GeekBrains

Обзор стандартной библиотеки Python

Урок 15

Содержание урока

План курса

1. Основы Python
2. Простые типы данных
3. Коллекции
4. Функции
5. Итераторы и генераторы
6. Модули
7. Файлы и файловая система
8. Сериализация
9. Декораторы
10. ООП. Начало
11. ООП. Особенности Python
12. ООП. Финал
13. Исключения
14. Основы тестирования
15. Обзор стандартной библиотеки Python

Что будет на уроке сегодня

- 📌 Узнаем о составе стандартной библиотеки Python
- 📌 Разберёмся в настройках логирования
- 📌 Изучим работу с датой и временем
- 📌 Узнаем ещё пару полезных структур данных
- 📌 Изучим способы парсинга аргументов при запуске скрипта с параметрами

Обзор библиотеки целиком

Обзор библиотеки целиком

The Python Standard Library

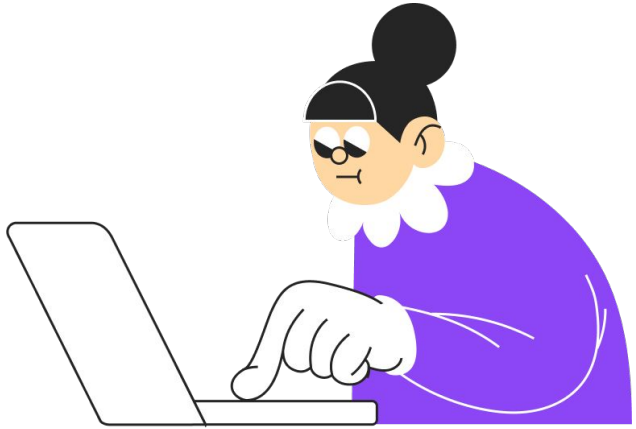
- Встроенные функции
- Встроенные константы
- Встроенные типы
- Встроенные исключения
- Услуги обработки текста
- Службы двоичных данных
- Типы данных
- Числовые и математические модули

Обзор библиотеки

- Модули функционального программирования
- Доступ к файлам и каталогам
- Сохранение данных
- Сжатие данных и архивирование
- Форматы файлов
- Криптографические услуги
- Общие службы операционной системы
- Параллельное выполнение
- Сеть и межпроцессное взаимодействие

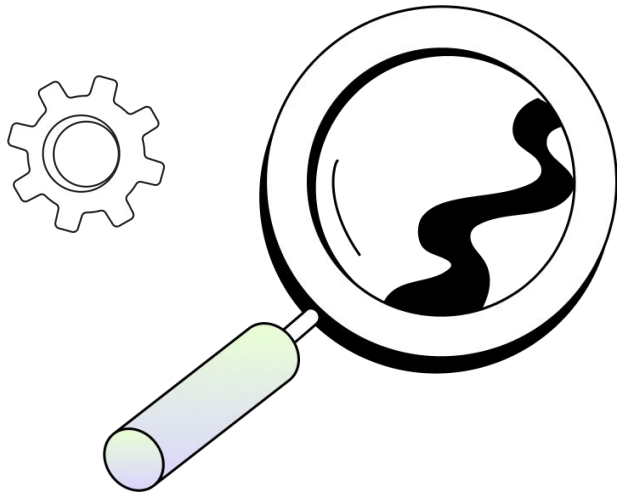
Обзор библиотеки

- Обработка данных в Интернете
- Инструменты обработки структурированной разметки
- Интернет-протоколы и поддержка
- Мультимедийные услуги
- Интернационализация
- Программные фреймворки
- Графические пользовательские интерфейсы с Tk
- Инструменты разработки



Обзор библиотеки

- Отладка и профилирование
- Упаковка и распространение программного обеспечения
- Службы выполнения Python
- Пользовательские интерпретаторы Python
- Импорт модулей
- Языковые службы Python
- Специальные службы MS Windows
- Специальные службы Unix
- Замененные (устаревшие) модули



Модуль logging

Уровни логирования

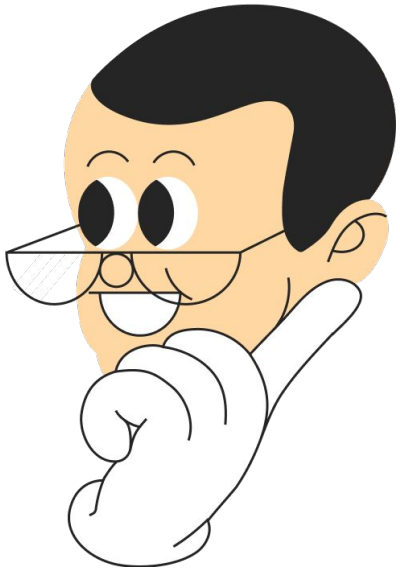
Уровень debug — отличная альтернатива функции print()

