Przykład 1

Tworzenie przykładowego obiektu aktualnej daty. ***`datetime`***

In [14]:
```
# Import datetime from the datetime module
from datetime import datetime

# Compute the local datetime: local_dt
local_dt = datetime.now()

# Print the local datetime
print(local_dt)

date_object = datetime.today()
print(date_object)

# dir() pozwala wydrukować wszystkie atrybuty w module
print(dir(datetime))

print(date_object.ctime())
```

```
2020-08-26 14:10:35.119911
2020-08-26 14:10:35.119911
['__add__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__ini
t__', '__init_subclass__', '__le__', '__lt__', '__ne__', '__new__', '__r
add__', '__reduce__', '__reduce_ex__', '__repr__', '__rsub__', '__setatt
r__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', 'astimezon
e', 'combine', 'ctime', 'date', 'day', 'dst', 'fold', 'fromisocalendar',
'fromisoformat', 'fromordinal', 'fromtimestamp', 'hour', 'isocalendar',
'isoformat', 'isoweekday', 'max', 'microsecond', 'min', 'minute', 'month
', 'now', 'replace', 'resolution', 'second', 'strftime', 'strptime', 'ti
me', 'timestamp', 'timetuple', 'timetz', 'today', 'toordinal', 'tzinfo',
'tzname', 'utcfromtimestamp', 'utcnow', 'utcoffset', 'utctimetuple', 'we
ekday', 'year']
Wed Aug 26 14:10:35 2020
```

Przykład 2

Tworzenie przykładowego obiektu daty.

In [2]:
```
# Import date
from datetime import date

d = datetime.date(2019, 4, 13)
print(d)

# Create Dates
two_dates = [date(2016, 10, 7), date(2017, 6, 21)] # dwie daty w tablicy
```

Przykład 3

Atrybuty obiektu daty.

In [4]:
```
# Import date
from datetime import date

# Create Dates
two_dates = [date(2016, 10, 7), date(2017, 6, 21)]

print(two_dates[0].year)
print(two_dates[0].month)
print(two_dates[0].day)
```

```
2016
10
7
```

Przykład 4

***Data z sygnatury czasowej.* `timestamp`**

In [15]:
```
from datetime import date

timestamp = date.fromtimestamp(1326244364)
print("Date =", timestamp)
```

```
Date = 2012-01-11
```

Przykład 5

Obiekt czasu. `time`

In [16]:
```
from datetime import time

# time(hour = 0, minute = 0, second = 0)
a = time()
print("a =", a)

# time(hour, minute and second)
b = time(11, 34, 56)
print("b =", b)

# time(hour, minute and second)
c = time(hour = 11, minute = 34, second = 56)
print("c =", c)

# time(hour, minute, second, microsecond)
d = time(11, 34, 56, 234566)
print("d =", d)
```

```
a = 00:00:00
b = 11:34:56
c = 11:34:56
d = 11:34:56.234566
```

Przykład 6

Drukowanie godziny, minuty, sekundy i mikrosekundy.

In [17]:
```
from datetime import time

a = time(11, 34, 56)

print("hour =", a.hour)
print("minute =", a.minute)
print("second =", a.second)
print("microsecond =", a.microsecond)
```

```
hour = 11
minute = 34
second = 56
microsecond = 0
```

Przykład 7

In [ ]:
```
from datetime import datetime

a = datetime(2017, 11, 28, 23, 55, 59, 342380)
print("year =", a.year)
print("month =", a.month)
print("hour =", a.hour)
print("minute =", a.minute)
print("timestamp =", a.timestamp())
```

Przykład 8

Delta czasu. Różnica pomiędzy dwoma datami. `datetime.timedelta`

In [18]:
```
from datetime import datetime, date

t1 = date(year = 2018, month = 7, day = 12)
t2 = date(year = 2017, month = 12, day = 23)
t3 = t1 - t2
print("t3 =", t3)

t4 = datetime(year = 2018, month = 7, day = 12, hour = 7, minute = 9, seco
nd = 33)
t5 = datetime(year = 2019, month = 6, day = 10, hour = 5, minute = 55, sec
ond = 13)
t6 = t4 - t5
print("t6 =", t6)

print("type of t3 =", type(t3))
print("type of t6 =", type(t6))
```

```
t3 = 201 days, 0:00:00
t6 = -333 days, 1:14:20
type of t3 = <class 'datetime.timedelta'>
type of t6 = <class 'datetime.timedelta'>
```

