datetime module

Documentation: https://docs.python.org/3/library/datetime.html

LinkedIn: https://www.linkedin.com/in/sandeeppreetam/

Github: https://github.com/sandeeppreetam/python_notebooks/blob/main/datetime.ipynb

importing module

In [1]:
```
import datetime
```

Creating Date and Time Objects

In [2]:
```
# Types: date, time, datetime

today = datetime.date.today()
print('today:', today)

birthday = datetime.date(1999,7,12)
print('birthday:', birthday)

specific_datetime = datetime.datetime(2023,8,23,18,4,0)
print('specific_datetime:', specific_datetime)
```

```
today: 2023-10-28
birthday: 1999-07-12
specific_datetime: 2023-08-23 18:04:00
```

Formatting and Parsing

Directive	Meaning	Example
%Y	Year with century as a decimal number.	2013, 2014
%y	Year without century as a zero-padded decimal number.	00, 01, ..., 99
%m	Month as a zero-padded decimal number.	01, 02, ..., 12
%b	Month as locale's abbreviated name.	Jan, Feb, ..., Dec
%B	Month as locale's full name.	January, February
%d	Day of the month as a zero-padded decimal number.	01, 02, ..., 31
%H	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, ..., 23
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, ..., 12
%M	Minute as a zero-padded decimal number.	00, 01, ..., 59
%S	Second as a zero-padded decimal number.	00, 01, ..., 59
%f	Microsecond as a decimal number, zero-padded to 6 digits.	000000, 000001
%p	Locale's equivalent of either AM or PM.	AM, PM
%a	Weekday as locale's abbreviated name.	Sun, Mon
%A	Weekday as locale's full name.	Sunday, Monday
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, ..., 6
%z	UTC offset in the form ±HHMM[SS[.ffffff]] (empty string if the object is naive).	-0400, +1030, -030712.345216
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, GMT
%j	Day of the year as a zero-padded decimal number.	001, 002, ..., 366
%U	Week number of the year (Sunday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.	00, 01, ..., 53
%W	Week number of the year (Monday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Monday are considered to be in week 0.	00, 01, ..., 53
%%	A literal '%' character.	%

In [3]:
```
print('today:' ,today)

formatted_date = today.strftime("On %d %B, %Y at %I:%M:%S")
print(formatted_date)
```

```
today: 2023-10-28
On 28 October, 2023 at 12:00:00
```

In [4]:
```
date_string = "2023-10-26 00:00:00"
date_format = "%Y-%m-%d %H:%M:%S"

parsed_datetime = datetime.datetime.strptime(date_string, date_format)
print(parsed_datetime)
```

```
2023-10-26 00:00:00
```

Accessing Components of a datetime Object

In [5]:
```
print('specific_datetime:' ,specific_datetime)

print('year:' ,specific_datetime.year)
print('month:' ,specific_datetime.month)
print('day:' ,specific_datetime.day)
print('hour:' ,specific_datetime.hour)
print('minute:' ,specific_datetime.minute)
print('second:' ,specific_datetime.second)
```

```
specific_datetime: 2023-08-23 18:04:00
year: 2023
month: 8
day: 23
hour: 18
minute: 4
second: 0
```

Replacing Components of a datetime Object

In [6]:
```
#datetime_object.replace(year,month,day,hour,minute,second,microsecond,tzinfo)

print('Birthday:' ,birthday)

print('Birthday in 2023:' ,birthday.replace(year=2023))
```

```
Birthday: 1999-07-12
Birthday in 2023: 2023-07-12
```

Date and Time Arithmetic

In [7]:
```
# datetime.timedelta(weeks=0, days=0, hours=0, minutes=0, seconds=0, milliseconds=0, microseconds=0)

print('today:' ,today)

yesterday = today - datetime.timedelta(days=1)
tomorrow = today + datetime.timedelta(days=1)


print('yesterday:' ,yesterday)
print('tomorrow:' ,tomorrow)
```

```
today: 2023-10-28
yesterday: 2023-10-27
tomorrow: 2023-10-29
```

Sample Use Cases

Date Formatting:

