9.7 Строки в памяти компьютера, таблица символов Unicode

Шаг 1

Тема урока: представление строк в памяти компьютера, ASCII и Unicode

1. Представление строк в памяти компьютера
2. Таблица символов ASCII
3. Таблица символов Unicode
4. Функция `ord()`
5. Функция `chr()`

Аннотация. Представление строк в памяти компьютера.

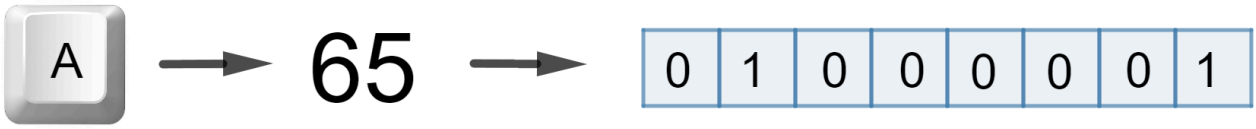
Представление строк в памяти компьютера

Любой набор данных в оперативной памяти (https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) компьютера должен храниться в виде двоичного числа. Это относится и к строкам, которые состоят из символов (буквы, знаки препинания и так далее). Когда символ сохраняется в памяти, он сначала преобразуется в цифровой код. И затем этот цифровой код сохраняется в памяти как двоичное число.

За прошедшие годы для представления символов в памяти компьютера были разработаны различные схемы кодирования. Исторически самой важной из этих схем кодирования является схема кодирования ASCII (American Standard Code for Information Interchange – американский стандартный код обмена информацией).

Таблица символов ASCII

ASCII представляет собой набор из 128 цифровых кодов, которые обозначают английские буквы, различные знаки препинания и другие символы. Например, код ASCII для прописной английской буквы «A» (латинской) равняется 65. Когда на компьютерной клавиатуре вы набираете букву «A» в верхнем регистре, в памяти сохраняется число 65 (как двоичное число, разумеется).



Код ASCII для английской «B» в верхнем регистре равняется 66, для «C» в верхнем регистре – 67 и так далее. **На один символ в ASCII отводится ровно 7 бит.**

Аббревиатура ASCII произносится «аски».

ASCII table

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Набор символов ASCII был разработан в начале 1960-х годов и в конечном счете принят почти всеми производителями компьютеров. Однако схема кодирования ASCII имеет ограничения, потому что она определяет коды только для 128 символов. Для того чтобы это исправить, в начале 1990-х годов был разработан набор символов Юникода (Unicode). Это широкая схема кодирования, совместимая с ASCII, которая может также представлять символы многих языков мира. Сегодня Юникод быстро становится стандартным набором символов, используемым в компьютерной индустрии.

Таблица символов Unicode

Таблица символов Юникод представляет собой набор цифровых символов, которые включают в себя знаки почти всех письменных языков мира. Стандарт предложен в 1991 году некоммерческой организацией «Консорциум Юникода». Применение этого стандарта позволяет закодировать очень большое число символов из разных систем письменности: в документах, закодированных по стандарту Юникод, могут соседствовать китайские иероглифы, математические символы, буквы греческого алфавита, латиницы и кириллицы, символы музыкальной нотации.

Стандарт состоит из двух основных частей: универсального набора символов и семейства кодировок (Unicode transformation format, UTF). Универсальный набор символов перечисляет допустимые по стандарту Юникод символы и присваивает каждому символу код в виде неотрицательного целого числа. Семейство кодировок определяет способы преобразования кодов символов для хранения на компьютере и передачи.

В Юникод все время добавляются новые символы, а сам размер этой таблицы не ограничен и будет только расти, поэтому сейчас при хранении в памяти одного юникод-символа может потребоваться от 1 до 8 байт. Отсутствие ограничений привело к тому, что стали появляться символы на все случаи жизни.

В Python строки хранятся в виде последовательности юникод символов.

Примечания

Примечание 1. Официальный сайт таблицы символов Unicode (https://home.unicode.org/).

Примечание 2. Юникод – это не кодировка. Это именно таблица символов. То, как символы с соответствующими кодами будут храниться в памяти компьютера, зависит от конкретной кодировки, базирующейся на Юникоде, например UTF-8.

