

DateTime in Python

Course Contents

- Introduction
- `datetime.date`
- `datetime.time`
- `datetime.datetime`
- `datetime.timedelta`
- Python format datetime
- Handling timezone in Python

Introduction

- Python has a module named **datetime** to work with dates and times. Following are simple programs to understand date and time.

```
import datetime
```

```
# Get Current Date and Time
```

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

```
datetime.datetime(2019, 12, 10, 21, 3, 58, 467962)
```

```
# Get Current Date
```

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

```
datetime.date(2019, 12, 10)
```

- To check what are the methods available in datetime module.

```
dir(datetime)
```

```
['MAXYEAR',  
 'MINYEAR',  
 '__builtins__',  
 '__cached__',  
 'date',  
 'datetime',  
 'datetime_CAPI',  
 'time',  
 'timedelta',  
 'timezone',  
 'tzinfo']
```

Classes in DateTime Module

- date Class
- time Class
- datetime Class
- timedelta Class

datetime.date Class

- We can instantiate date objects from the date class.
A date object represents a date (year, month and day).

```
d = datetime.date(2019, 4, 13)  
d
```

```
datetime.date(2019, 4, 13)
```

```
t = datetime.date.today()
```

```
t.year
```

```
2019
```

```
t.month
```

```
12
```

```
t.day
```

```
10
```

Get date from a timestamp

- A Unix timestamp is the number of seconds between a particular date and January 1, 1970 at UTC.

```
timestamp = datetime.date.fromtimestamp(1326244364)  
timestamp
```

```
datetime.date(2012, 1, 11)
```

datetime.time class

- A time object instantiated from the time class represents the local time.

```
# time(hour = 0, minute = 0, second = 0)
a = datetime.time()
a
```

```
datetime.time(0, 0)
```

```
# time(hour, minute and second)
b = datetime.time(11, 34, 56)
b
```

```
datetime.time(11, 34, 56)
```

```
# time(hour, minute and second)
c = datetime.time(hour=11, minute=34, second=56)
c
```

```
datetime.time(11, 34, 56)
```

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

```
datetime.time(11, 34, 56, 234566)
```

- Once we create a time object, we can easily print its attributes such as hour, minute etc.

```
a = datetime.time(11, 34, 56)
```

```
a.hour
```

```
11
```

```
a.second
```

```
56
```

```
a.minute
```

```
34
```

```
a.microsecond
```

```
0
```

datetime.datetime class

- The datetime module has a class named **datetime** that can contain information from both date and time objects.

```
#datetime(year, month, day)  
a = datetime.datetime(2018, 11, 28)  
a
```

```
datetime.datetime(2018, 11, 28, 0, 0)
```

```
# datetime(year, month, day, hour, minute, second, microsecond)  
b = datetime.datetime(2017, 11, 28, 23, 55, 59, 342380)  
b
```

```
datetime.datetime(2017, 11, 28, 23, 55, 59, 342380)
```

datetime.datetime class

- Print year, month, hour, minute and timestamp

```
a = datetime.datetime(2017, 11, 28, 23, 55, 59, 342380)
```

```
a.year
```

```
2017
```

```
a.month
```

```
11
```

```
a.hour
```

```
23
```

```
a.minute
```

```
55
```

```
a.timestamp()
```

```
1511893559.34238
```


datetime.timedelta class

- A timedelta object represents the difference between two dates or datetimes.

```
t1 = datetime.date(year = 2018, month = 7, day = 12)
t2 = datetime.date(year = 2017, month = 12, day = 23)
```

```
t3 = t1 - t2
t3
```

```
datetime.timedelta(201)
```

```
t4 = datetime.datetime(year = 2018, month = 7, day = 12, hour = 7, minute = 9, second = 33)
t5 = datetime.datetime(year = 2019, month = 6, day = 10, hour = 5, minute = 55, second = 13)
```

```
t6 = t4 - t5
t6
```

```
datetime.timedelta(-333, 4460)
```

```
type(t3)
```

```
datetime.timedelta
```

```
type(t6)
```

```
datetime.timedelta
```

datetime.timedelta class

- Difference between two timedelta objects

```
t1 = datetime.timedelta(weeks = 2, days = 5, hours = 1, seconds = 33)
t2 = datetime.timedelta(days = 4, hours = 11, minutes = 4, seconds = 54)
```

```
t3 = t1 - t2
t3
```

```
datetime.timedelta(14, 50139)
```

