

Python DateTime, TimeDelta, Strftime(Format) with Examples

In Python, **date**, **time** and **datetime** classes provides a number of function to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps. Whenever you manipulate dates or time, you need to import datetime function.

The datetime classes in Python are categorized into main 5 classes.

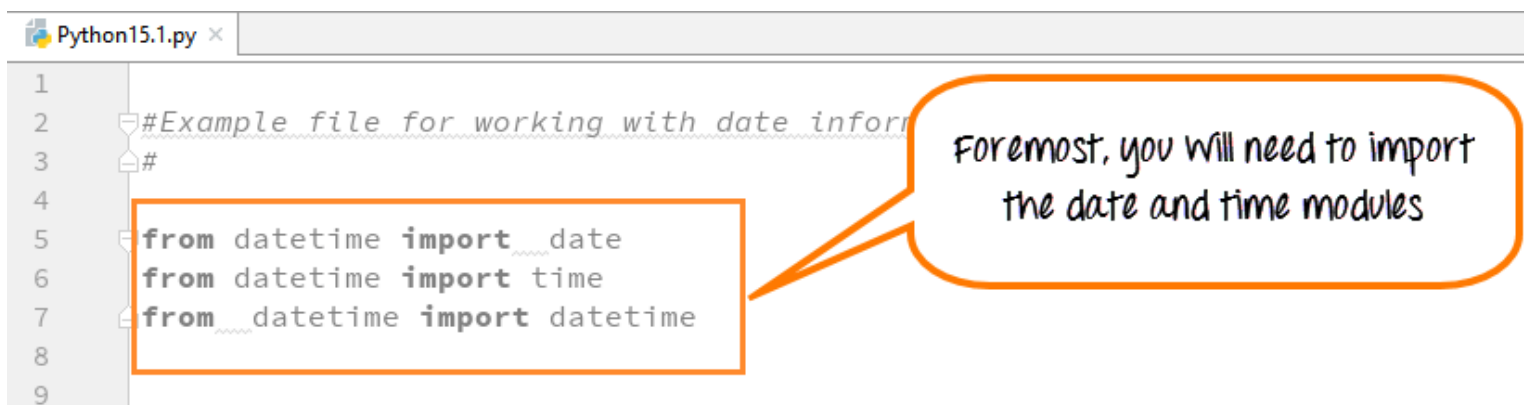
- **date** – Manipulate just date (Month, day, year)
- **time** – Time independent of the day (Hour, minute, second, microsecond)
- **datetime** – Combination of time and date (Month, day, year, hour, second, microsecond)
- **timedelta**— A duration of time used for manipulating dates
- **tzinfo**— An abstract class for dealing with time zones

In this tutorial, we will learn-

- [How to Use Date & DateTime Class](#)
- [Print Date using date.today\(\)](#)
- [Python Current Date and Time: now\(\) today\(\)](#)
- [How to Format Date and Time Output with Strftime\(\)](#)
- [How to use Timedelta Objects](#)

How to Use Date & DateTime Class

Step 1) Before you run the code for datetime, it is important that you import the date time modules as shown in the screenshot below.



The screenshot shows a code editor window titled 'Python15.1.py'. The code contains the following lines:

```
1
2 #Example file for working with date inform
3 #
4
5 from datetime import date
6 from datetime import time
7 from datetime import datetime
8
9
```

An orange box highlights the import statements on lines 5, 6, and 7. A speech bubble points to this box with the text: "Foremost, you will need to import the date and time modules".

