

Тема урока: модуль time

1. Модуль `time`
2. Измерение времени выполнения программы

Аннотация. Урок посвящен модулю `time`, который позволяет удобным образом работать со временем.

Модуль time

В Python помимо встроенного модуля `datetime` есть еще модуль `time`, который обычно используется при работе с текущим временем.

Работа функций модуля `time` основывается на общепринятой системе описания времени. Согласно ее концепции, текущее время представляется в виде обыкновенного вещественного значения в секундах, прошедших с момента начала эпохи и до сегодняшнего дня. С тех пор это число постоянно растет, позволяя людям работать с различными видами дат в максимально точном представлении.



Начало эпохи — это полночь 1 января 1970 года (00:00:00 UTC), когда счетчик секунд имел полностью нулевое значение.

Модуль `time` из стандартной библиотеки языка Python содержит массу полезных функций для работы со временем. С его помощью можно получать информацию о текущих дате и времени, выводить эти сведения в необходимом формате, а также управлять ходом выполнения программы, добавляя задержки по таймеру.

Модуль `time` предоставляет только функции, позволяющие работать со временем.

Использование модуля `time` дает возможность:

- отображать информацию о времени, прошедшем с начала эпохи
- преобразовывать значение системного времени к удобному виду
- прерывать выполнение программы (установка паузы) на заданное количество секунд
- измерять время выполнения программы целиком или ее отдельных модулей



Обратите внимание, если требуется сравнивать или производить арифметические операции со временем, то нужно использовать модуль `datetime`, а не `time`.

Функция time()

Для того чтобы получить количество секунд, прошедших с момента начала эпохи, необходимо использовать одноименную функцию `time()` из модуля `time`.

Приведенный ниже код:

```
import time

seconds = time.time()    # получаем количество прошедших секунд в виде float числа
print('Количество секунд с начала эпохи =', seconds)
```

выводит (на момент запуска 31 августа 2021 года):

```
Количество секунд с начала эпохи = 1630387918.354396
```

Таким образом, с 1 января 1970 года прошло уже более 1.6 миллиардов секунд.



В Python 3.7 добавили функцию `time_ns()`, которая возвращает целочисленное значение, представляющее то же время, прошедшее с эпохи, но в наносекундах, а не в секундах.

Функция ctime()

Представление времени на основе количества прошедших секунд с момента начала эпохи не очень удобно для человека (хотя и удобно для компьютера). Для того чтобы получить текущую дату в более удобном для человека виде, нужно использовать функцию `ctime()`.

Функция `ctime()` принимает в качестве аргумента количество секунд, прошедших с начала эпохи, и возвращает строку, представляющую собой **местное (локальное) время**.



Представление времени в зависимости от вашего физического местоположения называется местным (локальным) временем и использует концепцию часовых поясов.

Приведенный ниже код:

```
import time

seconds = 1630387918.354396
local_time = time.ctime(seconds)

print('Местное время:', local_time)
```

ВЫВОДИТ:

```
Местное время: Tue Aug 31 08:31:58 2021
```

Таким образом, функция `ctime()` возвращает **строковое представление** о местном (локальном) времени, которое включает в себя:

- день недели: Tue (Tuesday)
- название месяца: Aug (August)
- день месяца: 31
- часы, минуты, секунды: 08:31:58
- год: 2021

Вызывать функцию `ctime()` можно и без аргументов, в этом случае в качестве аргумента подставляется значение вызова функции `time()`. Таким образом, приведенный ниже код:

```
import time

local_time = time.ctime()          # вызов функции без аргумента
print('Местное время:', local_time)
```

равнозначен коду:

```
import time

seconds = time.time()
local_time = time.ctime(seconds)
print('Местное время:', local_time)
```

Обратите внимание на то, что результат работы функции `ctime()` зависит от вашего географического положения.

Функция `sleep()`

Функция `sleep()` используется для добавления задержки в выполнении программы. Эта функция принимает в качестве аргумента количество секунд (`secs`) и добавляет задержку в выполнении программы на указанное количество секунд.

Рассмотрим программный код:

```
import time

print('Before the sleep statement')
time.sleep(3)
print('After the sleep statement')
```

Если вы запустите приведенный выше код, то увидите, что вторая печать выполняется примерно через 3 секунды.

Аргумент `secs` может быть числом с плавающей точкой (`float`), для указания более точного времени приостановки. Например, мы можем сделать задержку на 700 миллисекунд, что составляет 0.7 секунды, как показано ниже:

```
import time

print('Before the sleep statement')
time.sleep(0.7)
print('After the sleep statement')
```



Время приостановки может быть дольше, чем запрошено, на произвольную величину из-за планирования других действий в операционной системе.

Иногда может потребоваться задержка на разное количество секунд. Сделать это можно следующим образом:

```
import time

for i in [0.7, 0.5, 1.0, 2.5, 3.3]:
    print(f'Waiting for {i} seconds')
    time.sleep(i)
print('The end')
```

Такая программа будет выполняться примерно $0.7 + 0.5 + 1.0 + 2.5 + 3.3 = 8.0$ секунд.



Функция `sleep()` нередко используется для тестирования кода и намеренного внесения задержек на различных этапах выполнения программы.

Примечания

Примечание 1. Документация по модулю `time` на английском языке доступна по [ссылке](#).

Примечание 2. Документация по модулю `time` на русском языке доступна по [ссылке](#).

Примечание 3. Понятие часового пояса зависит от нашего физического местоположения, однако вы можете изменить его в настройках компьютера без фактического перемещения.

Примечание 4. Поскольку местное время связано с нашим языковым стандартом, временные метки часто учитывают специфические для локали детали, такие как порядок элементов в строке и перевод сокращений дня и месяца. Функция `ctime()` игнорирует эти детали.

Примечание 5. Обратите внимание на то, что функция `sleep()` фактически останавливает выполнение только текущего потока, а не всей программы. О потоках можно почитать [тут](#).

Примечание 6. Хорошая статья о модуле `time` доступна по [ссылке](#).