Datetime

If i want to deal with date and time related issues we must use **datetime** Module or Library.
In Python, date and time are not a data type of their own, but a module named datetime can be imported to work with the date as well as time.
Date Syntax:-**datetime.date(year, month, day)**
By Default Date format:- yyyy-m-dd
Time Syntax:-**datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0)**

In [102]:

```
import datetime
```

In [11]:

```
print(dir(datetime)) # All the functions in the date time Module.
```

```
['MAXYEAR', 'MINYEAR', '__all__', '__builtins__', '__cached__', '__doc__', '_
_file__', '__loader__', '__name__', '__package__', '__spec__', 'date', 'datet
ime', 'datetime_CAPI', 'sys', 'time', 'timedelta', 'timezone', 'tzinfo']
```

'date', 'datetime', 'time', 'timedelta', ('timezone', 'tzinfo''MAXYEAR', 'MINYEAR', 'datetime_CAPI', 'sys')

In [103]:

```
datetime.date.today()
```

Out[103]:

```
datetime.date(2023, 3, 31)
```

In [104]:

```
# I want to print today date
today_date=datetime.date.today()
print(today_date)
```

```
2023-03-31
```

In [107]:

```
# I want to print today date with time
date_time=datetime.datetime.today()
date_time
```

Out[107]:

```
datetime.datetime(2023, 3, 31, 12, 51, 22, 622120)
```

In [108]:

```
# I want to print today date with time
date_time=datetime.datetime.today()
print(date_time)
```

```
2023-03-31 12:51:35.757009
```

```
In [36]: # Always writing datetime datetime i don't like it. Simply i import date module
         from datetime import date
         # here all date related functions in this
```

Date

```
In [65]: date()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13868\2647489711.py in <module>
----> 1 date()

TypeError: function missing required argument 'year' (pos 1)
```

```
In [34]: # Now see Today date simply
         dt=date.today(); print(dt)
```

```
2023-03-31
```

```
In [42]: # If i want to enter a particular hard coded date like date of birthdate then w
         independence_day=date(1947,8,15) # (yyyy,m,dd)
         print("Independence_day",independence_day)
```

```
Independence_day 1947-08-15
```

```
In [45]: # I want to extract year, month, day from today date for particular date.
         y=dt.year;m=dt.month;d=dt.day
         print("year",y)
         print("month",m)
         print("day",d)
```

```
year 2023
month 3
day 31
```

```
In [46]: y=independence_day.year;m=independence_day.month;d=independence_day.day
         print("year",y)
         print("month",m)
         print("day",d)
```

```
year 1947
month 8
day 15
```

Time

In [82]:

```
# here all time related functions in this
from datetime import time
```

In [67]:

```
time() # We don't have direct time fetching function in time module. but we car
```

Out[67]:

```
datetime.time(0, 0)
```

In [94]:

```
# Now we can enter manually
tim=time(10,59,8,2534)# format(hh,mm,ss)
print("birth_time",tim)
print("hour",tim.hour)
print("minute",tim.minute)
print("second",tim.second)
print("microsecond",tim.microsecond)
```

```
birth_time 10:59:08.002534
hour 10
minute 59
second 8
microsecond 2534
```

Date time combination

In [109]:

```
from datetime import datetime
```

In [126]:

```
# Everything return date and time
date_today=datetime.today()
date_now=datetime.now()
dt=date_now
dt=date_time
dt=date_today
print("date_today",date_today)
date_today

print("date_now",date_now)
date_now
```

```
date_today 2023-03-31 13:01:43.737887
date_now 2023-03-31 13:01:43.737886
```

Out[126]:

```
datetime.datetime(2023, 3, 31, 13, 1, 43, 737886)
```

In [130]:

```
# Manual Entry date with time
# date of bith with time
dt=datetime(2020,7,22);print(dt)
dt1=datetime(2020,7,22,8,52,0);print(dt1)
```

```
2020-07-22 00:00:00
2020-07-22 08:52:00
```

