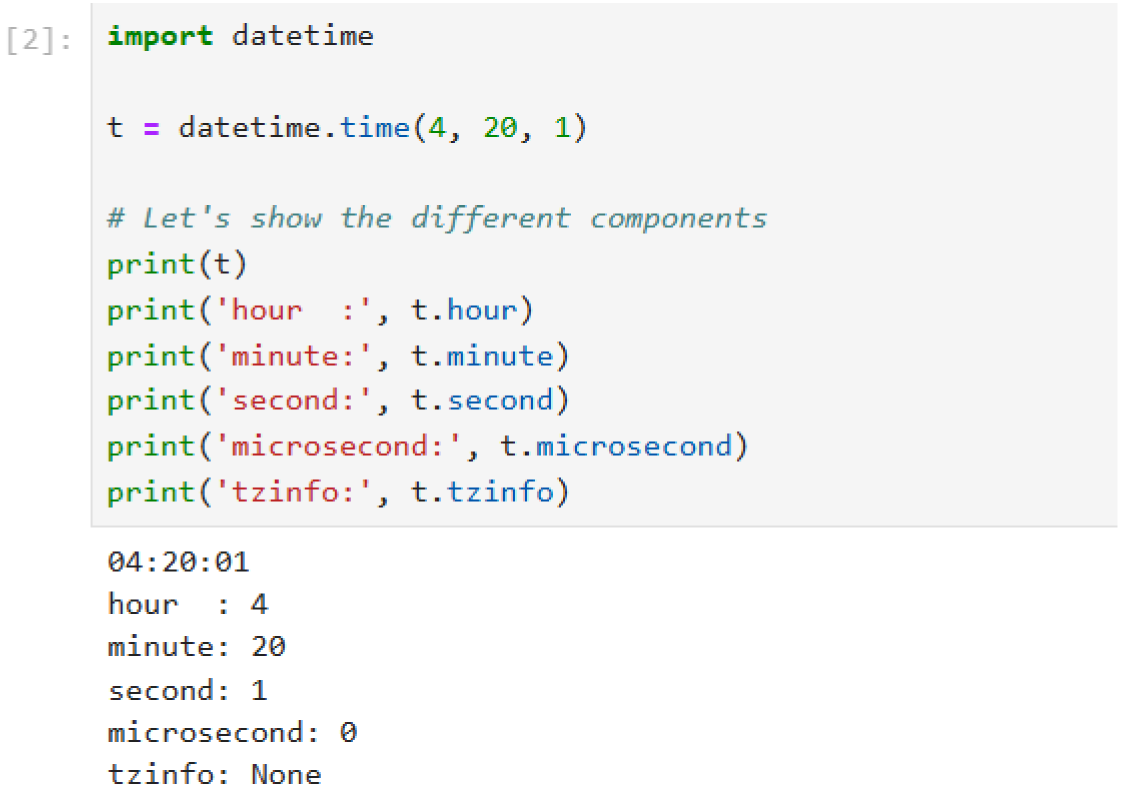
😊 Date Time Module

1. Now we are going to look at the Date Time module in Python.
2. Python has the datetime module to help deal with timestamps in your code. Time values are represented with the time class. Times have attributes for hour, minute, second, and microsecond. They can also include time zone information. The arguments to initialize a time instance are optional, but the default of 0 is unlikely to be what you want.
3. This code uses the datetime module to create a time object representing 4:20:01 (with microseconds defaulting to 0 and no timezone information). It then prints the entire time object and separately prints its individual components: hour, minute, second, microsecond, and tzinfo.



[2]:

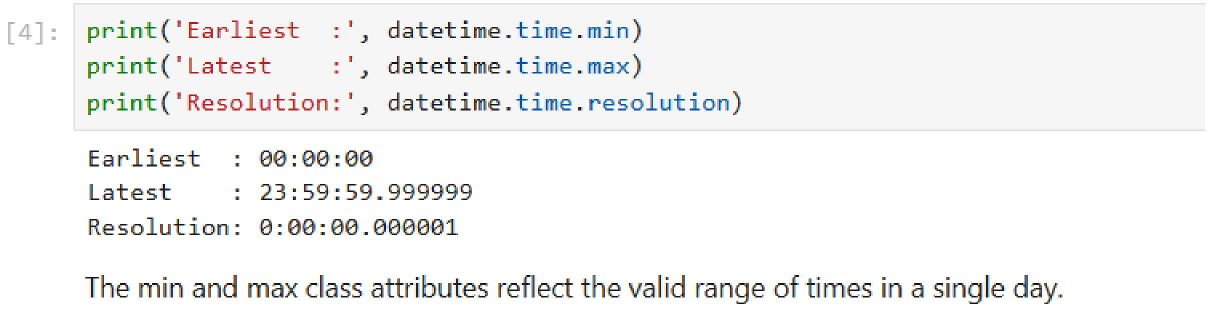
```
import datetime

t = datetime.time(4, 20, 1)

# Let's show the different components
print(t)
print('hour  :', t.hour)
print('minute:', t.minute)
print('second:', t.second)
print('microsecond:', t.microsecond)
print('tzinfo:', t.tzinfo)
```

```
04:20:01
hour  : 4
minute: 20
second: 1
microsecond: 0
tzinfo: None
```

