СОВРЕМЕННЫЙ СТАНДАРТ КОДИРОВАНИЯ ТЕКСТОВОЙ ИНФОРМАЦИИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

Информатика

Тихонов Е. Б.

г. Тверь, Тверской лицей, 9 класс

Научный руководитель: Наумова А.И., г. Тверь, преподаватель информатики высшей категории, Тверской лицей

Введение

Данная статья является реферативным изложением основной работы. Полный текст научной работы, иллюстрации и иные дополнительные материалы доступны на сайте XX Международного конкурса научно-исследовательских и творческих работ учащихся “Старт в науке” по ссылке: https://school-science.ru/20/4/56245.

В данной работе представлен материал по кодированию информации на языке программирования Python (Пайтон). Тема достаточно *актуальная* и представляет *повышенный интерес для учащихся предпрофильных и профильных классов.*

Цель данной работы заключается в том, чтобы получить *дополнительные знания* по этой теме.

Задача состоит в том, чтобы подобрать *соответствующий материал* с последующей систематизацией, обобщением и иллюстрацией текста.

Работа состоит из двух частей: *описательной* (дана характеристика истории создания кода UTF-8, основных принципов кодирования информации) и *практической* (приведён пример разработки программ (скриптов) по кодированию (декодированию) символов и строк с последующим компьютерным экспериментом и анализом полученных результатов).

Основные принципы кодирования информации

Кодирование заключается в том, что каждому символу ставится в соответствие *уникальный* десятичный код от 0 до 255 или соответствующий ему

двоичный код от 00000000 до 11111111. Таким образом, человек различает символы по их *начертаниям*, а компьютер – по их *кодам*.

При вводе в компьютер текстовой информации происходит её *двоичное кодирование*, изображение символа преобразуется в его двоичный код. Пользователь нажимает на клавиатуре клавишу с символом, и в компьютер поступает определённая последовательность из *восьми электрических импульсов* (двоичный код символа). Код символа хранится в оперативной памяти компьютера, где занимает *один байт*.

В процессе вывода символа на экран компьютера производится обратный процесс – *декодирование*, то есть преобразование кода символа в его изображение.

Важно, что присвоение символу конкретного кода – это вопрос соглашения, которое фиксируется в кодовой таблице.

В настоящее время широкое распространение получил новый международный стандарт Unicode, который отводит на каждый символ не один байт, а два, поэтому с его помощью можно закодировать не 256 символов, а $N = 2^{16} = 65536$ различных символов.

Каждая кодировка задаётся своей *собственной* кодовой таблицей [3].

История создания кода

Когда компьютеры только появились, у них была кодировка только для букв латинского алфавита и некоторых знаков — всего 7 бит и 128 символов. С развитием технологий многие страны сделали себе *альтернативные восьмибитные кодировки* — в них можно было хранить уже 256 символов. Кроме латиницы, в таких кодировках записывали буквы *национальных алфавитов* и другие нужные символы. Это сработало в тех странах, где алфавит состоит из небольшого числа букв (20—40), но не решило проблему с иероглифами. Тогда страны Азии сделали свои кодировки. В итоге всё это привело к тому, что файл с одного компьютера мог *не прочитаться* на другом компьютере, если там не было нужной кодировки. Для решения этих проблем сделали *Юникод — универсальную таблицу*, в которую можно поместить

1112064 символа. Сейчас в Юникоде записаны символы почти всех языков мира, но свободных позиций там осталось ещё около 80%.

Получается, что *Юникод — универсальное решение проблемы совместимости текста*. Текстовый файл, записанный в таком формате, можно прочитать на любом современном компьютере. Поддержка Юникода есть во всех новых операционных системах последних лет.

Чтобы пользоваться Юникодом, нужна была новая кодировка, которая бы *определяла правила хранения информации о каждом символе*. Такой кодировкой стала UTF (Unicode Transformation Format). Сейчас самая популярная разновидность UTF-кодировки ― UTF-8 (*табл. 1*) .