In [169]:

```
print("Date",dt1.date(),end=' ');print("year",dt1.year,end=" ");print("month",d
print("Time",dt1.time(),end=" ");print("Hour",dt1.hour,end=" ");print("Minute",
print("Second",dt1.second,end=" ");print("MicroSecond",dt1.microsecond)
print("Timestamp",dt1.timestamp())
print("Time Zone",dt1.astimezone())
print("ctime",dt1.ctime())
print("timetuple",dt1.timetuple())

print("Now",dt1.now(),end=" ")
print("Min date time",dt1.min,end=" ")
print("Max date time",dt1.max)
```

```
Date 2020-07-22 year 2020 month 7 day 22
Time 08:52:00 Hour 8 Minute 52 Second 0 MicroSecond 0
Timestamp 1595388120.0
Time Zone 2020-07-22 08:52:00+05:30
ctime Wed Jul 22 08:52:00 2020
timetuple time.struct_time(tm_year=2020, tm_mon=7, tm_mday=22, tm_hour=8, tm_
min=52, tm_sec=0, tm_wday=2, tm_yday=204, tm_isdst=-1)
Now 2023-03-31 15:47:34.214477 Min date time 0001-01-01 00:00:00 Max date tim
e 9999-12-31 23:59:59.999999
```

In [224]:

```
# previously we faced issue in time function we don't have direct time fetching
```

```
2023-03-31 16:15:37.703976
```

In [181]:

```
print("Timestamp",dt1.timestamp()) # It return date into seconds.
```

```
Timestamp 1595388120.0
```

Time Stamp means Seconds===> 1 day=24 hours, 1 hr =60 min, 1 min= 60 sec. how many seconds for one day?

sec_day=24 x 60 x 60

sec_day=86400 Seconds.

In [183]:

```
from datetime import date
```

In [186]:

```
timestamp=date.fromtimestamp(1595388120)
print(timestamp)
# It is matching with dt1 date of birth time date.
```

```
2020-07-22
```

In [188]:

```
timestamp=date.fromtimestamp(1)
print(timestamp) # By default it returns 1970-01-01. if i want next day add one
```

```
1970-01-01
```

In [200]:

```
timestamp=date.fromtimestamp(86400)
print(timestamp) # Yes exactly working. Lets cross verify
```

```
1970-01-02
```

In [201]:

```
# Lets check reduce 60 sec
timestamp=date.fromtimestamp(86340)
print(timestamp) # It is giving same answer! why because we are +5:30 ahead of
```

```
1970-01-02
```

In [199]:

```
5*3600+0.5*3600
```

Out[199]:

```
19800.0
```

In [206]:

```
(86400-19800)
```

Out[206]:

```
66600
```

In [208]:

```
# Lets check reduce 5hr:30 mins in sec
timestamp=date.fromtimestamp(66600)
print(timestamp)
```

```
1970-01-02
```

In [209]:

```
timestamp=date.fromtimestamp(66599)
print(timestamp) # Yes it is working
```

```
1970-01-01
```

In [213]:

```
(66600+86400)
```

Out[213]:

```
153000
```

In [214]:

```
timestamp=date.fromtimestamp(153000)
print(timestamp) # Yes it is working
```

```
1970-01-03
```

In [215]:

```
timestamp=date.fromtimestamp(153000-1)
print(timestamp) # Yes it is working
```

```
1970-01-02
```

Now we will see difference between two dates and times

In [235]:

```
# Difference between 2 dates
from datetime import datetime,date
d1=date(2023,3,1)
d2=date(2022,3,1)
print(d1-d2)
d1-d2
```

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

Out[235]:

```
datetime.timedelta(days=365)
```

In [236]:

```
# Difference between 2 dates with time interval
from datetime import datetime,time
dt1=datetime(2023,3,1,18,30,0)
dt2=datetime(2022,3,1,17,30,0)
print(dt1-dt2)
dt1-dt2
```

```
365 days, 1:00:00
```

Out[236]:

```
datetime.timedelta(days=365, seconds=3600)
```

If i want difference weeks and days we use timedeltafunctions

In [238]:

```
from datetime import timedelta
wd1=timedelta(weeks=2,days=5,hours=1,seconds=30)
wd2=timedelta(days=5,hours=1,seconds=30)
print(wd1-wd2)
```

```
14 days, 0:00:00
```

In [240]:

```
# Converting days time into seconds.
t6=timedelta(days=7,hours=1,seconds=33)
t6.total_seconds()
```

Out[240]:

```
608433.0
```

In [1]:

```
7*86400+3600+33
```

Out[1]:

```
608433
```

In [4]:

```
# If i want to convert seconds into date by using timestamp as we seen in top.
from datetime import date
d=date.fromtimestamp(608433)
print(d)
```

