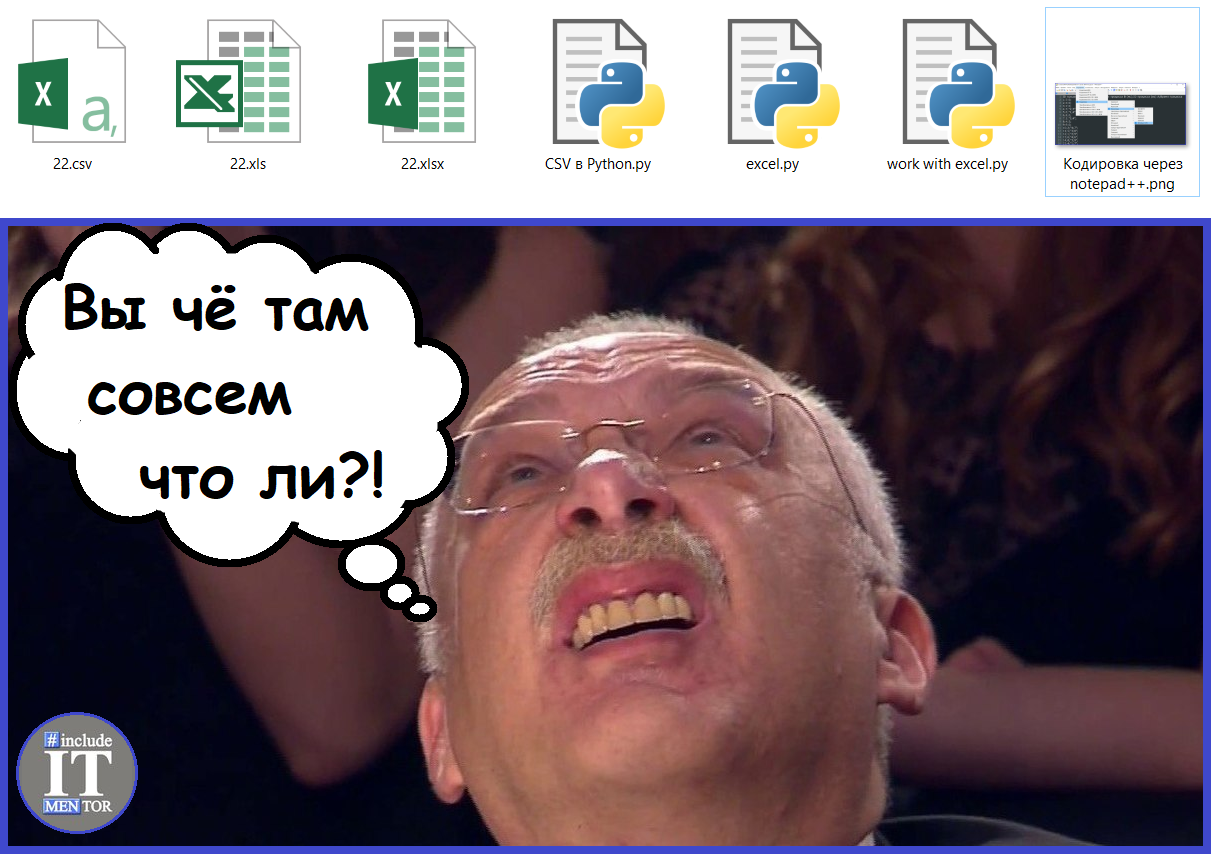
**Парсинг Excel-файлов на Python на примере задачи из ЕГЭ по информатике**

|  |  |  |
| --- | --- | --- |
| [**Информатика**](https://dzen.ru/itmentor) | [**Репетитор IT mentor**](https://t.me/mentor_it) | [**Кирилл Хало**](https://vk.com/impressive_i) |
| **https://dzen.ru/itmentor** | **https://t.me/mentor\_it** | **https://vk.com/impressive\_i** |

****

Сегодня разберем с вами довольное сложное 22 задание. Здесь стоит оговориться. Горе-эксперты ЕГЭ каждый раз выдумывают что-то новое, поэтому никогда не знаешь, что ожидать в следующий раз. Недавно на занятиях с учениками попалась задача, которая не решается обычными формулами, встроенными в Excel (во всяком случае я не знаю, как её автоматизировать средствами ТОЛЬКО Excel). И тут повезло, что таблица была небольшой, поэтому можно было решить задачу руками. Но я сразу же задумался над тем, а что если записей в ней было бы гораздо больше? Что если руками решать было бы не целесообразно, потому что это заняло бы бесконечно большое время, которого нет на экзамене? Как же тогда автоматизировать решение? Об этом мы сегодня с вами и поговорим... Заваривайте чай, здесь нужно будет посидеть и подумать.