- **NOTSET, 0** — уровень не установлен. Регистрируются все события.
- **DEBUG, 10** — подробная информация, обычно представляющая интерес только при диагностике проблем.
- **INFO, 20** — подтверждение того, что все работает так, как ожидалось.
- **WARNING, 30** — указание на то, что произошло что-то неожиданное, или указание на какую-то проблему в ближайшем будущем (например, «недостаточно места на диске»). Программное обеспечение по-прежнему работает, как ожидалось.
- **ERROR, 40** — из-за более серьезной проблемы программное обеспечение не может выполнять некоторые функции.
- **CRITICAL, 50** — серьезная ошибка, указывающая на то, что сама программа не может продолжать работу.

Базовые регистраторы

В официальной документации указано, что работать с регистраторами напрямую запрещено.

```
import logging

logging.basicConfig(level=logging.NOTSET)
logger = logging.getLogger(__name__)
```



Подробнее о basicConfig

Уровень логирования

- level

Файл журнала

- filename
- filemode
- encoding
- errors

Формат сохранения события

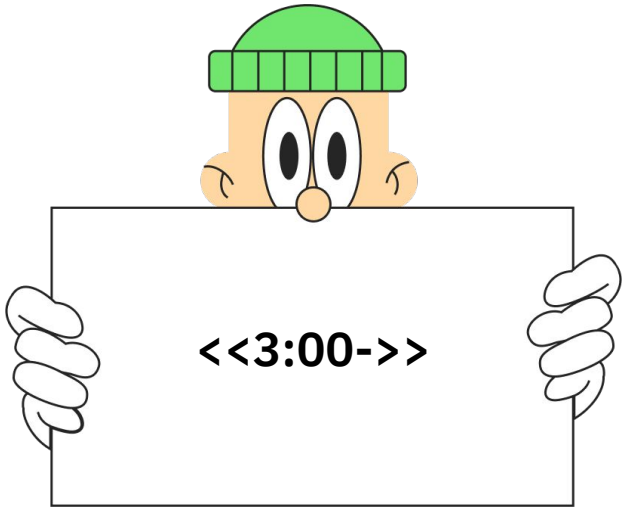
- “%”
- “{“
- “$”

Перед вами несколько
строк кода. Напишите в чат что они
вернут не запуская программу.

У вас 3 минуты.

Модуль logging

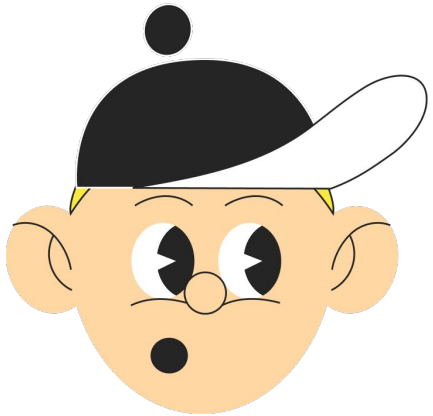
```
import logging

logging.basicConfig(
    filename="log/log.log",
    encoding='utf-8',
    format='{asctime} {levelname}
{funcName}->{lineno}: {msg}',
    style='{',
    level=logging.WARNING
)
```



Модуль datetime

Дата, время, дата-время, разница во времени

```
from datetime import time, date, datetime, timedelta


d = date(year=2007, month=6, day=15)

t = time(hour=2, minute=14, second=0, microsecond=24)

dt = datetime(year=2007, month=6, day=15, hour=2,
minute=14, second=0, microsecond=24)

delta = timedelta(weeks=1, days=2, hours=3, minutes=4,
seconds=5, milliseconds=6, microseconds=7)
```



Свойства и методы для работы с datetime

Для получения доступа к свойствам используется точечная нотация.

Вывод по формату:

`dt.timestamp()`

`dt.isoformat()`

`dt.strftime(FORMAT)`

Получение из текстового формата:

`datetime.fromtimestamp(text)`

`datetime.fromisoformat(number)`

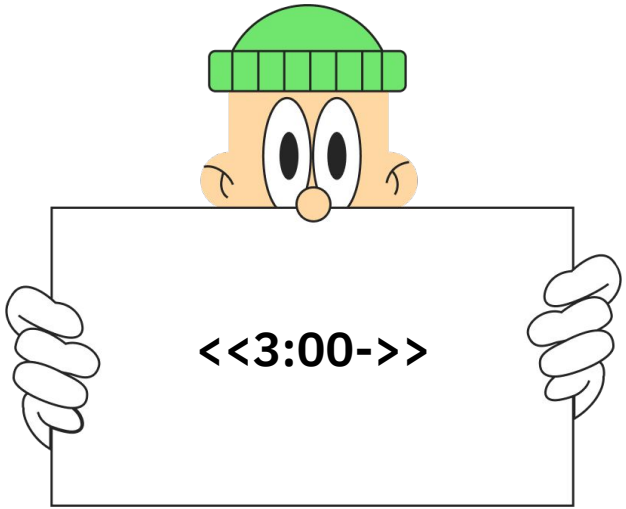
`datetime.strptime(text, FORMAT)`

Перед вами несколько
строк кода. Напишите в чат что они
вернут не запуская программу.