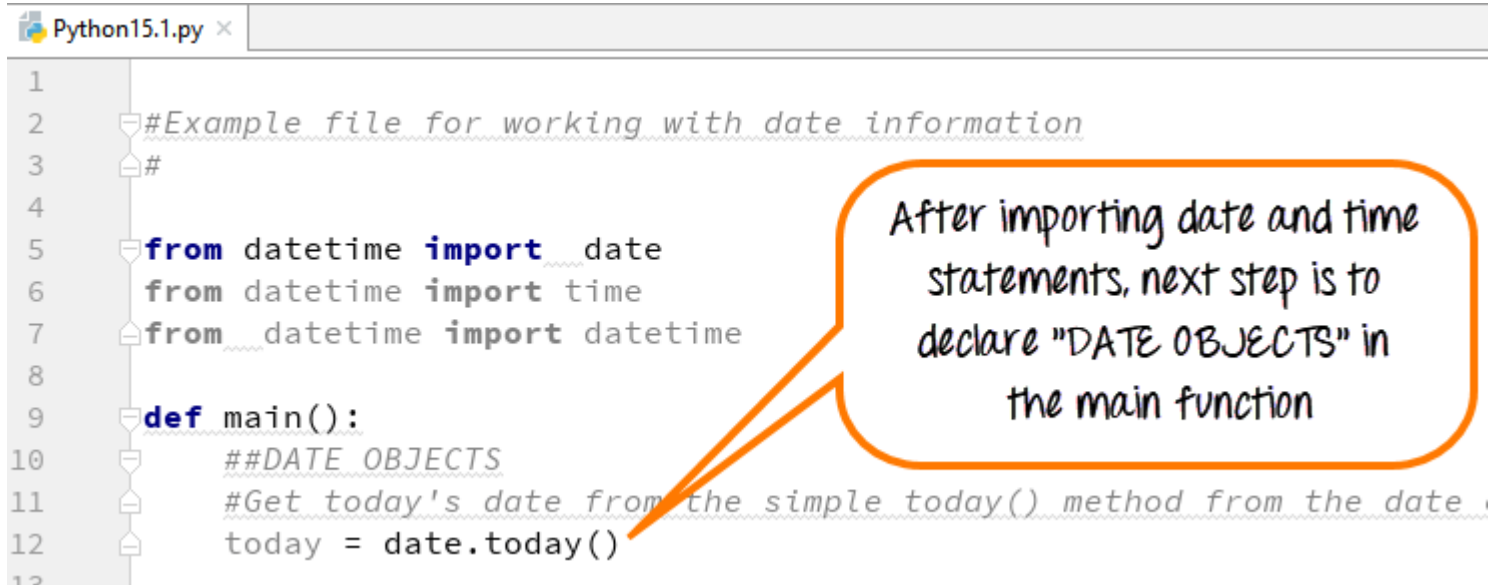
These import statements are pre-defined pieces of functionality in the Python library that let you manipulate dates and times, without writing any code.

Consider the following points before executing the datetime code

```
from datetime import date
```

This line tells the Python interpreter that from the datetime module import the date class We are not writing the code for this date functionality alas just importing it for our use

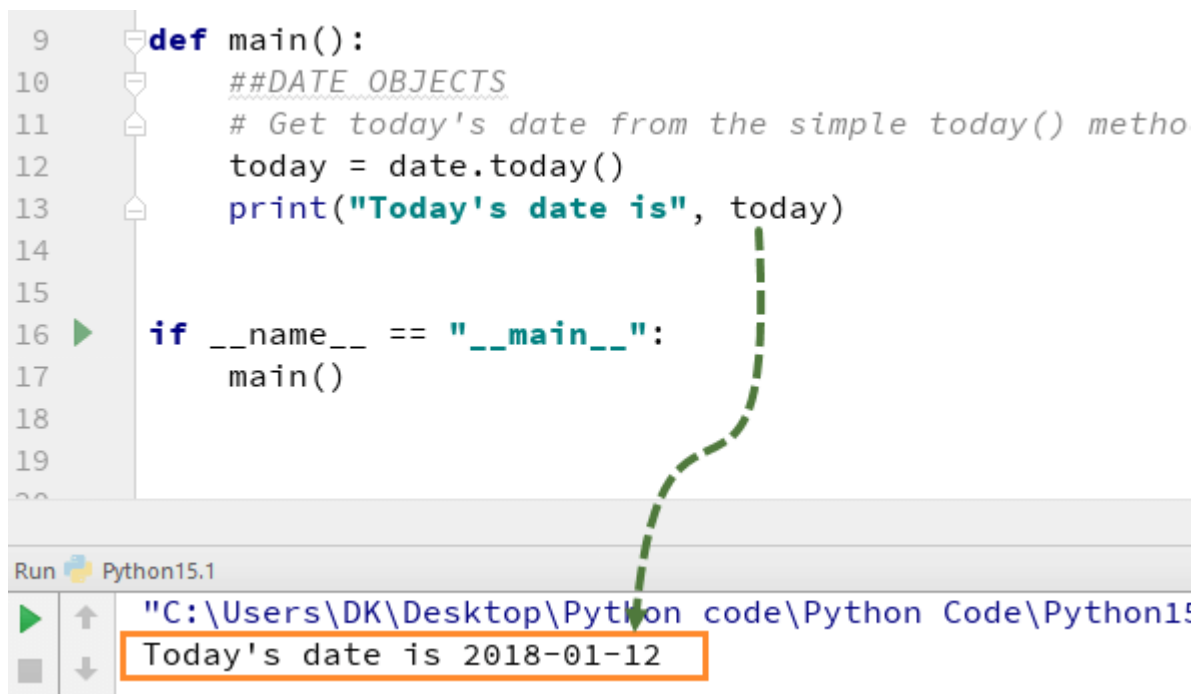
Step 2) Next, we create an instance of the date object.



```
1
2 #Example file for working with date information
3 #
4
5 from datetime import date
6 from datetime import time
7 from datetime import datetime
8
9 def main():
10     ##DATE OBJECTS
11     #Get today's date from the simple today() method from the date
12     today = date.today()
13
```

