How to print the current date.

```
# import datetime class from datetime module
from datetime import datetime

# get current date
datetime_object = datetime.now()
print(datetime_object)
print('Type :- ',type(datetime_object))
```

```
2020-09-10 10:59:57.397334
Type :-  <class 'datetime.datetime'>
```

'datetime' includes two methods, strptime() and strftime(), for converting objects from strings to datetime objects and vice versa. strptime() can read strings with date and time information and convert them to datetime objects, and strftime() converts datetime objects back into strings.

```
my_string = '2019-10-31'

# Create date object in given time format yyyy-mm-dd
my_date = datetime.strptime(my_string, "%Y-%m-%d")

print(my_date)
print('Type: ',type(my_date))
```

```
2019-10-31 00:00:00
Type:  <class 'datetime.datetime'>
```

Separate month and year

```
my_date = datetime.strptime(my_string, "%Y-%m-%d")
print('Month: ', my_date.month) # To Get month from date
print('Year: ', my_date.year) # To Get month from year
```

```
Month:  10
Year:  2019
```

Here is how to get the day of the month and the day of the week from my_date. Datetime will give us the day of the week as a number using its .weekday() function, but we can convert this to a text format (i.e. Monday, Tuesday, Wednesday…) using the calendar module and a method called day_name.

```
# import calendar module
import calendar
print('Day of Month:', my_date.day)

# to get name of day(in number) from date
print('Day of Week (number): ', my_date.weekday())

# to get name of day from date
print('Day of Week (name): ', calendar.day_name[my_date.weekday()])
```

```
Day of Month: 31
Day of Week (number):  3
Day of Week (name):  Thursday
```

Keep in mind that Monday is '0'.

```
j = 0
for i in calendar.day_name:
    print(j,'-',i)
    j+=1
```

```
0 - Monday
1 - Tuesday
2 - Wednesday
3 - Thursday
4 - Friday
5 - Saturday
6 - Sunday
```

Extract the hours and minutes from datetime object

```
from datetime import datetime
todays_date = datetime.now()

# to get hour from datetime
print('Hour: ', todays_date.hour)

# to get minute from datetime
print('Minute: ', todays_date.minute)
```

```
Hour:  11
Minute:  7
```

We can get the year, week of the year, and day of the week from a datetime object with the .isocalendar() function.

```
# Return a 3-tuple, (ISO year, ISO week number, ISO weekday).
todays_date.isocalendar()
```

```
(2020, 37, 4)
```

In programming, it is not uncommon to encounter time and date data that is stored as a timestamp, or to want to store your own data in Unix timestamp format.

We can do that using datetime's built-in timestamp() function, which takes a datetime object as an argument and returns that date and time in timestamp format

```
#import datetime
from datetime import datetime
# get current date
now = datetime.now()

# convert current date into timestamp
timestamp = datetime.timestamp(now)

print("Date and Time :", now)
print("Timestamp:", timestamp)
```

```
Date and Time : 2020-09-10 11:23:56.845432
Timestamp: 1599737036.845432
```

We can do the reverse conversion using fromtimestamp(). This is a datetime function that takes a timestamp (in float format) as an argument and returns a datetime object, as below

```
#import datetime
from datetime import datetime
timestamp = 1599737036.845432

#convert timestamp to datetime object
dt_object = datetime.fromtimestamp(timestamp)

print("dt_object:", dt_object)
print("type(dt_object): ", type(dt_object))
```

```
dt_object: 2020-09-10 11:23:56.845432
type(dt_object):  <class 'datetime.datetime'>
```

We may want to measure a span of time, or a duration, using Python datetime. We can do this with its built-in timedelta class. A timedelta object represents the amount of time between two dates or times. We can use this to measure time spans, or manipulate dates or times by adding and subtracting from them, etc.

```
#import datetime
from datetime import timedelta
# create timedelta object with difference of 2 weeks
d = timedelta(weeks=2)

print(d)
print(type(d))
print(d.days)
```