**Задача**

В файле 22.xlsx содержится информация о совокупности *N* вычислительных процессов, которые могут выполняться параллельно или последовательно. Будем говорить, что процесс B зависит от процесса A, если для выполнения процесса B необходимы результаты выполнения процесса A. В этом случае процессы могут выполняться только последовательно.

Информация о процессах представлена в файле в виде таблицы. В первой строке таблицы указан идентификатор процесса (ID), во второй строке таблицы  — время его выполнения в миллисекундах, в третьей строке перечислены с разделителем «;» ID процессов, от которых зависит данный процесс. Если процесс является независимым, то в таблице указано значение 0.

Определите минимальное время, через которое завершится выполнение всей совокупности процессов, при условии, что все независимые друг от друга процессы могут выполняться параллельно.

Типовой пример организации данных в файле:

|  |  |  |
| --- | --- | --- |
| ID процесса B | Время выполнения процесса B (мс) | ID процесса (ов) A |
| 1 | 4 | 0 |
| 2 | 3 | 0 |
| 3 | 1 | 1; 2 |
| 4 | 7 | 3 |

**Решение:**

Рассмотрим упрощенную таблицу. По первым двум строкам видно, что процессы ID=1 и ID=2 независимые, они могут выполняться одновременно (параллельно), поэтому завершатся оба эти процесса за 4 миллисекунды. Процесс ID=3 зависит от выполнения процессов ID=1 и ID=2. Поэтому 3й процесс может выполнить только завершения 1го и 2го. Значит на завершение 3го процесса нужно 1 мс + 4 мс = 5 мс. Процесс ID=4 зависит от процесса ID=3, то есть ждет его завершения. Поэтому время до завершения 4го процесса складывается из времени, необходимого для его выполнения (это 7 мс), и времени ожидания зависимых процессов (в нашем случае 5 мс – ожидание завершения 3го процесса). Тогда время до завершения 4го будет 5 мс + 7 мс = 12 мс. Итак, самый длительный процесс 4й, поэтому все процессы завершатся за 12 мс.

Получается, что решить задачу можно с помощью создания дополнительного столбца, в котором будем записывать время до завершения процесса от начала запуска программы, инициализирующей все процессы.

Таблица с данными в задании предоставляется нам в виде файла с расширением .xlsx. Для решения необходимо будет отсортировать данные таким образом, чтобы процессы, от которых зависит наш текущий процесс, были описаны в таблице выше. Ну а в самом начале при такой сортировке должны оказаться несколько процессов, которые на зависят от других. Выполнить такую сортировку в Excel можно с помощью функции «Фильтр».

Тогда посчитать четвертый столбец (время полного завершения) можно с помощью функции, описанной алгоритмически примерно так:

- функция полного выполнения, возвращает время полного завершения процесса.  


Таблица в примере более сложная. Давайте посмотрим на неё.

|  |  |  |  |
| --- | --- | --- | --- |
| ID процесса B | Время выполнения процесса B (мс) | ID процесса (ов) A | Время процесса |
| 1 | 7 | 0 | 7 |
| 2 | 4 | 0 | 4 |
| 3 | 7 | 0 | 7 |
| 4 | 3 | 2;3 | 3 + max(4; 7) = 10 |
| 5 | 6 | 1;4 | 6 + max(7; 10) = 16 |
| 6 | 8 | 0 | 8 |
| 7 | 2 | 2;4 | 2 + max(4; 10) = 12 |
| 8 | 4 | 5 | 4 + max(16) = 20 |
| 9 | 9 | 0 | 9 |
| 10 | 2 | 4;7 | 2 + max(10; 12) = 14 |
| 11 | 5 | 3;8 | 5 + max(4; 20) = 25 |
| 12 | 1 | 2;9 | 1 + max(4; 9) = 10 |
| 13 | 6 | 4;6 | 6 + max(10; 8) = 16 |
| 14 | 8 | 3;9 | 8 + max(7; 9) = 17 |
| 15 | 4 | 1;7 | 4 + max(7;12) = 16 |

Как мы видим, можно решить «вручную», потому что таблица не слишком большая. Но что же делать, если таблица окажется очень большой? Тогда не получится сделать всё это от руки. Попробуем автоматизировать процесс с помощью Python. Нам понадобится считывать данные из **excel**-файла.

Первый метод, который можно осуществить, это парсинг с помощью библиотеки xlrd.   
**xlrd** – это инструмент для чтения **.xls** файлов. Но тогда нам предварительно потребуется сохранить файл в расширении **.xls** вместо **.xlsx**.

