

# Python Essential

ВВОД И ВЫВОД

# Python Essential

После урока обязательно



Повторите этот урок в видео формате на [ITVDN.com](http://itvdn.com)

Доступ можно получить через руководство вашего учебного центра



Проверьте как Вы усвоили данный материал на [TestProvider.com](http://testprovider.com)

# Python Essential

Тема

ВВОД И ВЫВОД

# ВВОД И ВЫВОД

## Понятие файлов и потоков



**Файл** – именованная область данных на носителе информации.

**Файловый объект (поток)** – объект, предоставляющий файл-ориентированный API (методы `read()`, `write()` и т.д.) для доступа к ресурсу. В зависимости от способа создания, файловый объект может предоставлять доступ к реальному файлу на диске или другому виду устройства хранения или передачи данных (стандартные потоки ввода/вывода, буферы в памяти, сокеты и т.д.).

Стандартные потоки:

- `sys.stdin`
- `sys.stdout`
- `sys.stderr`

# ВВОД И ВЫВОД

## Байтовые строки и Unicode-строки

	Python 2	Python 3
	В Python 2 класс str представляет собой строку как последовательность байтов, он же и используется в тех случаях, когда требуется представить в памяти последовательность байтов, которая не является строкой.	В Python 3 класс str представляет Unicode-строки, и именно Unicode является стандартной кодировкой для интерпретатора.
Байтовые строки	str	bytes
Unicode-строки	unicode	str

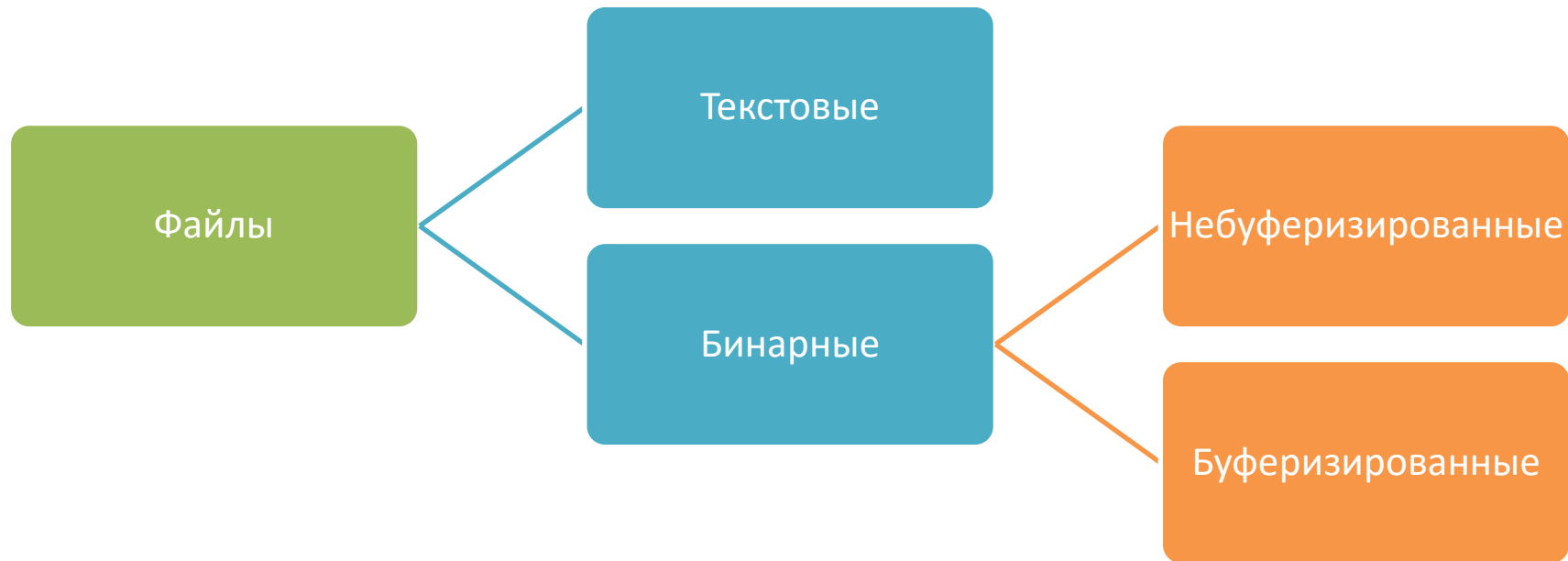
# ВВОД И ВЫВОД

## Изменяемые массивы байтов и символов

	Python 2	Python 3
Изменяемые массивы байтов		<code>bytearray</code>
Изменяемые массивы Unicode-символов		<code>array.array('u')</code>

