

*School of  
Computer  
Science*

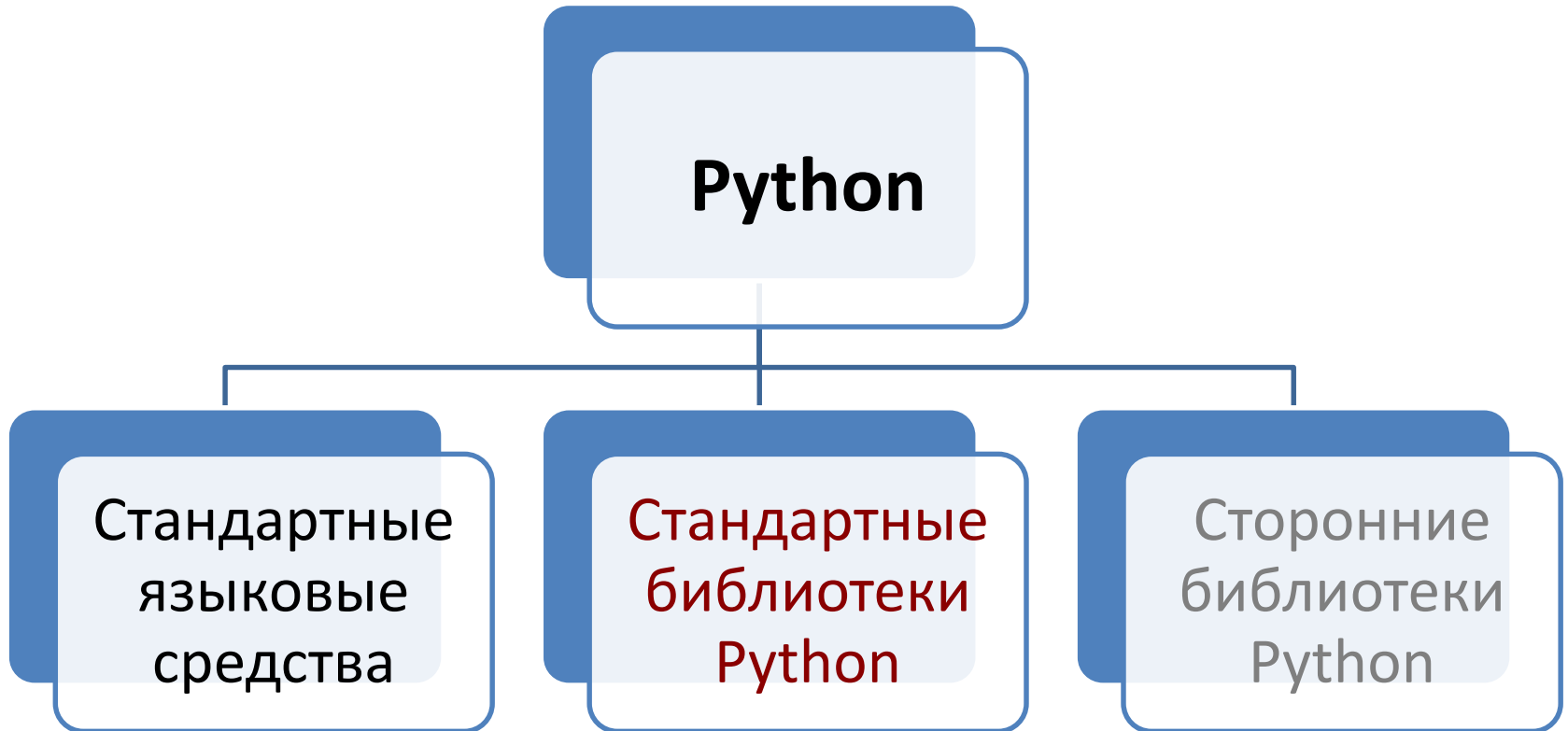
# СТАНДАРТНЫЕ БИБЛИОТЕКИ PYTHON

## ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы



# ДОСТУПНЫЕ СРЕДСТВА PYTHON. КЛАССИФИКАЦИЯ





# СТАНДАРТНЫЕ БИБЛИОТЕКИ PYTHON. СВОЙСТВА

- Стандартные библиотеки Python:
  - Предоставляют широкий спектр функций без необходимости установки пакетов
  - Включают средства для работы с разными типами данных, файлами, взаимодействия с ОС, поддержки сетевых протоколов и т.д.
- Базовая документация:  
<https://docs.python.org/3/library>
- Подключаются с помощью  
`import <имя модуля>`



