Тип данных struct_time

В модуле `time` имеется единственный тип данных, который называется `struct_time`. Данный тип является именованным кортежем, представляющий информацию о времени. Структура представления времени `struct_time` чем-то похожа на тип `datetime`, который изучался ранее.

Именованные кортежи будут изучаться позже в рамках этого курса. Именованные кортежи подобны обычным кортежам за тем исключением, что к их полям можно обращаться не только по индексу, но и по названию.

Именованный кортеж `struct_time` состоит из следующих атрибутов:

Номер индекса	Атрибут	Значение
0	`tm_year`	диапазон от 0000 до 9999
1	`tm_mon`	диапазон от 1 до 12
2	`tm_mday`	диапазон от 1 до 31
3	`tm_hour`	диапазон от 0 до 23
4	`tm_min`	диапазон от 0 до 59
5	`tm_sec`	диапазон от 0 до 61
6	`tm_wday`	диапазон от 0 до 6, понедельник = 0
7	`tm_yday`	диапазон от 1 до 366
8	`tm_isdst`	значения -1, 0, 1
N/A	`tm_zone`	сокращение названия часового пояса
N/A	`tm_gmtoff`	смещение к востоку от UTC в секундах

Создавать объекты типа `struct_time` можно на основе кортежа:

```
import time

time_tuple = (2021, 8, 31, 5, 31, 58, 1, 243, 0)
time_obj = time.struct_time(time_tuple)
```

На практике редко приходится собственноручно создавать объекты типа `struct_time`. Обычно используют функции модуля `time`, которые сами создают и оперируют ими. Такие функции как `localtime()`, `gmtime()`, `asctime()` и другие,

принимают объект `time.struct_time` в качестве аргумента или возвращают его.

Функция localtime()

Функция `localtime()` принимает в качестве аргумента количество секунд, прошедших с начала эпохи, и возвращает кортеж `struct_time` в **локальном времени**.

 Если функции `localtime()` передан аргумент `None`, то вернется значение `time()`.

Приведенный ниже код:

```
import time

result = time.localtime(1630387918)
print('Результат:', result)
print('Год:', result.tm_year)
print('Месяц:', result.tm_mon)
print('День:', result.tm_mday)
print('Час:', result.tm_hour)
```

выводит:

```
Результат: time.struct_time(tm_year=2021, tm_mon=8, tm_mday=31, tm_hour=8,
tm_min=31, tm_sec=58, tm_wday=1, tm_yday=243, tm_isdst=0)
Год: 2021
Месяц: 8
День: 31
Час: 8
```

Обратите внимание на то, что мы можем обращаться к данным именованного кортежа `struct_time` и по индексам.

Приведенный ниже код:

```
import time

result = time.localtime(1630387918)
print('Результат:', result)
print('Год:', result[0])
print('Месяц:', result[1])
print('День:', result[2])
print('Час:', result[3])
```

аналогичен коду выше, однако менее нагляден.

Функция gmtime()

Функция `gmtime()` принимает в качестве аргумента количество секунд, прошедших с начала эпохи, и возвращает кортеж `struct_time` **в UTC**.

 Если функции `gmtime()` передан аргумент `None`, то вернется значение `time()`.

Приведенный ниже код:

```
import time

result = time.gmtime(1630387918)
print('Результат:', result)
print('Год:', result.tm_year)
print('Месяц:', result.tm_mon)
print('День:', result.tm_mday)
print('Час:', result.tm_hour)
```

выводит:

```
Результат: time.struct_time(tm_year=2021, tm_mon=8, tm_mday=31, tm_hour=5,
tm_min=31, tm_sec=58, tm_wday=1, tm_yday=243, tm_isdst=0)
Год: 2021
Месяц: 8
День: 31
Час: 5
```

 Обратите внимание на разницу в часах. В Москве используется сдвиг UTC+3:00, поэтому количество часов в локальном времени на 3 больше, чем по UTC.

Функция mktime()

Функция `mktime()` принимает `struct_time` (или кортеж, содержащий 9 значений, относящихся к `struct_time`) в качестве аргумента и возвращает количество секунд, прошедших с начала эпохи, в местном времени.

Приведенный ниже код:

```
import time

time_tuple = (2021, 8, 31, 5, 31, 58, 1, 243, 0)
time_obj = time.mktime(time_tuple)
print('Локальное время в секундах:', time_obj)
```

выводит:

```
Локальное время в секундах: 1630377118.0
```

Функция `mktime()` является обратной к функции `localtime()`. Следующий пример показывает их связь:

```
import time

seconds = 1630377118

time_obj = time.localtime(seconds)            # возвращает struct_time
print(time_obj)

time_seconds = time.mktime(time_obj)          # возвращает секунды из
struct_time
print(time_seconds)
```

и выводит:

```
time.struct_time(tm_year=2021, tm_mon=8, tm_mday=31, tm_hour=5, tm_min=31,
tm_sec=58, tm_wday=1, tm_yday=243, tm_isdst=0)
1630377118.0
```

Когда кортеж с неправильной длиной или имеющий элементы неправильного типа передается функции, ожидающей `struct_time`, возникает ошибка `TypeError`.

Функция asctime()

Функция `asctime()` принимает `struct_time` (или кортеж, содержащий 9 значений, относящихся к `struct_time`) в качестве аргумента и возвращает строку, представляющую собой дату и время.

