

АаDSaPз_fZtH. Занятие 3

Часть 1. Двоичная система счисления

Емельянов Антон
login-const@mail.ru

Двоичная система счисления

- Двоичная система счисления — позиционная система счисления с основанием 2. Благодаря непосредственной реализации в цифровых электронных схемах на логических вентилях, двоичная система используется практически во всех современных компьютерах и прочих вычислительных электронных устройствах.
- В двоичной системе счисления используются всего две цифры 0 и 1. Другими словами, поэтому двойка является основанием двоичной системы счисления. (Аналогично у десятичной системы основание 10.)

Двоичная система счисления

- В десятичной системе счисления мы располагаем десятью знаками-цифрами (от 0 до 9). Когда счет достигает 9, то вводится новый разряд (десятки), а единицы обнуляются и счет начинается снова. После 19 разряд десятков увеличивается на 1, а единицы снова обнуляются. И так далее. Когда десятки доходят до 9, то потом появляется третий разряд - сотни.
- Двоичная система счисления аналогична десятичной за исключением того, что в формировании числа участвуют всего лишь две знака-цифры: 0 и 1. Как только разряд достигает своего предела (т.е. единицы), появляется новый разряд, а старый обнуляется.

- Попробуем считать в двоичной системе:
 - 0 - это ноль
 - 1 - это один (и это предел разряда)
 - 10 - это два
 - 11 - это три (и это снова предел)
 - 100 - это четыре
 - 101 - пять
 - 110 - шесть
 - 111 - семь
 - 1000 - восемь и т.д.
 - 11111111 - это?

- Попробуем считать в двоичной системе:
 - 0 - это ноль
 - 1 - это один (и это предел разряда)
 - 10 - это два
 - 11 - это три (и это снова предел)
 - 100 - это четыре
 - 101 - пять
 - 110 - шесть
 - 111 - семь
 - 1000 - восемь и т.д.
 - 11111111 - 255.

Двоичная система счисления

- Не трудно заметить, что в двоичной системе счисления длины чисел с увеличением значения растут быстрыми темпами. Как определить, что значит вот это: 10001001? Непривычный к такой форме записи чисел человеческий мозг обычно не может понять сколько это. Неплохо бы уметь переводить двоичные числа в десятичные.
- В десятичной системе счисления любое число можно представить в форме суммы единиц, десятков, сотен и т.д. Например:
 - $1476 = 1000 + 400 + 70 + 6$
- Можно пойти еще дальше и разложить так:
 - $1476 = 1 * 10^3 + 4 * 10^2 + 7 * 10^1 + 6 * 10^0$

Двоичная система счисления

- Аналогично можно разложить и любое двоичное число. Только основание здесь будет 2:

- $10001001 = 1*2^7 + 0*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$

- Если посчитать сумму составляющих, то в итоге мы получим десятичное число, соответствующее 10001001:

- $1*2^7 + 0*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 128 + 0 + 0 + 0 + 8 + 0 + 0 + 1 = 137$


- Т.е. число 10001001 по основанию 2 равно числу 137 по основанию 10.

Записать это можно так:

- $10001001_2 = 137_{10}$

Двоичная система счисления

- Может потребоваться перевести десятичное число в двоичное. Один из способов - это деление на два и формирование двоичного числа из остатков. Например, нужно получить из числа 77 его двоичную запись:
- $77 / 2 = 38$ (1 остаток)
- $38 / 2 = 19$ (0 остаток)
- $19 / 2 = 9$ (1 остаток)
- $9 / 2 = 4$ (1 остаток)
- $4 / 2 = 2$ (0 остаток)
- $2 / 2 = 1$ (0 остаток)
- $1 / 2 = 0$ (1 остаток)
- Собираем остатки вместе, начиная с конца: 1001101. Это и есть число 77 в двоичном представлении. Проверим:
- $1001101 = 1*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 64 + 0 + 0 + 8 + 4 + 0 + 1 = 77$



