# Python: импортируемые типы данных | Я.Шпора

## Модуль decimal

Позволяет работать с дробями с заданной точностью. При импорте этого модуля появляется возможность использовать числовой тип данных `Decimal`.

При объявлении переменной типа `Decimal` дробное числовое значение передаётся в виде строки, в кавычках:

```
# Импорт типа данных Decimal из модуля decimal.
from decimal import Decimal

bank_account_1 = Decimal('100.01')  # Числовое значение - в кавычках!

# Уточнение типа данных.
print('Тип переменной bank_account_1:', type(bank_account_1))

bank_account_2 = Decimal('100.01')
bank_account_3 = Decimal('100.01')

bank_account_new = bank_account_1 + bank_account_2 + bank_account_3

print(bank_account_new)

# Вывод в терминал:
# Тип переменной bank_account_1: <class 'decimal.Decimal'>
# 300.03
```

Модуль *decimal* позволяет задать точность вычислений с помощью атрибута `getcontext().prec`:

```
from decimal import Decimal, getcontext

getcontext().prec = 5
result = Decimal('10.12345') + Decimal('0.00006')
```

```
print(result)

# Вывод в терминал: 10.124
```

При значении `getcontext().prec = 5` результат вычислений был сокращён до 5 значащих цифр.

## Дата и время

Для работы с датами и временем в Python используется модуль *datetime*.

**Тип данных `date`**

Предназначен для работы с датами. В значениях этого типа можно хранить год, месяц и день. Часы, минуты и секунды в этом типе данных не учитываются.

Чтобы объявить дату, нужно:

- импортировать тип данных `date` из модуля *datetime*;
- объявить переменную типа `date`, передав параметры через запятую: первым аргументом — год, вторым — месяц, третьим — день.

```
from datetime import date

deadline = date(2023, 11, 6)

print('Ваш дедлайн:', deadline)
print(type(deadline))

# Вывод в терминал:
# Ваш дедлайн: 2023-11-06
# <class 'datetime.date'>
```

## Тип данных `time`

Используется для работы с секундами, минутами и часами. В переменную этого типа не получится записать ни год, ни месяц, ни день.

При описании переменной первым аргументом можно передать час, вторым минуту, третьим секунду. Также можно передать количество микросекунд.

```
from datetime import time

meeting_time = time(9, 15, 00)

print('Презентация проекта состоится в', meeting_time)
print(type(meeting_time))

# Вывод в терминал:
# Презентация проекта состоится в 09:15:00
# <class 'datetime.time'>
```

## Тип данных `datetime`

Позволяет сохранить дату и время как единое значение. Порядок аргументов такой: год, месяц, день, часы, минуты, секунды и микросекунды.

```
from datetime import datetime

exact_deadline = datetime(2023, 11, 6, 9, 15, 00)

print('Ваш точный дедлайн:', exact_deadline)
print(type(exact_deadline))

# Вывод в терминал:
# Ваш точный дедлайн: 2023-11-06 09:15:00
# <class 'datetime.datetime'>
```

При необходимости из `datetime` можно получить типы `date` и `time`:

```
from datetime import datetime

exact_deadline = datetime(2023, 11, 6, 9, 15, 00)

# Из типа datetime получен тип date.
date_from_datetime = datetime.date(exact_deadline)
print('Ваш дедлайн:', date_from_datetime)

# Из типа datetime получен тип time.
```

```
time_from_datetime = datetime.time(exact_deadline)
print('Презентация проекта состоится в', time_from_datetime)

# Вывод в терминал:
# Ваш дедлайн: 2023-11-06
# Презентация проекта состоится в 09:15:00
```

# Работа со временем

С типами `datetime`, `date` можно выполнять арифметические операции вычитания, чтобы найти разницу во времени. Также можно сравнивать значения дат и времени.

## Разница во времени

Между объектами типа `date` можно определить разницу во времени:

```
from datetime import date

deadline = date(2023, 11, 6)
today = date(2023, 10, 12)
print('Дней до дедлайна:', deadline - today)

# Вывод в терминал: Дней до дедлайна: 25 days, 0:00:00
```

> Вычитать из значений типа `datetime` значения типа `date` и наоборот нельзя.