Przykład 9

In [19]:
```
from datetime import timedelta

t1 = timedelta(weeks = 2, days = 5, hours = 1, seconds = 33)
t2 = timedelta(days = 4, hours = 11, minutes = 4, seconds = 54)
t3 = t1 - t2

print("t3 =", t3)
```

```
t3 = 14 days, 13:55:39
```

Przykład 10

Czas w sekundach. ***`total_seconds()`***

In [20]:
```
from datetime import timedelta

t = timedelta(days = 5, hours = 1, seconds = 33, microseconds = 233423)
print("total seconds =", t.total_seconds())
```

```
total seconds = 435633.233423
```

Możesz również wykonać sumę dwóch dat i godzin za pomocą operatora `+` *. Możesz także pomnożyć i podzielić obiekt timedelta przez liczby całkowite i zmiennoprzecinkowe.*

Formatowanie daty.

Sposób przedstaw ienia daty i godziny może być różny w różnych miejscach, organizacjach itp. W Stanach Zjednoczonych używ a się częściej `mm/dd/rrrr` , podczas gdy w Polsce częściej używ a się `dd/mm/rrrr` . Python ma do tego metody `strftime()` i `strptime()` .

Przykład 11

Metoda ***`strftime()`***

In [21]:
```
from datetime import datetime

# current date and time
now = datetime.now()

t = now.strftime("%H:%M:%S")
print("time:", t)

s1 = now.strftime("%m/%d/%Y, %H:%M:%S")
# mm/dd/YY H:M:S format
print("s1:", s1)

s2 = now.strftime("%d/%m/%Y, %H:%M:%S")
# dd/mm/YY H:M:S format
print("s2:", s2)
```

```
time: 14:31:03
s1: 08/26/2020, 14:31:03
s2: 26/08/2020, 14:31:03
```

Przykład 12

Metoda `strptime()`

Zamiana tekstu na obiekt daty

In [27]:
```
from datetime import datetime

date_string = "21 June, 2018"
print("date_string =", date_string)

date_object = datetime.strptime(date_string, "%d %B, %Y")
print("date_object =", date_object)
```

```
date_string = 21 June, 2018
date_object = 2018-06-21 00:00:00
```

Przykład 13

Wyznaczanie stref czasowych przy przy pomocy biblioteki `pytz`

In [28]:
```
! python -m pip install pytz

from datetime import datetime
import pytz

local = datetime.now()
print("Local:", local.strftime("%m/%d/%Y, %H:%M:%S"))

tz_NY = pytz.timezone('America/New_York')
datetime_NY = datetime.now(tz_NY)
print("NY:", datetime_NY.strftime("%m/%d/%Y, %H:%M:%S"))

tz_London = pytz.timezone('Europe/London')
datetime_London = datetime.now(tz_London)
print("London:", datetime_London.strftime("%m/%d/%Y, %H:%M:%S"))
```

```
Collecting pytz
  Using cached pytz-2020.1-py2.py3-none-any.whl (510 kB)
Installing collected packages: pytz
Successfully installed pytz-2020.1
Local: 08/26/2020, 14:37:12
NY: 08/26/2020, 08:37:13
London: 08/26/2020, 13:37:13
```

Przykład 14

Zamiana daty na tekst. `datetime`

In [30]:
```
from datetime import datetime

now = datetime.now() # current date and time

year = now.strftime("%Y")
print("year:", year)

month = now.strftime("%m")
print("month:", month)

day = now.strftime("%d")
print("day:", day)

time = now.strftime("%H:%M:%S")
print("time:", time)

date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
print("date and time:",date_time)

print(type(date_time))
```

```
year: 2020
month: 08
day: 26
time: 14:40:36
date and time: 08/26/2020, 14:40:36
<class 'str'>
```