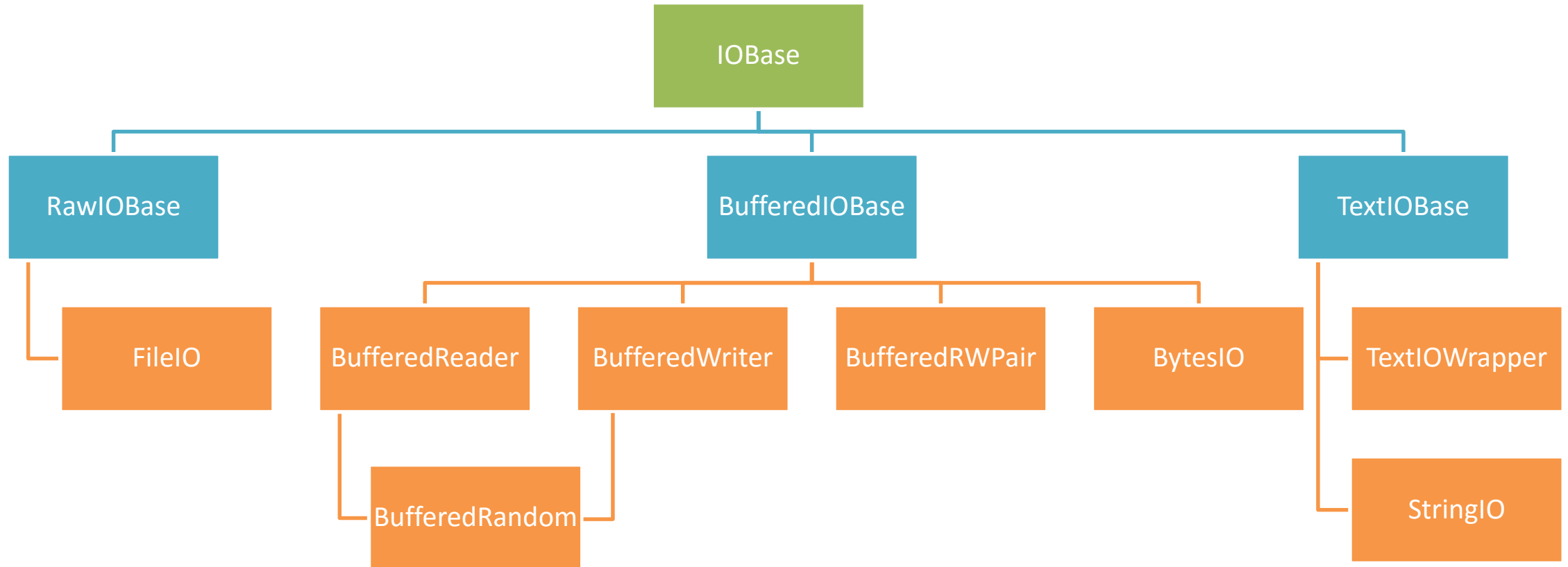
# ВВОД И ВЫВОД

## Виды файлов в Python



# ВВОД И ВЫВОД

## Иерархия классов модуля io





# ВВОД И ВЫВОД

## Открытие файла

Python 2 `open(file, mode='r', buffering=-1)`

Python 3 `open(file, mode='r', buffering=-1,  
encoding=None, errors=None,  
newline=None, closefd=True,  
opener=None)`

Основные параметры:

- `file` – имя файла или файловый дескриптор;
- `mode` – режим открытия файла;
- `encoding` – кодировка файла;
- `buffering` – использовать ли буферизацию и если да, каким должен быть размер буфера



**Чаще всего функция `open()` используется с двумя параметрами**

# ВВОД И ВЫВОД

## Режимы открытия файлов

	Символ	Значение
первый символ	'r'	открыть для чтения (по умолчанию)
	'w'	открыть для записи, удалив содержимое файла
	'x'	открыть для исключительного создания (ошибка, если файл существует)
	'a'	открыть для записи, добавляя содержимое в конец файла, если он существует
второй или третий символ	'b'	бинарный режим
	't'	текстовый режим (по умолчанию)
	'+'	открыть для чтения и записи



