When uploading files to a web server using an automated FTP method, it might be helpful to have a time stamp in the file names. That is, the date and time could be incorporated into the file name. Below you will find a copy of a Python shell session that shows a method for creating a file name containing a time stamp. When you write your own programs, you can leave out the print statements as they are included here only to help you understand certain details.

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.

>>> from datetime import datetime

>>> now = datetime.now()
      # the object named now contains the current date and time

>>> print(now)
2018-01-17 09:06:37.201419
      # here we print out the contents of now so that we can see exactly the
      format.  Note that the date is contained within the first ten
      characters.  Then there is a space character.  The hour is contained in
      the next 2 characters, followed by a colon, then the minutes in the
      next two characters, followed by a colon, then the seconds, to 6 places
      past the decimal point.  It is not good practice to have spaces in a
      file name if it is to be stored on a web server.  In addition, it is
      not good practice to use the colon character in a file name (the colon
      is a reserved character for a special use, e.g., http://lafavre.us).
      Therefore, we need to remove the colon characters and the space.

>>> type(now)
<class 'datetime.datetime'>
      # you have already learned about three types of objects: int or
      integer, float or floating point number and str or string.  In Python
      you can check the type with this command: type(put object name here).
      Notice that the object named now is not a string object, it is a
      datetime.datetime object.  Before we can use the object now as part of
      a file name, is must be of the string type.  Therefore, we will start
      over and create a new object.

>>> now = str(datetime.now())
      # notice here that the datetime.now()object is converted to a string as
      it is assigned to the object name now

>>> print(now)
2018-01-17 09:12:36.640507
      # we print just to check the new contents of now

>>> type(now)
<class 'str'>
      # we check the type again and this time we see it is a string, the
      correct type for our needs.  We can proceed.  We want to extract only
      certain portions of the string in now, which we can do by indexing.
      Characters are indexed in the string using a position specification.
      The first position is zero.  In the now string the first character is a
      2 and it occupies position zero.  The second character, 0, occupies
      position one, etc.  Notice in the next Python line the part [0:10].
      This means to extract from the string starting with the first character
```

```
      and ending with the tenth character (the tenth character is in position
      nine)

>>> date = now[0:10]
      # the object named date will contain the first 10 characters of the
      string contained in the object named now.

>>> print(date)
2018-01-17
      # here we print out contents of date to confirm we got it right

>>> hour = now[11:13]
      # the first ten characters in the string now are the date and the
      eleventh character is a space.  Therefore, the first character of the
      hour is the twelfth character, which occupies position 11 (remember we
      start counting at 0, not 1).  The hour is contained in two characters,
      in positions 11 and 12.  Therefore, [11:13] will extract the characters
      in positions 11 and 12.

>>> print(hour)
09
      # we print out contents of hour to confirm we got it right

>>> minutes = now[14:16]
      # the minutes are contained in positions 14 and 15 (there is a colon at
      position 13 that we don't want to include)

>>> print(minutes)
12
      # checking to see we captured the minutes correctly

>>> seconds = now[17:]
      # there is a colon at position 16 and the seconds start at position 17.
      Since we don't specify an ending position, we get the characters all
      the way to the end of the string, starting at position 17.

>>> print(seconds)
36.640507
      # confirms we captured the seconds to 6 places past the decimal point.
      We don't really need to add the seconds to the file name, but I
      included it just so you could see how to retrieve the seconds.

>>> filename = "Log" + date + "_" + hour + "_" + minutes + ".html"
      # here we assemble the file name by concatenation, using the +
      operator.  In order to concatenate this way, all items must be of the
      same type.  Any text delimited by quotes will automatically be of the
      string type.  The three objects named date, hour and minutes are also
      of the string type because our code specified it in this line:
      now = str(datetime.now())

>>> print(filename)
Log2018-01-17_09_12.html
      # notice that the file name contains underscore (_) characters to
      separate elements of the time stamp.  All characters used in the file
      name are legal.
```