After importing date and time statements, next step is to declare "DATE OBJECTS" in the main function

Step 3) Next, we print the date and run the code.



```
9 def main():
10     ##DATE OBJECTS
11     # Get today's date from the simple today() metho
12     today = date.today()
13     print("Today's date is", today)
14
15
16 if __name__ == "__main__":
17     main()
18
19
20
```

Run Python15.1

"C:\Users\DK\Desktop\Python code\Python Code\Python15.1"

Today's date is 2018-01-12

The output is as expected.

Print Date using date.today()

`date.today` function has several properties associated with it. We can print individual day/month/year and many other things

Let's see an example

```
9 def main():
10     ##DATE OBJECTS
11     # Get today's date from the simple today() method from the date class
12     today = date.today()
13     print("Today's date is", today)
14
15
16     # print out the date's individual component
17     today = date.today()
18     print("Date Components:", today.day, today.month, today.year)
19
20     if __name__ == "__main__":
21         main()
22
```

Run Python15.1

"C:\Users\DK\Desktop\Python code\Python
Date Components: 12 1 2018
Today's date is 2018-01-12

Individual date components can also be executed in Python. Here in our code we have mentioned date, month and year separately. The output is printed out as expected

Today's Weekday Number

The `date.today()` function also gives you the weekday number. Here is the Weekday Table which start with Monday as 0 and Sunday as 6

Day	WeekDay Number
Monday	0
Tuesday	1
Wednesday	2
Thursday	3
Friday	4
Saturday	5
Sunday	6

Weekday Number is useful for arrays whose index is dependent on the Day of the week.

```

Python15.1.py
17
18 #retrive today's weekday (0=Monday, 6=Sunday)
19 print("Today's Weekday#:", today.weekday())
20
21
22 if __name__ == "__main__":
23     main()
24
25
26

```

Run Python15.1

"C:\Users\DK\Desktop\Python code\Python Code\Python15\venv\Scripts\python.exe" "C:\Users\DK\Desktop\Python code\Python Code\Python15.py"

Date Components: 12 1 2018

Today's Weekday#: 4

Today's date is 2018-01-12

today.weekday() will return the weekday number which in our case is 4

Python Current Date and Time: now() today()

Step 1) Like Date Objects, we can also use "DATETIME OBJECTS" in Python. It gives date along with time in **hours, minutes, seconds and milliseconds**.

```

4 from datetime import date
5 from datetime import time
6 from datetime import datetime
7
8 def main():
9     ##DATE OBJECTS
10    #Get today's date from the datetime class
11    today = datetime.now()
12    print("The current date and time is", today)
13
14
15 if __name__ == "__main__":
16     main()
17

```

Run Python15.1

"C:\Users\DK\Desktop\Python code\Python Code\Python15\venv\Scripts\python.exe" "C:\Users\DK\Desktop\Python code\Python Code\Python15.py"

The current date and time is 2018-01-12 11:59:35.976715

Like DATE OBJECTS We have "DATE TIME OBJECTS" in python, it allows you to use time function with date. In the output we printed out date along with time

When we execute the code for datetime, it gives the output with current date and time.