We can get our time duration in days by using the timedelta class attribute .days.

```
year = timedelta(days=365)
print(year)
```

```
365 days, 0:00:00
```

```
#import datetime
from datetime import datetime, timedelta
# get current time
now = datetime.now()
print ("Today's date: ", str(now))

#add 15 days to current date
future_date_after_15days = now + timedelta(days = 15)
print('Date after 15 days: ', future_date_after_15days)

#subtract 2 weeks from current date
two_weeks_ago = now - timedelta(weeks = 2)
print('Date two weeks ago: ', two_weeks_ago)
print('two_weeks_ago object type: ', type(two_weeks_ago))
```

```
Today's date:  2020-09-10 11:29:13.536552
Date after 15 days:  2020-09-25 11:29:13.536552
Date two weeks ago:  2020-08-27 11:29:13.536552
two_weeks_ago object type:  <class 'datetime.datetime'>
```

We can also subtract one date from another date to find the timespan between them using datetime.

Because the result of this math is a duration, the object produced when we subtract one date from another will be a timedelta object.

```
# import datetime
from datetime import date
# Create two dates
date1 = date(2012, 8, 18)
date2 = date(2019, 9, 23)

# Difference between two dates
delta = date2 - date1
print("Difference: ", delta.days)
print('delta object type: ', type(delta))
```

```
Difference:  2592
delta object type:  <class 'datetime.timedelta'>
```

Here is another way to compute duration.

```
# import datetime
from datetime import datetime
# create two dates with year, month, day, hour, minute, and second
date1 = datetime(2017, 6, 21, 18, 25, 30)
date2 = datetime(2017, 5, 16, 8, 21, 10)

# Difference between two dates
diff = date1-date2
print("Difference: ", diff)
```

```
Difference:  36 days, 10:04:20
```

Formatting time.

```
# import datetime
from datetime import datetime
date_string = "1 August, 2019"

# format date
date_object = datetime.strptime(date_string, "%d %B, %Y")

print("date_object: ", date_object)
```

```
date_object:  2019-08-01 00:00:00
```

Manipulating date and time and formatting them.

```
# import datetime
from datetime import datetime
dt_string = "12/11/2018 09:15:32"
# Considering date is in dd/mm/yyyy format
dt_object1 = datetime.strptime(dt_string, "%d/%m/%Y %H:%M:%S")
print("dt_object1:", dt_object1)
# Considering date is in mm/dd/yyyy format
dt_object2 = datetime.strptime(dt_string, "%m/%d/%Y %H:%M:%S")
print("dt_object2:", dt_object2)

# Convert dt_object2 to Unix Timestamp
timestamp = datetime.timestamp(dt_object2)
print('Unix Timestamp: ', timestamp)

# Convert back into datetime
date_time = datetime.fromtimestamp(timestamp)
d = date_time.strftime("%c")
print("Output 1:", d)
d = date_time.strftime("%x")
print("Output 2:", d)
d = date_time.strftime("%X")
print("Output 3:", d)
```

```
dt_object1: 2018-11-12 09:15:32
dt_object2: 2018-12-11 09:15:32
Unix Timestamp:  1544519732.0
Output 1: Tue Dec 11 09:15:32 2018
Output 2: 12/11/18
Output 3: 09:15:32
```

Format dates and time

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

# get year from date
year = now.strftime("%Y")
print("Year:", year)

# get month from date
month = now.strftime("%m")
print("Month;", month)

# get day from date
day = now.strftime("%d")
print("Day:", day)

# format time in HH:MM:SS
time = now.strftime("%H:%M:%S")
print("Time:", time)

# format date
date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
print("Date and Time:",date_time)
```

```
Year: 2020
Month; 09
Day: 10
Time: 11:35:20
Date and Time: 09/10/2020, 11:35:20
```

Handling Time Zones

```
# import timezone from pytz module
from pytz import timezone
# Create timezone US/Eastern
east = timezone('US/Eastern')
# Localize date
loc_dt = east.localize(datetime(2011, 11, 2, 7, 27, 0))
print(loc_dt)

# Convert localized date into Asia/Kolkata timezone
kolkata = timezone("Asia/Kolkata")
print(loc_dt.astimezone(kolkata))

# Convert localized date into Australia/Svdnev timezone
```

```
14  au_tz = timezone('Australia/Sydney')
15  print(loc_dt.astimezone(au_tz))
```

```
2011-11-02 07:27:00-04:00
2011-11-02 16:57:00+05:30
2011-11-02 22:27:00+11:00
```

Working with pandas. pandas has both datetime and timedelta objects for specifying dates and times and durations, respectively.

We can convert date, time, and duration text strings into pandas Datetime objects using these functions:

- to_datetime(): Converts string dates and times into Python datetime objects.
- to_timedelta(): Finds differences in times in terms of days, hours, minutes, and seconds.

