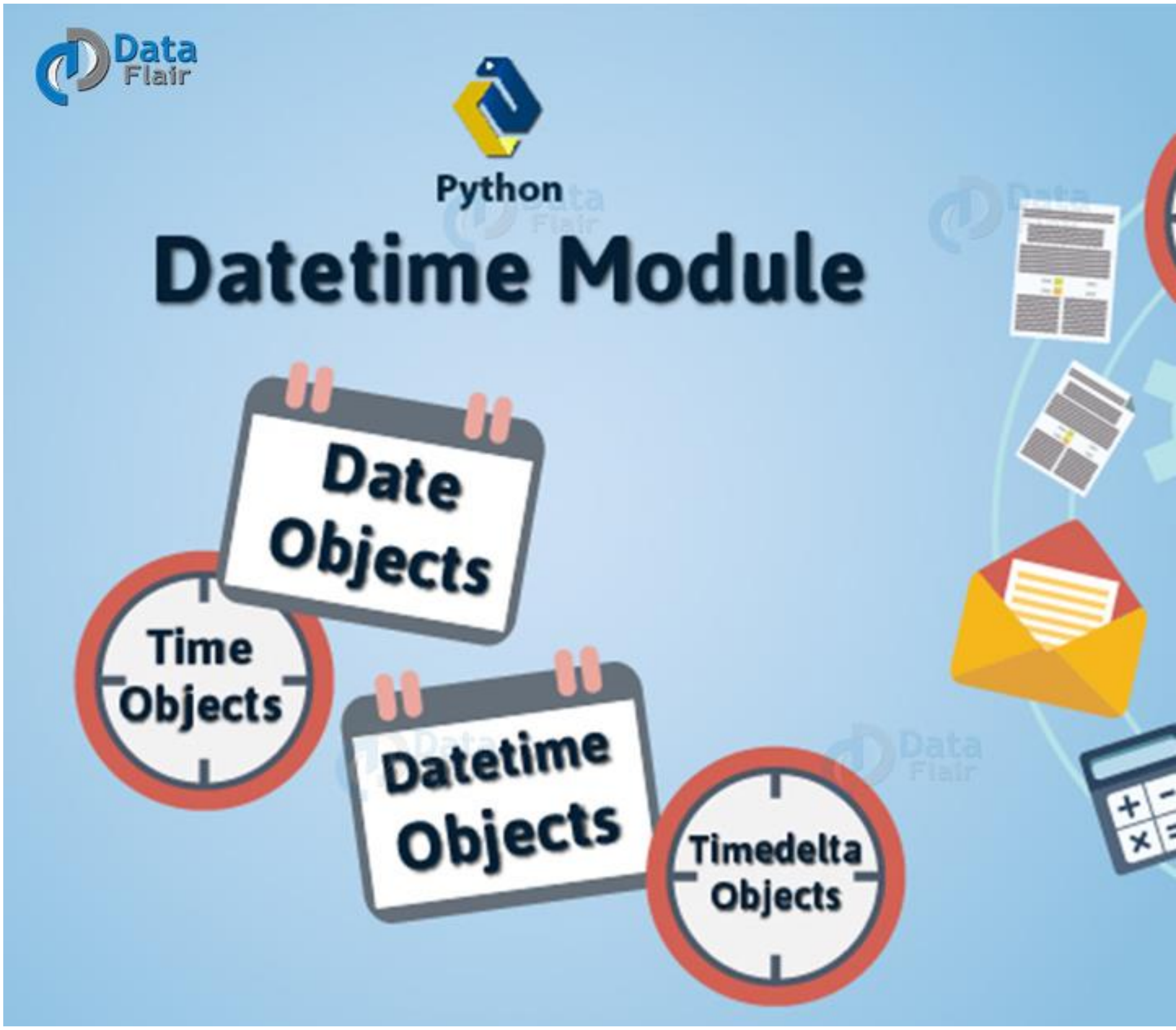
1. Python Datetime Module

With the ever-complex lifestyle we have adopted, date and time are extremely important. It is like we're all slaves to time. In this tutorial, we will see how to work with Python Datetime Module.

Before we begin, you should refer to Python Date and Time.