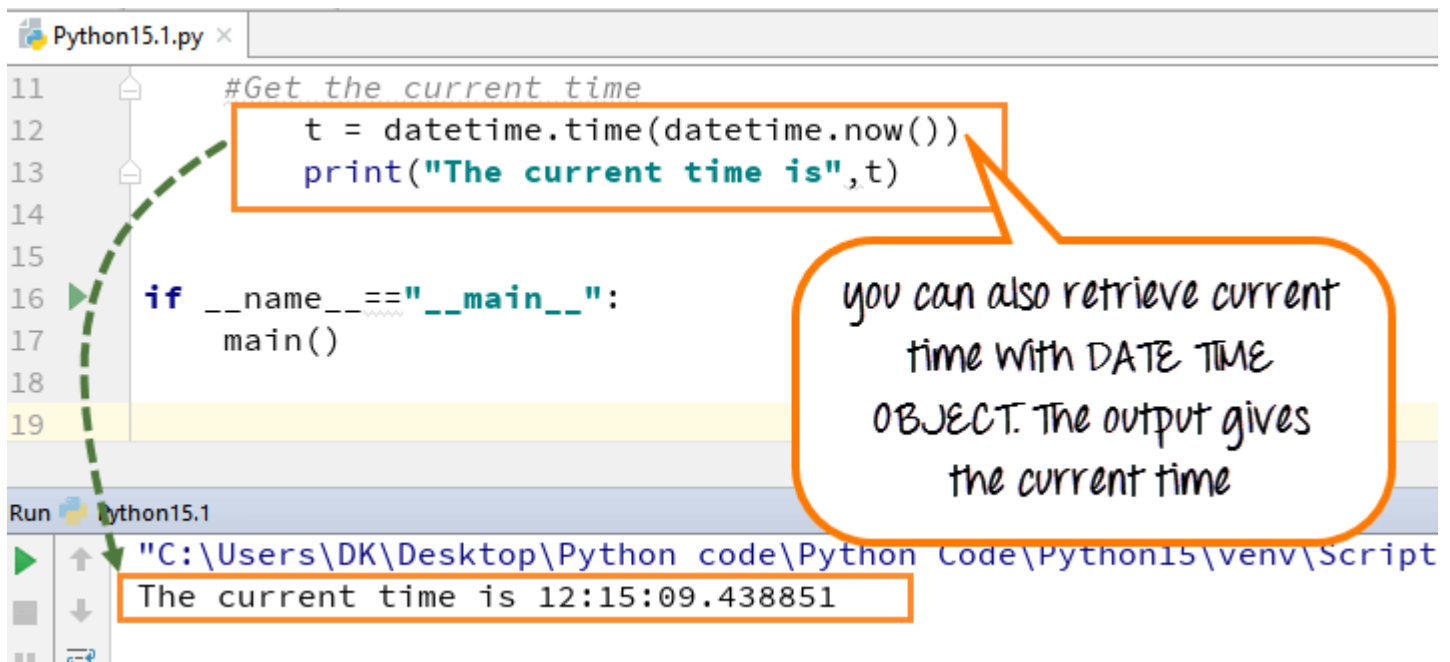
Step 2) With "DATETIME OBJECT", you can also call time class.

Suppose we want to print just the current time without the date.

```
t = datetime.time(datetime.now())
```

- We had imported the time class. We will be assigning it the current value of time using datetime.now()
- We are assigning the value of the current time to the variable t.

And this will give me just the time. So let's run this program.