# ДОПОЛНИТЕЛЬНЫЕ ТИПЫ ДАННЫХ

## РАБОТА СО СТРОКАМИ

Модуль	Описание и возможное использование
<b>string</b>	Сравнение со строковыми константами, например, digits, whitespace, punctuation
<b>re</b>	Поиск и замена текста с использованием регулярных выражений
<b>struct</b>	Работа с двоичными данными. Чтение и запись структурированных данных в файлы
<b>difflib</b>	Вычисление расхождений, поиск различий между строками и последовательностями, создание файлов различий и исправлений
<b>textwrap</b>	Перенос и заполнение текста, форматирование текста посредством разбиения строк или добавления пробелов



# ДОПОЛНИТЕЛЬНЫЕ ТИПЫ ДАННЫХ

## РАБОТА С ДАТОЙ, ВРЕМЕНЕМ. КОНСТАНТЫ

Модуль	Описание и возможное использование
<b>datetime</b>	Работа с датой и временем
<b>time</b>	Доступ к системному времени и преобразования времени
<b>calendar</b>	Работа с календарем
<b>enum</b>	Создание перечислений, связывающих символические имена с константами
<b>pprint</b>	Структурированная печать данных



# РАБОТА С ДАТАМИ И ВРЕМЕНЕМ.

## МОДУЛЬ DATETIME

- Используемые типы данных:
  - `date`, `datetime` и `time` – для хранения даты, даты и времени, только времени
  - `timedelta` – для выполнения арифметических действий с датами и временем
- Используемые методы:
  - `today()`, `now()`, `date()`, `time()` – получение дат или времени
  - `strptime()`, `strftime()` – форматирование дат по шаблону (см. <http://strftime.org>)
- Базовая документация:  
<https://docs.python.org/3/library/datetime.html>
- Пример использования: см. функцию `get_date()` в `console.py`



# ДОПОЛНИТЕЛЬНЫЕ ТИПЫ ДАННЫХ

## РАБОТА С КОЛЛЕКЦИЯМИ

Модуль	Описание и возможное использование
<b>collections</b>	Дополнительные контейнерные типы defaultdict, OrderedDict, namedtuple, Counter, deque, ...
<b>array</b>	Эффективная реализация типизированных массивов с числовыми элементами
<b>copy</b>	Операции глубокого и поверхностного копирования данных
<b>typing</b>	Поддержка разметки кода рекомендациями относительно типа объектов, особенно параметров функций и возвращаемых значений
<b>types</b>	Словари «только для чтения» (MappingProxyType) и атрибутный доступ по ключу в словаре (SimpleNamespace)



# КОЛЛЕКЦИОННЫЕ ТИПЫ ДАННЫХ.

## КАКИЕ ТИПЫ ДАННЫХ ИСПОЛЬЗОВАТЬ?

- Всего несколько полей – 2 или 3?
  - Порядок следования полей легко запоминается
  - Имена полей излишни
  - Используем обыкновенный кортеж
- Нужны неизменяемые поля?
- Нужно избавиться от индексов полей, т.к. их сложно запоминать?
  - Используем `collections.namedtuple` или `typing.NamedTuple`





# КОЛЛЕКЦИОННЫЕ ТИПЫ ДАННЫХ.

## КАКИЕ ТИПЫ ДАННЫХ ИСПОЛЬЗОВАТЬ?

- Не хотим усложнять?
  - Нужен удобный, «массиво-подобный» синтаксис
  - Нужна сериализация в JSON
    - Используем обыкновенный словарь
- Нужен полный контроль над собственной структурой данных?
  - Делаем собственный класс с методами доступа и `@property`



# КОЛЛЕКЦИОННЫЕ ТИПЫ ДАННЫХ.

## КАКИЕ ТИПЫ ДАННЫХ ИСПОЛЬЗОВАТЬ?

- Нужно добавить в объект поведение (методы)?
  - Написать собственный класс с методами или
  - Расширить `collections.namedtuple` / `typing.NamedTuple`
- Нужно плотно упаковать данные?
  - Важна передача данные по сети или компактное хранение на диске
  - Используем `struct.Struct`



# ДОПОЛНИТЕЛЬНЫЕ ТИПЫ ДАННЫХ

## ЧИСЛОВЫЕ И МАТЕМАТИЧЕСКИЕ МОДУЛИ

Модуль	Описание и возможное использование
<b>math,</b> <b>cmath</b>	Математические операции для работы с вещественными и комплексными числами
<b>decimal</b>	Точные операции для вещественных десятичных чисел
<b>statistics</b>	Функции для вычисления величин из области математической статистики
<b>fractions</b>	Рациональные числа
<b>random</b>	Генерация псевдослучайных чисел и вариантов
<b>functools</b>	Функции более высокого порядка и операции с вызываемыми объектами
<b>operator</b>	Стандартные операторы как функции