Порядок символов 'b'/'t' и '+' не имеет значения

# ВВОД И ВЫВОД

## Заккрытие файлов



После окончания работы с файлом следует обязательно его закрыть при помощи метода `close()`, особенно если он был открыт для записи. При использовании буферизированного вывода данные, которые записываются в файл, не попадают в него сразу, а записываются в буфер. Содержимое буфера записывается в файл при его заполнении или вызове методов `flush()` или `close()`. Кроме того, если файл открыт для записи, он будет заблокирован для открытия для записи другими процессами до момента закрытия. Все открытые файлы автоматически закрываются при удалении соответствующих файловых объектов из памяти сборщиком мусора интерпретатора Python и при завершении работы самого интерпретатора, однако следует держать файлы открытыми минимально требуемое время.

# ВВОД И ВЫВОД

## Оператор with



Файловые объекты являются менеджерами контекста.

**Менеджер контекста (context manager)** – это объект, который описывает определённый контекст, который устанавливается при выполнении оператора with. Менеджеры контекста используются для сохранения и восстановления глобального состояния, блокирования и разблокирования ресурсов, автоматического закрытия файлов и т.д.

Таким образом, при открытии файла таким образом:

```
with open(filename, mode) as file:
```

```
...
```

```
    # perform actions on file
```

```
...
```

он гарантированно будет закрыт, как только перестанет быть нужен.

# ВВОД И ВЫВОД

## Чтение файлов

Чтение одной строки текста	<code>file.readline()</code>
Чтение списка строк текста	<code>file.readlines()</code>
Построчное чтение	<code>for line in file:     pass</code>
Чтение заданного или максимально возможного количества данных (символов или байтов)	<code>file.read(100) file.read()</code>
Чтение данных бинарного файла в изменяемый массив	<code>file.readinto(arr)</code>

# ВВОД И ВЫВОД

## Запись в файлы

Запись строк текста	<code>file.writelines(lines)</code>
Запись данных (символов или байтов)	<code>file.write(data)</code>
Запись данных в текстовый файл при помощи функции или оператора print	<code>print(*args, file=text_file) # Python 3</code> <code>print &gt;&gt;text_file, arg1, arg2, ... # Python 2</code>

# ВВОД И ВЫВОД

## Управление позицией чтения/записи

Получение текущей позиции (отступ от начала файла в символах или байтах)	<code>file.tell()</code>
Установка позиции как смещения относительно начала файла	<code>file.seek(offset)</code> <code>file.seek(offset, o.SEEK_SET)</code>
Установка позиции как смещения от текущей	<code>file.seek(offset, io.SEEK_CUR)</code>
Установка позиции как смещения от конца файла	<code>file.seek(offset, io.SEEK_END)</code>

# ВВОД И ВЫВОД

## Сериализация объектов



**Сериализация** – это процесс сохранения объектов в двоичном или строковом виде для хранения, передачи и восстановления. Обратный процесс называется **десериализацией**.



# ВВОД И ВЫВОД

## Модуль json



- Для сохранения разнообразных данных в текстовом виде можно использовать формат *JSON* (JavaScript Object Notation), который является стандартом де-факто в веб-приложениях и довольно популярен в других областях.
- Он поддерживается (на уровне языка, стандартной библиотеки языка или, по крайней мере, существующих сторонних библиотек) всеми современными языками программирования и знаком многим разработчикам, что делает его отличным выбором для сохранения данных для передачи между приложениями, написанными на разных языках.
- Модуль `json` позволяет сериализовать словари, списки, кортежи, строки, целые и вещественные числа, перечисления, булевские значения и `None`. Для поддержки других типов данных следует расширить классы `JSONEncoder` и `JSONDecoder`.

# ВВОД И ВЫВОД

## Модуль pickle



- Модуль pickle реализует бинарные протоколы для сериализации и десериализации объектов Python.
- Pickle поддерживает сериализацию огромного количества типов данных, включая созданные пользователем (многие из которых поддерживаются автоматически, но в более сложных случаях объект должен реализовать определённые специальные методы), корректно обрабатывает ссылки на уже сериализованные объекты и сериализацию рекурсивных типов данных.
- В отличие от JSON, который может быть прочитан где угодно, pickle специфичен для Python и не может быть использован для обмена данными с приложениями, написанными на других языках.



