Date Time

1. In Python, date and time are not a data type of their own, but a module named **datetime**.
2. It can be imported to work with the date as well as time
3. Python Datetime module supplies classes to work with date and time.
4. These classes provide a number of functions to deal with dates, times and time intervals.
5. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

The date time module categorized into 6 main classes

Date,datetime,time,timedelta,tzinfo,timezone

1. [date](https://www.geeksforgeeks.org/python-datetime-date-class/)

An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Its attributes are year, month and day.

Syntax: class datetime.date(year, month, day)

*import datetime*

*my\_date=datetime.date(2022,3,30)*

*print("passed date is: " , my\_date)*

*#OR*

*from datetime import date*

*my\_date=date(2022,3,30)*

*print("passed date is: " , my\_date)*

Let’s see the attributes provided by this class –

|  |  |
| --- | --- |
| min | The minimum representable date |
| max | The maximum representable date |
| resolution | The minimum possible difference between date objects |
| year | The range of year must be between MINYEAR and MAXYEAR |
| month | The range of month must be between 1 and 12 |
| day | The range of day must be between 1 and number of days in the given month of the given year |

*#min max attribute of class date*

*from datetime import date*

*# Getting min date*

*mindate = date.min*

*print("Min Date supported", mindate)*

*# Getting max date*

*maxdate = date.max*

*print("Max Date supported", maxdate)*

*from datetime import date*

*# creating the date object*

*Date = date(2020, 12, 11)*

*# Accessing the attributes*

*print("Year:", Date.year)*

*print("Month:", Date.month)*

*print("Day:", Date.day)*

List of functions from Date class

| Function Name | Description |
| --- | --- |
| ctime() | Return a string representing the date (o/p= Tue Sep 20 00:00:00 2022  ) |
| fromisocalendar() | Returns a date corresponding to the ISO calendar This date is defined by the year, the week number, and the day of the week  syntax : fromisocalender(year,week,day) |
| fromisoformat() | Returns a date object from the string representation of the date |
| fromordinal() | Returns a date object from the proleptic Gregorian ordinal, where January 1 of year 1 has ordinal 1 |
| fromtimestamp() | Returns a date object from the POSIX timestamp |
| isocalendar() | Returns a tuple year, week, and weekday |
| isoformat() | Returns the string representation of the date |
| isoweekday() | Returns the day of the week as integer where Monday is 1 and Sunday is 7 |
| replace() | Changes the value of the date object with the given parameter |
| strftime() | Returns a string representation of the date with the given format |
| timetuple() | Returns an object of type time.struct\_time |
| today() | Returns the current local date |
| toordinal() | Return the proleptic Gregorian ordinal of the date, where January 1 of year 1 has ordinal 1 |
| weekday() | Returns the day of the week as integer where Monday is 0 and Sunday is 6 |

1. [time](https://www.geeksforgeeks.org/python-datetime-time-class/)

An idealized time, independent of any particular day, assuming that every day has exactly 24\*60\*60 seconds. Its attributes are hour, minute, second, microsecond, and tzinfo.

Synatx –

*class datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None, \*, fold=0)*

All the arguments are optional. tzinfo can be None otherwise all the attributes must be integer

Class attributes as follows

| Attribute Name | Description |
| --- | --- |
| min | Minimum possible representation of time |
| max | Maximum possible representation of time |
| resolution | The minimum possible difference between time objects |
| hour | The range of hour must be between 0 and 24 (not including 24) |
| minute | The range of minute must be between 0 and 60 (not including 60) |
| second | The range of second must be between 0 and 60 (not including 60) |
| microsecond | The range of microsecond must be between 0 and 1000000 (not including 1000000) |
| tzinfo | The object containing timezone information |
| fold | Represents if the fold has occurred in the time or not |

#attributes of the class time

*from datetime import time*

*mintime = time.min*

*print("Min Time supported", mintime)*

*maxtime = time.max*

*print("Max Time supported", maxtime)*

*# Creating Time object*

*Time = time(12,24,36,1212)*

*print("Hour:", Time.hour)*

*print("Minutes:", Time.minute)*

*print("Seconds:", Time.second)*

*print("Microseconds:", Time.microsecond)*

List of functions from class Time

| Function Name | Description |
| --- | --- |
| dst() | Returns tzinfo.dst() is tzinfo is not None |
| fromisoformat() | Returns a time object from the string representation of the time |
| isoformat() | Returns the string representation of time from the time object |
| replace() | Changes the value of the time object with the given parameter |
| strftime() | Returns a string representation of the time with the given format |
| tzname() | Returns tzinfo.tzname() is tzinfo is not None |
| utcoffset() | Returns tzinfo.utcffsets() is tzinfo is not None |

