

Занятие 10

Работа с файлами

Что напечатает программа?

```
dct = dict.fromkeys([0,-1,1,0,-1,1,0,-1,1],0)
print(len(dct))
print(sorted(dct, key = lambda x:-abs(x)))
#-----
tcid = {1:{1:{1:{1:1111}}}}
print(len(tcid))
print(tcid[1])
print(tcid[1][1][1][1])
```

Задача 9-1

Дан генетический код ДНК (строка, состоящая из букв G, C, T, A)

Постройте РНК, используя принцип замены букв:

- $G \rightarrow C$;
- $C \rightarrow G$;
- $T \rightarrow A$;
- $A \rightarrow T$

Напишите функцию, которая на вход получает ДНК, и возвращает РНК, для этого постройте словарь для замены букв.

Например:

Вход: GGCTAA Результат: CCGATT

Задача 9-2

Напишите программу, которая определяет и печатает «похожие» слова.

Слово называется **похожим** на другое слово, если его гласные буквы находятся там же, где находятся гласные буквы другого слова, например: дорога и пароход - похожие слова (гласные буквы на втором, четвертом и шестом местах), станок и прыжок - похожие слова, питон и удав непохожие слова. Считаем, что в русском языке 10 гласных букв (а, у, о, ы, и, э, я, ю, ё, е).

Ввод: x – первое слово, например, питон.

n – количество слов для сравнения, например 6.

Дальше вводятся 6 слов, например: поросенок, титан, итог, лавка, погост, кино.

Вывод - слова, **похожие** на питон:

титан, погост, кино

Задание 9-3

Произвести частотный анализ текста из файла.

Сосчитать с помощью словаря и функции `get` сколько раз встречается каждый символ в тексте (включая буквы, цифры и служебные символы, включая пробелы), не учитывая регистр.

Отсортировать по убыванию и напечатать первые 10 символов, и их частоты. При равенстве частот отсортировать символы в алфавитном порядке

Например, текст «Мама мыла раму»:

а – 4

м – 4

л – 1

И т.д.

import collections

collections.Counter - вид словаря, который позволяет нам считать количество неизменяемых объектов (в большинстве случаев, [строки](#)).

collections.deque(iterable, [maxlen]) - создаёт очередь из итерируемого объекта с максимальной длиной maxlen. Очереди очень похожи на списки, за исключением того, что добавлять и удалять элементы можно либо справа, либо слева. (appendleft, popleft)

collections.OrderedDict - ещё один похожий на словарь объект, но он помнит порядок, в котором ему были даны ключи (после версии 3.6 любой словарь помнит этот порядок)

collections.namedtuple позволяет создать тип данных, ведущий себя как кортеж, с тем дополнением, что каждому элементу присваивается имя, по которому можно в дальнейшем получать доступ

`collections.Counter`

```
import collections
```

```
a = collections.Counter('aabbbbccccdddeeeeabcdef')
```

```
print(f"{a=}")
```

```
b = dict(a)
```

```
print(f"{b=}")
```

```
a=Counter({'c': 5, 'd': 5, 'e': 5, 'b': 4, 'a': 3, 'f': 1})
```

```
b={'a': 3, 'b': 4, 'c': 5, 'd': 5, 'e': 5, 'f': 1}
```

Вложенные словари

Ключами (keys) словаря могут быть только неизменяемые объекты: числа, строки, кортежи, True, False и некоторые другие.

```
{True:1, False:2, 1:2, '1':2, frozenset((1,2)):[1,2]}
```

Значениями (values) может быть все, что угодно: числа, строки, кортежи, True, False, а также: списки, множества, функции, лямбда функции...

```
a = {True:abs, False:(lambda x:x * x), 3:{1,2,3}, '1':[1,2], 'Город':'Санкт-Петербург'}
```


Задание

Дан словарь, имеет **три** уровня вложенности, например:

```
dct = {1 : {11 : {111 : 1111}}, 2 : {22 : {222 : 2222}}, 3: {33 : 333}, 4:444}
```

Напечатайте все ключи и элементы всех словарей.

Например:

```
1 : {11 : {111 : 1111}}
```

```
11 : {111 : 1111}
```

```
111 : 1111
```

и т.д.

Проверить, является ли элемент словарем, можно с помощью функции

```
if type(v) == dict
```

Работа с файлами

TXT

.txt — это формат файлов, который содержит текст, упорядоченный по строкам.