Если эта библиотека отсутствует, то её можно установить с помощью команды (в терминал или командную строку, запущенную от имени администратора):

**pip install xlrd**

Документация по библиотеке здесь: <https://xlrd.readthedocs.io/en/latest/api.html#xlrd>

Или здесь: <http://ilnurgi1.ru/docs/python/modules_user/xlrd.html#>

**Замечание**: Чтобы программа работала, файл (.xls) должен находиться в одной папке с программой.

**Код реализации решения через XLRD**

Открытие файла происходит через библиотеку xlrd ([**почитать тут**](http://ilnurgi1.ru/docs/python/modules_user/xlrd.html)или [**почитать тут**](https://xlrd.readthedocs.io/en/latest/api.html#xlrd)). Далее формируется список процессов, где каждый процесс также представляет собой список, в котором: *1 элемент = ID/номер процесса   
2 элемент = собственное время выполнения   
3 элемент = строка, в которой написаны процессы, от которых зависит текущий процесс (разделяется ; )   
4 элемент = полное время до выполнения с учетом всех зависимостей.*

В начале программы заносятся данные из **.xls** файла, а уже затем производится расчет четвертого элемента каждого процесса. Затем ищется максимальное время. Время до завершения самого длительного процесса будет определять время завершения всех процессов задачи.

**import** **xlrd**

workbook = xlrd.open\_workbook("22.xls")

worksheet = workbook.sheet\_by\_index(**0**)

#Внесение данных в cписок

processes = []

**for** row **in** range(**1**, **16**):

process = [**0**,**0**,**0**,**0**]

**for** col **in** range(**0**,**3**):

buf = str(worksheet.cell(row, col).value)

**if** ';' **in** buf:

process[col] = buf

**else**:

process[col] = str(int(float(buf)))

processes.append(process)

# Заполнение времени до завершения процесса

**for** row **in** range(**0**, len(processes)):

print("Текущий процесс ID = {}".format(processes[row][**0**]))

**if** processes[row][**2**] == "0": # Если процесс не зависит от других

processes[row][**3**] = int(processes[row][**1**])

print("{} не зависит от процессов".format(processes[row][**0**]))

**else**: # если процесс зависит от других процессов

# время процесса = время\_выполнения + max(задержек\_зависимых)

# Список процессов от которых зависит текущий рассматриваемый

list\_dependent = processes[row][**2**].split(';')

print("{} зависит от процессов: {}".

format(processes[row][**0**], list\_dependent))

# Список временных задержек зависимых процессов

times\_dependent = [ int(processes[int(ID)-**1**][**3**])

**for** ID **in** list\_dependent ]

# Расчет полного времени текущего процесса

processes[row][**3**] = int(processes[row][**1**]) + max(times\_dependent)

print("Список временных задержек для {}: {}"

.format(processes[row][**0**], times\_dependent))

print("Полное время текущего процесса {}: {}**\n**".

format(processes[row][**0**], processes[row][**3**]))

# Вывод информации

**for** row **in** range(**0**, len(processes)):

print(processes[row])

# Время выполнения всех процессов

execution\_time = max([p[**3**] **for** p **in** processes])

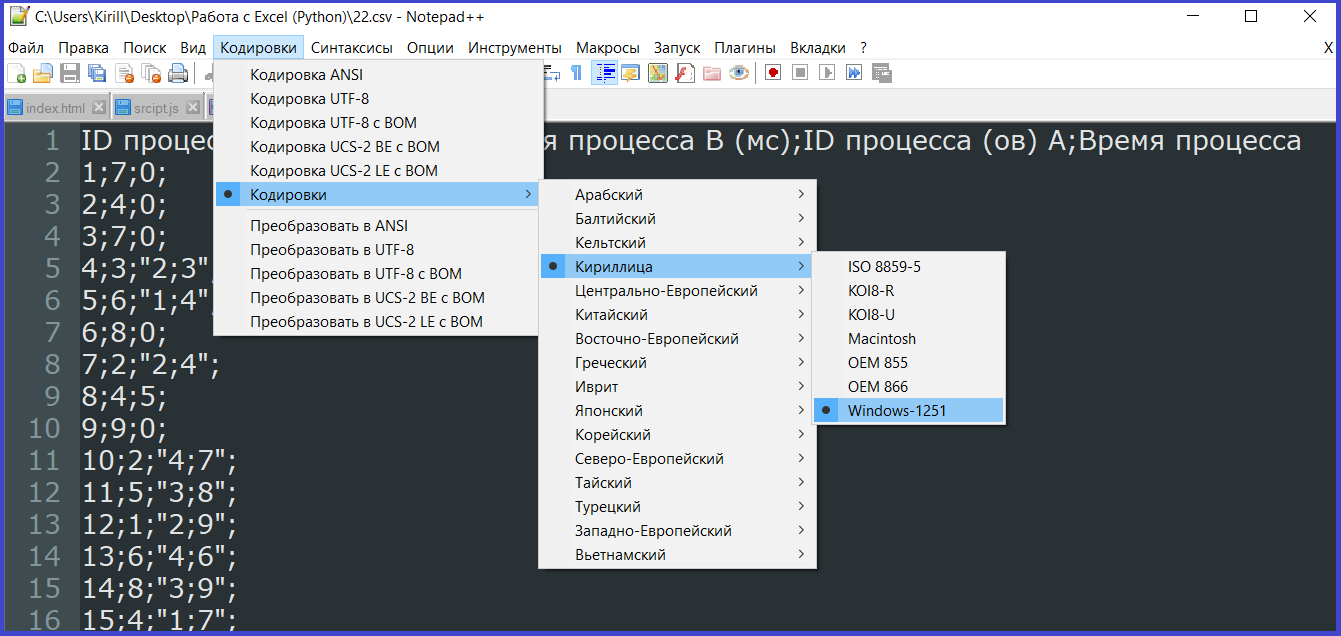
print("Время выполнения всех процессов: {}".format(execution\_time))