- Printing negative timedelta object



```
t1 = datetime.timedelta(seconds = 33)
t2 = datetime.timedelta(seconds = 54)
```

```
t3 = t1 - t2
t3
```

```
datetime.timedelta(-1, 86379)
```

```
abs(t3)
```

```
datetime.timedelta(0, 21)
```

Python format datetime

- The way date and time is represented may be different in different places, organizations etc.
- It's more common to use **mm/dd/yyyy** in the US, whereas **dd/mm/yyyy** is more common in the UK.
- Python has **strftime()** and **strptime()** methods to handle this.

Python strftime() - datetime object to string

- The strftime() method is defined under classes date, datetime and time.
- The method creates a formatted string from a given date, datetime or time object.

```
# current date and time  
n = datetime.datetime.now()
```

```
# H:M:S format  
t = n.strftime("%H:%M:%S")  
t
```

```
'22:06:00'
```

```
# mm/dd/YY H:M:S format  
s1 = n.strftime("%m/%d/%Y, %H:%M:%S")  
s1
```

```
'12/10/2019, 22:06:00'
```

```
# dd/mm/YY H:M:S format  
s2 = n.strftime("%d/%m/%Y, %H:%M:%S")  
s2
```

```
'10/12/2019, 22:06:00'
```

Python's <code>strftime</code> directives		
Note: Examples are based on <code>datetime.datetime(2013, 9, 30, 7, 6, 5)</code>		
Code	Meaning	Example
<code>%a</code>	Weekday as locale's abbreviated name.	Mon
<code>%A</code>	Weekday as locale's full name.	Monday
<code>%w</code>	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	1
<code>%d</code>	Day of the month as a zero-padded decimal number.	30
<code>%-d</code>	Day of the month as a decimal number. (Platform specific)	30
<code>%b</code>	Month as locale's abbreviated name.	Sep
<code>%B</code>	Month as locale's full name.	September
<code>%m</code>	Month as a zero-padded decimal number.	09
<code>%-m</code>	Month as a decimal number. (Platform specific)	9
<code>%y</code>	Year without century as a zero-padded decimal number.	13
<code>%Y</code>	Year with century as a decimal number.	2013
<code>%H</code>	Hour (24-hour clock) as a zero-padded decimal number.	07

Python strptime() - string to datetime

- The strptime() method creates a datetime object from a given string (representing date and time).

```
date_string = "21 June, 2018"  
date_string
```

```
'21 June, 2018'
```

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

```
datetime.datetime(2018, 6, 21, 0, 0)
```

Handling timezone in Python

- Suppose, we are working on a project and need to display date and time based on their timezone. Rather than trying to handle timezone ourself, we use a third-party **pytz** module.

```
import pytz
local = datetime.datetime.now()
```

```
local.strftime("%m/%d/%Y, %H:%M:%S")

'12/10/2019, 22:14:56'
```

```
tz_NY = pytz.timezone('America/New_York')
```

```
datetime_NY = datetime.datetime.now(tz_NY)
```

```
datetime_NY.strftime("%m/%d/%Y, %H:%M:%S")

'12/10/2019, 11:46:11'
```

```
pytz.all_timezones
```

```
['Africa/Abidjan',
 'Africa/Accra',
 'Africa/Addis_Ababa',
 'Africa/Algiers',
 'Africa/Asmara',
 'Africa/Asmera',
```