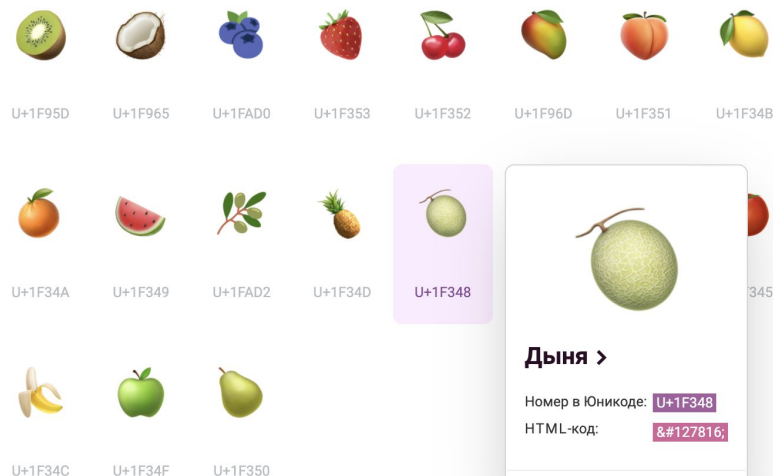
АаDSaPз_fZtH. Занятие 3

Часть 2. К😊дир😊вки

Емельянов Антон
login-const@mail.ru

Способ хранения строк

- **Кодировка** (часто называемая также *кодовой страницей*) - это набор числовых значений, которые ставятся в соответствие группе алфавитно-цифровых символов, знаков пунктуации и специальных символов.
- **Символ** - минимальная компонента текста.
- **Стандарт**, в котором перечислены все возможные символы - **Unicode** (<https://unicode-table.com/>).



ASCII кодировка

- ASCII (American Standard Code for Interchange of Information).
- В данной кодировке всего

- ASCII (American Standard Code for Interchange of Information).
- В данной кодировке всего $2^8=256$ (1 байт).
- В ASCII первые 128 символов всех кодовых страниц состоят из базовой таблицы символов (латинский алфавит и управляющие символы). Первые 32 кода базовой таблицы, начиная с нулевого, размещают управляющие коды.
- Символы с номерами от 128 до 255 представляют собой таблицу расширения и варьируются в зависимости от набора скриптов, представленных кодировкой символов.

ASCII кодировка

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	32	20	(sp)	64	40	@
1	1	SOH	33	21	!	65	41	A
2	2	STX	34	22	"	66	42	B
3	3	ETX	35	23	#	67	43	C
4	4	EOT	36	24	\$	68	44	D
5	5	ENQ	37	25	%	69	45	E
6	6	ACK	38	26	&	70	46	F
7	7	BEL	39	27	'	71	47	G
8	8	BS	40	28	(72	48	H
9	9	TAB	41	29)	73	49	I
10	A	LF	42	2A	*	74	4A	J
11	B	VT	43	2B	+	75	4B	K
12	C	FF	44	2C	,	76	4C	L
13	D	CR	45	2D	-	77	4D	M
14	E	SO	46	2E	.	78	4E	N
15	F	SI	47	2F	/	79	4F	O
16	10	DLE	48	30	0	80	50	P
17	11	DC1	49	31	1	81	51	Q
18	12	DC2	50	32	2	82	52	R
19	13	DC3	51	33	3	83	53	S
20	14	DC4	52	34	4	84	54	T
21	15	NAK	53	35	5	85	55	U
22	16	SYN	54	36	6	86	56	V
23	17	ETB	55	37	7	87	57	W
24	18	CAN	56	38	8	88	58	X
25	19	EM	57	39	9	89	59	Y
26	1A	SUB	58	3A	:	90	5A	Z
27	1B	ESC	59	3B	;	91	5B	[
28	1C	FS	60	3C	<	92	5C	\
29	1D	GS	61	3D	=	93	5D]
30	1E	RS	62	3E	>	94	5E	^
31	1F	US	63	3F	?	95	5F	_

ASCII кодировка - пример

- Закодируем строку «ok» (англ.) в кодировке ASCII.

ASCII кодировка - пример

- Закодируем строку «ok» (англ.) в кодировке ASCII.
- Символ «o» (англ.) имеет позицию 111 в десятичном виде и *6F* в шестнадцатиричном. Переведем это в двоичную систему — **01101111**.

ASCII кодировка - пример

- Закодируем строку «ok» (англ.) в кодировке ASCII.
- Символ «o» (англ.) имеет позицию 111 в десятичном виде и *6F* в шестнадцатиричном. Переведем это в двоичную систему — **01101111**.
- Символ «k» (англ.) — позиция 107 в десятичной и *6B* в шестнадцатиричной, переводим в двоичную — **01101011**.

