

Занятие 11

Работа с файлами

time, datetime, calendar

Что напечатает?

```
f_out = open('test.txt', 'w')  
f_out.write('1')  
f_out.close()  
f_in = open('test.txt')  
print(111, len(f.read()))  
print(222, len(f.read()))
```

Задача 10-1

Есть текстовый файл 'test1.txt'. Надо прочитать содержимое этого файла и создать файл 'test2.txt'. В нем должны быть строки из первого файла, но в обратном порядке. В каждой строке должен быть поменян порядок слов на противоположный.

Например, исходный файл:

Мой дядя самых честных правил

Когда не в шутку занемог

Результат:

занемог шутку в не Когда

правил честных самых дядя Мой

Задача 10-2

Дан эксельный файл. На странице находится список сотрудников и их ЗП.

Надо создать еще одну страницу в этом файле и поместить туда отсортированный список фамилий и их ЗП, а в последней строчке поместить слово ИТОГО: и сумму всех ЗП. Сортировка по убыванию ЗП сотрудников.

Например, исходная страница файла:

Сидоров 100

Петров 200

Иванов 300

Результат:

Иванов 300

Петров 200

Сидоров 100

ИТОГО: 600

Задача 10-3

Дан исходный эксельный файл со списком людей и их ЗП.

Следует создать еще одну страницу со статистическими данными исходного списка.

Например, исходная страница:

Сидоров	100
Петров	200
Иванов	300
Федоров	1000

Результат:

Максимальное значение	1000
Минимальное значение	100
Сумма	1600
Среднее арифметическое	400
Медиана	250

Работа с файлами

Общий алгоритм работы с текстовыми файлами

1. Открыть файл
2. Считать информацию в какую-то конструкция Python, словарь, список, кортеж и т.д.
3. Обработать информацию, напечатать результат, ввести результат в этот или другой файл
4. Закрыть файл

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 test1.txt.

Записать в файл test2.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 все буквы всех слов
и дописать их в конец файла, каждая буква на своей строке.

Мама мыла раму

аааа

л

мммм

р

у

ы

Модуль openpyxl

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

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`

Общий алгоритм работы с эксельными файлами

1. Открываем книгу (wb) (или уже существующий, или делаем новый файл)
2. Выбираем лист (ws) (или создаем новый, или делаем активным по имени)
3. Считываем данные из ячеек и помещаем их в список или в словарь или куда-то еще
4. Делаем нужные действия (вычисляем, сортируем, что-то еще), т.е. храним данные в Excel, а обрабатываем их в Python
5. Записываем их в какую-то страницу
6. Сохраняем wb

Создание книги Excel (workbook)

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


Рабочие листы

```
import openpyxl
wb = openpyxl.load_workbook("test.xlsx")
print(wb.sheetnames)           # список листов
ws = wb.active                 # Кто активный рабочий лист?
print(ws.title)               # Посмотрим
ws.title = "NewPage"          # Изменим его имя
print(ws.title)               # Проверим
ws3 = wb["NewPage"]           # Другой лист
print(ws3)
print(wb.sheetnames)          # список листов
wb.active = wb['NewPage']      # назначим активный лист
wb.remove(ws)                  # удаление рабочего листа
print(wb.sheetnames)
wb.save('test.xlsx')
```

Доступ к ячейкам

Пусть sheet один из листов

Retrieve the value of a certain cell

sheet['A1'].value # Значение, которое хранится в ячейке

Select element 'B2' of your sheet

c = sheet['B2'] # выбрать ячейку

Retrieve the row number of your element

c.row # номер строки

Retrieve the column letter of your element

c.column # номер колонки

Retrieve the coordinates of the cell

c.coordinate # координаты ячейки

Работа с ячейками

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 ---')

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)
```

Как можно напечатать все ячейки всех листов?