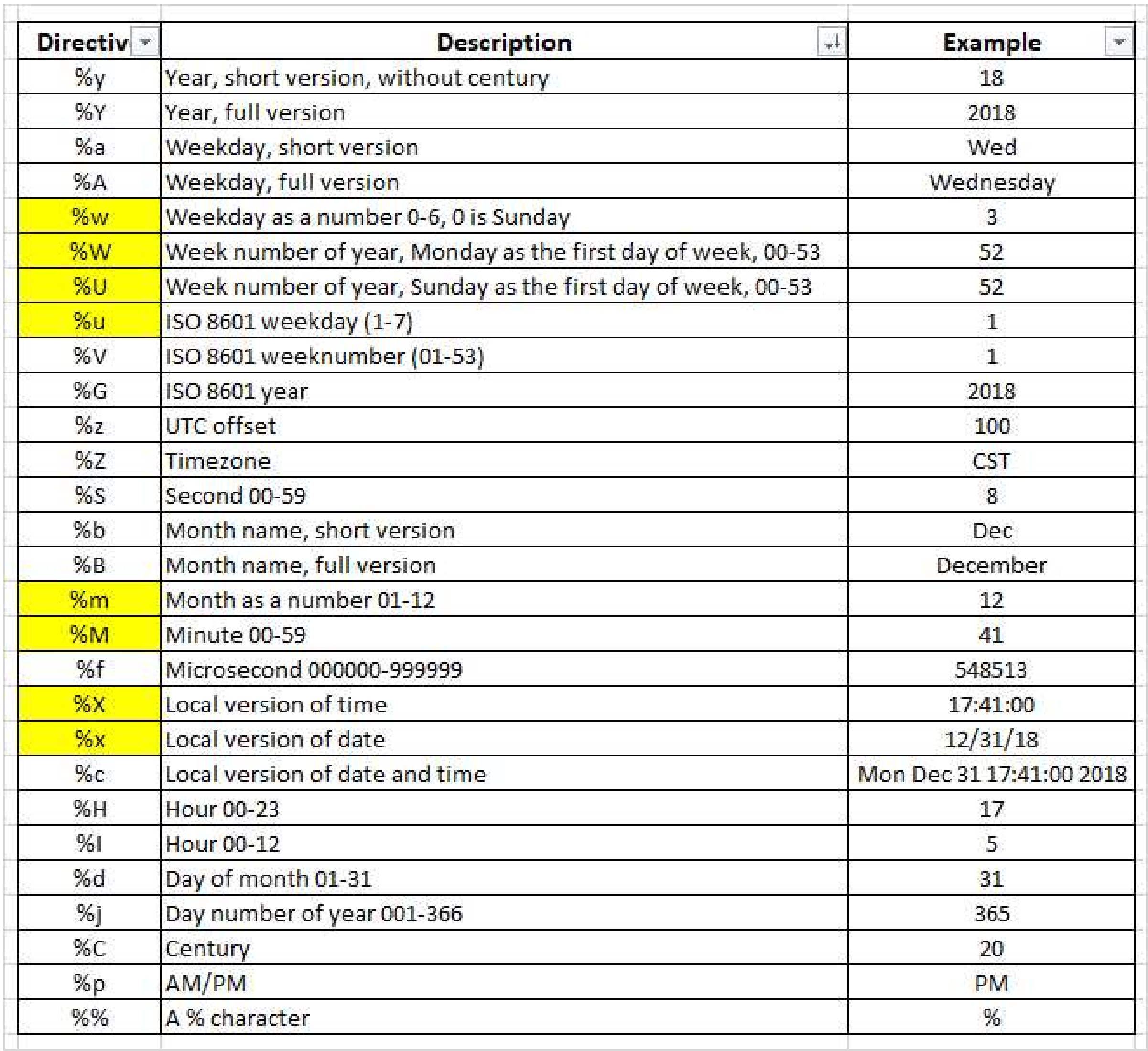
```
1970-01-08
```

But i don't like the numbers i want in month names like jan,feb or July, Auguest then we must format the date and time

Date and Time Formatting.

If we want to format the time and date we must have to understand the two functions.

1. strftime( )===> System time to user required format (For formatting date objects into readable strings.)
2. strptime( )===> User format to system time format.