The screenshot shows a Python IDE window titled "Python15.1.py". The code is as follows:

```
11 #Get the current time
12 t = datetime.time(datetime.now())
13 print("The current time is",t)
14
15
16 if __name__=="__main__":
17     main()
18
19
```

Line 12 is highlighted with a green dashed arrow pointing to a callout box that contains the code: `t = datetime.time(datetime.now())` and `print("The current time is",t)`. Another callout box points to line 12 with the text: "you can also retrieve current time with DATE TIME OBJECT. The output gives the current time".

Below the code editor, the output console shows the command prompt "Run Python15.1" and the output: `"C:\Users\DK\Desktop\Python code\Python Code\Python15\venv\Script` and `The current time is 12:15:09.438851`. The output line is highlighted with an orange box.

Okay, so you can see that here I got the date and time. And then the next line, I've got just the time by itself

Step 3) We will apply our weekday indexer to our weekday's arrayList to know which day is today

- Weekdays operator (wd) is assigned the number from (0-6) number depending on what the current weekday is. Here we declared the array of the list for days (Mon, Tue, Wed...Sun).
- Use that index value to know which day it is. In our case, it is #2, and it represents Wednesday, so in the output it will print out "Which is a Wednesday."

```

12  #weekday returns 0 (monday) through 6 (sunday)
13  wd = date.weekday(today)
14  #Days start at 0 for monday
15  days= ["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
16  print ("Today is day number %d" % wd)
17  print ("which is a " + days[wd])
18
19  if __name__ == "__main__":
20      main()
21
22

```

Run Python15.1

"C:\Users\DK\Desktop\Python code" \Scripts\python.exe "C:/P

Today is day number 4
which is a friday

We use indexer to know which day is today and based on that we retrieve today's day

Here is the complete code to get current date and time using datetime now

Here is the complete code to get current date and time using datetime now

```

from datetime import date
from datetime import time
from datetime import datetime
def main():
    ##DATETIME OBJECTS
    #Get today's date from datetime class
    today=datetime.now()
    #print (today)
    # Get the current time
    #t = datetime.time(datetime.now())
    #print "The current time is", t
    #weekday returns 0 (monday) through 6 (sunday)
    wd=date.weekday(today)
    #Days start at 0 for monday
    days= ["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
    print("Today is day number %d" % wd)
    print("which is a " + days[wd])

if __name__ == "__main__":
    main()

```

How to Format Date and Time Output with Strftime()

As of now we have learned, how to use datetime and date object in Python. We will advance a step further and learn how to use a formatting function to format Time and Date.

Step 1) First we will see a simple step of how to format the year. It is better to understand with an example.

```

1  #
2  #Example file for formatting time and date output
3  #
4  from datetime import datetime
5  def main():
6      #Times and dates can be formatted using a set of predefined string
7      #Control codes
8      now= datetime.now() #get the current date and time
9
10     ##### Date Formatting #####
11     #%%y%%Y - Year, %a/%A- weekday, %b/%B- month, %d - day of month
12     print (now.strftime("%Y")) #full year with century
13
14     if __name__ == "__main__":
15         main()
16
17
18
19
20

```

The "Str-f-time" function is used to format various parameter of time, date, month and year. Here we have displayed year with century i.e "2018" and going to change this to just "18" in next step

2018

- We used the "**strftime function**" for formatting.
- This function uses different **control code** to give an output.
- Each control code resembles different parameters like year, month, weekday and date [(**%y/%Y - Year**), (**%a/%A- weekday**), (**%b/%B- month**), (**%d - day of month**)] .
- In our case, it is ("**%Y**") which resembles year, it prints out the full year with the century (e.g., 2018).