**Недостатком pickle является его небезопасность: десериализовать можно только те данные в формате pickle, которые получены из надёжного источника, иначе есть риск выполнения произвольного кода.**

# Смотрите наши уроки в видео формате

ITVDN.com



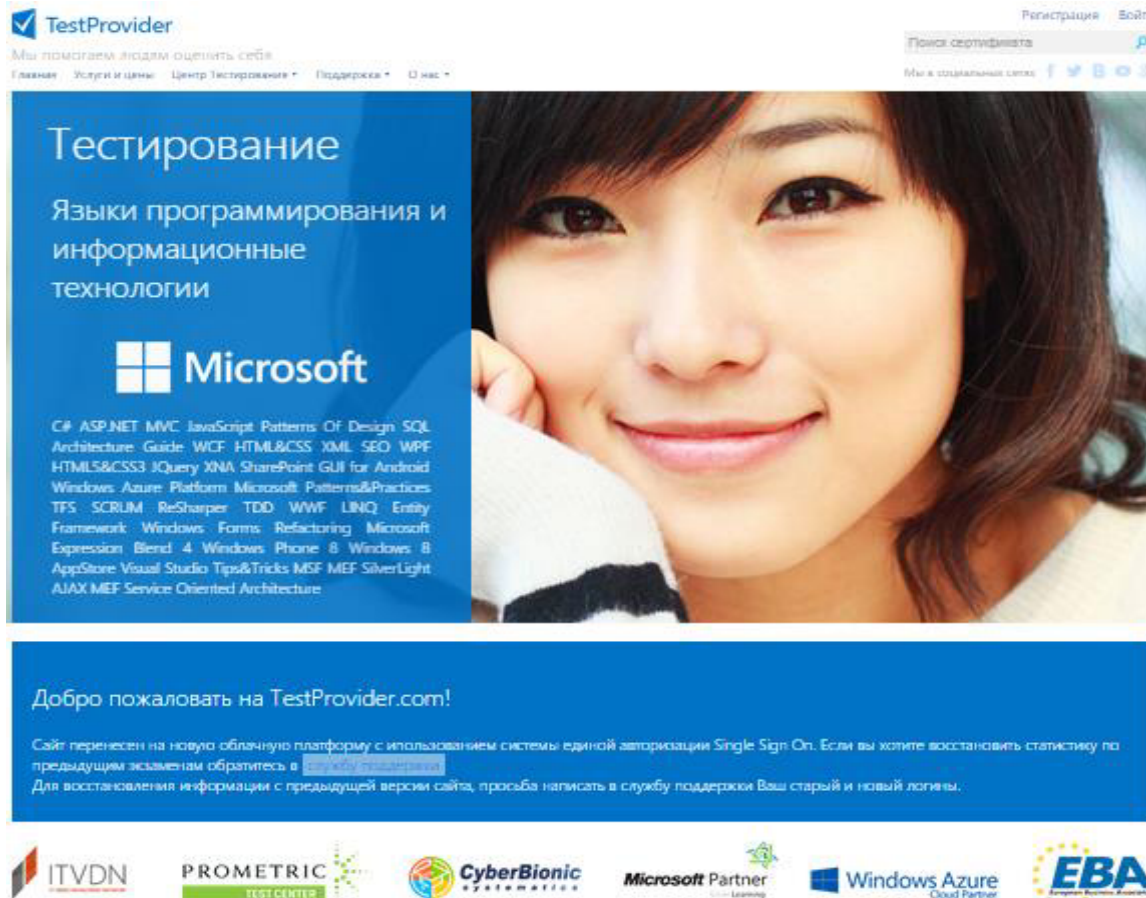
Посмотрите этот урок в видео формате на образовательном портале [ITVDN.com](http://itvdn.com) для закрепления пройденного материала.

Все курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics



# Проверка знаний

## TestProvider.com



TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на [TestProvider.com](http://TestProvider.com)

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.



# Python Essential

Q&A

# Информационный видеосервис для разработчиков программного обеспечения