*#functions of Time class*

*from datetime import time*

*# Creating Time object*

*Time = time(12,24,36,1212)*

*# Converting Time object to string*

*Str = Time.isoformat()*

*print("String Representation:", Str)*

*print(type(Str))*

*Time = "12:24:36.001212"*

*# Converting string to Time object*

*Time = time.fromisoformat(Str)*

*print("\nTime from String", Time)*

*print(type(Time))*

*Time = time(12,24,36,1212)*

*print("Original time:", Time)*

*# Replacing hour*

*Time = Time.replace(hour = 13, second = 12)*

*print("New Time:", Time)*

*# Formatting Time*

*Ftime = Time.strftime("%I:%M %p")*

*print("Formatted time", Ftime)*

1. [datetime](https://www.geeksforgeeks.org/python-datetime-datetime-class/)

Its a combination of date and time along with the attribute’s year, month, day, hour, minute, second, microsecond, and tzinfo.

DateTime class of the DateTime module as the name suggests contains information on both date as well as time

it contains information regarding time zone as well.

Syntax: *class datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None, \*, fold=0)*

It does have all the attributes from date and time class

Functions of Datetime class

| Function Name | Description |
| --- | --- |
| astimezone() | Returns the DateTime object containing timezone information. |
| combine() | Combines the date and time objects and return a DateTime object |
| ctime() | Returns a string representation of date and time |
| date() | Return the Date class object |
| fromisoformat() | Returns a datetime object from the string representation of the date and time |
| fromordinal() | Returns a date object from the proleptic Gregorian ordinal, where January 1 of year 1 has ordinal 1. The hour, minute, second, and microsecond are 0 |
| fromtimestamp() | Return date and time from POSIX timestamp |
| isocalendar() | Returns a tuple year, week, and weekday |
| isoformat() | Return the string representation of date and time |
| isoweekday() | Returns the day of the week as integer where Monday is 1 and Sunday is 7 |
| now() | Returns current local date and time with tz parameter |
| replace() | Changes the specific attributes of the DateTime object |
| strftime() | Returns a string representation of the DateTime object with the given format |
| strptime() | Returns a DateTime object corresponding to the date string |
| time() | Return the Time class object |
| timetuple() | Returns an object of type time.struct\_time |
| timetz() | Return the Time class object |
| today() | Return local DateTime with tzinfo as None |
| toordinal() | Return the proleptic Gregorian ordinal of the date, where January 1 of year 1 has ordinal 1 |
| tzname() | Returns the name of the timezone |
| utcfromtimestamp() | Return UTC from POSIX timestamp |
| utcoffset() | Returns the UTC offset |
| utcnow() | Return current UTC date and time |
| weekday () | Returns the day of the week as integer where Monday is 0 and Sunday is 6 |

#DateTime class

*from datetime import datetime*

*# Getting Today's Datetime*

*today = datetime.now()*

*print("Today's date using now() method:", today)*

*today = datetime.today()*

*print("Today's date using today() method:", today)*

*date\_time = datetime.fromtimestamp(1887639468)*

*print("Datetime from timestamp:", date\_time)*

*# Getting Datetime from ordinal*

*date\_time = datetime.fromordinal(737994)*

*print("Datetime from ordinal:", date\_time)*

1. [timedelta](https://www.geeksforgeeks.org/python-datetime-timedelta-class/)

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

class is used for calculating differences between dates and represents a duration. The difference can both be positive as well as negative.

Syntax:

*class datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)*

*class attributes are*

*min*

*max*

*resolution*

*timedelta program*

*#timedelta class*

*# Timedelta function demonstration*

*from datetime import datetime, timedelta*

*# creating datetime objects*

*date1 = datetime(2020, 1, 3)*

*date2 = datetime(2020, 2, 3)*

*# difference between dates*

*diff = date2 - date1*

*print("Difference in dates:", diff)*

*# Adding days to date1*

*date1 += timedelta(days = 4)*

*print("Date1 after 4 days:", date1)*

*# Subtracting days from date1*

*date1 -= timedelta(15)*

*print("Date1 before 15 days:", date1)*

*Class functions*

Timedelta class provides only one function which is **total\_seconds().**This method returns the duration provided by the timedelta object in the number of seconds.