2. The Datetime Module



Introduction to Python Datetime Module

The datetime module offers functions and classes for working with date and time parsing, formatting, and arithmetic. Lets start with Python Date time with Examples.

Two kinds of date and time objects exist- naïve and aware.

A naïve object is easier to deal with, and doesn't have enough information to unambiguously locate itself relative to other date/time objects. An aware object is more realistic, it knows applicable algorithmic and political time adjustments.

To use this module, we must first import it.

It has the following constants:

a. MAXYEAR

MAXYEAR tells us the maximum value a year can take, which is 9999.

```
9999
```

b. MINYEAR

The minimum value of year is 1. This is what MINYEAR returns.

```
1
```

Both MAXYEAR and MINYEAR are of the type integer.

1. >>> type(datetime.MAXYEAR),type(datetime.MINYEAR)

```
(<class ‘int’>, <class ‘int’>)
```

Bonus- type() is of the type type.

```
<class ‘type’>
```

Other than these constants, datetime has these datetime class types:

1. **class datetime.date**

Python date is an idealized naïve date considering the current Gregorian calendar.

Attributes: year, month, and day.

2. **class datetime.time**

Python time is an idealized time, independent of any particular day. Here, we assume that each day is made of exactly 24*60*60 seconds (no leap seconds).

Attributes: hour, minute, second, microsecond, and tzinfo.

3. **class datetime.datetime**

When you combine a date and a time, you get a datetime.

1. >>> issubclass(datetime.datetime,datetime.date)

True

1. >>> issubclass(datetime.datetime,datetime.time)

False

Attributes: year, month, day, hour, minute, second, microsecond, and tzinfo.

4. **class datetime.timedelta**

A timedelta is a duration expressing the difference between two date, time, or datetime instances to microsecond resolution.

5. **class datetime.tzinfo**

tzinfo is an abstract base class we use for time zone information objects. The date and time classes use it to provide a customizable notion of time adjustment (for example, to account for time zone and/or DST(daylight saving time)).

6. **class datetime.timezone**

timezone implements the tzinfo abstract base class as a fixed offset from the UTC.

Such objects are immutable. Also, objects of the type ‘date’ is naïve, but those of the types ‘time’ or ‘datetime’ may be aware or naïve. The next in Python Datetime Module Tutorial is Date Objects

3. date Objects

A date object represents a date with year, month, and day, according to the ideal Gregorian calendar.

With a date object, we have the following methods:

a. date(year, month, day)

This method will create an object of type ‘date’.

```
1. >>> d=datetime.date(2018,2,28)
```

Here, year can be from MAXYEAR to MINYEAR, and month can be from 1 to 12. The day can be from 1 to the number of days in the given month for the given year.

b. today()

today() will return the current local date.

```
1. >>> datetime.date.today()
```

```
datetime.date(2018, 2, 28)
```

c. fromtimestamp(timestamp)

fromtimestamp will return the date for the Python timestamp provided.

```
1. >>> import time
2. >>> time.time()
```

```
1519818814.358453
```

```
1. >>> datetime.date.fromtimestamp(time.time())
```

```
datetime.date(2018, 2, 28)
```

```
1. >>> datetime.date.fromtimestamp(time.time()+999999)
```

```
datetime.date(2018, 3, 12)
```

The date class also has these attributes:

a. date.min

It returns the earliest representable date.

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

b. date.max

Like min, max returns the latest representable date.

```
datetime.date(9999, 12, 31)
```

c. date.resolution

resolution returns the smallest possible difference between non-equal date objects.

```
>>> datetime.date.resolution
```

```
datetime.timedelta(1)
```

We also have the following instance attributes:

a. year

This returns the year from a date object.

```
>>> d=datetime.date(2018,2,28)
>>> d.year
```

```
2018
```

b. month

month returns the month from a date object.

```
2
```

c. day

This returns the day from a date object.

```
28
```

And then, we have the following instance methods:

a. replace(year, month, day)

This will let us update any or all of these three values.

```
>>> d=d.replace(month=3,day=1)
```

```
datetime.date(2018, 3, 1)
```

b. timetuple()

timetuple() returns a tuple of attributes for the current local time.

time.struct_time(tm_year=2018, tm_mon=2, tm_mday=28, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=2, tm_yday=59, tm_isdst=-1)

c. weekday()

weekday() will return the day of the week, where 0 denotes Monday.