```
for sh in wb.sheetnames:  
    ws = wb[sh]  
    print(ws.title, '-----')  
    for i in range(ws.max_row):  
        for j in range(ws.max_column):  
            print(i + 1, j + 1, ws.cell(i + 1, j + 1).value)
```

Выполните эту программу. Что она напечатает?

Основные функции

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

Еще немного

##	Команда	Что делает
20	<pre>source_page = wb.active target_page = wb.copy_worksheet(source_page)</pre>	Копирование листа
21	<pre>ws.append([111, 'Текст', 333])</pre>	Добавление списка в строчку после последней
22	<pre>ws.append({1:'123', 3:'345'})</pre>	Добавление словаря в строчку после последней. Ключи – номера столбцов
23	<pre>col_range = ws['C:D']</pre>	Срез. Все доступные ячейки в колонках от C до D
24	<pre>row_range = ws[5:10]</pre>	Срез. Все доступные ячейки в строках от 5 до 10

Добавьте строку и словарь в эксельный файл и напечатайте их

Задание

Введите два числа n – количество рядов матрицы, m – количество столбцов матрицы.

Заполните страницу Excel файла двумерной матрицей.

a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18} a_{19} a_{110} a_{111} ... a_{1m}

a_{21} a_{22} a_{23} a_{2m}

.....

a_{n1} a_{n2} a_{n3} a_{nm}

Time, Datetime, Calendar

Time – удобен для оценки длительности программы, нахождения самых медленных и неэффективных ее частей

Datetime – огромный набор классов и функций для решения разнообразных задач с датами и временем

Calendar – модуль для работы с календарями

Модуль time

`import time` # Удобен для технической работы со временем.

`t0 = time.time()`

`t1 = time.time()`

`print(t1 - t0, t0, t1)` # Выполните эти команды

`import time` # Например, оценить время работы программы:

`x = 100000`

`for i in range(0, 10):`

`t0 = time.time()`

`su = 0`

`for j in range(i * x):`

`su += j`

`t1 = time.time()`

`print(i, t1 - t0)`

Выполните эту программу, увеличивая значение x, чтобы визуальнo увидеть замедление

Некоторые функции модуля time

##	Функция	Что делает
1	<code>import time</code>	Импортирует модуль
2	<code>time.time()</code>	Текущее время в секундах
3	<code>time.ctime()</code>	Текущее время
4	<code>time.ctime(t)</code>	Дата и время из количества секунд t
5	<code>time.sleep()</code>	Приостанавливает выполнение программы на ... секунд

Напишите программу, которая запоминает момент начала программы, напечатайте дату и время с помощью функции `ctime`, затем «усыпите» программу на 5 секунд, потом запомните момент завершения программы, напечатайте дату и время. И напечатайте разницу между моментами окончания и начала.

Задание

Напишите программу, в которой используется две функции. В одной программа «спит» 2 секунды, в другой – 3 секунды.

Пусть каждая функция возвращает время, которое она «проспала».

Главная программа запускает цикл от 0 до 10, если число четное, то запускает функцию с 2 секундами, если нечетное, то функцию с 3 секундами.

Накапливает сон обеих функций отдельно и печатает две суммы.

Модуль datetime

- Класс **datetime.date**(year, month, day) - стандартная дата. Атрибуты: year, month, day. Неизменяемый объект.
- Класс **datetime.time**(hour=0, minute=0, second=0, microsecond=0) - стандартное время, не зависит от даты. Атрибуты: hour, minute, second, microsecond.
- Класс **datetime.timedelta**(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0) - разница между двумя моментами времени, с точностью до микросекунд.
- Класс **datetime.datetime**(year, month, day, hour=0, minute=0, second=0, microsecond=0) - комбинация даты и времени.
- **datetime.datetime.today()** - объект datetime из текущей даты и времени.
- Сохраните в переменную b «сегодня» и напечатайте все его атрибуты, год, месяц, день и т.д.

Преобразование строки в дату – `strptime()`

- # Воспользуемся библиотекой `datetime` методом `strptime`
from `datetime` **import** `datetime`
`print(datetime.strptime('22 04 2020 19:33', '%d %m %Y %H:%M'))`
- 2020-04-22 19:33:00

Преобразование даты и времени в строки – strftime()

```
from datetime import date, time
```

```
my_date = date(2021, 8, 10)
```

```
my_time = time(7, 18, 34)
```

```
print(my_date)                # вывод в ISO формате
```

```
print(my_time)                # вывод в ISO формате
```

```
print(my_date.strftime('%d/%m/%y'))    # форматированный вывод даты
```

```
print(my_date.strftime('%A %d, %B %Y')) # форматированный вывод даты
```

```
print(my_time.strftime('%H.%M.%S'))    # форматированный вывод
```

Выполните эти операторы. Поменяйте разделители на другие.
Замените буквы на y на Y, A на a, B на b

Получение текущей даты и времени

Одним из классов, определенных в модуле `datetime`, является класс `datetime`.

После импортирования класса мы можем применить метод `now()` для создания объекта `datetime`, содержащего текущие локальные дату и время.

```
from datetime import datetime
datetime_object = datetime.now()
t0 = datetime_object
# После небольшой паузы
datetime_object = datetime.now()
t1 = datetime_object
print(t0, t1, t0 == t1, t0 < t1)
```

Что напечатает?

Создание даты и времени из кортежа

```
from datetime import datetime  
a = datetime(2017, 11, 28, 23, 55, 59)  
print("year =", a.year)  
print("month =", a.month)  
print("day =", a.day)  
  
print("hour =", a.hour)  
print("minute =", a.minute)  
print("second =", a.second)
```

Выполните это для сегодня

Получение текущей даты – еще один способ

Метод `today()`, определенный в классе `date`, чтобы получить объект `date`, содержащий текущую локальную дату.