Примечание 3. Первые 128 кодов таблицы символов Unicode совпадают с ASCII.

Шаг 2

Функция ord()

Функция `ord()` позволяет определить код некоторого символа в таблице символов Unicode. Аргументом данной функции является одиночный символ.

Приведённый ниже код:

```
num1 = ord('A')
num2 = ord('B')
num3 = ord('a')
print(num1, num2, num3)
```

выводит:

```
65 66 97
```

Обратите внимание, что функция `ord()` принимает именно **одиночный символ**. Если попытаться передать строку, содержащую более одного символа, мы получим ошибку времени выполнения.

Приведённый ниже код:

```
num = ord('Abc')
print(num)
```

приводит к возникновению ошибки:

```
TypeError: ord() expected a character, but string of length 3 found
```

Название функции `ord()` происходит от английского слова «order» — порядок.

Функция chr()

Функция `chr()` позволяет определить по коду символа сам символ. Аргументом данной функции является численный код.

Приведённый ниже код:

```
chr1 = chr(65)
chr2 = chr(75)
chr3 = chr(110)

print(chr1, chr2, chr3)
```

выводит:

```
A K n
```

Название функции `chr()` происходит от английского слова «character» — символ.

Функции `ord()` и `chr()` часто работают в паре. Мы можем использовать следующий код для вывода всех заглавных букв английского алфавита:

```
for i in range(26):
    print(chr(ord('A') + i))
```

Приведённый выше код выводит:

```
A
B
C
...
X
Y
Z
```

Вызов функции `ord('A')` возвращает код символа `A`, который равен 65. Далее на каждой итерации цикла к данному коду прибавляется значение переменной `i = 0, 1, 2, ..., 25`, а затем полученный код преобразуется в символ с помощью вызова функции `chr()`.

Примечания

Примечание 1. Функции `ord()` и `chr()` являются **взаимно обратными**. Для них выполнены равенства:

`chr(ord(<символ>)) = <символ>`

`ord(chr(<код символа>)) = <код символа>`

Например, `ord('A')` возвращает число `65`, а `chr(65)` возвращает символ `'A'`. То есть `chr(ord('A'))` возвращает символ `A`. В обратную сторону это также работает: `ord(chr(65))` возвращает число `65`.

Приведённый ниже код:

```
print(chr(ord('A')) == 'A')
print(ord(chr(65)) == 65)
```

выводит:

```
True
True
```

Примечание 2. Обратите внимание, что некоторые символы могут выглядеть одинаково, но на самом деле иметь **разные коды** в таблице Unicode. Это в первую очередь касается букв, которые имеют одинаковое написание на разных языках.

Приведённый ниже код:

```
print(ord('a'))     # английская буква «a»
print(ord('а'))     # русская буква «а»
```

выводит:

```
97
1072
```

Также можно заметить, что в таблице Unicode русские буквы находятся гораздо **дальше** английских.

❤️ Happy Pythoning!

Шаг 3

Соотнесите символы с их кодами в таблице Unicode (ASCII).

Примечание 1. Гарантируется, что все буквенные символы в данной задаче являются буквами английского алфавита.

Примечание 2. Чтобы узнать код символа в таблице Unicode (ASCII), можно воспользоваться функцией `ord()`.

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/3

Шаг 4

Что покажет приведённый ниже код?

```
print(chr(ord('ᘏ')))
print(ord(chr(128013)))
```

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/4

Шаг 5

Что покажет приведённый ниже код?

```
print(ord('foo'))
```

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/5

Шаг 6

Какая следующая буква? SOON

На вход программе подаётся некоторая буква русского алфавита в верхнем регистре. Найдите следующую за ней букву и выведите её на экран. Если введённая буква является последней в алфавите, то выведите текст «Дальше букв нет» (без кавычек).

Формат входных данных
На вход программе подаётся одна буква русского алфавита в верхнем регистре.

Формат выходных данных
Программа должна вывести одну букву в верхнем регистре или текст «Дальше букв нет» (без кавычек) в соответствии с условием задачи.

Примечание. Будем считать, что буквы `Ё` нет в русском алфавите.

