Манипуляция датой и временем

«Засыпание» скрипта

Функция sleep(<Время в секундах>) из модуля time прерывает выполнение скрипта на указанное время, по истечении которого скрипт продолжит работу. В качестве параметра можно указать целое или вещественное число.

```
import time
print("Один!")
time.sleep(4)
print("Два!")
```

Модуль **datetime**:
манипуляции датой и временем

Модуль **datetime**

Модуль **datetime** позволяет манипулировать датой и временем: выполнять арифметические операции, сравнивать даты, выводить дату и время в различных форматах и др. Прежде чем использовать классы из этого модуля, необходимо подключить модуль с помощью инструкции:

```
import datetime
```

Модуль содержит пять классов:

timedelta — дата в виде количества дней, секунд и микросекунд. Экземпляр этого класса можно складывать с экземплярами классов date и datetime. Кроме того, результат вычитания двух дат будет экземпляром класса timedelta;

date — представление даты в виде объекта;

time — представление времени в виде объекта;

datetime — представление комбинации даты и времени в виде объекта;

tzinfo — абстрактный класс, отвечающий за зону времени. За подробной информацией по этому классу обращайтесь к документации по модулю datetime.

Класс **timedelta**

Класс timedelta из модуля datetime позволяет выполнять операции над датами: складывать, вычитать, сравнивать и др. Конструктор класса имеет следующий формат:

timedelta(days, seconds, microseconds, milliseconds, minutes, hours, weeks)

Все параметры не являются обязательными и по умолчанию имеют значение 0.

```
import datetime

d = datetime.timedelta(hours=1, days=2, milliseconds=1)

print("Время:",d)
```

Первые три параметра считаются основными:

days — дни (диапазон -999999999 <= days <= 999999999);

seconds — секунды (диапазон 0 <= seconds < 3600*24);

microseconds — микросекунды (диапазон 0 <= microseconds < 1000000).

Все остальные параметры автоматически преобразуются в следующие значения:

milliseconds — миллисекунды (одна миллисекунда преобразуется в 1000 микросекунд)

minutes — минуты (одна минута преобразуется в 60 секунд)

hours — часы (один час преобразуется в 3600 секунд)

weeks — недели (одна неделя преобразуется в 7 дней):

Над экземплярами класса timedelta можно производить арифметические операции **+, -, /, //, %** и ***, использовать унарные операторы **+** и **-**, а также получать абсолютное значение с помощью функции **abs()**.

```
import datetime


d1 = datetime.timedelta(days=2)
d2 = datetime.timedelta(days=7)


print('Разность дней = ',  d1 - d2, "Модуль = ", abs(d1-d2))
print('Сумма дней:', d1 + d2)
print('Деление дней:', d2 / d1)
print('Умножение дней:', d2 * 2)
```

Кроме того, можно использовать логические операторы сравнения

==, !=, <, <=, > и >= :

```
import datetime

d1 = datetime.timedelta(days=2)
d2 = datetime.timedelta(days=7)

print('d1 > d2 ? :',  d1 > d2 )
print('d1 < d2', d1 < d2)
print('d1 == 2', d1 == 2)
```

Также можно получать строковое представление экземпляра класса timedelta с помощью
функций str() и repr():

```
import datetime


d = datetime.timedelta(hours = 25, minutes = 5, seconds = 27)
print(str(d))
print(repr(d))
```

```
 1 day, 1:05:27
datetime.timedelta(days=1, seconds=3927)
```

Еще поддерживаются следующие атрибуты класса:

min — минимальное значение, которое может иметь объект timedelta;

max — максимальное значение, которое может иметь объект timedelta;

resolution — минимальное возможное различие между значениями timedelta.

```
print("min =", datetime.timedelta.min)

print("max =", datetime.timedelta.max)

print("resolution =", datetime.timedelta.resolution)
```

```
min = -999999999 days, 0:00:00
max = 999999999 days, 23:59:59.999999
resolution = 0:00:00.000001
```

Класс **Date**

Класс **date**

Класс **date** из модуля datetime позволяет выполнять операции над датами.
Конструктор класса имеет следующий формат:

date(Год, Месяц, День)

Все параметры являются обязательными. В параметрах можно указать следующие диапазоны значений:

Год — в виде числа, расположенного в диапазоне между значениями, хранящимися в константах MINYEAR и MAXYEAR класса datetime (о нем речь пойдет позже).
Месяц — от 1 до 12 включительно;
День — от 1 до количества дней в месяце.