Текстовые файлы отличаются от двоичных файлов, содержащих данные, не предназначенные для интерпретирования в качестве текста (закодированный звук или изображение).

.py — это тоже текстовые файлы)

Что мы можем делать с файлом?

- Открыть

- Прочитать

- Дописать

- Переписать

- Заккрыть!!!

Открытие файла

Прежде, чем работать с файлом, его надо открыть.

Для этой задачи есть встроенная функция `open`:

```
f = open("test.txt", encoding="utf-8")
```

Результатом работы функция `open` возвращает специальный объект, который позволяет работать с файлом (файловый дескриптор)

Создайте в PyCharm текстовый файл `test.txt`, введите туда 4-5 строк:

First string

Second string

Третья строка

Четвертая строка

Синтаксис функции open()

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

Параметры:

- **file** - абсолютное или относительное значение пути к файлу или файловый дескриптор открываемого файла.
- **mode** - необязательно, строка, которая указывает режим, в котором открывается файл. По умолчанию 'r'.
- **buffering** - необязательно, целое число, используемое для установки политики буферизации.
- **encoding** - необязательно, кодировка, используемая для декодирования или кодирования файла.
- **errors** - необязательно, строка, которая указывает, как должны обрабатываться ошибки кодирования и декодирования. Не используется в бинарном режиме
- **newline** - необязательно, режим перевода строк. Варианты: None, '\n', '\r' и '\r\n'. Следует использовать только для текстовых файлов.
- **closefd** - необязательно, bool, флаг закрытия файлового дескриптора.
- **opener** - необязательно, пользовательский объект, возвращающий открытый дескриптор файла.

Имя файла, какой файл, что делать

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, mode, в котором мы будем открывать файл. Режим обычно состоит из двух букв, первой является тип файла - текстовый или бинарный, в котором мы хотим открыть файл, а второй указывает, что именно мы хотим сделать с файлом.

Третий аргумент — кодировка файла

Первая буква режима:

"b" - открытие в двоичном режиме.

"t" - открытие в текстовом режиме (является значением по умолчанию).

Второй буква режима:

"r" - открытие на чтение (является значением по умолчанию).

"w" - открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.

"x" - эксклюзивное создание (открытие на запись), бросается исключение `FileExistsError`, если файл уже существует.

"a" - открытие на дозапись, информация добавляется в конец файла.

"+" - открытие на чтение и запись

Примеры

Режим "w" открывает файл только для записи.

Перезаписывает файл, если файл существует.

Если файл не существует, создает новый файл для записи.

```
f = open("test.txt", mode="w" encoding="utf-8")
```

Открывает файл в бинарном режиме для записи и чтения.

Перезаписывает существующий файл, если файл существует.

Если файл не существует, создается новый файл для чтения и записи.

```
f = open("music.mp3", mode="wb+")
```

По всем режимам см. [документацию open\(\)](#)

Заккрыть файл

После того как вы сделали всю необходимую работу с файлом - его следует закрыть.

```
f = open("text.txt", encoding="utf-8")  
# какие-то действия  
f.close()
```


Чтение файла

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).

```
f = open("test.txt", "r")
```

```
print(f.read(5))  
print(f.read(5))  
print(f.read(4))  
print(f.read())
```

```
f.close()
```

Функция `readlines()`

Файлы можно читать не только целиком или посимвольно, но и построчно.

Для этого у объекта файла есть метод `readlines`, который возвращает список из строк файла.

```
f = open("test.txt", "rt")  
print(f.readlines())  
f.close()
```

Обратите внимания, что каждая строка в списке имеет в конце символ ``\\n``.

Задание

Прочитайте содержимое файла с помощью функции `readlines()`

Присвойте ее результат переменной `lst`

Напечатайте пронумерованный список строк.

Постарайтесь избавиться от лишних пустых строк.

Функция `readline()`

Функция `readlines()` загружает все строки целиком и хранит их в оперативной памяти, что может быть очень накладно, если файл занимает много места на жёстком диске.

Можно читать файл построчно с помощью функции `readline()`

```
f = open("text.txt", "rt")  
print(f.readline())  
print(f.readline())  
f.close()
```

Также обратите внимание, что возвращённые строки имеют в конце символ ``\\n``.

Задание

Прочитайте содержимое файла с помощью функции `readline()`
Напечатайте пронумерованный список строк.