Чаще всего упоминание UTF-8 можно встретить в самом начале HTML-кода, когда объявляется кодировка в заголовке страницы. Строка <meta charset=”utf-8”> как раз говорит о том, что всё текстовое содержимое страницы нужно отображать по формату UTF-8 [1].

Код UTF-8

UTF-8 (от англ. *Unicode Transformation Format, 8-bit* — “формат преобразования Юникода, 8-бит”) — *распространённый стандарт кодирования символов*, позволяющий *более компактно* хранить и передавать символы Юникода, используя *переменное количество байт* (от 1 до 4), и обеспечивающий полную обратную совместимость с 7-битной кодировкой ASCII. Стандарт UTF-8 официально закреплён в документах RFC 3629 и ISO/IEC 10646 Annex D.

Кодировка UTF-8 сейчас является *доминирующей* в веб-пространстве.

Формат UTF-8 был разработан 2 сентября 1992 года *Кеном Томпсоном* и *Робом Пайком*.

UTF-8, по сравнению с UTF-16, наибольший выигрыш в компактности даёт для текстов на латинице, поскольку латинские буквы без диакритических знаков, цифры и наиболее распространённые знаки препинания кодируются в UTF-8 лишь одним байтом, и коды этих символов соответствуют их кодам в ASCII [2].

Таблица № 1. Кодовая таблица UTF-8

Символ	Десятичный код	Символ	Десятичный код	Символ	Десятичный код	Символ	Десятичный код
Пробел	32	+	43	0	48	5	53
!	33	,	44	1	49	6	54
#	35	-	45	2	50	7	55
$	36	.	46	3	51	8	56
*	42	/	47	4	52	9	57
Латинские буквы							
Символ	**Десятичный код**	**Символ**	**Десятичный код**	**Символ**	**Десятичный код**	**Символ**	**Десятичный код**
A	65	N	78	a	97	n	110
B	66	O	79	b	98	o	111
C	67	P	80	c	99	p	112
D	68	Q	81	d	100	q	113
E	69	R	82	e	101	r	114
F	70	S	83	f	102	s	115
G	71	T	84	g	103	t	116
H	72	U	85	h	104	u	117
I	73	V	86	i	105	v	118
J	74	W	87	j	106	w	119
K	75	X	88	k	107	x	120
L	76	Y	89	l	108	y	121
M	77	Z	90	m	109	z	122
Русские буквы							
Символ	**Десятичный код**	**Символ**	**Десятичный код**	**Символ**	**Десятичный код**	**Символ**	**Десятичный код**
А	1040	Р	1056	а	1072	р	1088
Б	1041	С	1057	б	1073	с	1089
В	1042	Т	1058	в	1074	т	1090
Г	1043	У	1059	г	1075	у	1091
Д	1044	Ф	1060	д	1076	ф	1092
Е	1045	Х	1061	е	1077	х	1093
Ж	1046	Ц	1062	ж	1078	ц	1094
З	1047	Ч	1063	з	1079	ч	1095
И	1048	Ш	1064	и	1080	ш	1096
Й	1049	Щ	1065	й	1081	щ	1097
К	1050	Ъ	1066	к	1082	ъ	1098
Л	1051	Ы	1067	л	1083	ы	1099
М	1052	Ь	1068	м	1084	ь	1100
Н	1053	Э	1069	н	1085	э	1101
О	1054	Ю	1070	о	1086	ю	1102
П	1055	Я	1071	п	1087	я	1103

Примечание:

1. Коды с номерами от 0 до 32 соответствуют не символам, а операциям (перевод строки, ввод пробела и так далее).
2. Коды с номерами от 33 до 127 являются *интернациональными* и соответствуют символам латинского алфавита, цифрам, знакам арифметических операций и знакам препинания.
3. Коды с 128 до 255 являются *национальными*, то есть в национальных кодировках *одному и тому же коду соответствуют различные символы* [3].
4. Десятичный код каждой строчной (малые) латинской и русской буквы на 32 больше кода соответствующей прописной (заглавные) буквы.
5. Алфавит латинского языка начинается с десятичного кода – 65 (прописные буквы) и с 97 (строчные буквы).
6. Алфавит русского языка начинается с десятичного кода – 1040 (прописные буквы) и с 1072 (строчные буквы)