У вас 3 минуты.

Модуль datetime

```
from datetime import datetime, timedelta

d = datetime.now()
td = timedelta(hours=1)
for i in range(24*7):
    if d.weekday() == 6:
        break
    else:
        d = d + td
print(i)
```



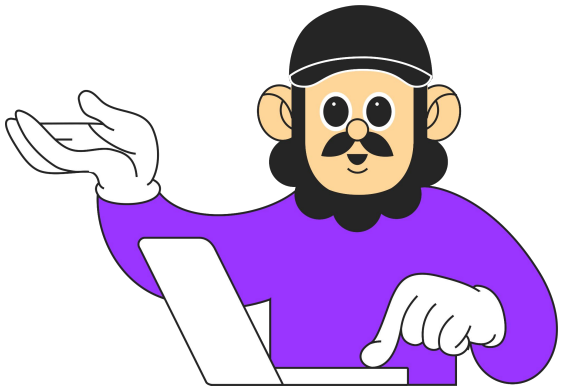
Пара полезных структур данных

Фабричная функция namedtuple из модуля collections

namedtuple принимает пару обязательных значений:

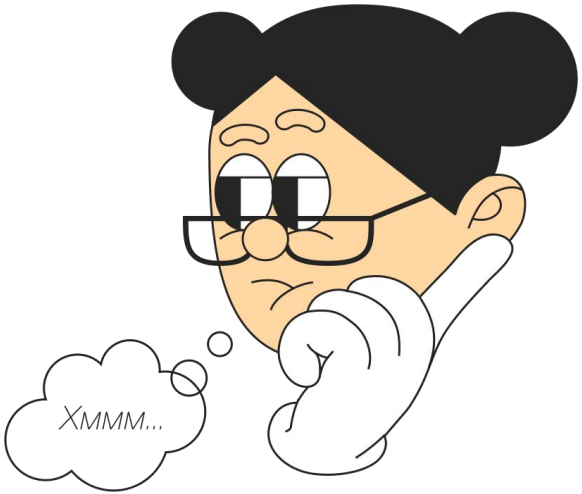
1. Имя класса. Это строка, содержащее точно такое же имя как и переменная слева от знака равно.

2. Список строк или строка с пробелами в качестве разделителей. Имена из списка превращаются в свойства класса.



Модуль array

- 'b' — целое со знаком, 1 байт
- 'B' — целое без знака, 1 байт
- 'u' — Юникод-символ в 2 или 4 байта
- 'h' — целое со знаком, 2 байта
- 'H' — целое без знака, 2 байта
- 'i' — целое со знаком, 4 байта
- 'I' — целое без знака, 4 байта
- 'l' — целое со знаком, 4 байта
- 'L' — целое без знака, 4 байта
- 'q' — целое со знаком, 8 байт
- 'Q' — целое без знака, 8 байт
- 'f' — вещественное обычной точности, 4 байта
- 'd' — вещественное двойной точности, 8 байт

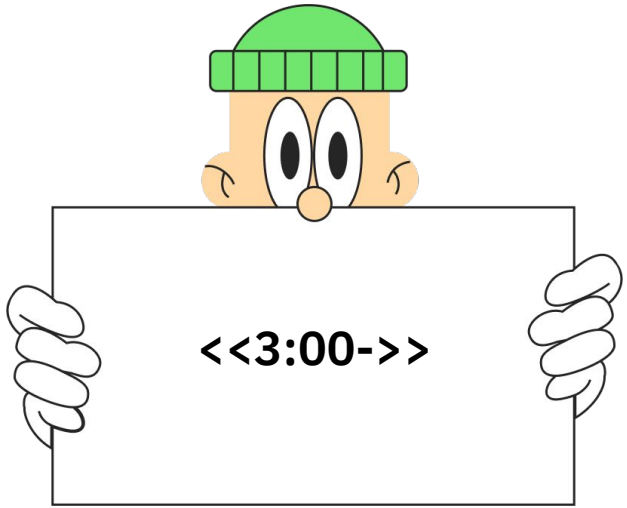


Перед вами несколько
строк кода. Напишите в чат что они
вернут не запуская программу.

У вас 3 минуты.