Directiv	Description	Example
%y	Year, short version, without century	18
%Y	Year, full version	2018
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%W	Week number of year, Monday as the first day of week, 00-53	52
%U	Week number of year, Sunday as the first day of week, 00-53	52
%u	ISO 8601 weekday (1-7)	1
%V	ISO 8601 weeknumber (01-53)	1
%G	ISO 8601 year	2018
%z	UTC offset	100
%Z	Timezone	CST
%S	Second 00-59	8
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%M	Minute 00-59	41
%f	Microsecond 000000-999999	548513
%X	Local version of time	17:41:00
%x	Local version of date	12/31/18
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%H	Hour 00-23	17
%I	Hour 00-12	5
%d	Day of month 01-31	31
%j	Day number of year 001-366	365
%C	Century	20
%p	AM/PM	PM
%%	A % character	%

Converting System time format to user define format then we use

strftime() function

_s==>System
_u==>User

In [2]:

```
# What is current Date and time
from datetime import datetime
now=datetime.now()
print(now)
```

```
2023-04-03 11:43:06.117056
```

In [79]:

```
# From now i want only date then
date_s=now.date()
print("Default python date format",date_s);print(type(date_s))

# I want to convert system format into my desire format then,
date_u=now.strftime("%d-%m-%y")
print("Custmized user date format",date_u,"I changed as day-month-year and Year
date_u=now.strftime("%d-%m-%Y")
print("Custmized user date format",date_u,"Year is long  %Y")
date_u=now.strftime("%d-%b-%Y")
print("Custmized user date format",date_u,"Month is short %b")
date_u=now.strftime("%d-%B-%Y")
print("Custmized user date format",date_u,"Month is Long %B")
date_u=now.strftime("%a %d-%m-%y")
print("Custmized user date format",date_u,"short week name  %a")
date_u=now.strftime("%A %d-%m-%y")
print("Custmized user date format",date_u,"long week name  %A")

date_u=now.strftime("%Y-%m-%d %W week from year and %w week from month")
print("Custmized user date format",date_u,"%W,%w")

date_u=now.strftime("%j From starting of year to till date number of days")
print("Custmized user date format",date_u,"%j")
```

```
Default python date format 2023-04-03
<class 'datetime.date'>
Custmized user date format 03-04-23 I changed as day-month-year and Year is s
hort %d-%m-%y
Custmized user date format 03-04-2023 Year is long  %Y
Custmized user date format 03-Apr-2023 Month is short %b
Custmized user date format 03-April-2023 Month is Long %B
Custmized user date format Mon 03-04-23 short week name  %a
Custmized user date format Monday 03-04-23 long week name  %A
Custmized user date format 2023-04-03 14 week from year and 1 week from month
%W,%w
Custmized user date format 093 From starting of year to till date number of d
ays %j
```

In [74]:

```
# From now i want only time then
time_s=now.time()
print("Default python time format",time_s)
time_u=now.strftime("%H:%M:%S.%f")
print("Custmized user time format",time_u,"24-hrs format %H")
time_u=now.strftime("%I:%M:%S")
print("Custmized user time format",time_u,"12-hrs format %I")
time_u=now.strftime("%I:%M:%S %p")
print("Custmized user time format",time_u,"(AM/PM) %p")
time_u=now.strftime("%H:%M")
print("Custmized user time format",time_u,"Only hrs and min")
```

```
Default python time format 12:29:51.855766
Custmized user time format 12:29:51.855766 24-hrs format %H
Custmized user time format 12:29:51 12-hrs format %I
Custmized user time format 12:29:51 PM (AM/PM) %p
Custmized user time format 12:29 Only hrs and min
```

In [61]:

```
# Current date and time
from datetime import datetime
now=datetime.now(); print("Default python datetime format",now)

date_time=now.strftime("%d-%m-%Y, %H:%M:%S.%f 24-hrs format")
print("Custmized user datetime format",date_time,"%H")

date_time=now.strftime("%d-%m-%Y, %I:%M:%S.%f 12-hrs format")
print("Custmized user datetime format",date_time,"%I")

date_time=now.strftime("%d-%m-%Y, %I:%M:%S %p 12-hrs format with (AM/PM)")
print("Custmized user datetime format",date_time,"%p")
```

```
Default python datetime format 2023-04-03 12:29:51.855766
Custmized user datetime format 03-04-2023, 12:29:51.855766 24-hrs format %H
Custmized user datetime format 03-04-2023, 12:29:51.855766 12-hrs format %I
Custmized user datetime format 03-04-2023, 12:29:51 PM 12-hrs format with (A
M/PM) %p
```