# РАБОТА С ФАЙЛАМИ В СТИЛЕ ОС

Модуль	Описание и возможное использование
<code>os.path</code>	Стандартные манипуляции с полными именами файлов
<code>pathlib</code>	Объектно-ориентированная работа с полными именами файлов
<code>fileinput</code>	Перебор строк из разных входных потоков
<code>filecmp</code>	Сравнение файлов и каталогов
<code>tempfile</code>	Генерирование временных файлов и каталогов
<code>glob</code> , <code>fnmatch</code>	Работа с путями и именами файлов с использованием шаблонов в стиле UNIX
<code>shutil</code>	Высокоуровневые операции с файлами



# РАБОТА С ФАЙЛАМИ

## ХРАНЕНИЕ, ОБРАБОТКА И РАЗНЫЕ ФОРМАТЫ

Модуль	Описание / Использование
<code>linecache</code>	Произвольный доступ к строкам из текстового файла с кэшированием
<code>pathlib</code>	Объектно-ориентированная работа с полными именами файлов
<code>pickle, shelve</code>	Сериализация и хранение объектов Python в бинарных файлах собственного формата
<code>sqlite3</code>	Работа с локальными базами данных SQLite
<code>zlib, gzip, bz2, zipfile, tarfile</code>	Работа с архивными файлами и сжатием данных
<code>csv</code>	Чтение и запись файлов в формате CSV
<code>configparser</code>	Разбор конфигурационных файлов; чтение/запись ini-файлов в стиле Windows





# РАБОТА С ОС

## СЕРВИСНЫЕ ФУНКЦИИ ОС

Модуль	Описание и возможное использование
<code>os</code>	Различные интерфейсы операционной системы
<code>io</code>	Основные средства для работы с потоками
<code>otpparse</code>	Мощный парсер параметров командой строки
<code>logging</code>	Средства журналирования событий
<code>platform</code>	Работа с идентификационными данными платформы
<code>ctypes</code>	Библиотека для работы с внешними функциями
<code>threading</code>	Высокоуровневый интерфейс программных потоков
<code>subprocess</code>	Управление подпроцессами



# ИНТЕРНЕТ ПРОТОКОЛЫ И ФОРМАТЫ

## СЕРВИСНЫЕ МОДУЛИ

Модуль	Описание и возможное использование
socket, ssl	Низкоуровневые сетевые интерфейсы и обертка SSL для объектов сокетов
email	Пакет для работы с электронной почтой и MIME
json	Кодирование и декодирование формата JSON
mailbox	Работа с почтовыми ящиками в разных форматах
html.parser	Разбор HTML
base64, binhex, binascii, uu	Кодирование / декодирование файлов или потоков с различными кодировками
http.server	Серверы HTTP



# СРЕДСТВА РАЗРАБОТКИ И ОТЛАДКИ

## ВСПОМОГАТЕЛЬНЫЕ МОДУЛИ ПРОГРАММИСТА

Модуль	Описание и возможное использование
<b>pydoc</b>	Генератор документации и электронной справки
<b>doctest</b>	Тестирование интерактивных примеров Python
<b>unittest</b>	Инфраструктура модульного тестирования
<b>pdb</b>	Консольный отладчик Python
<b>cProfile</b>	Профилировщик Python
<b>timeit</b>	Измерение времени выполнения небольших фрагментов кода
<b>sys</b>	Системные параметры и функции
<b>__future__</b>	Определения будущих команд – возможностей, которые будут добавлены в Python





# ПРОЕКТ «РОССИЙСКАЯ ИСТОРИЯ»

## DEADLINE = 16.12.2019

- Рассмотрите файл `russian_history.json` с описанием событий из истории России
- Загрузите данные из этого файла в Python и определите структуру полученных данных
- Напишите программу для проверки знаний Российской истории у школьников. Требования:
  - Вопросы задаются в случайном порядке – при каждом запуске разные и неповторяющиеся для сеанса работы
  - Количество задаваемых вопросов для каждого сеанса работы программы должно быть ограничено
  - Используйте свой механизм расчета результатов тестирования в баллах, исходя из структуры данных
- Для запроса даты исторических событий по шаблону необходимо использовать функцию `get_date()` из модуля `console.py`

**СПАСИБО ЗА ВНИМАНИЕ !**  
**ВОПРОСЫ ?**



*School of  
Computer  
Science*