How to print the current date.

1

# import datetime class from datetime module 2

from datetime import datetime

3

4

# get current date

5

datetime\_object = datetime.now()

6

print(datetime\_object)

7

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.

1

my\_string = '2019-10-31'

2

3

# Create date object in given time format yyyy-mm-dd 4

my\_date = datetime.strptime(my\_string, "%Y-%m-%d") 5

6

print(my\_date)

7

print('Type: ',type(my\_date))

2019-10-31 00:00:00

Type: <class 'datetime.datetime'>

Separate month and year

1

my\_date = datetime.strptime(my\_string, "%Y-%m-%d") 2

print('Month: ', my\_date.month) # To Get month from date

o

3

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.

1

# import calendar module

2

import calendar

3

print('Day of Month:', my\_date.day)

4

5

# to get name of day(in number) from date

6

print('Day of Week (number): ', my\_date.weekday())

7

8

# to get name of day from date

9

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'.

1

j = 0

2

for i in calendar.day\_name: 3

    print(j,'-',i)

4

    j+=1

0 - Monday

1 - Tuesday

2 - Wednesday

3 - Thursday

4 - Friday

5 - Saturday

6 - Sunday

Extract the hours and minutes from datetime object

1

from datetime import datetime

2

todays\_date = datetime.now()

3

4

# to get hour from datetime

t

5

print('Hour: ', todays\_date.hour) 6

7

# to get minute from datetime

8

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.

1

# Return a 3-tuple, (ISO year, ISO week number, ISO weekday). 2

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

1

#import datetime

2

from datetime import datetime

3

# get current date

4

now = datetime.now()

5

6

# convert current date into timestamp 7

timestamp = datetime.timestamp(now) 8

9

print("Date and Time :", now)

10

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 oat format) as an argument and returns a datetime object, as below

1

#import datetime

2

from datetime import datetime

3

timestamp = 1599737036.845432

4

5

#convert timestamp to datetime object 6

dt\_object = datetime.fromtimestamp(timestamp) 7

8

print("dt\_object:", dt\_object)

9

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.

1

#import datetime

2

from datetime import timedelta

t

3

# create timedelta object with difference of 2 weeks 4

d = timedelta(weeks=2)

5

6

print(d)

7

print(type(d))

8

print(d.days)

14 days, 0:00:00

<class 'datetime.timedelta'>

We can get our time duration in days by using the timedelta class attribute .days. 14

1

year = timedelta(days=365)

2

print(year)y

365 days, 0:00:00

1

#import datetime

2

from datetime import datetime, timedelta

3

# get current time

4

now = datetime.now()

5

print ("Today's date: ", str(now))

6

7

#add 15 days to current date

8

future\_date\_after\_15days = now + timedelta(days = 15) 9

print('Date after 15 days: ', future\_date\_after\_15days) 10

11

#subtract 2 weeks from current date

12

two\_weeks\_ago = now - timedelta(weeks = 2)

13

print('Date two weeks ago: ', two\_weeks\_ago)

14

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 nd 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.

1

# import datetime

2

from datetime import date

3

# Create two dates

4

date1 = date(2012, 8, 18)

5

date2 = date(2019, 9, 23)

6

7

# Difference between two dates

8

delta = date2 - date1

9

print("Difference: ", delta.days)

10

print('delta object type: ', type(delta))

Difference: 2592

delta object type: <class 'datetime.timedelta'>

Here is another way to compute duration.

1

# import datetime

2

from datetime import datetime

3

# create two dates with year, month, day, hour, minute, and second 4

date1 = datetime(2017, 6, 21, 18, 25, 30)

5

date2 = datetime(2017, 5, 16, 8, 21, 10)

6

7

# Difference between two dates

8

diff = date1-date2

9

print("Difference: ", diff)

Difference: 36 days, 10:04:20

Formatting time.

1

# import datetime

2

from datetime import datetime

3

date\_string = "1 August, 2019"

4

5

# format date

6

date\_object = datetime.strptime(date\_string, "%d %B, %Y") 7

8

print("date\_object: ", date\_object)

date\_object: 2019-08-01 00:00:00

Manipulating date and time and formatting them.

1

# import datetime

2

from datetime import datetime

3

dt\_string = "12/11/2018 09:15:32"

4

# Considering date is in dd/mm/yyyy format

5

dt\_object1 = datetime.strptime(dt\_string, "%d/%m/%Y %H:%M:%S") 6

print("dt\_object1:", dt\_object1)

7

# Considering date is in mm/dd/yyyy format

8

dt\_object2 = datetime.strptime(dt\_string, "%m/%d/%Y %H:%M:%S") 9

print("dt\_object2:", dt\_object2)

10

11

# Convert dt\_object2 to Unix Timestamp

12

timestamp = datetime.timestamp(dt\_object2)

13

print('Unix Timestamp: ', timestamp)

14

15

# Convert back into datetime

16

date\_time = datetime.fromtimestamp(timestamp)

17

d = date\_time.strftime("%c")

18

print("Output 1:", d)

19

d = date\_time.strftime("%x")

20

print("Output 2:", d)

21

d = date\_time.strftime("%X")

22

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

1

# current date and time

2

now = datetime.now()

3

4

# get year from date

5

year = now.strftime("%Y")

6

print("Year:", year)

7

8

# get month from date

9

month = now.strftime("%m")

10

print("Month;", month)

11

12

# get day from date

13

day = now.strftime("%d")

14

print("Day:", day)

15

16

# format time in HH:MM:SS

17

time = now.strftime("%H:%M:%S")

18

print("Time:", time)

19

20

# format date

21

date\_time = now.strftime("%m/%d/%Y, %H:%M:%S") 22

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

1

# import timezone from pytz module

2

from pytz import timezone

3

# Create timezone US/Eastern

4

east = timezone('US/Eastern')

d

5

# Localize date

6

loc\_dt = east.localize(datetime(2011, 11, 2, 7, 27, 0)) 7

print(loc\_dt)

8

9

# Convert localized date into Asia/Kolkata timezone 10

kolkata = timezone("Asia/Kolkata")

11

print(loc\_dt.astimezone(kolkata))

12

13

# Convert localized date into Australia/Sydney timezone

13

# Convert localized date into Australia/Sydney 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)a

2019-09-08 00:00:00

We create a series of twelve dates starting from the day we dened above. Then we create a different series of dates starting from a predened 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

p

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()d

**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, lik **4** 2019-08-14 2019 8 14 0 e the da 0 y of the week and the day of the year, from its datetime objects.

1

# get Weekday and Day of Year. Assign all these date component to new column. 2

df['weekday'] = df['date'].dt.weekday

3

df['dayofyear'] = df['date'].dt.dayofyear

d

4

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.

1

# Assign date column to dataframe index

2

df.index = df.date

3

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