Алгоритм кодирования

Алгоритм кодирования в UTF-8 стандартизирован в RFC 3629 и состоит из 3 этапов:

1. Определить количество байтов, требуемых для кодирования символа. Номер символа берётся из стандарта Юникода.
2. Установить старшие биты первого байта в соответствии с необходимым количеством байтов, определённом на первом этапе.
3. Установить значащие биты байтов в соответствии с номерами символа Юникода, выраженном в двоичном виде [2].

Примеры кодирования

<table>
<tr><td>Русский алфавит</td><td>Десятичный код</td><td colspan="2">Двоичный код</td></tr>
<tr><td rowspan="3">А</td><td rowspan="3">1040</td><td colspan="2">10000010000</td></tr>
<tr><td>1-й байт</td><td>2-й байт</td></tr>
<tr><td>00000100</td><td>00010000</td></tr>
</table>

Маркер UTF-8

Для указания, что файл или поток содержит символы Юникода, в начале файла или потока может быть вставлен *маркер последовательности байтов* (англ. *Byte order mark, BOM*), который в случае кодирования в UTF-8 принимает форму трёх байтов: EF BB BF_{16}.

Использование этого символа, согласно спецификации Юникод, не является обязательным, однако оно широко распространено, так как позволяет легко избежать неверного декодирования текстовой информации [2].

	1-й байт	2-й байт	3-й байт
Двоичный код	1110 1111	1011 1011	1011 1111
Шестнадцатеричный код	EF	BB	BF

Пятый и шестой байты

Изначально кодировка UTF-8 допускала использование до *шести байтов* для кодирования *одного символа*, однако в ноябре 2003 года стандарт REC 3629 запретил использование *пятого и шестого байтов*, а диапазон кодируемых символов был ограничен символом U+10FFFF. Это было сделано для обеспечения совместимости с UTF-16 [2].

Возможные ошибки декодирования

- Недопустимый байт.
- Байт продолжения (10xxxxxx) без начального байта.
- Отсутствие нужного количества байтов продолжения 10xxxxxx — например, двух после 1110xxxx).
- Строка обрывается посреди символа.
- Неэкономное кодирование — например, кодирование символа тремя байтами, когда можно двумя.
- Последовательность байтов, декодирующаяся в недопустимую кодовую позицию [2].

Разработка программ на языке Python с использованием кодировки UTF-8

В данной работе дана разработка *двух программ (скриптов)* по кодированию и декодированию информации: *работа с символами* и *работа со строками*.

В программах по кодированию информации на языке Python используются *различные методы*. Наиболее популярные представлены в таблице № 2 [5].

Таблица № 2. Методы кодирования и декодирования информации

Название	Характеристика
encode и decode	Методы, отвечающие за кодирование и декодирование строки в нужном формате.
chr и ord	Взаимообратные операции. Первая демонстрирует Unicode-символ, соответствующий введённому числовому значению. Вторая вернет числовой аналог конкретной символьной записи.
hex, bin, int, oct	Функции, позволяющие переводить числа в различные системы счисления.
bytes	Работает так же, как и метод encode. Отличается расширенными возможностями.
str	Перевод байтовых строк в обычные с использованием указанной ранее кодировки.
Unicodedata	Модуль, умеющий работать с базами данных всех Unicode-элементов.

Для кодирования и декодирования символов в первой программе использованы методы *ord* и *chr*. В программном коде по работе со строками использованы методы *encode* и *decode*.