Итерирование файла

Ещё один способ прочитать файл построчно - использовать файл как итератор.
Такой вариант считается самым оптимизированным

```
f = open("text.txt")  
for line in f:  
    print(line)  
  
f.close()
```

Запись

Теперь рассмотрим запись в файл.

Для того чтобы можно было записывать информацию в файл, нужно открыть файл в режиме записи.

Для записи в файл используется функция `write`.

При открытии файла на запись из него полностью удаляется предыдущая информация.

```
fout = open("test.txt", "wt")  
fout.write("New string")  
fout.write("Another string")  
fout.close()
```

Если вы откроете файл в текстовом редакторе, то увидите, что строки "New string" и "Another string" склеились.

Так произошло, потому что между ними нет символа перевода строки.

writelines()

Также в файлах, открытых на запись, есть метод `writelines`, который позволяет записать несколько строк в файл

```
f = open("text.txt", "wt")
lines = [
    "New string\n",
    "Another string\n",
]
f.writelines(lines)
f.close()
```


Задание

Прочитать информацию из файла test.txt.

Записать в файл test1.txt, только те строки, которые содержат цифры.

Например:

Hello!

This is the 1st letter.

Bye

Результат: This is the 1st letter.

Задание

Откройте текстовый файл.

Каждый второй знак этого файла перенесите в другой файл.

Задание

Прочитать строки текста из одного файла,
отсортировать слова внутри строки по возрастанию и
записать обновленные строки в другой файл.

print(..., file = f)

Можно использовать print(), если указать **file = файловый объект**

```
f = open('text.txt', 'w', encoding = 'utf-8')
```

```
print(*objects, sep=' ', end='\n', file=f)
```

По умолчанию стандартный вывод на экран, а можно указать file = f

Напишите программу, которая печатает в текстовый файл строки из числа и его квадрата, т.е.

0 0

1 1

2 4

3 9 и т.д.

```
fi = open('file.txt', 'w', encoding = 'utf-8')
```

```
for i in range(5):
```

```
    print(i, i * i, file = fi)
```

Дозапись

Если нужно записать в конец файла какую-то информацию, то можно сделать это, открыв файл в режиме дозаписи.

Все методы, доступные в режиме записи также доступны в режиме дозаписи.

```
f = open("text.txt", "at")
```

```
f.write("First string\n")
```

```
lines = [  
    "Second string\n",  
    "Third string\n",  
]
```

```
f.writelines(lines)
```

```
f.close()
```

```
# Давайте проверим это
```

Запись с возможностью чтения

Иногда нужно открыть файл с возможностью и записи, и чтения.

В Python есть два режима:

- * Запись с возможностью чтения ("w+")
- * Чтение с возможностью записи ("r+")

На первый взгляд кажется, что они ничем не отличаются, но это не так.

При открытии файла на запись (w+) с возможностью чтения из файла полностью удаляется вся информация.

with ... as ... - менеджер контекста

```
with open('file.txt', 'r', encoding = 'utf-8') as fi:
```

```
    print(fi.readlines())
```

файл закрывается автоматически

```
# откройте файл с помощью with и напечатайте его содержимое
```

```
print(fi.readlines() )
```

Задание

Прочитать строки текста из одного файла,
отсортировать слова внутри строки по возрастанию и
записать обновленные строки в другой файл.

Модуль openpyxl

[Модуль openpyxl](#) - это библиотека Python для чтения/записи форматов Office Open XML (файлов Excel 2010) с расширениями xlsx/xlsm/xltx/xltn.