Чтобы узнать время до дедлайна вплоть до секунд, используется значение типа `datetime`:

```
from datetime import datetime, date

deadline = date(2023, 11, 6)
today = date(2023, 10, 12)

deadline_detail = datetime(2023, 11, 6, 9, 15, 00)
today_detail = datetime(2023, 10, 12, 22, 13, 55)
```

```
print('Разность дат:', deadline - today, type(deadline - today))
print('Разность дат и времени:', deadline_detail - today_detail, type(d
eadline_detail - today_detail))

# Вывод в терминал:
# Разность дат: 25 days, 0:00:00 <class 'datetime.timedelta'>
# Разность дат и времени: 24 days, 11:01:05 <class 'datetime.timedelt
a'>
```

## Промежуток времени: тип timedelta

Разность значений типов `date` или `datetime` возвращает значение типа `timedelta`. Этот тип хранит продолжительность некоего отрезка времени.

Значение с типом `timedelta` можно вычесть или прибавить к `date` или `datetime`:

```
from datetime import datetime

deadline_detail = datetime(2023, 11, 6, 9, 15, 00)
today_detail = datetime(2023, 10, 12, 22, 13, 55)

delta = deadline_detail - today_detail

print(delta)

# Выведется:
# 24 days, 11:01:05
```

С помощью `total_seconds()` можно преобразовать значение типа `timedelta` в секунды:

```
from datetime import datetime, timedelta

deadline_detail = datetime(2023, 11, 6, 13, 00, 00)
today_detail = datetime(2023, 10, 12, 22, 13, 55)

print(timedelta.total_seconds(deadline_detail - today_detail))
```

```
# Вывод в терминал: 2126765.0
```

## Сравнение времени

Значения даты и времени можно сравнивать:

```
from datetime import datetime

deadline = datetime(2023, 11, 6, 13, 00, 00)
today = datetime(2023, 10, 12)

print(deadline > today)

# Вывод в терминал: True
```

> При сравнении типы операндов должны совпадать: нельзя тип `datetime` сравнить с типом `date`.

## Замена значения даты

Чтобы изменить дату, час, месяц, год или другие значения в данных типа `datetime`, используется `replace()`:

```
from datetime import datetime

datetime1 = datetime(2023, 8, 13, 3, 0, 0)  # 13 августа 2023 года, 3 утр
datetime2 = datetime.replace(datetime1, year=2019)  # 13 августа 2019 год

print('Исходная дата:', datetime1)
print('Изменённая дата:', datetime2)

# Вывод в терминал:
# Исходная дата: 2023-08-13 03:00:00
# Изменённая дата: 2019-08-13 03:00:00
```

### Текущая дата и время

При работе с типом `datetime` можно получить текущую дату и время, а при работе с типом `date` — текущую дату:

```
from datetime import datetime, date

# Получить текущую дату и время.
now = datetime.now()
# Получить текущую дату.
today_1 = datetime.today()
today_2 = date.today()
```

## Форматирование времени

При выводе даты и времени на печать можно настроить свой формат. Для этого используется функция `strftime()`. Эта функция преобразует время в строку по заданному шаблону:

```
from datetime import datetime

deadline = datetime(2023, 11, 6, 13, 0, 0)
# Синтаксис: datetime.strftime(<дата>, 'шаблон_для_форматирования')
deadline_as_str = datetime.strftime(deadline, '%Y/%m/%d %H:%M')
print('Ваш дедлайн:', deadline_as_str)

# Вывод в терминал: Ваш дедлайн: 2023/11/06 13:00
```

Строка `%Y/%m/%d %H:%M` — это шаблон, где:

- `%Y` — год,
- `%m` — месяц,
- `%d` — день,
- `%H` — часы,
- `%M` — минуты.

В шаблоне можно указать любые текстовые символы, например разделители `/` и `.` или обозначения `ч.`, `мин.` и `сек.`.

Все значения, которые можно использовать в шаблоне, можно найти в официальной документации Python