ASCII кодировка - пример

- Закодируем строку «ок» (англ.) в кодировке ASCII.
- Символ «о» (англ.) имеет позицию 111 в десятичном виде и *6F* в шестнадцатиричном. Переведем это в двоичную систему — **01101111**.
- Символ «к» (англ.) — позиция 107 в десятичной и *6B* в шестнадцатиричной, переводим в двоичную — **01101011**.
- Итого строка «ок» закодированная в ASCII будет выглядеть так —
 - **01101111 01101011**.

ASCII кодировка - пример

- Закодируем строку «ok» (англ.) в кодировке ASCII.
- Символ «o» (англ.) имеет позицию 111 в десятичном виде и *6F* в шестнадцатиричном. Переведем это в двоичную систему — **01101111**.
- Символ «k» (англ.) — позиция 107 в десятичной и *6B* в шестнадцатиричной, переводим в двоичную — **01101011**.
- Итого строка «ok» закодированная в ASCII будет выглядеть так —
 - **01101111 01101011**.
- Процесс декодирования будет обратный. Берем по 8 бит, переводим их в десятичную кодировку, получаем номер символа, смотрим по таблице что это за символ.

- **Unicode** — именно эта таблица и есть (это не кодировка, а именно таблица символов). Она состоит из 1 114 112 позиций.
- Большинство этих позиций пока не заполнены символами, так что вряд ли понадобится это пространство расширять.
- Разделено это общее пространство на **17 блоков**, по 65 536 символов в каждом. Каждый блок содержит свою группу символов. Нулевой блок — базовый, там собраны наиболее употребляемые символы всех современных алфавитов. Во втором блоке находятся символы вымерших языков. Есть два блока отведенные под частное использование. Большинство блоков пока не заполнены.
- Итого емкость символов юникода составляет от 0 до 10FFFF (в шестнадцатиричном виде).
- Записываются символы в шестнадцатиричном виде с приставкой «U+». Например первый базовый блок включает в себя символы от U+0000 до U+FFFF (от 0 до 65 535), а последний семнадцатый блок от U+100000 до U+10FFFF (от 1 048 576 до 1 114 111).

- Кодировки на основе Unicode:
 - ucs1 - байт, ucs2 - 2 байта, ucs4 - 4 байта. Кодировки фиксированной длины
 - utf8 - использует переменное количество байт.
 - utf16 использует переменное количество байт.

Подробнее: <https://habr.com/ru/post/478636/>

Кодировка в python

- Размер внутреннего представления символа меняется в зависимости от того, какой диапазон номеров символов в строке (ucs1, ucs2, ucs4).

```
import sys
import chardet
```

```
easy = "easy"
изич = "изич"
易易易易 = "易易易易"
```


```
[sys.getsizeof(easy[:index]) for index in range(len(easy) + 1)]
[49, 50, 51, 52, 53]
```

```
[sys.getsizeof(изич[:index]) for index in range(len(изич) + 1)]
[49, 76, 78, 80, 82]
```

```
[sys.getsizeof(易易易易[:index]) for index in range(len(易易易易) + 1)]
[49, 76, 78, 80, 82]
```

```
sys.getsizeof(изич + easy), sys.getsizeof(易易易易 + изич + easy)
(90, 98)
```

```
chardet.detect(easy.encode()), chardet.detect(изич.encode()), chardet.detect(易.encode())
({'confidence': 1.0, 'encoding': 'ascii'},
 {'confidence': 0.938125, 'encoding': 'utf-8'},
 {'confidence': 0.73, 'encoding': 'windows-1252'})
```