2

d. isoweekday()

Here, Monday is 1.

3

e. isocalendar()

This method returns a tuple of three things- ISO year, ISO week number, and ISO weekday.

(2018, 9, 3)

f. isoformat()

This returns the date in the ISO format.

‘2018-02-28’

g. __str__()

This will return the current local date as a string.

‘2018-02-28’

‘2018-02-28’

h. ctime()

ctime returns the current local date in a string format.

‘Wed Feb 28 00:00:00 2018’

We can also perform some operations on a date object:

```
>>> d=datetime.date(2018,12,30)
>>> td=datetime.timedelta(0,99999)
```

3. >>> d1=d+td
4. >>> d1

datetime.date(2018, 12, 31)

False

Any doubt yet in Date time module in Python or Examples? Please comment.

The Next in Python Datetime Module Tutorial is Time Objects

4. time Objects

A time object represents a local time of day. Its arguments are- hour, minute, second, microsecond, and tzinfo.

1. >>> t=datetime.time(11,59,59,99999)

The class time has the following attributes:

a. min

This returns the earliest representable time.

datetime.time(0, 0)

b. max

This returns the latest representable time.

datetime.time(23, 59, 59, 999999)

c. resolution

resolution returns the smallest possible difference between non-equal time objects.

datetime.timedelta(0, 0, 1)

Now, let’s look at some non-readable instance attributes:

a. hour

This tells us the hour from the object.

b. minute

This returns the minute.

59

c. second

This returns the second.

59

d. microsecond

This returns the microsecond.

99999

e. tzinfo

This returns whatever we pass as tzinfo to the object while creating it. If we didn’t, it returns None.

None

f. fold

This disambiguates wall times during a repeated interval.

0

We have the following instance methods:

a. replace()

This does the same as for ‘date’ objects.

```
>>> t=t.replace(hour=22,second=7)
```

b. isoformat()

This method returns the time in the object in the ISO format.

’22:59:07.099999′

c. __str__()

This does the same as it does for 'date' objects.

’22:59:07.099999′

Now in the Python Datetime Module Tutorial Lets move ahead with datetime Objects

5. datetime Objects

A datetime objects knows about the date and the time. Arguments include year, month, day, hour, minute, second, microsecond, tzinfo.

1. >>> d=datetime.datetime(1995,12,31,10,0)

We have the following methods:

a. today()

This method returns the current local datetime.

1. >>> datetime.datetime.today()

datetime.datetime(2018, 2, 28, 19, 9, 40, 751445)

b. now()

now() returns the current local date and time, but is more precise than today().

1. >>> datetime.datetime.now()

datetime.datetime(2018, 2, 28, 19, 12, 45, 710542)

c. utcnow()

utcnow() returns the current UTC time.

1. >>> datetime.datetime.utcnow()

datetime.datetime(2018, 2, 28, 13, 49, 14, 486367)

d. fromtimestamp()

This returns the date and time for the provided timestamp.

1. >>> datetime.datetime.fromtimestamp(time.time())

datetime.datetime(2018, 2, 28, 19, 20, 24, 55451)

e. utctimestamp()

This is like the previous one, but it returns the UTC time.

1. >>> datetime.datetime.utcfromtimestamp(time.time())

datetime.datetime(2018, 2, 28, 13, 51, 56, 615793)

Now, let’s look at the class attributes:

a. min

Like for everything else, this returns the earliest representable datetime.

1. >>> datetime.datetime.min

datetime.datetime(1, 1, 1, 0, 0)

b. max

max returns the latest representable datetime.

1. >>> datetime.datetime.max

datetime.datetime(9999, 12, 31, 23, 59, 59, 999999)

c. resolution

Resolution is about the smallest possible difference between non-equal datetime objects.

1. >>> datetime.datetime.resolution

datetime.timedelta(0, 0, 1)

Now, let’s see some read-only instance attributes:

a. year

This returns the year.

1995

b. month

This returns the month.

```
12
```

c. day

This returns the day.

```
31
```

d. hour

This returns the hour.

```
10
```

e. minute

This returns the minute.

```
0
```

f. second

This returns the second.

```
0
```

g. microsecond

This returns the microsecond.

```
0
```

h. tzinfo

This returns the tzinfo; None, if we passed none while creating the object.

```
None
```

i. fold

This is the same as we discussed before.