▶ Тестовые данные

Sample Input 1:

```
А
```

Sample Output 1:

```
Б
```

Sample Input 2:

```
Я
```

Sample Output 2:

```
Дальше букв нет
```

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/6

Шаг 7

Символы в диапазоне

На вход программе подаются два числа a и b $(a < b)$. Напишите программу, которая для каждого кодового значения в диапазоне от a до b (включительно) выводит соответствующий ему символ из таблицы символов Unicode.

Формат входных данных
На вход программе подаются два натуральных числа, каждое на отдельной строке.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные

Sample Input 1:

```
65
70
```

Sample Output 1:

```
A B C D E F
```

Sample Input 2:

```
97
110
```

Sample Output 2:

```
a b c d e f g h i j k l m n
```

Sample Input 3:

```
48
64
```

Sample Output 3:

```
0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
```

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/7

Шаг 8

Простой шифр

На вход программе подаётся строка текста. Напишите программу, которая переводит каждый ее символ в соответствующий ему код из таблицы символов Unicode.

Формат входных данных

На вход программе подаётся строка текста.

Формат выходных данных

Программа должна вывести кодовые значения символов строки разделенных одним символом пробела.

Примечание. Проверяющая система примет ваше решение, даже если в конце будет лишний пробел. 😇

▶ Тестовые данные 🟢

Sample Input:

```
Hello world!
```

Sample Output:

```
72 101 108 108 111 32 119 111 114 108 100 33
```

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/8

Шаг 9

Самое тяжёлое слово 🗿

Под "тяжестью" слова будем понимать сумму кодов по таблице Unicode всех символов этого слова. Напишите программу, которая принимает 4 слова и находит среди них **самое тяжёлое** слово. Если самых тяжёлых слов будет несколько, то программа должна вывести первое из них.

Формат входных данных

На вход программе подаются 4 слова, каждое на отдельной строке.

Формат выходных данных

Программа должна вывести самое тяжёлое слово в строке.

▶ Тестовые данные 🟢

Sample Input 1:

```
строки
списки
кортежи
множества
```

Sample Output 1:

```
множества
```

Sample Input 2:

```
az
by
cx
122
```

Sample Output 2:

```
az
```

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/9

Шаг 10

Стоимость ответа 💬

Модератору Сэму **за каждый символ** его сообщений в комментариях Тимур платит в 🐝 (пчёлках-coin) по следующему тарифу:

$$<\text{код символа в таблице Unicode}> \times 3 🐝$$

А стоимость всего сообщения складывается из суммы стоимостей всех символов. Сэму захотелось подсчитать, сколько 🐝 он зарабатывает за свои ответы в комментариях, и просит вас помочь ему.

На вход программе подаётся строка текста. Требуется написать программу, которая находит стоимость сообщения Сэма в 🐝 и выводит текст в следующем формате:

```
Текст сообщения: '<текст сообщения Сэма>'
Стоимость сообщения: <стоимость сообщения Сэма>🐝
```

Формат входных данных
На вход программе подаётся строка текста – очередной ответ Сэма в комментариях.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

Примечание. 🐝 (пчёлка-coin) – виртуальная валюта команды BEEGEEK, которой Тимур расплачивается со своими сотрудниками.

▶ Тестовые данные 🟢

Sample Input 1:

```
@кодер 666, пишите в комментариях по делу, не засоряйте чат бредом
```

Sample Output 1:

```
Текст сообщения: '@кодер 666, пишите в комментариях по делу, не засоряйте чат бредом'
Стоимость сообщения: 164457🐝
```

Sample Input 2:

```
@тот самый Гвидо, у вас программа выводит лишний пробел в конце первой строки
```

Sample Output 2:

```
Текст сообщения: '@тот самый Гвидо, у вас программа выводит лишний пробел в конце первой строки'
Стоимость сообщения: 206064🐝
```

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/10

Накручиваем стоимость ответа ⬆️ 🌶️

Как вы помните из прошлой задачи, модератору Сэму **за каждый символ** его сообщений в комментариях Тимур платит в 🐝 (пчёлках-coin) по следующему тарифу:

$$<\text{код символа в таблице Unicode}> \times 3 🐝$$

А стоимость всего сообщения складывается из суммы стоимостей всех символов. На этот раз Сэму захотелось схитрить и попробовать увеличить стоимость своего сообщения, заменив в нем некоторые английские буквы на русские. Как вы помните, русские буквы в таблице Unicode находятся после английских.