```
1  # import pandas module as pd
2  import pandas as pd
3  # create date object using to_datetime() function
4  date = pd.to_datetime("8th of sep, 2019")
5  print(date)
```

```
2019-09-08 00:00:00
```

We create a series of twelve dates starting from the day we defined above. Then we create a different series of dates starting from a predefined date using pd.date_range():

```
1  import numpy as np
2  # Create date series using numpy and to_timedelta() function
3  date_series = date + pd.to_timedelta(np.arange(12), 'D')
4  print(date_series)
5
6  # Create date series using date_range() function
7  date_series = pd.date_range('08/10/2019', periods = 12, freq ='D')
8  print(date_series)
```

```
DatetimeIndex(['2019-09-08', '2019-09-09', '2019-09-10', '2019-09-11',
               '2019-09-12', '2019-09-13', '2019-09-14', '2019-09-15',
               '2019-09-16', '2019-09-17', '2019-09-18', '2019-09-19'],
              dtype='datetime64[ns]', freq=None)
DatetimeIndex(['2019-08-10', '2019-08-11', '2019-08-12', '2019-08-13',
               '2019-08-14', '2019-08-15', '2019-08-16', '2019-08-17',
               '2019-08-18', '2019-08-19', '2019-08-20', '2019-08-21'],
              dtype='datetime64[ns]', freq='D')
```

Make a quick DataFrame using one of the Series we created above.

```
1  import pandas as pd
2  # Create a DataFrame with one column date
3  df = pd.DataFrame()
4  df['date'] = date_series
5  df.head()
```

	date
0	2019-08-10
1	2019-08-11
2	2019-08-12
3	2019-08-13
4	2019-08-14

Create separate columns for each element of the date by using the relevant Python datetime (accessed with dt) attributes:

```
1  # Extract year, month, day, hour, and minute. Assign all these date component to new column.
2  df['year'] = df['date'].dt.year
3  df['month'] = df['date'].dt.month
4  df['day'] = df['date'].dt.day
5  df['hour'] = df['date'].dt.hour
6  df['minute'] = df['date'].dt.minute
7  df.head()
```

	date	year	month	day	hour	minute
0	2019-08-10	2019	8	10	0	0
1	2019-08-11	2019	8	11	0	0
2	2019-08-12	2019	8	12	0	0
3	2019-08-13	2019	8	13	0	0

pandas is also capable of getting other elements, like the day of the week and the day of the year, from its datetime objects.

```
# get Weekday and Day of Year. Assign all these date component to new column.
df['weekday'] = df['date'].dt.weekday
df['dayofyear'] = df['date'].dt.dayofyear
df.head()
```

	date	year	month	day	hour	minute	weekday	dayofyear
0	2019-08-10	2019	8	10	0	0	5	222
1	2019-08-11	2019	8	11	0	0	6	223
2	2019-08-12	2019	8	12	0	0	0	224
3	2019-08-13	2019	8	13	0	0	1	225
4	2019-08-14	2019	8	14	0	0	2	226

Use pandas to make a datetime column into the index of our DataFrame.

```
# Assign date column to dataframe index
df.index = df.date
df.head()
```

	date	year	month	day	hour	minute	weekday	dayofyear
date								
2019-08-10	2019-08-10	2019	8	10	0	0	5	222
2019-08-11	2019-08-11	2019	8	11	0	0	6	223
2019-08-12	2019-08-12	2019	8	12	0	0	0	224
2019-08-13	2019-08-13	2019	8	13	0	0	1	225
2019-08-14	2019-08-14	2019	8	14	0	0	2	226