АаDSaPз_fZtH. Занятие 3

Часть 3. Работа с текстовой информацией в python

Емельянов Антон
login-const@mail.ru

Строки

- Тип `str` — базовый тип представляющий из себя неизменяемую последовательность символов; `str` от «string» — «строка».
- Имеется в виду последовательность кодовых точек Unicode.

```
str.py x
1 s1 = input()
2 s2 = "world"
3 s3 = s1 + " " + s2
4 s4 = str(10)
5
6 print(s3, s3[1], 'Кот' 'обус', sep="\t")
7
8 # Одиночные кавычки. Часто встречаемый вариант записи.
9 my_str = 'а внутри "можно" поместить обычные'
10 # Кавычки.
11 my_str = "а внутри 'можно' поместить одиночные"
12 # Три одиночных кавычки. Удобно для записей в несколько строк
13 my_str = '''В трёх одиночных
14 кавычках'''
15 # Тройные кавычки. Общепринятый способ для строк документации.
16 my_str = """Three double quotes"""
```

```
hi
```

```
hi world
```

```
i
```

```
Котобус
```

Базовые операции над строками

str_base_op.py

```
1 S1 = 'spam'
2 S2 = 'eggs'
3 # Конкатенация (сложение)
4 print(S1 + S2)
5 # Дублирование строки
6 print('spam' * 3)
7 # Длина строки (функция len)
8 print(len('spam'))
9 # Доступ по индексу и срез
10 print(S1[0], S1[1:3])
11
```

```
spameggs
spamspamspam
4
s pa
```


Функции и методы строк

Функция или метод	Назначение
<code>S = 'str'; S = "str"; S = ""str""; S = """"str""""</code>	Литералы строк
<code>S = "s\np\ta\nbbbb"</code>	Экранированные последовательности
<code>S = r"C:\temp\new"</code>	Неформатированные строки (подавляют экранирование)
<code>S = b"byte"</code>	Строка байтов
<code>S1 + S2</code>	Конкатенация (сложение строк)

Функции и методы строк

S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза
len(S)	Длина строки
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
S.rfind(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или -1

Функции и методы строк

S.index (str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
S.rindex (str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
S.replace (шаблон, замена)	Замена шаблона
S.split (символ)	Разбиение строки по разделителю
S.isdigit ()	Состоит ли строка из цифр
S.isalpha ()	Состоит ли строка из букв

Функции и методы строк

S.isalnum()	Состоит ли строка из цифр или букв
S.islower()	Состоит ли строка из символов в нижнем регистре
S.isupper()	Состоит ли строка из символов в верхнем регистре
S.isspace()	Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая строка" ('\n'), "перевод каретки" ('\r'), "горизонтальная табуляция" ('\t') и "вертикальная табуляция" ('\v'))

Функции и методы строк

S.istitle()	Начинаются ли слова в строке с заглавной буквы
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.startswith(str)	Начинается ли строка S с шаблона str
S.endswith(str)	Заканчивается ли строка S шаблоном str

Функции и методы строк

S.join (список)	Сборка строки из списка с разделителем S
ord (символ)	Символ в его код ASCII
chr (число)	Код ASCII в символ
S.capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
S.center (width, [fill])	Возвращает отцентрированную строку, по краям которой стоит символ fill (пробел по умолчанию)

Функции и методы строк

S.count (str, [start],[end])	Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)
S.expandtabs ([tabsize])	Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам
S.lstrip ([chars])	Удаление пробельных символов в начале строки
S.rstrip ([chars])	Удаление пробельных символов в конце строки
S.strip ([chars])	Удаление пробельных символов в начале и в конце строки

Функции и методы строк

S.partition (шаблон)	Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки
S.rpartition (sep)	Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку
S.swapcase ()	Переводит символы нижнего регистра в верхний, а верхнего – в нижний
S.title ()	Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний

S.zfill (width)	Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями
S.ljust (width, fillchar=" ")	Делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar
S.rjust (width, fillchar=" ")	Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar



str_format.py

```
1 a = 10
2 b = "ten"
3 print(str(a) + " is " + b)
4 print("%d is %s" % (a, b))
5 print("{} is {}".format(a, b))
6 print("{1} is {0}".format(a, b))
7 print("{a} is {b}".format(a=a, b=b))
8 print(f"{a} is {b}")
```

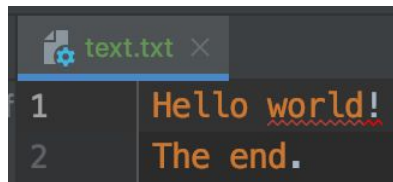
```
10 is ten
10 is ten
10 is ten
ten is 10
10 is ten
10 is ten
```

Работа с файлами. Открытие

- Прежде, чем работать с файлом, его надо открыть. С этим замечательно справится встроенная функция `open`:
 - `f = open('L3/titanik_train.csv', 'r')`
- У функции `open` много параметров, нам пока важны 3 аргумента:
 - Первый, это имя файла. Путь к файлу может быть относительным или абсолютным.
 - Второй аргумент, это режим, в котором мы будем открывать файл.
 - И последний аргумент, `encoding`, нужен только в текстовом режиме чтения файла. Этот аргумент задает кодировку.