Работа с символами

Описание формальной модели

Написать программу на языке Python, в которой для введённой с клавиатуры буквы (на русском и латинском языках) на экран выводится её код. Затем на экран выводится строка, представляющая собой последовательность из трёх букв используемой кодовой таблицы: исходная буква и буквы,

следующие за исходной, а также слова, составленные из соответствующих кодов на русском и латинском языках.

Компьютерная модель

Программа (скрипт)

```
#По умолчанию в Python используется кодировка UTF-8,
#что означает: Формат преобразования Юникода (8 бит)
print('Кодирование информации (код и строка)')
print('Введите букву (символ) на латинском языке')
#функция ord() определяет код символа (параметр – символ)
s1 = input()
kod1 = ord(s1)
print('Код введённой буквы', s1, '-', kod1)
print('Введите букву (символ) на русском языке')
s2 = input()
kod2 = ord(s2)
print('Код введённой буквы', s2, '-', kod2)
#функция chr() по коду определяет символ (параметр – код)
st1 = s1 + chr(kod1+1) + chr(kod1+2)
print('Строка на латинском языке: ', st1)
st2 = s2 + chr(kod2+1) + chr(kod2+2)
print('Строка на русском языке: ', st2)
print('Используя коды латинских букв, составьте слово')
print('Слово -',chr(99)+chr(111)+chr(100)+chr(101))
print('Используя коды русских букв, составьте слово')
print('Слово -',(chr(1082))+(chr(1086))+(chr(1076)))
print('Задержка экрана при выполнении файла .exe')
input('Для выхода из программы нажмите на <Enter>')
print('Программа завершена!')
```

Для выполнения программы (скрипта) тестовые примеры даны на *русском* и *латинском* языках (*табл. № 3*).

Тестовые примеры

Таблица № 3. Тестовые примеры для первого задания

Исходная буква (символ)			
Латинский язык	Код	Русский язык	Код
a	97	а	1072
Строки (исходные коды – первые три буквы алфавита)			
Латинский язык		Русский язык	
abc		абв	
Слова (исходные коды)			
Латинский язык		Русский язык	
99, 111, 100, 101		1082, 1086, 1076	
Полученное слово		Полученное слово	
code		код	

Выполнение

Метод ord() преобразует введённый символ в код, а метод chr() преобразует код в символ. Слова формируются с помощью *конкатенации (объединения) кодов*, а метод chr() переводит их в символы (буквы) (*рис. 1*).

```
Кодирование информации (код и строка)
Введите букву (символ) на латинском языке
a
Код введённой буквы a - 97
Введите букву (символ) на русском языке
а
Код введённой буквы а - 1072
Строка на латинском языке:  abc
Строка на русском языке:  абв
Используя коды латинских букв, составьте слово
Слово - code
Используя коды русских букв, составьте слово
Слово - код
Задержка экрана при выполнении файла .exe
Для выхода из программы нажмите на <Enter>
Программа завершена!
```

Рис. 1. Выполнение скрипта по работе с символами (буквами)

Работа со строками

Описание формальной модели

Написать программу на языке Python, в которой для введённой с клавиатуры строки на экран выводится её код. Затем на экран выводится исходная строка (на русском или латинском языках).

Компьютерная модель

Программа (скрипт)

```
print('Кодирование информации. Код UTF-8')
print("Введите количество вводимых строк:")
n=int(input())
print('Кодирование и декодирование на русском языке:')
for I in range(n):
#Перевести информацию в байты (encode())
    print('Введите строку:')
    st1 = input()
    st1_utf = st1.encode()
    print('Перевод строки в байты:')
    print(st1_utf)
#Перевести байты в текстовую информацию (decode())
    st1_utf = st1_utf.decode()
    print('Перевод из байт в строку:',st1_utf)
print('Кодирование и декодирование на латинском языке:')
#Перевести информацию в байты (encode())
for I in range(n):
    print('Введите строку:')
    st2 = input()
    print('Original string:',st2)
    st2_utf = st2.encode()
    print('Encoded string:',st2_utf)
#Перевести байты в текстовую информацию (decode())
    st2_utf = st2_utf.decode()
    print('Перевод из байт в строку:',st2_utf)
print('Задержка экрана при выполнении файла .exe')
input('Для выхода из программы нажмите на <Enter>')
print('Программа завершена!')
```

Для выполнения программы (скрипта) тестовые примеры даны на *русском* и *латинском* языках (*табл. № 4*).