Не входит в стандартную библиотеку, необходимо его установить

IDLE: `pip install openpyxl`

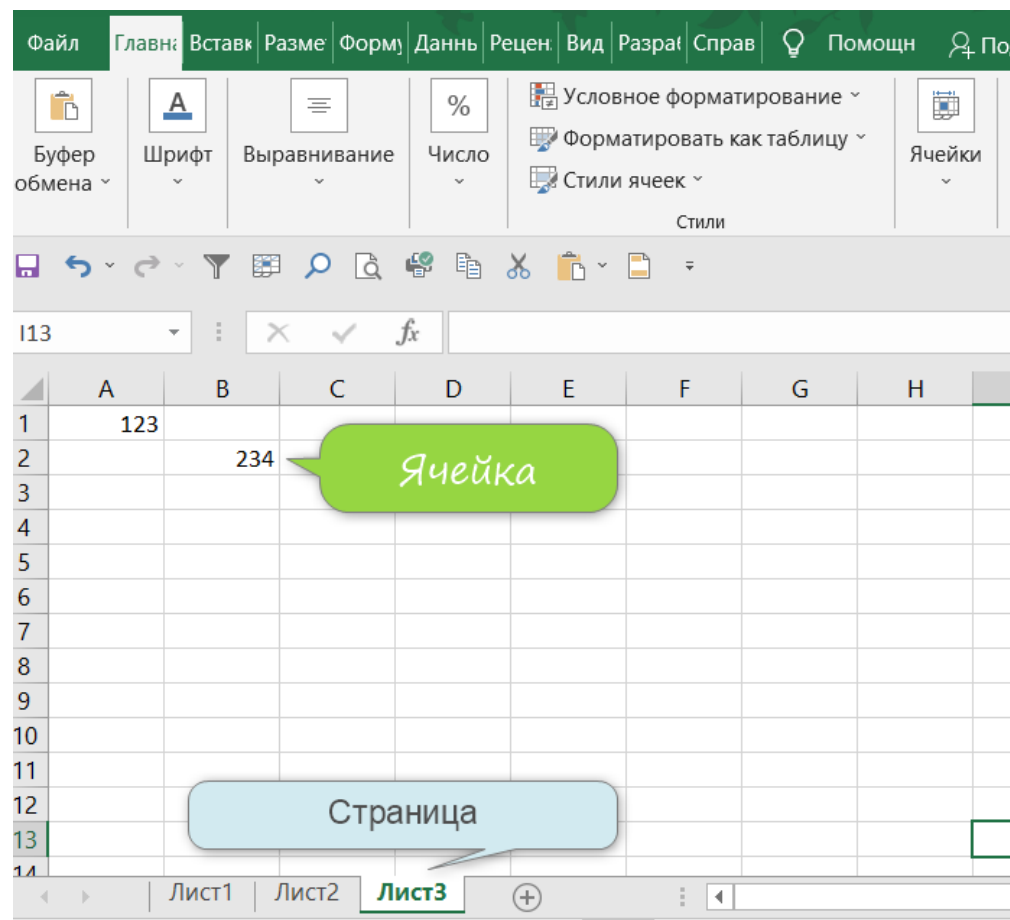
PyCharm: File/Settings/Project.../Python Interpreter/ + / набрать openpyxl / Install Package

Попробуйте в консоли `import openpyxl`

Excel файл - книга

Функции openruхl:

- Книга
- Страницы
- Ячейки



Создание книги Excel (workbook)

```
import openpyxl  
from openpyxl import Workbook  
wb = Workbook() # создаем экземпляр класса Workbook  
wb.save('test.xlsx') # сразу его записываем пустой
```

Создайте файл

Рабочие страницы

```
import openpyxl

wb = openpyxl.load_workbook('test.xlsx') # Открываем книгу
print(wb.sheetnames)                    # Список листов

ws = wb.active                          # Кто активный рабочий лист?
print(ws.title)                         # Каков его title?

wb.create_sheet('New')                  # Создаем новый лист
ws3 = wb["New"]                         # Другой лист
print(ws3)                             # Посмотрим
```

Рабочие страницы

```
print(wb.sheetnames)
```

список листов

```
wb.active = wb['New']
```

назначим другой активный лист

```
wb.remove(ws)
```

удаление рабочего листа

```
print(wb.sheetnames)
```

```
wb.save('test.xlsx')
```

сохраняем книгу

Доступ к ячейкам - cell

`sheet['A1'].value` # Значение, которое хранится в ячейке (Retrieve the value of a certain cell)

`sheet['A1'].value = 123` # Поместить число в ячейку

`c = sheet['B2']` # выбрать ячейку (Select element 'B2' of your sheet)

`c.row` # номер строки (Retrieve the row number of your element)

`c.column` # номер колонки (Retrieve the column letter of your element)

`c.coordinate` # координаты ячейки

#Запишите в ячейку A1 число 100, в ячейку B2 – 200, сложите их значения
и результат запишите в C3

Работа с ячейками - cell

Retrieve cell value

sheet.cell(row=1, column=2).value # значение ячейки по номеру строки и колонки

Print out values in column 2 / печатаем колонку номер 2

for i in range(1, 4):

 print(i, sheet.cell(row=i, column=2).value)