Работа с файлами. чтение

- Открыли мы файл, а теперь мы хотим прочитать из него информацию. Для этого есть несколько способов, но большого интереса заслуживают лишь два из них.
- Первый - метод **read**, читающий весь файл целиком, если был вызван без аргументов, и `n` символов, если был вызван с аргументом (целым числом `n`).



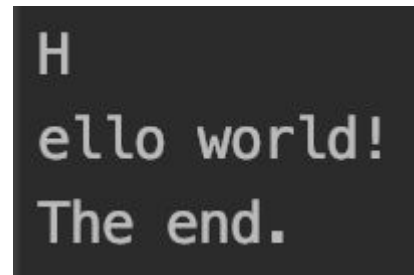
text.txt

```
1 Hello world!  
2 The end.
```



read_file.py

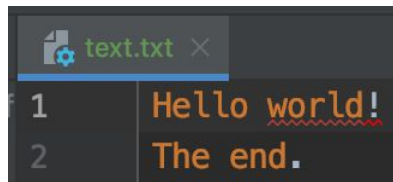
```
1 f = open('L3/text.txt')  
2 print(f.read(1))  
3 print(f.read())  
4
```



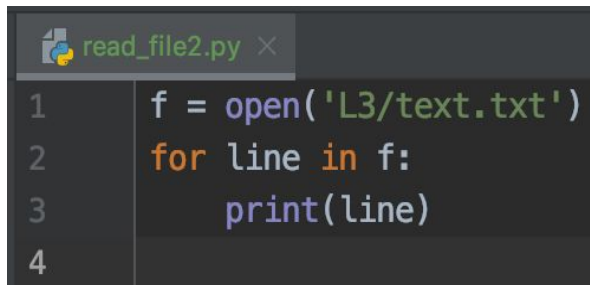
```
H  
ello world!  
The end.
```

Работа с файлами. чтение

- Открыли мы файл, а теперь мы хотим прочитать из него информацию. Для этого есть несколько способов, но большого интереса заслуживают лишь два из них.
- Ещё один способ сделать это - прочитать файл построчно, воспользовавшись циклом **for**.

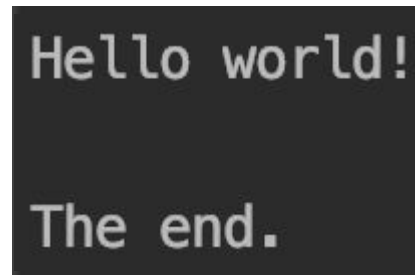


A screenshot of a text editor window titled 'text.txt'. The editor shows two lines of text: '1 Hello world!' and '2 The end.' The text is in a monospaced font, with 'Hello world!' on the first line and 'The end.' on the second line.



```
1 f = open('L3/text.txt')
2 for line in f:
3     print(line)
4
```

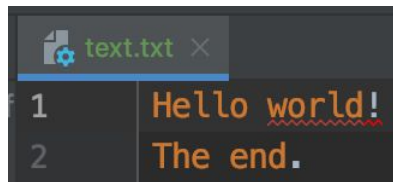
A screenshot of a Python script editor window titled 'read_file2.py'. The script contains three lines of code: 'f = open('L3/text.txt')', 'for line in f:', and 'print(line)'. The code is in a monospaced font, with line numbers 1, 2, 3, and 4 visible on the left side.



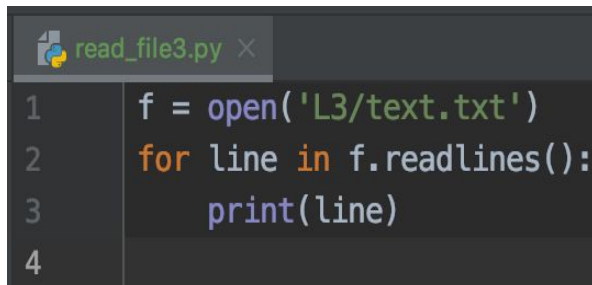
A screenshot of the output of the Python script. It shows two lines of text: 'Hello world!' and 'The end.' The text is in a monospaced font, with 'Hello world!' on the first line and 'The end.' on the second line.