Тестовые примеры

Таблица № 4. Тестовые примеры для второго задания

NN	Русский язык	Латинский язык	Произношение
1.	Первый среди равных	Primus inter pares	[При́мус и́нтэр па́рэс]
2.	Опыт — лучший учитель	Usus est optĭmus magister	[У́зус эст о́птимус маги́стэр].

Выполнение

Задача encode() – представить строку в виде объекта типа bytes (*предваряется литералом b*). Если знак относится к ASCII, то его байтовое представление будет выглядеть как оригинальный символ. В случае, когда он выходит за пределы ASCII, то заменяется байтовым представлением (\x – эскейп – последовательность для обозначения 16-ричных чисел в языке Python).

Метод decode() преобразует последовательность байтов в *привычную* нам строку (*рис. 2*) [4].

```
Кодирование информации. Код UTF-8
Введите количество вводимых строк:
1
Кодирование и декодирование на русском языке:
Введите строку:
Первый среди равных
Перевод строки в байты:
b'\xd0\x9f\xd0\xb5\xd1\x80\xd0\xb2\xd1\x8b\xd0\xb9 \xd1\x
81\xd1\x80\xd0\xb5\xd0\xb4\xd0\xb8 \xd1\x80\xd0\xb0\xd0\x
b2\xd0\xbd\xd1\x8b\xd1\x85'
Перевод из байт в строку: Первый среди равных
Кодирование и декодирование на латинском языке:
Введите строку:
Usus est optimus magister
Original string: Usus est optimus magister
Encoded string: b'Usus est optimus magister'
Перевод из байт в строку: Usus est optimus magister
Задержка экрана при выполнении файла .exe
Для выхода из программы нажмите на <Enter>
Программа завершена!
```

Рис. 2. Выполнение скрипта по работе со строками

Анализ полученных результатов

Результаты, полученные при проведении компьютерного эксперимента (*рис.1 и рис. 2*) *полностью* соответствуют результатам в тестовых примерах (*табл. 3 и табл. 4*).

Заключение

В данной работе показано *кодирование текстовой информации* с использованием кодировки UTF-8 на языке программирования Python. Достаточно подробно рассмотрены правила *кодирования и декодирования*, приведены конкретные примеры с использованием соответствующих методов. Проведён *анализ планируемых и полученных результатов* компьютерного эксперимента по работе с символами и строками.

В ходе разработки проекта получены дополнительные знания по теме “Кодирование информации”. Проведённый компьютерный эксперимент наглядно показывает *практическую значимость* проведённых исследований.

Python (Пайтон) – *современный развивающийся язык*, изучение которого начинается в *предпрофильных и профильных классах* общеобразовательных учреждений на примере *использования его основных конструкций и методов*, что достаточно *наглядно* показано в данной работе.

Программное обеспечение

1. Операционная система Windows 10
2. Среда программирования Python – 3.8.0
3. Приложение Microsoft Office Word 2010

Список использованных источников и литературы

1. UTF – универсальная кодировка для всего [Электронный ресурс]. – Режим доступа: https://thecode.media/unicode-2-2/
2. UTF-8 – Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/UTF-8
3. Информатика и информационные технологии. Учебник для 10-11 классов / Н. Д. Угринович. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2006. – 511 с.
4. Кодирование строк – ASCII, Unicode, UTF-8 [Электронный ресурс]. – Режим доступа: https://smartiqa.ru/blog/python-encoding
5. Кодировки в Python и Unicode [Электронный ресурс]. – Режим доступа: https://otus.ru/journal/kodirovki-v-python-i-unicode/