Если значения выходят за диапазон, возбуждается исключение **ValueError**

Для создания экземпляра класса date также можно воспользоваться следующими методами
этого класса:

today() — возвращает текущую дату;

fromtimestamp(Количество секунд) — возвращает дату, соответствующую количеству секунд, прошедших с начала эпохи;

fromordinal(Количество дней с 1-го года) — возвращает дату, соответствующую количеству дней, прошедших с первого года. В качестве параметра указывается число от 1 до datetime.date.max.toordinal()

```
print("Сегодня: ",datetime.date.today())
print("1503333334 = ",datetime.date.fromtimestamp(1503333334))
print(datetime.date.fromordinal(365205))
```

Получить значение объекта date можно с помощью следующих атрибутов:

year — год (число в диапазоне от MINYEAR до MAXYEAR);

month — месяц (число от 1 до 12);

day — день (число от 1 до количества дней в месяце).

Пример:

```
d = datetime.date.today()
print("Дата: ",d.year, d.month, d.day)
```

Над экземплярами класса date можно производить следующие операции:
date2 = date1 + timedelta — прибавляет к дате указанный период в днях.

date2 = date1 — timedelta — вычитает из даты указанный период в днях.

timedelta = date1 — date2 — возвращает разницу между датами (период в днях).
Значения атрибутов timedelta.seconds и timedelta.microseconds игнорируются;

```
d = datetime.date.today()
print("Дата: ", d )
print("Дата + 3 дня: ", d + datetime.timedelta(3))
```

Можно также сравнивать две даты с помощью операторов сравнения:

==, !=, <, <=, > и >= :

```
d1 = datetime.date.today()
d2 = d1 + datetime.timedelta(3)
print("d1 > d2 :", d1 > d2 )
print("d1 == d2 :", d1 == d2 )
```

Класс date поддерживает следующие методы:

replace(year, month, day) — возвращает дату с обновленными значениями. Значения можно указывать через запятую в порядке следования параметров или присвоить значение названию параметра. Все параметры не обязательные.

```
d = datetime.date(2017, 11, 21)
print(d)
print( d.replace(2016, 12))
print(d.replace(year=2018, month=1, day=31))
print(d.replace(day=30))
```

Класс date поддерживает следующие методы:

strftime(Строка формата) — возвращает отформатированную строку. В строке формата можно задавать комбинации специальных символов, которые используются в функции strftime() из модуля time.

```
d = datetime.date(2017, 11, 21)
print(d.strftime("%d.%m.%Y"))
```

Класс date поддерживает следующие методы:

isoformat() — возвращает дату в формате ГГГГ-ММ-ДД:

```
d = datetime.date(2017, 11, 21)
print(d.isoformat())
```

Класс date поддерживает следующие методы:

ctime() — возвращает строку формата "%a %b %d %H:%M:%S %Y":

```
d = datetime.date(2017, 11, 21)
print(d.ctime())
```

Класс date поддерживает следующие методы:

timetuple() — возвращает объект struct_time с датой и временем:

```
d = datetime.date(2017, 11, 21)
print(d.timetuple())
```

Класс date поддерживает следующие методы:

toordinal() — возвращает количество дней, прошедших с 1-го года:

```
d = datetime.date(2017, 11, 21)
print(d.toordinal())
```

Класс date поддерживает следующие методы:

weekday() — возвращает порядковый номер дня в неделе (0 — для понедельника, 6 — для воскресенья)

```
d = datetime.date(2017, 11, 21)
print(d.weekday())
```

Класс date поддерживает следующие методы:

isoweekday() — возвращает порядковый номер дня в неделе (1 — для понедельника, 7 — для воскресенья):

```
d = datetime.date(2017, 11, 21)
print(d.isoweekday())
```

Класс date поддерживает следующие методы:

isocalendar() — возвращает кортеж из трех элементов (год, номер недели в году и порядковый номер дня в неделе):

```
d = datetime.date(2017, 11, 21)
print(d.isocalendar())
```

Класс date поддерживает следующие методы:

min — минимально возможное значение даты;
max — максимально возможное значение даты;
resolution — минимальное возможное различие между значениями даты

```
print(datetime.date.min)

print( datetime.date.max)

print(datetime.date.resolution)
```