Przykład 15

Tworzenie daty w formie tekstowej z `timestamp` Metoda `fromtimestamp()`

In [31]:
```
from datetime import datetime

timestamp = 1528797322
date_time = datetime.fromtimestamp(timestamp)

print("Date time object:", date_time)

d = date_time.strftime("%m/%d/%Y, %H:%M:%S")
print("Output 2:", d)

d = date_time.strftime("%d %b, %Y")
print("Output 3:", d)

d = date_time.strftime("%d %B, %Y")
print("Output 4:", d)

d = date_time.strftime("%I%p")
print("Output 5:", d)
```

```
Date time object: 2018-06-12 11:55:22
Output 2: 06/12/2018, 11:55:22
Output 3: 12 Jun, 2018
Output 4: 12 June, 2018
Output 5: 11AM
```

In [ ]:
```
# %a    Skrócona nazwa dnia tygodnia.   Nie, pon, ...
# %A    Pełna nazwa dnia tygodnia.      Niedziela poniedziałek, ...
# %w    Dzień tygodnia jako liczba dziesiętna.  0, 1, ..., 6
# %d    Dzień miesiąca jako ułamek dziesiętny z wypełnieniem zerowym.   0
1, 02, ..., 31
# %-d   Dzień miesiąca jako liczba dziesiętna.  1, 2, ..., 30
# %b    Skrócona nazwa miesiąca.        Jan, luty, ..., grudzień
# %B    Pełna nazwa miesiąca.   Styczeń luty, ...
# %m    Miesiąc jako liczba dziesiętna z zerami.        01, 02, ..., 12
# %-m   Miesiąc jako liczba dziesiętna. 1, 2, ..., 12
# %y    Rok bez wieku jako liczba dziesiętna z uzupełnieniem zerowym.   0
0, 01, ..., 99
# %-y   Rok bez wieku jako liczba dziesiętna.   0, 1, ..., 99
# %Y    Rok z podaniem wieku jako liczby dziesiętnej.   2013, 2019 itd.
# %H    Godzina (zegar 24-godzinny) jako liczba dziesiętna z zerami.    0
0, 01, ..., 23
# %-H   Godzina (zegar 24-godzinny) jako liczba dziesiętna.     0, 1, ...,
23
# %I    Godzina (zegar 12-godzinny) jako liczba dziesiętna z zerami.    0
1, 02, ..., 12
# %-I   Godzina (zegar 12-godzinny) jako liczba dziesiętna.     1, 2, ...
12
# %p    Lokalne AM lub PM.      AM, PM
# %M    Minuta jako liczba dziesiętna z zerami. 00, 01, ..., 59
# %-M   Minuta jako liczba dziesiętna.  0, 1, ..., 59
# %S    Po drugie jako liczba dziesiętna z zerami.      00, 01, ..., 59
# %-S   Drugi jako liczba dziesiętna.   0, 1, ..., 59
# %f    Mikrosekunda jako liczba dziesiętna, dopełniona zerami po lewej st
ronie.  000000 - 999999
# %z    Przesunięcie czasu UTC w postaci + GGMM lub -GGMM.
# %Z    Nazwa strefy czasowej.
# %j    Dzień roku jako liczba dziesiętna z zerami.     001, 002, ..., 366
# %-j   Dzień roku jako liczba dziesiętna.      1, 2, ..., 366
# %U    Numer tygodnia w roku (niedziela jako pierwszy dzień tygodnia). Ws
zystkie dni w nowym roku poprzedzające pierwszą niedzielę są uważane za ty
dzień 0.        00, 01, ..., 53
# %W    Numer tygodnia w roku (poniedziałek jako pierwszy dzień tygodnia).
Wszystkie dni w nowym roku poprzedzające pierwszy poniedziałek przypadają
na tydzień 0.   00, 01, ..., 53
# %c    Odpowiednia reprezentacja daty i czasu w ustawieniach regionalnyc
h.      Poniedziałek, 30 września, 07:06:05 2013
# %x    Właściwa reprezentacja daty dla lokalizacji.    30.09.13
# %X    Odpowiednia reprezentacja czasu w języku lokalnym.      07:06:05
# %%    Dosłowny znak „%”.      %
```

Przykład 16

In [ ]:
```
from datetime import datetime

timestamp = 1528797322
date_time = datetime.fromtimestamp(timestamp)

d = date_time.strftime("%c")
print("Output 1:", d)

d = date_time.strftime("%x")
print("Output 2:", d)

d = date_time.strftime("%X")
print("Output 3:", d)
```