4. **This code prints three attributes of Python's time objects:**
    - The earliest representable time.
    - The latest representable time.
    - The smallest difference between distinct time values (the resolution).
5. These attributes provide the boundaries and granularity of time objects in the datetime module.

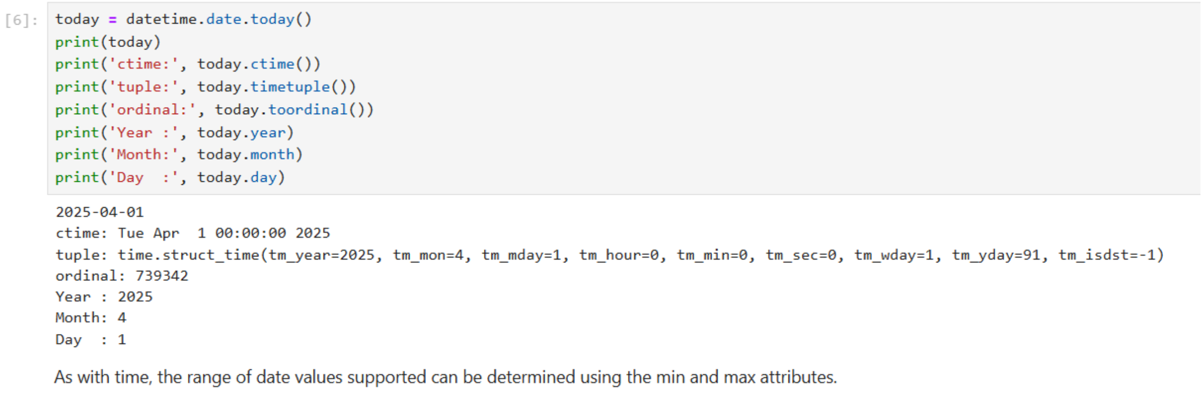


```python
print('Earliest  :', datetime.time.min)
print('Latest    :', datetime.time.max)
print('Resolution:', datetime.time.resolution)
```

```
Earliest  : 00:00:00
Latest    : 23:59:59.999999
Resolution: 0:00:00.000001
```

The min and max class attributes reflect the valid range of times in a single day.

6. The code below uses the datetime module to obtain the current date. It then prints today's date and displays various representations and components of that date. Specifically, it shows a human-readable string format, a tuple format of the date, the ordinal number representing the date, and the individual year, month, and day values.

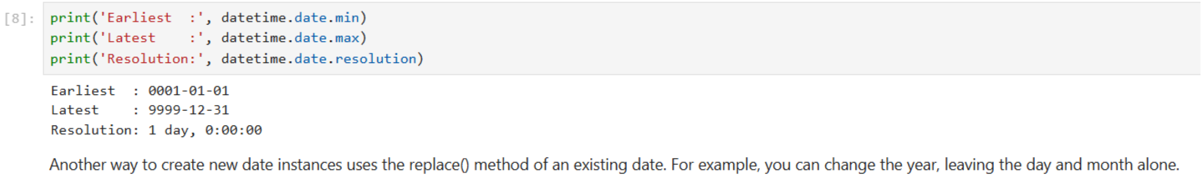


```python
today = datetime.date.today()
print(today)
print('ctime:', today.ctime())
print('tuple:', today.timetuple())
print('ordinal:', today.toordinal())
print('Year :', today.year)
print('Month:', today.month)
print('Day  :', today.day)
```

```
2025-04-01
ctime: Tue Apr  1 00:00:00 2025
tuple: time.struct_time(tm_year=2025, tm_mon=4, tm_mday=1, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=1, tm_yday=91, tm_isdst=-1)
ordinal: 739342
Year : 2025
Month: 4
Day  : 1
```

As with time, the range of date values supported can be determined using the min and max attributes.