Сэм хочет заменить следующие **английские** буквы:

```
eyopaxcETOPAHXCBM
```

на соответствующие им **русские** буквы:

```
eyopaxcETOPAHXCBM
```

Тимур визуально разницу не заметит, а Сэм сможет зарабатывать больше пчёлок-coin. 😊

На вход программе подаётся строка текста. Требуется написать программу, которая находит стоимость старого и нового сообщений Сэма в 🐝 и выводит текст в следующем формате:

```
Старая стоимость: <стоимость старого сообщения>🐝
Новая стоимость: <стоимость нового сообщения>🐝
```

Формат входных данных

На вход программе подаётся строка текста – очередной ответ Сэма в комментариях.

Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

Примечание. Обратите внимание, что если в строке не удаётся заменить буквы, то стоимость сообщения не изменится. 😳

▶ Тестовые данные 🟢

Sample Input 1:

```
@coder666, пишите в комментариях по делу, не засоряйте чат бредом
```

Sample Output 1:

```
Старая стоимость: 149709🐝
Новая стоимость: 158532🐝
```

Sample Input 2:

```
@Тимур Гуев, ХОЧУ БОЛЬШЕ ПЧЕЛОК, ПЧЕЛОК, ПЧЕЛОК, ПЧЕЛОК
```

Sample Output 2:

```
Старая стоимость: 137946🐝
Новая стоимость: 137946🐝
```

Шаг 12

Шифр Цезаря 🌶

Легион Цезаря, созданный в 23 веке на основе Римской Империи не изменяет древним традициям и использует шифр Цезаря (https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F). Это их и подвело, ведь данный шифр очень простой. Однако в постапокалипсисе люди плохо знают все тонкости довоенного мира, поэтому ученые из НКР не могут понять, как именно нужно декодировать данные сообщения. Напишите программу для декодирования этого шифра.

Формат входных данных
В первой строке подаётся число n $(1 \leq n \leq 25)$ – сдвиг, во второй строке даётся закодированное сообщение в виде строки со строчными латинскими буквами.

Формат выходных данных
Программа должна вывести одну строку – декодированное сообщение. Обратите внимание, что нужно декодировать сообщение, а не закодировать.

▶ Тестовые данные 🟢

Sample Input 1:

```
1
bwfusvfupdbftbs
```

Sample Output 1:

```
avetruetocaesar
```

Sample Input 2:

```
14
fsfftsfufksttskskt
```

Sample Output 2:

```
rerrfergrweffewewf
```

Чтобы решить это задание откройте https://stepik.org/lesson/313439/step/12

Шаг 13

Сбой в системе ⚠️ 🌶 🌶

После недавнего сбоя в операционной системе от компании «Oursoft» у Гвидо сбилась кодировка на компьютере. Теперь все буквы **русского алфавита** отображаются в некорректном виде:

```
[u-<номер символа в таблице Unicode>]
```

Гвидо ещё не научился читать символы в таком формате, поэтому просит вас написать программу, которая будет "расшифровывать" для него все тексты на компьютере.

На вход программе подаётся строка текста. Расшифруйте текст, заменив все конструкции `[u-<номер символа в таблице Unicode>]` на соответствующие буквы русского алфавита, и выведите его.

Формат входных данных
На вход программе подаётся строка текста, в которой могут быть зашифрованы символы русского алфавита.

Формат выходных данных

Программа должна вывести строку текста, расшифровав символы русского алфавита.

Примечание. Будем считать, что буквы `Ё` нет в русском алфавите. 🤔

▶ Тестовые данные 🟢

Sample Input 1:

```
Hello, my name is [u-1061][u-1072][u-1082][u-1080]!
```

Sample Output 1:

```
Hello, my name is Хаки!
```

Sample Input 2:

```
Username: [u-1042][u-1072][u-1089][u-1103]; City: [u-1050][u-1072][u-1079][u-1072][u-1085][u-1100]
```

Sample Output 2:

```
Username: Вася; City: Казань
```

Sample Input 3:

```
Because I didn't know what I had until it was gone! All right?
```

Sample Output 3:

```
Because I didn't know what I had until it was gone! All right?
```