =", date_object)
```

When you run the program, the result is:

```
date_string = 7 March, 2019
date_object = 2019-03-07 00:00:00
```

Display time zone in Python

Suppose, you are working on a project and need to display the date and time based on the time zone you need. Instead of trying to search and search for the time zone yourself, you should use the **module pytZ** as follows:

```
from datetime import datetime
import pytz

local = datetime.now()
print("Local:", local.strftime("%m/%d/%Y, %H:%M:%S"))

tz_NY = pytz.timezone('America/New_York')
datetime_NY = datetime.now(tz_NY)
print("NY:", datetime_NY.strftime("%m/%d/%Y, %H:%M:%S"))

tz_London = pytz.timezone('Europe/London')
datetime_London = datetime.now(tz_London)
print("London:", datetime_London.strftime("%m/%d/%Y, %H:%M:%S"))
```

Run the program, the result is:

```
Local time: 2019-03-07 13:10:44.260462
America/New_York time: 2019-03-07 13:10:44.260462
Europe/London time: 2019-03-07 13:10:44.260462
```

In this example, *datetime_NY* and *datetime_London* are *datetime* objects that contain the current date and time of the corresponding time zone.