Работа с файлами. чтение

- Открыли мы файл, а теперь мы хотим прочитать из него информацию. Для этого есть несколько способов, но большого интереса заслуживают лишь два из них.
- Ещё один способ сделать это - прочитать файл построчно, воспользовавшись циклом **for**.

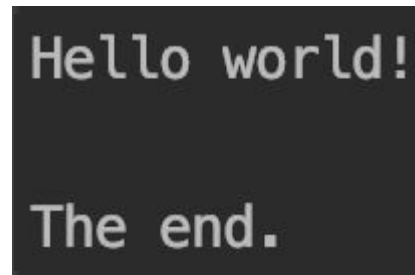


A screenshot of a text editor window titled 'text.txt'. The editor shows two lines of text: 'Hello world!' on the first line and 'The end.' on the second line. The text is in a monospaced font, with 'Hello world!' in orange and 'The end.' in a lighter orange.



```
1 f = open('L3/text.txt')
2 for line in f.readlines():
3     print(line)
4
```

A screenshot of a Python script editor window titled 'read_file3.py'. The script contains four lines of code: line 1 is 'f = open('L3/text.txt')', line 2 is 'for line in f.readlines():', line 3 is ' print(line)', and line 4 is an empty line.



A screenshot showing the output of the Python script. It displays two lines of text: 'Hello world!' on the first line and 'The end.' on the second line. The text is in a monospaced font, with 'Hello world!' in orange and 'The end.' in a lighter orange.

Работа с файлами. Запись

- После окончания работы с файлом его *обязательно нужно закрыть(?)* с помощью метода **close**.

```
write_file.py ×
1 f = open("L3/text2.txt", "w")
2 lst = [str(i) + " " + str(i-1) for i in range(20)]
3 for idx in lst:
4     f.write(idx + '\n')
5 f.close()
6
```

```
text2.txt ×
1 0 -1
2 1 0
3 2 1
4 3 2
5 4 3
6 5 4
7 6 5
8 7 6
9 8 7
10 9 8
11 10 9
```

Работа с файлами. Титаник

- Набор данных включает в себя два CSV-файла. Файл **train.csv** представляет собой обучающий набор, файл **test.csv** — тестовый набор. Обучающий набор содержит признак **Survived** для каждого пассажира, обозначающий, выжил данный пассажир или нет (0 для умерших, 1 для выживших).
- Каждая строка наборов данных содержит следующие поля:
 - **Pclass** — класс пассажира (1 — высший, 2 — средний, 3 — низший);
 - **Name** — имя;
 - **Sex** — пол;
 - **Age** — возраст;
 - **SibSp** — количество братьев, сестер, сводных братьев, сводных сестер, супругов на борту титаника;
 - **Parch** — количество родителей, детей (в том числе приемных) на борту титаника;
 - **Ticket** — номер билета;
 - **Fare** — плата за проезд;
 - **Cabin** — каюта;
 - **Embarked** — порт посадки (C — Шербур; Q — Квинстаун; S — Саутгемптон).

В поле **Age** приводится количество полных лет. Для детей меньше 1 года — дробное. Если возраст не известен точно, то указано примерное значение в формате **xx.5**.

Домашнее задание

- TODO
- TODO

Источники

- <http://pythontutor.ru/>
- <https://e-maxx.ru/bookz/files/cormen.pdf>
- <https://unicode-table.com/ru/emoji/>
- <https://habr.com/ru/post/478636/>
- <https://inf1.info/binarynotation>
- <https://pythonz.net/references/named/str/>
- <https://pythonworld.ru/typy-dannyx-v-python/stroki-funkcii-i-metody-strok.htm>
- <https://www.kaggle.com/hesh97/titanicdataset-traincsv#>
- <https://pythonworld.ru/typy-dannyx-v-python/fajly-rabota-s-fajlami.html>
- <https://ansmirnov.ru/kaggle-titanic-dataset/>