7. This code prints three key attributes of Python's date objects:
   - The earliest date that can be represented (date.min).
   - The latest date that can be represented (date.max).
   - The resolution of date objects, which is the smallest difference between two distinct dates.

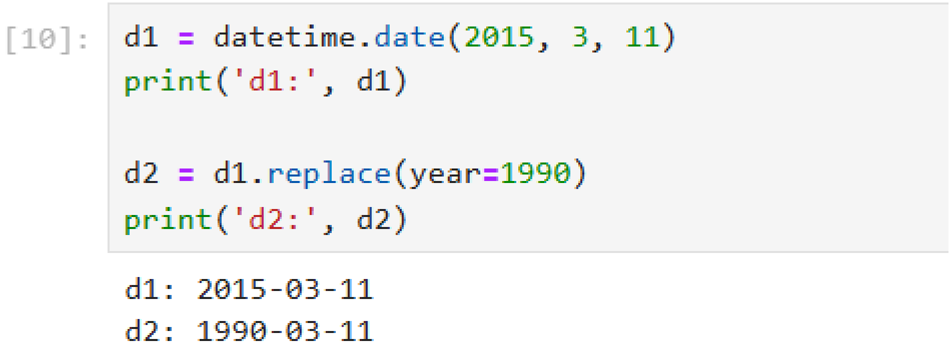


```python
print('Earliest  :', datetime.date.min)
print('Latest    :', datetime.date.max)
print('Resolution:', datetime.date.resolution)
```

```
Earliest  : 0001-01-01
Latest    : 9999-12-31
Resolution: 1 day, 0:00:00
```

Another way to create new date instances uses the replace() method of an existing date. For example, you can change the year, leaving the day and month alone.

8. The code creates a date object representing March 11, 2015, and assigns it to d1. Then, using the replace () method, it creates a new date object (d2) by changing only the year to 1990 while keeping the same month and day.

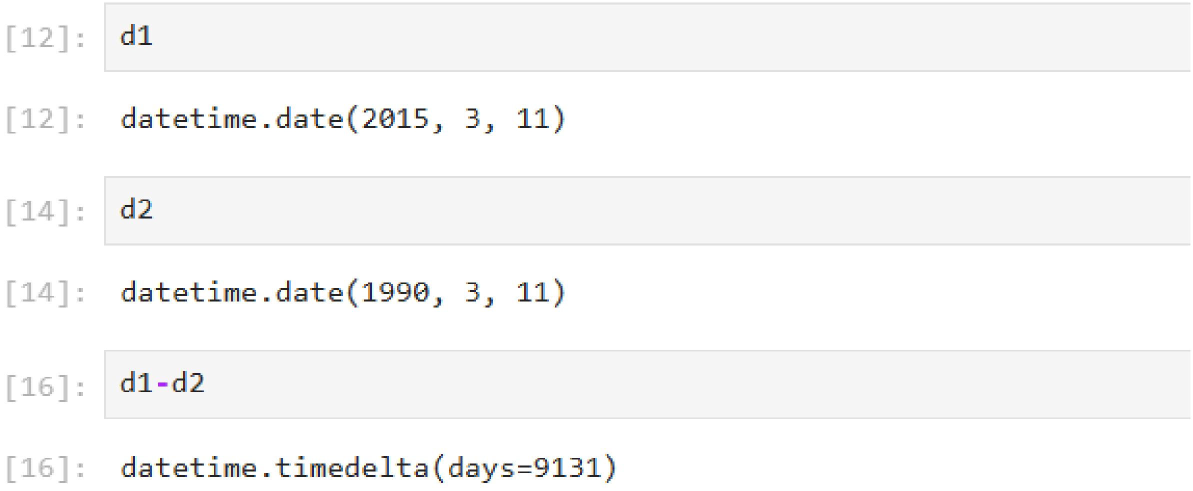


```
[10]: d1 = datetime.date(2015, 3, 11)
      print('d1:', d1)

      d2 = d1.replace(year=1990)
      print('d2:', d2)

      d1: 2015-03-11
      d2: 1990-03-11
```

9. We can also perform arithmetic on date objects to check time differences. This gives us the difference in days between the two dates. You can use the timedelta method to specify various units of times (days, minutes, hours, etc.)



```
[12]: d1

[12]: datetime.date(2015, 3, 11)

[14]: d2

[14]: datetime.date(1990, 3, 11)

[16]: d1-d2

[16]: datetime.timedelta(days=9131)
```