In [8]:
```
# Convert the current date to a string in the format "YYYY-MM-DD".
today = datetime.date.today()
print(today)
print(today.strftime('%Y-%m-%d'))
```

```
# Convert the current time to a string in the format "HH:MM:SS".
current_time = datetime.datetime.now().time()
print(current_time)
print(current_time.strftime('%H:%M:%S'))
```

```
2023-10-28
2023-10-28
22:12:34.024815
22:12:34
```

Date Parsing:

In [9]:
```
# Parse the date string "2023-10-26" into a datetime object.
date = '2023-10-26'
pdate = datetime.datetime.strptime(date, '%Y-%m-%d')
print(pdate)


# Parse the time string "14:30:00" into a time object.
time = '14:30:00'
ptime = datetime.datetime.strptime(time, '%H:%M:%S').time()
print(ptime)
```

```
2023-10-26 00:00:00
14:30:00
```

Date Arithmetic:

In [10]:
```
# Calculate the date that is 30 days from the current date.
print(datetime.date.today())
print(datetime.date.today() + datetime.timedelta(days = 30))

# Calculate the date and time that is 6 hours and 45 minutes from now.
print(datetime.datetime.now())
print(datetime.datetime.now() + datetime.timedelta(hours = 6, minutes = 45))
```

```
2023-10-28
2023-11-27
2023-10-28 22:12:34.047086
2023-10-29 04:57:34.047341
```

Day of the Week

In [11]:
```
# Determine the day of the week for a specific date (e.g., "2023-10-26") and display it as a string (e.g., "Wednesd

specific_date = '2023-10-26'
dt = datetime.datetime.strptime(specific_date, '%Y-%m-%d')

print('Day of the week:', dt.strftime('%A'))
```

```
Day of the week: Thursday
```

Age Calculation

In [12]:
```
# Given a birthdate, calculate the age of a person in years.

age = datetime.date.today() - birthday

print(age)

print(age.days / 365)
```

```
8874 days, 0:00:00
24.312328767123287
```

Date Difference:

In [13]:
```
# Calculate the number of days between two specific dates.

sd1 = datetime.date(2023,7,12)
sd2 = datetime.date(1999,7,12)
```

```
print(sd1 - sd2)

# Calculate the number of hours between two specific times on the same day.

t1 = '17:00'
t2 = '9:00'

print((datetime.datetime.strptime(t1, '%H:%M') - datetime.datetime.strptime(t2, '%H:%M')))
```

```
8766 days, 0:00:00
8:00:00
```

Event Scheduling

In [14]:
```
# Schedule an event to occur at a specific date and time in the future,
# and calculate the time remaining until that event.

event = datetime.datetime(2025, 9, 20, 9,0,0)
print(event - datetime.datetime.now())
```

```
692 days, 10:47:25.916363
```

Date Range

In [15]:
```
# Generate a list of dates within a specific date range (e.g., all the dates in a given month).

date = datetime.date(2023,9,1)
start_date = date
date_list = []

while start_date.month == date.month:
    date_list.append(start_date)
    start_date = start_date + datetime.timedelta(days = 1)

print(date_list)
```

```
[datetime.date(2023, 9, 1), datetime.date(2023, 9, 2), datetime.date(2023, 9, 3), datetime.date(2023, 9, 4), datetim
e.date(2023, 9, 5), datetime.date(2023, 9, 6), datetime.date(2023, 9, 7), datetime.date(2023, 9, 8), datetime.date(2
023, 9, 9), datetime.date(2023, 9, 10), datetime.date(2023, 9, 11), datetime.date(2023, 9, 12), datetime.date(2023,
9, 13), datetime.date(2023, 9, 14), datetime.date(2023, 9, 15), datetime.date(2023, 9, 16), datetime.date(2023, 9, 1
7), datetime.date(2023, 9, 18), datetime.date(2023, 9, 19), datetime.date(2023, 9, 20), datetime.date(2023, 9, 21),
datetime.date(2023, 9, 22), datetime.date(2023, 9, 23), datetime.date(2023, 9, 24), datetime.date(2023, 9, 25), date
time.date(2023, 9, 26), datetime.date(2023, 9, 27), datetime.date(2023, 9, 28), datetime.date(2023, 9, 29), datetim
e.date(2023, 9, 30)]
```