Converting System time format to user define format then we use

strptime() function

_u==>User
_s==>System

In [84]:

```
from datetime import datetime
date_u="22-August-2023"
print(date_u)
print(type(date_u))
```

```
22-August-2023
<class 'str'>
```

In [85]:

```
date_s=datetime.strptime(date_u,"%d-%B-%Y")
print(date_s)
```

```
2023-08-22 00:00:00
```

In [87]:

```
import pandas as pd
import numpy as np
from datetime import datetime
```

In [97]:

```
date=pd.Series(pd.date_range("2022-07-01 20:30:45",periods=5))
df=pd.DataFrame(dict(date_given=date))
df
```

Out[97]:

	date_given
0	2022-07-01 20:30:45
1	2022-07-02 20:30:45
2	2022-07-03 20:30:45
3	2022-07-04 20:30:45
4	2022-07-05 20:30:45

In [107]:

```
df["today_date"]=datetime.now()
print(df)
```

```
           date_given                 today_date
0 2022-07-01 20:30:45 2023-04-03 13:14:49.291198
1 2022-07-02 20:30:45 2023-04-03 13:14:49.291198
2 2022-07-03 20:30:45 2023-04-03 13:14:49.291198
3 2022-07-04 20:30:45 2023-04-03 13:14:49.291198
4 2022-07-05 20:30:45 2023-04-03 13:14:49.291198
```

In [108]:

```
df
```

Out[108]:

	date_given	today_date
0	2022-07-01 20:30:45	2023-04-03 13:14:49.291198
1	2022-07-02 20:30:45	2023-04-03 13:14:49.291198
2	2022-07-03 20:30:45	2023-04-03 13:14:49.291198
3	2022-07-04 20:30:45	2023-04-03 13:14:49.291198
4	2022-07-05 20:30:45	2023-04-03 13:14:49.291198

In [109]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   date_given  5 non-null      datetime64[ns]
 1   today_date  5 non-null      datetime64[ns]
dtypes: datetime64[ns](2)
memory usage: 208.0 bytes
```

In [114]:
```
df["Year_given"]=df["date_given"].dt.year
```

In [116]:
```
df["Month_given"]=df["date_given"].dt.month
```

In [117]:
```
df["Day_given"]=df["date_given"].dt.day
```

In [118]:
```
df
```

Out[118]:

	date_given	today_date	Year_given	Month_given	Day_given
0	2022-07-01 20:30:45	2023-04-03 13:14:49.291198	2022	7	1
1	2022-07-02 20:30:45	2023-04-03 13:14:49.291198	2022	7	2
2	2022-07-03 20:30:45	2023-04-03 13:14:49.291198	2022	7	3
3	2022-07-04 20:30:45	2023-04-03 13:14:49.291198	2022	7	4
4	2022-07-05 20:30:45	2023-04-03 13:14:49.291198	2022	7	5

In [121]:
```
df["diff"]=df["today_date"]-df["date_given"] # It returns in days
```

In [122]:
```
df
```

Out[122]:

	date_given	today_date	Year_given	Month_given	Day_given	diff
0	2022-07-01 20:30:45	2023-04-03 13:14:49.291198	2022	7	1	275 days 16:44:04.291198
1	2022-07-02 20:30:45	2023-04-03 13:14:49.291198	2022	7	2	274 days 16:44:04.291198
2	2022-07-03 20:30:45	2023-04-03 13:14:49.291198	2022	7	3	273 days 16:44:04.291198
3	2022-07-04 20:30:45	2023-04-03 13:14:49.291198	2022	7	4	272 days 16:44:04.291198
4	2022-07-05 20:30:45	2023-04-03 13:14:49.291198	2022	7	5	271 days 16:44:04.291198

In [123]:

```
df["date_given_dt"]=pd.to_datetime(df["date_given"]).dt.date
df
```

Out[123]:

	date_given	today_date	Year_given	Month_given	Day_given	diff	date_given_
0	2022-07-01 20:30:45	2023-04-03 13:14:49.291198	2022	7	1	275 days 16:44:04.291198	2022-07-0
1	2022-07-02 20:30:45	2023-04-03 13:14:49.291198	2022	7	2	274 days 16:44:04.291198	2022-07-0
2	2022-07-03 20:30:45	2023-04-03 13:14:49.291198	2022	7	3	273 days 16:44:04.291198	2022-07-0
3	2022-07-04 20:30:45	2023-04-03 13:14:49.291198	2022	7	4	272 days 16:44:04.291198	2022-07-0
4	2022-07-05 20:30:45	2023-04-03 13:14:49.291198	2022	7	5	271 days 16:44:04.291198	2022-07-0

In [124]:

```
df["today_dt"]=pd.to_datetime(df["today_date"]).dt.date
df
```

Out[124]:

	date_given	today_date	Year_given	Month_given	Day_given	diff	date_given_
0	2022-07-01 20:30:45	2023-04-03 13:14:49.291198	2022	7	1	275 days 16:44:04.291198	2022-07-0
1	2022-07-02 20:30:45	2023-04-03 13:14:49.291198	2022	7	2	274 days 16:44:04.291198	2022-07-0
2	2022-07-03 20:30:45	2023-04-03 13:14:49.291198	2022	7	3	273 days 16:44:04.291198	2022-07-0
3	2022-07-04 20:30:45	2023-04-03 13:14:49.291198	2022	7	4	272 days 16:44:04.291198	2022-07-0
4	2022-07-05 20:30:45	2023-04-03 13:14:49.291198	2022	7	5	271 days 16:44:04.291198	2022-07-0

In [126]:

```
df["diff2"]=df["today_dt"]-df["date_given_dt"] # It returns in days
df
```

Out[126]:

	date_given	today_date	Year_given	Month_given	Day_given	diff	date_given_
0	2022-07-01 20:30:45	2023-04-03 13:14:49.291198	2022	7	1	275 days 16:44:04.291198	2022-07-0
1	2022-07-02 20:30:45	2023-04-03 13:14:49.291198	2022	7	2	274 days 16:44:04.291198	2022-07-0
2	2022-07-03 20:30:45	2023-04-03 13:14:49.291198	2022	7	3	273 days 16:44:04.291198	2022-07-0
3	2022-07-04 20:30:45	2023-04-03 13:14:49.291198	2022	7	4	272 days 16:44:04.291198	2022-07-0
4	2022-07-05 20:30:45	2023-04-03 13:14:49.291198	2022	7	5	271 days 16:44:04.291198	2022-07-0

In [130]:

```
# how to convert days into months
df["diff_mnt"]=(df["diff2"]/np.timedelta64(1,"M"))
df
```

Out[130]:

	date_given	today_date	Year_given	Month_given	Day_given	diff	date_given_
0	2022-07-01 20:30:45	2023-04-03 13:14:49.291198	2022	7	1	275 days 16:44:04.291198	2022-07-(
1	2022-07-02 20:30:45	2023-04-03 13:14:49.291198	2022	7	2	274 days 16:44:04.291198	2022-07-(
2	2022-07-03 20:30:45	2023-04-03 13:14:49.291198	2022	7	3	273 days 16:44:04.291198	2022-07-(
3	2022-07-04 20:30:45	2023-04-03 13:14:49.291198	2022	7	4	272 days 16:44:04.291198	2022-07-(
4	2022-07-05 20:30:45	2023-04-03 13:14:49.291198	2022	7	5	271 days 16:44:04.291198	2022-07-(

In [131]:

```
# how to convert days into years
df["diff_year"]=(df["diff2"]/np.timedelta64(1,"Y"))
df
```

Out[131]:

	date_given	today_date	Year_given	Month_given	Day_given	diff	date_given_
0	2022-07-01 20:30:45	2023-04-03 13:14:49.291198	2022	7	1	275 days 16:44:04.291198	2022-07-(
1	2022-07-02 20:30:45	2023-04-03 13:14:49.291198	2022	7	2	274 days 16:44:04.291198	2022-07-(
2	2022-07-03 20:30:45	2023-04-03 13:14:49.291198	2022	7	3	273 days 16:44:04.291198	2022-07-(
3	2022-07-04 20:30:45	2023-04-03 13:14:49.291198	2022	7	4	272 days 16:44:04.291198	2022-07-(
4	2022-07-05 20:30:45	2023-04-03 13:14:49.291198	2022	7	5	271 days 16:44:04.291198	2022-07-(

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```