```
from datetime import date  
today = date.today()  
print("Current date =", today)  
2022-04-22
```

Конструирование даты

```
import datetime
```

```
dt = datetime.date(2020, 6, 29)
```

```
print(dt)
```

```
2020-06-29
```

```
# получение значений
```

```
print("Current year:", dt.year)
```

```
print("Current month:", dt.month)
```

```
print("Current day:", dt.day)
```

Форматирование даты

```
from datetime import datetime
```

```
now = datetime.now()
```

```
t = now.strftime("H:M:S")
```

```
print("time:", t)
```

```
time: 15:00:24
```

```
s1 = now.strftime("%m/%d/%Y, H:M:S")
```

```
print("s1:", s1)
```

```
s1: 04/22/2022, 15:00:24
```

```
s2 = now.strftime("%d/%m/%Y, H:M:S")
```

```
# dd/mm/YY H:M:S format
```

```
print("s2:", s2)
```

Основные коды для определения формата:

%Y — год [0001, ..., 2018, 2019, ..., 9999]

%m — месяц [01, 02, ..., 11, 12]

%d — день [01, 02, ..., 30, 31]

%H — час [00, 01, ..., 22, 23]

%M — минута [00, 01, ..., 58, 59]

%S — секунда [00, 01, ..., 58, 59]

Локализация

```
from datetime import date  
import locale
```

```
locale.setlocale(locale.LC_ALL, "ru") # иногда используется 'ru_RU.UTF-8'
```

```
my_date = date(2021, 8, 10)  
print(my_date.strftime("%A %d, %B %Y"))
```

Задание

```
from datetime import date  
import locale
```

```
locale.setlocale(locale.LC_ALL, 'ru') #, 'en_EN.UTF-8')  
birthday = date(1992, 10, 6)
```

```
print('Название месяца:', birthday.strftime('%B'))  
print('Название дня недели:', birthday.strftime('%A'))  
print('Год:', birthday.strftime('%Y'))  
print('Месяц:', birthday.strftime('%m'))  
print('День:', birthday.strftime('%d'))
```

Поставьте свой день рождения и выполните эти операторы

import calendar

Функция **weekday(year, month, day)** возвращает день недели в виде целого числа (по умолчанию 0 – понедельник, 6 – воскресенье) для заданной даты.

Функция **monthrange(year, month)** возвращает день недели первого дня месяца и количество дней в месяце в виде кортежа для указанного года year и месяца month

Функция **monthcalendar(year, month)** возвращает матрицу, представляющую календарь на месяц. Каждая строка матрицы представляет неделю.

Функция **month(year, month, w=0, l=0)** возвращает календарь на месяц в многострочной строке. Аргументами функции являются: year (год), month (месяц), w (ширина столбца даты) и l (количество строк, отводимые на неделю).

Функция **calendar(year, w=2, l=1, c=6, m=3)** возвращает календарь на весь год в виде многострочной строки. Аргументами функции являются: year (год), w (ширина столбца даты) и l (количество строк, отводимые на неделю), c (количество пробелов между столбцом месяца), m (количество столбцов).

Функция **isleap(year)** определяет, является ли год високосным.

Задание

Напишите программу, которая принимает в качестве аргумента номер года.

В качестве результата формирует словарь, в котором считается количество понедельников, вторников и т.д. в этом году.

Дни недели года

```
import calendar
year = int(input("Введите номер года:"))
dw = {0:0, 1:0, 2:0, 3:0, 4:0, 5:0, 6:0}
for month in range(1, 13):
    for day in range(1, calendar.monthrange(year, month)[1] + 1):
        dd = calendar.weekday(year, month, day)
        dw[dd] += 1 # dw[dd] = dw.get(dd, 0) + 1
print(dw)
```

Задача 11-1

Каждый третий четверг каждого месяца билеты в Эрмитаж бесплатны.

Напечатайте список дат в 2024 году, когда вход бесплатен.

Задача 11-2

Дан файл с расширением .txt, содержащий в каждой строке следующую информацию: номер, фамилия, имя, компания, зарплата, разделенные запятыми.

Создайте Эксельный файл, в который перенесите эту информацию, предварительно отсортировав этот список по компании, по фамилии и по имени.

В конце списка добавьте строку: ИТОГО и суммарное значение всех зарплат.

Задача 11-3

Напишите функцию, которая переводит арабские числа в римские.

Например: 2023 -> MMXXIII