Пара полезных структур данных

```
from collections import namedtuple

Data = namedtuple('Data', ['mathematics', 'chemistry',
'physics', 'genetics'], defaults=[5, 5, 5, 5])

d = {
    'Ivanov': Data(4, 5, 3, 5),
    'Petrov': Data(physics=4, genetics=3),
    'Sidorov': Data(),
}
```

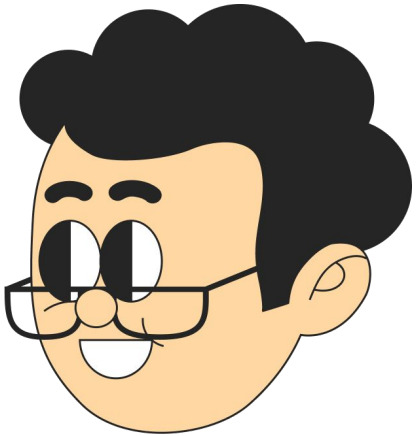


Модуль argparse

Модуль argparse

Ключ --help или -h добавляется по умолчанию

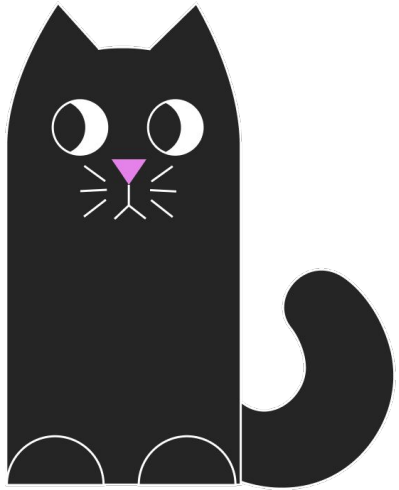
```
import argparse

parser = argparse.ArgumentParser(description='My first
argument parser')

parser.add_argument('numbers', metavar='N', type=float,
nargs='*', help='press some numbers')

args = parser.parse_args()
```



Создаём парсер, ArgumentParser

Большинство настроек отлично работают со значениями по умолчанию

- **prog** — заменяет название файла в первой строке справки на переданное имя,
- **description** — описание в начале справки
- **epilog** — описание в конце справки



Выгружаем результаты, parse_args

`parse_args` принимает два аргумента

Строка для анализа

Значение по умолчанию `sys.argv`

Объект для сохранения результата

Значение по умолчанию класс `Namespace`

Добавляем аргументы, add_argument

- **metavar** — имя, которое выводится с справке
- **type** — тип, для преобразования аргумента.
- **nargs** — указывает на количество значений, которые надо собрать из командной строки и собрать результат в список list.
- **help** — вывод подсказки об аргументе.
- **action** — принимает одно из строковых значений и срабатывает при наличии в строке вызова скрипта соответствующего параметра.
    - **store_const** — передаёт в args ключ со значением из параметра const
    - **store_true или store_false** — сохраняет в ключе истину или ложь
    - **append** — ищет несколько появлений ключа и собирает все значения после него в список
    - **append_const** — добавляет значение из ключа в список, если ключ вызван.

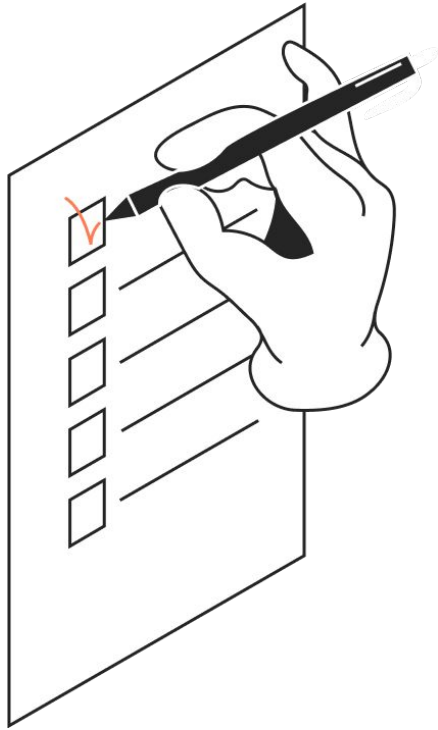
Итоги занятия

На этой лекции мы

- Узнали о составе стандартной библиотеки Python.
- Разобрались в настройках логирования
- Изучили работу с датой и временем
- Узнали ещё пару полезных структур данных
- Изучили способы парсинга аргументов при запуске скрипта с параметрами

Задание

Обратитесь к официальной документации по стандартной библиотеке Python. Выберите любой раздел по вашему желанию. Почитайте о модуле, попробуйте запустить примеры из документации.



Финальная лекция курса пройдена

Спасибо за внимание

Погружение в Python