Что если нет возможности скачать и установить библиотеку xlrd ? Тогда можно воспользоваться стандартной библиотекой в Python. Для этого нам понадобиться наш файл .xlsx сохранить с расширением .csv. Структурно файл не изменится.

**CSV** – это вид файла, который позволит структурировать большие объемы файлов. Этот файл по умолчанию в windows открывается через Excel, но его можно легко открывать через текстовый редактор. Потому что по своей сути CSV является текстовым файлом, в котором каждый элемент отделяется от предыдущего каким-то разделителем (это может быть «;» или «,»). Данные CSV легко экспортировать в электронные таблицы обратно.

**Замечание**: Для парсинга/чтения CSV-файла необходимо знать какой именно разделить используется. Откройте файл в текстовом редакторе и посмотрите сами.

**Замечание**: Чтобы открыть файл через Python, нужно быть уверенным, что кодировки совпадут. В моем случае через Excel файл сохранился в .csv в кодировке Windows-1251. Посмотреть кодировку можно, например, в Notepad++:



**Код реализации через встроенную библиотеку CSV**

Здесь логика такая же, как в предыдущем варианте. Только работать с CSV удобнее. Не нужно устанавливать эту библиотеку, она встроенная. Работаем с файлом как с обычным текстовым, в котором есть разделитель в виде точки с запятой. Открываем и считываем. Во втором варианте для хранения информации я выбрал список словарей. ([Почитать про словари можно здесь](https://pythonworld.ru/tipy-dannyx-v-python/slovari-dict-funkcii-i-metody-slovarej.html)). Потому что доступ по ключам в виде слов делает программу более понятной. Есть одна сложность при работе с ID. Нумерация ID начинается с 1, а нумерация элементов списка с нуля, поэтому образуется такой вот сложный индекс на строке 26.

Еще стоит заметить, что в программе выводится информация для отладки и написаны комментарии, которые делают код больше. Но зато этот код становится более понятным, чем предложенные решения такой задачи от авторов решу-егэ. Про работу с CSV можно [почитать здесь в документации](https://docs.python.org/3/library/csv.html) или [здесь в статье на русскоязычном сайте.](https://all-python.ru/osnovy/csv.html)

**import** **csv**

processes = [] # структура под хранение всех процессов

**with** open("22.csv", encoding = 'Windows-1251') **as** data:

reader\_csv = csv.reader(data, delimiter = ";")

count\_row = **0**

**for** row **in** reader\_csv:

**if** count\_row == **0**: # выводим заголовки (один раз)

print(f'Файл содержит столбцы: {"| ".join(row)}')

**else**: # выводим строки, которые удалось распознать из файла

print(f' {row[0]} {row[1]} {row[2]}')

process = dict(ID = int(row[**0**]),

time = int(row[**1**]),

depend = str(row[**2**]),

exectime = **0**)

processes.append(process)

count\_row += **1**

print(f'Всего в файле {count\_row} строк')

# Расчет времен полного завершения процессов

**for** p **in** processes:

**if** p['depend'] == '0':

p['exectime'] = p['time']

**else**:

list\_depends = p['depend'].split(';')

times\_depends = [ processes[int(ID)-**1**]['exectime'] **for** ID **in** list\_depends ]

p['exectime'] = p['time'] + max(times\_depends)

# Вывод информации

**for** row **in** range(**0**, len(processes)):

print(processes[row])

max\_exec\_time = **0**

**for** p **in** processes:

**if** p['exectime'] > max\_exec\_time:

max\_exec\_time = p['exectime']

print(f'Время исполнение всех процессов {max\_exec\_time}')