Приведенный ниже код:

```
import time

time_tuple = (2021, 8, 31, 5, 31, 58, 1, 243, 0)

result = time.asctime(time_tuple)
print('Результат:', result)
```

выводит:

```
Результат: Tue Aug 31 05:31:58 2021
```

Форматированный вывод

Функции `ctime()` и `asctime()` имеют практически одинаковый функционал, за тем исключением, что первая функция принимает количество прошедших от начала эпохи секунд, а вторая принимает `struct_time` (или соответствующий кортеж). Обе функции представляют время в более удобном виде, благодаря автоматическому форматированию.

Приведенный ниже код:

```
import time

seconds = 1530377118
time_tuple = (2021, 8, 31, 5, 31, 58, 1, 243, 0)

print(time.ctime(seconds))
print(time.asctime(time_tuple))
```

выводит:

```
Sat Jun 30 19:45:18 2018
Tue Aug 31 05:31:58 2021
```

С форматированием мы уже сталкивались при работе с типами данных `date, time, datetime`.

Автоматическое форматирование не всегда то, что нужно поскольку может показаться чересчур сложным для восприятия либо же недостаточно информативным. Именно поэтому функции `strftime()` и `strptime()` модуля `time` позволяют создавать свои уникальные типы форматирования.

Функция strftime()

Функция `strftime` принимает строку с некоторым набором правил для форматирования и объект `struct_time` (или соответствующий кортеж) в качестве аргументов и возвращает строку с датой в зависимости от использованного формата.

Приведенный ниже код:

```
import time

time_obj = time.localtime()
result = time.strftime('%d.%m.%Y, %H:%M:%S', time_obj)
print(result)
```

выводит (на момент создания урока):

```
01.09.2021, 08:50:32
```

Функция strptime()

Функция `strptime()` делает разбор строки в зависимости от использованного формата и возвращает объект `struct_time`.

Приведенный ниже код:

```
import time

time_string = '1 September, 2021'
result = time.strptime(time_string, '%d %B, %Y')
print(result)
```

выводит:

```
time.struct_time(tm_year=2021, tm_mon=9, tm_mday=1, tm_hour=0, tm_min=0,
tm_sec=0, tm_wday=2, tm_yday=244, tm_isdst=-1)
```

Обратите внимание, что строка `time_string` должна полностью соответствовать формату `%d %B, %Y`, в противном случае возникнет исключение `ValueError`.

Примечания

Примечание 1. Модуль `time` оперирует двумя основными типами объектов: `struct_time` и секундами с начала эпохи.

Примечание 2. Таблица для форматированного вывода:

Формат	Значение	Пример
`%a`	Сокращенное название дня недели	Sun, Mon, ..., Sat (en_US) Пн, Вт, ..., Вс (ru_RU)
`%A`	Полное название дня недели	Sunday, Monday, ..., Saturday (en_US) понедельник, ..., воскресенье (ru_RU)
`%w`	Номер дня недели [0, ..., 6]	0, 1, ..., 6 (0=воскресенье, 6=суббота)
`%d`	День месяца [01, ..., 31]	01, 02, ..., 31

Формат	Значение	Пример
`%b`	Сокращенное название месяца	Jan, Feb, ..., Dec (en_US); янв, ..., дек (ru_RU)
`%B`	Полное название месяца	January, February, ..., December (en_US); Январь, ..., Декабрь (ru_RU)
`%m`	Номер месяца [01, ...,12]	01, 02, ..., 12
`%y`	Год без века [00, ..., 99]	00, 01, ..., 99
`%Y`	Год с веком	0001, 0002, ..., 2013, 2014, ..., 9999 В Linux год выводится без ведущих нулей: 1, 2, ..., 2013, 2014, ..., 9999
`%H`	Час (24-часовой формат) [00, ..., 23]	00, 01, ..., 23
`%I`	Час (12-часовой формат) [01, ..., 12]	01, 02, ..., 12
`%p`	До полудня или после (при 12-часовом формате)	AM, PM (en_US)
`%M`	Число минут [00, ..., 59]	00, 01, ..., 59
`%S`	Число секунд [00, ..., 59]	00, 01, ..., 59
`%f`	Число микросекунд	000000, 000001, ..., 999999
`%z`	Разница с UTC в формате ±HHMM[SS[.ffffff]]	+0000, -0400, +1030, +063415, ...
`%Z`	Временная зона	UTC, EST, CST
`%j`	День года [001,366]	001, 002, ..., 366
`%U`	Номер недели в году (нулевая неделя начинается с воскр.) [00, ..., 53]	00, 01, ..., 53
`%W`	Номер недели в году (нулевая неделя начинается с пон.) [00, ..., 53]	00, 01, ..., 53

Формат	Значение	Пример
`%c`	Дата и время в текущей локали	Tue Aug 16 21:30:00 1988 (en_US); 03.01.2019 23:18:32 (ru_RU)
`%x`	Дата в текущей локали	08/16/88 (None); 08/16/1988 (en_US); 03.01.2019 (ru_RU)
`%X`	Время в текущей локали	21:30:00

Примечание 3. На практике, модуль `time` используется не так часто. В основном для приостановки программы с помощью функции `sleep()` и для измерения времени выполнения программы.