Step 2) Now if you replace ("%Y") with lowercase, i.e., ("%y") and execute the code the output will display only (18) and not (2018). The century of the year will not display as shown in the screenshot below


```

10 ##### Date Formatting #####
11
12 # %y%Y - Year, %a/%A- weekday, %b/%B- month, %d - day of month
13 print(now.strftime("%y")) #full year with century
14
15 if __name__ == "__main__":
16     main()
17
18
19
20

```

When we replaced "%Y" with lowercase "%y" the output changes from "2018" to just a year "18"

Run Python 15.1

18

Step 3) Strf function can declare the date, day, month and year separately. Also with small changes in the control code in strftime function you can format the style of the text.

```

1 #
2 #Example file for formatting time and date output
3 #
4 from datetime import datetime
5
6 def main():
7     #Times and dates can be formatted using a set of predefined string
8     #Control codes
9     now= datetime.now() #get the current date and time
10    print(now.strftime("%a,%d %B,%y"))
11
12
13 if __name__ == "__main__":
14     main()
15
16
17

```

Day

Month

Year

Date

With str-f-time function you can declare date, day, month, and year separately.

Run Python 15.1

Fri,12 January,18

Inside the strftime function if you replace (%a) with capital A, i.e., (%A) the output will print out as "Firday" instead of just an abbreviation "Fri".


```

7      #Control codes
8      now= datetime.now() #get the current date and time
9      print (now.strftime("%A %d %B,%y"))
10
11
12
13
14  if __name__ == "__main__":
15      main()
16
17
18

```

Run Python15.1

C:\Users\DK\Desktop\Python code\Python Code\Python15\venv' Friday 12 January,18

small (% a) is replaced by (% A). It will print out "Friday" instead of #abbr-"Fri"

Step 4) With the help of "Strftime" function we can also retrieve local system time, date or both.

1. %C- indicates the local date and time
2. %x- indicates the local date
3. %X- indicates the local time

```

8      #Control codes
9      now= datetime.now() #get the current date and time
10     # %c - local date and time, %x-local's date, %X- local's time
11     print(now.strftime("%c"))
12     print(now.strftime("%x"))
13     print(now.strftime("%X"))
14
15
16
17
18  if __name__ == "__main__":
19      main()
20
21
22

```

Run Python15.1

C:\Users\DK\Desktop\Python code\Python Code\Python15\venv' Fri Jan 12 13:08:33 2018
01/12/18
13:08:33

We can use "Str-f-time" function to retrieve local date and time, local date and local time of native system

In the output, you can see the result as expected

Step 5) The "strftime function" allows you to call the time in any format 24 hours or 12 hours.

```

8      #Control codes
9      now= datetime.now() #get the current
10     #%c - local date and time, %x-local
11     # print now.strftime("%c")
12     # print now.strftime("%x")
13     # print now.strftime("%X")
14
15
16     ##### Time Formatting #####
17
18     #%I/%H - 12/24 Hour, %M - minute, %S - second, %p - local's AM/PM
19     print (now.strftime("%I:%M:%S %p")) ① 12-Hour:Minute:Second:AM
20     print (now.strftime("%H:%M")) ② 24-Hour:Minute
21
22     if __name__ == "__main__":
23         main()
24

```

With "strftime" function you can do lot many things like printing out time in any formats (24 hrs or 12 hours)

① 12-Hour:Minute:Second:AM
② 24-Hour:Minute

Run Python15.1

"C:\Users\DK\Desktop\Python code\Python Code\Python15\venv\Scripts\python
01:19:18 PM ①
13:19 ②

Just by defining control code like %I/H for hour, % M for minute, %S for second, one can call time for different formats

12 hours time is declared [print now.strftime("%I:%M:%S %P)]

24 hours time is declared [print now.strftime("%H:%M")]

Here is the complete code to convert datetime to String object.

```

#
#Example file for formatting time and date output
#
from datetime import datetime
def main():
    #Times and dates can be formatted using a set of predefined string
    #Control codes
    now= datetime.now() #get the current date and time
    #%c - local date and time, %x-local's date, %X- local's time
    print(now.strftime("%c"))

```

```

print(now.strftime("%x"))
print(now.strftime("%X"))
##### Time Formatting #####
#%I/%H - 12/24 Hour, %M - minute, %S - second, %p - local's AM/PM
print(now.strftime("%I:%M:%S %p")) # 12-Hour:Minute:Second:AM
print(now.strftime("%H:%M")) # 24-Hour:Minute

if __name__ == "__main__":
    main()

```

How to use Timedelta Objects

With timedelta objects, you can estimate the time for both future and the past. In other words, it is a timespan to predict any special day, date or time.