Print row per row / печатаем строку за строкой

for cellObj in sheet['A1':'C3']:

 for cell in cellObj:

 print(cell.coordinate, cell.value)

 print('--- END ---')

#Напечатайте

max_row max_col

Retrieve the maximum amount of rows

sheet.max_row # максимальная строка, где есть информация

Retrieve the maximum amount of columns

sheet.max_column # максимальная колонка, где есть информация

Как напечатать все ячейки листа

```
for i in range(ws.max_row):  
    for j in range(ws.max_column):  
        print(i + 1, j + 1, ws.cell(row = i + 1, column = j + 1).value)
```

Выполните этот код

Основные функции

##		Команда	Что делает
1		<code>import openpyxl</code>	Импортирует модуль openpyxl
2	B	<code>from openpyxl import Workbook</code>	Загрузка класса Workbook
3	B	<code>wb = Workbook()</code>	Создаем рабочую книгу
4	B	<code>wb.save('test.xlsx')</code>	Сохраняем файл
5	B	<code>wb = openpyxl.load_workbook("test.xlsx")</code>	Загружаем файл
6	S	<code>ws = wb.active</code>	Определяем активный рабочий лист
7	S	<code>wb.active = ws</code>	Переопределяем активный рабочий лис
8	S	<code>ws.title</code>	Имя листа
9	B	<code>wb.sheetnames</code>	Список листов книги
10	S	<code>wb.create_sheet("Newsheet")</code>	Создание нового листа
11	S	<code>wb.remove(ws)</code>	Удаление листа
12	C	<code>ws['A1'].value</code>	Значение ячейки (cell)
13	C	<code>c = ws['B2']</code>	Присвоить с ячейку (не значение!!!)
14	C	<code>c.row</code>	Номер строки
15	C	<code>c.column</code>	Номер колонки
16	C	<code>c.coordinate</code>	Координаты ячейки ('A1')
17	C	<code>ws.cell(row = 1, column = 2).value</code>	Значение ячейки по номеру строки и колонки (нумерация с 1)
18	S	<code>ws.max_row</code>	Максимальная строка с данными
19	S	<code>ws.max_column</code>	Максимальная колонка с данными

Задание

1. Создайте Эксель файл

Откройте его с помощью openruhl

Напечатайте список его страниц

Создайте список кортежей (имя страницы, количество заполненных ячеек в странице)

Напечатайте его

2. Отсортируйте его по алфавиту названия страниц

3. Отсортируйте его по количеству заполненных ячеек в странице в убывающем порядке

Задание

Дан Эксельный файл.

В нем список людей и их ЗП.

После последней фамилии в списке создайте строчку ИТОГО и напечатайте сумму ЗП.

Задание

Дан эксельный файл, содержащий список людей и их премию.

Загрузите этот список, отсортируйте его по алфавиту.

Создайте еще одну страницу в файле и запишите туда отсортированный список с премиями.

Задача 10-1

Есть Excel файл, в первом листе которого находится информация по выработке каждого программиста.

В первой строке ФИО, во второй ячейке результат его работы за один день.

Дана информация за несколько дней по разным людям.

Необходимо во втором листе сформировать суммарные итоги по каждому сотруднику и ИТОГО по списку по всем сотрудникам.

Например:

Иванов 100

Петров 400

Иванов 200

Во втором листе должно быть:

Иванов 300

Петров 400

ИТОГО 700

Задача 10-2

Дан Excel файл. В первом листе фамилии и выработка по дням за один период времени.

Во втором тоже самое за другой период времени.

Фамилии могут быть в другом порядке.

Фамилии могут повторяться в каждом листе несколько раз, а могут быть разными.

Необходимо создать третий лист, который суммирует выработку из первых двух.

Список должен быть отсортирован по фамилиям.

Задача 10-3

Дан эксельный файл со списком людей и результатами их работы (люди не повторяются).

Необходимо сформировать еще один лист со следующей информацией:

- Минимальное значение
- Максимальное значение
- Среднее арифметическое
- Медиана (срединное значение).

Для списка с нечетным количеством членов, это значение посередине упорядоченного списка (1, 3, 5, 7, 9) -> 5.

Для четного количества – полусумма средних чисел (1, 3, 5, 7) -> $(3 + 5) / 2 = 4$