*from datetime import timedelta*

*# Getting minimum value*

*obj = timedelta(hours=1)*

*print(obj.total\_seconds())*

*obj = timedelta(minutes=1)*

*print(obj.total\_seconds())*

*obj = timedelta(days=1)*

*print(obj.total\_seconds())*

## Operations supported by Timedelta Class

| Operator | Description |
| --- | --- |
| Addition (+) | Adds and returns two timedelta objects |
| Subtraction (-) | Subtracts and returns two timedelta objects |
| Multiplication (\*) | Multiplies timedelta object with float or int |
| Division (/) | Divides the timedelta object with float or int |
| Floor division (//) | Divides the timedelta object with float or int and return the int of floor value of the output |
| Modulo (%) | Divides two timedelta object and returns the remainder |
| +(timedelta) | Returns the same timedelta object |
| -(timedelta) | Returns the resultant of -1\*timedelta |
| abs(timedelta) | Returns the +(timedelta) if timedelta.days > 1=0 else returns -(timedelta) |
| str(timedelta) | Returns a string in the form (+/-) day[s],  HH:MM:SS.UUUUUU |
| repr(timedelta) | Returns the string representation in the form of the constructor call |

*#operators supported by timedelta class*

*from datetime import timedelta*

*# creating the timedelta object*

*t1 = timedelta(days=1)*

*print("Original timedelta:", t1)*

*# multiplication*

*t2 = t1\*5.5*

*print("After Multiplication:", t2)*

*# Subtraction*

*res = t2 - t1*

*print("After Subtraction:", res)*

*# addition*

*res += t2*

*print("After Addition:", res)*

*# division*

*res = t2/2.5*

*print("After division:", res)*

*# floor division*

*res = t2 //2*

*print("After floor division:", res)*

*# Modulo*

*res = t2%timedelta(days=3)*

*print("After Modulo:", res)*

5.tzinfo

It provides time zone information objects.

6.timezone

A class that implements the tzinfo abstract base class as a fixed offset from the UTC (New in version 3.2).

Let's create a few simple programs related to date and time

1. Get current date and time

*import datetime*

*res=datetime.datetime.now()*

*print(res)*

Here, we have imported **datetime** module using import datetime statement.

One of the classes defined in the datetime module is datetime class. We then used now() method to create a datetime object containing the current local date and time.

### Get Current Date

*import datetime*

*date\_object = datetime.date.today()*

*print(date\_object)*

today() method defined in the date class to get a date object containing the current local date.

1. *We can use dir() to get the list of all the methods inside datetime*

*import datetime*

*print(dir(datetime))*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Date | Time | Datetime | Timedelta | Tzinfo | Timezone |
| Attributes | Min | Min | Min | Min |  |  |
|  | Max | Max | Max | Max |  |  |
|  | Resolution | Resolution | Resolution | resolution |  |  |
|  | Year | Hour | Year |  |  |  |
|  | month | Min | Month |  |  |  |
|  | day | Seconds | Day |  |  |  |
|  |  | Tzinfo | Hour |  |  |  |
|  |  | fold | Min |  |  |  |
|  |  |  | Seconds |  |  |  |
|  |  |  | Tzinfo |  |  |  |
|  |  |  | fold |  |  |  |
| Operations supported |  |  |  | +,-,\*,/,%,//,  +(timedelta), -(timedelta), abs(timedelta), str(timedelta), repr(timedelta) |  |  |
| methods | fromordinal() |  | fromordinal() | total\_seconds() |  |  |
|  | toordinal() |  | toordinal() |  |  |  |
|  | fromisocalendar() |  |  |  |  |  |
|  | isocalendar() |  | isocalendar() |  |  |  |
|  | fromisoformat() | fromisoformat() | fromisoformat() |  |  |  |
|  | isoformat() | isoformat() | isoformat() |  |  |  |
|  | isoweekday() |  | isoweekday() |  |  |  |
|  | weekday() |  | weekday() |  |  |  |
|  | fromtimestamp() |  | fromtimestamp() |  |  |  |
|  | replace() | replace() | replace() |  |  |  |
|  | today() |  | today() |  |  |  |
|  | timetuple() |  | timetuple() |  |  |  |
|  | strftime() | strftime() | strftime() |  |  |  |
|  | ctime() |  | ctime() |  |  |  |
|  |  | dst() |  |  |  |  |
|  |  | tzname() | tzname() |  |  |  |
|  |  | utcoffset() | utcoffset() |  |  |  |
|  |  |  | Utcnow() |  |  |  |
|  |  |  | utcfromtimestamp() |  |  |  |
|  |  |  | strptime() |  |  |  |
|  |  |  | Now() |  |  |  |
|  |  |  | Timetz() |  |  |  |
|  |  |  | Time() |  |  |  |
|  |  |  | Date() |  |  |  |
|  |  |  | Combin() |  |  |  |
|  |  |  | astimezone() |  |  |  |