Remember this function is not for printing out the time or date, but something to CALCULATE about the future or past. Let's see an example to understand it better.

Step 1) To run Timedelta Objects, you need to declare the import statement first and then execute the code

```

1 #
2 # Example file for working with timedelta objects
3 #
4 from datetime import date
5 from datetime import time
6 from datetime import datetime
7 from datetime import timedelta
8
9 # construct a basic timedelta and print it
10 print(timedelta(days=365, hours=8, minutes=15))
11
12
13
14

```

1) Write import statement including time delta
2) Print command for timedelta

3) Run the code. The timedelta represents a span of 365 days, 8 hrs and 15 minutes and prints the same

1. Write import statement for timedelta
2. Now write the code to print out object from time delta as shown in screen shot
3. Run the code. The timedelta represents a span of 365 days, 8 hrs and 15 minutes and prints the same

Confusing? Next step will help-

Step 2) Let's get today's date and time to check whether our import statement is working well. When code is executed, it prints out today's date which means our import statement is working well

```

9      # construct a basic timedelta and print it
10     print(timedelta(days=365, hours=8, minutes=15))
11     # print today's date
12     print("today is: " + str(datetime.now()))
13
14

```

Run Python15.1

"C:\Users\DK\Desktop\Python code\Python Code\Pytl
 365 days, 8:15:00
 today is: 2018-01-12 14:40:05.569077

Step 3) We will see how we can retrieve date a year from now through delta objects. When we run the code, it gives the output as expected.

```

8
9      # construct a basic timedelta and print it
10     print(timedelta(days=365, hours=8, minutes=15))
11
12     # print today's date
13     print("today is: " + str(datetime.now()))
14     # print today's date one year from now
15
16     print("one year from now it will be:" + str(datetime.now() + timedelta(days=365)))
17

```

Run Python15.1

"C:\Users\DK\Desktop\Python code\Python Code\Python15\venv\Scripts\python.exe" "C:/Pyt
 365 days, 8:15:00
 today is: 2018-01-12 14:43:18.407611
 one year from now it will be:2019-01-12 14:43:18.407611

Time delta adds 365 days to the current time

Step 4) Another example of how time delta can be used to calculate future date from current date and time

```

14
15 # print today's date one year from now
16 print("one year from now it will be:" + str(datetime.now() + timedelta(days=365)))
17
18 # create a timedelta that uses more than one argument
19 print("in one week and 4 days it will be " + str(datetime.now() + timedelta(weeks=1, days=4)))
20
21

```

Run Python15.1

```

"C:\Users\DK\Desktop\Python code\Python Code\Python15\venv\Scripts\python.exe"
365 days, 8:15:00
today is: 2018-01-12 14:48:25.567629
one year from now it will be:2019-01-12 14:48:25.567629
in one week and 4 days it will be 2018-01-23 14:48:25.567629

```

This code for time delta will print the date from one week and 4 days from the current

Step 5) Let's look into a more complex example. I would like to determine how many days past the New Year. Here is how we will proceed

- Using today= date.today() we will get today's date
- We know the new year is always on 1-Jan, but the year could be different. Using nyd= date(today.year,1,1) we store the new year in variable nyd
- if nyd < today: compares whether the current date is greater than the new year. If yes, it enters the while loop
- ((today-nyd).days) gives the difference between a current date and new year in DAYS

```

15
16 today = date.today() # get today's date
17 nyd = date(today.year, 1, 1) # get New Year Day for the same year
18 # use date comparison to see if New Year Day has already gone for this year
19 # if it has, use the replace() function to get the date for next year
20 if nyd < today:
21     print("New Year day is already went by %d days ago" % ((today - nyd).days))
22

```

Run Python15.1

```

"C:\Users\DK\Desktop\Python code\Python Code\Python15\venv\Scripts\python.exe"
365 days, 8:15:00
today is: 2018-01-12 14:56:35.604281
New Year day is already went by 11 days ago

```

With timedelta function you can calculate how many days is left or passed by for any special occasion. Here we have calculated the days passed by for the current New Year day

The output shows that "New Year Day already went by 11 days ago."