0

Finally, let’s try some instance methods.

a. replace()

Again, this is like we have been doing so far.

1. >>> d.replace(microsecond=7)

datetime.datetime(1995, 12, 31, 10, 0, 0, 7)

b. date()

date() returns a date object from the datetime object.

datetime.date(1995, 12, 31)

c. time()

time() returns a time object from the datetime object.

datetime.time(10, 0)

d. timetz()

Other than what time() does, timetz() also returns the tzinfo value.

datetime.time(10, 0)

e. astimezone()

This returns a datetime object with new tzinfo attribute tz, adjusting the date and time data so the result is the same UTC time as self, but in tz’s local time.

datetime.datetime(1995, 12, 31, 10, 0, tzinfo=datetime.timezone(datetime.timedelta(0, 19800), ‘India Standard Time’))

f. utcoffset()

If there is a UTC offset, it returns that, else, None.

None

g. dst()

If Daylight Savings Time applies, it returns its magnitude.

```
None
```

h. tzname

If you would have set tzinfo, this would return its name.

```
None
```

i. timetuple()

Like we saw earlier, this returns a time tuple of the object.

```
time.struct_time(tm_year=1995, tm_mon=12, tm_mday=31, tm_hour=10, tm_min=0, tm_sec=0,
tm_wday=6, tm_yday=365, tm_isdst=-1)
```

j. utctimetuple()

This returns a time tuple of the UTC time.

```
time.struct_time(tm_year=1995, tm_mon=12, tm_mday=31, tm_hour=10, tm_min=0, tm_sec=0,
tm_wday=6, tm_yday=365, tm_isdst=0)
```

k. timestamp()

This returns the timestamp of the object (in seconds).

```
820384200.0
```

l. weekday()

This returns the number of weekday for the object, where 0 is for Monday.

```
6

#A Sunday
```

m. isoweekday()

This returns the day of the week, but with Monday as 1.

```
7
```

n. isocalendar()

This returns a tuple of the following- ISO year, ISO week number, ISO weekday.

(1995, 52, 7)

o. isoformat()

This returns the date and time in the ISO format.

‘1995-12-31T10:00:00’

p. __str__()

This returns the date and time in a string representation.

‘1995-12-31 10:00:00’

‘1995-12-31 10:00:00’

q. ctime()

This, like str, returns the datetime as a string.

‘Sun Dec 31 10:00:00 1995’

Some operations that we can perform on a datetime object are:

```
>>> d=datetime.datetime(2018,12,30)
>>> td=datetime.timedelta(0,99999)
>>> d1=d+td
>>> d1
```

datetime.datetime(2018, 12, 31, 3, 46, 39)

True

6. timedelta Objects

A timedelta object represents the duration between two dates or times.

Attributes: days, seconds, microseconds, milliseconds, minutes, hours, weeks

Here, seconds can be from 0 to 86399.

```
>>> td=datetime.timedelta(9,3,9999,999,58,22,3)
```

These attributes belong to the timedelta class:

a. min

This returns the most negative timedelta object.

1. >>> datetime.timedelta.min

datetime.timedelta(-999999999)

b. max

This one returns the most positive timedelta object.

1. >>> datetime.timedelta.max

datetime.timedelta(999999999, 86399, 999999)

c. resolution

resolution returns the smallest possible difference between non-equal timedelta objects.

1. >>> datetime.timedelta.resolution

datetime.timedelta(0, 0, 1)

timedelta also supports one instance method:

a. total_seconds()

This returns the total number of seconds in the duration.

2674684.008999

You can perform arithmetic operations on timedelta objects.

1. >>> td=datetime.timedelta(0,333)
2. >>> td1=td*2
3. >>> td1

datetime.timedelta(0, 666)

This was All about the Python Datetime Module Tutorial.

7. Conclusion: Python Datetime Module Tutorial

We hope that after this article on Python datetime Module Tutorial, and after Date and Time, you will be able to handle all date and time easily. If you think this is a bit too much for once, don’t panic. Work on it twice or thrice. Practice will do it for you. And comment if you need any help on Python Datetime examples.