Here is the complete working code

```

#
# Example file for working with timedelta objects
#

```

```

from datetime import date
from datetime import time
from datetime import datetime
from datetime import timedelta

# construct a basic timedelta and print it
print (timedelta(days=365, hours=8, minutes=15))
# print today's date
print ("today is: " + str(datetime.now()))
# print today's date one year from now
print ("one year from now it will be:" + str(datetime.now() + timedelta(days=365)))
# create a timedelta that uses more than one argument
# print (in one week and 4 days it will be " + str(datetime.now() + timedelta(weeks=1, days=4)))
# How many days until New Year's Day?
today = date.today() # get today's date
nyd = date(today.year, 1, 1) # get New Year Day for the same year
# use date comparison to see if New Year Day has already gone for this year
# if it has, use the replace() function to get the date for next year
if nyd < today:
    print ("New Year day is already went by %d days ago" % ((today - nyd).days))

```

Python 2 Example

```

from datetime import date
from datetime import time
from datetime import datetime
def main():
    ##DATETIME OBJECTS
    #Get today's date from datetime class
    today=datetime.now()
    #print today
    # Get the current time
    #t = datetime.time(datetime.now())
    #print "The current time is", t
    #weekday returns 0 (monday) through 6 (sunday)
    wd = date.weekday(today)
    #Days start at 0 for monday
    days= ["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]
    print "Today is day number %d" % wd
    print "which is a " + days[wd]

if __name__ == "__main__":
    main()

```

```

#
#Example file for formatting time and date output

```

```
#
from datetime import datetime
def main():
    #Times and dates can be formatted using a set of predefined string
    #Control codes
    now= datetime.now() #get the current date and time
    #%c - local date and time, %x-local's date, %X- local's time
    print now.strftime("%c")
    print now.strftime("%x")
    print now.strftime("%X")
##### Time Formatting #####
    #%I/%H - 12/24 Hour, %M - minute, %S - second, %p - local's AM/PM
    print now.strftime("%I:%M:%S %p") # 12-Hour:Minute:Second:AM
    print now.strftime("%H:%M") # 24-Hour:Minute

if __name__ == "__main__":
    main()
```

```
#
# Example file for working with timedelta objects
#
from datetime import date
from datetime import time
from datetime import datetime
from datetime import timedelta

# construct a basic timedelta and print it
print timedelta(days=365, hours=8, minutes=15)
# print today's date
print "today is: " + str(datetime.now())
# print today's date one year from now
print "one year from now it will be:" + str(datetime.now() + timedelta(days=365))
# create a timedelta that uses more than one argument
# print "in one week and 4 days it will be " + str(datetime.now() + timedelta(weeks=1, days=4))
# How many days until New Year's Day?
today = date.today() # get today's date
nyd = date(today.year, 1, 1) # get New Year Day for the same year
# use date comparison to see if New Year Day has already gone for this year
# if it has, use the replace() function to get the date for next year
if nyd < today:
    print "New Year day is already went by %d days ago" % ((today - nyd).days)
```

Summary

For manipulating dates and times in both simple and complex ways datetime module supplies different classes or categories like

htt

- date – Manipulate just date (Month, day, year)

- time – Time independent of the day (Hour, minute, second, microsecond)
- datetime – Combination of time and date (Month, day, year, hour, second, microsecond)
- timedelta— A duration of time used for manipulating dates
- tzinfo— An abstract class for dealing with timezones

Using datetime objects

- Importing datetime objects before executing the code is mandatory
- Using date.today function for printing individual date/month/year as well as indexing the day
- Using date.time object to get time in hours, minutes, seconds and milliseconds

Formatting Time-Out with "str f time function"

- Use "str f time function" to change the format of the year
- Print day, date, month and year separately,
- Call out time for any format 12 hrs or 24 hrs

Timedelta Objects

- With timedelta objects, you can estimate the time for both future and the past
- Calculate the total days left for the special day(birthday) from the current time
- Calculate the total days passed for special day(birthday) from the current time