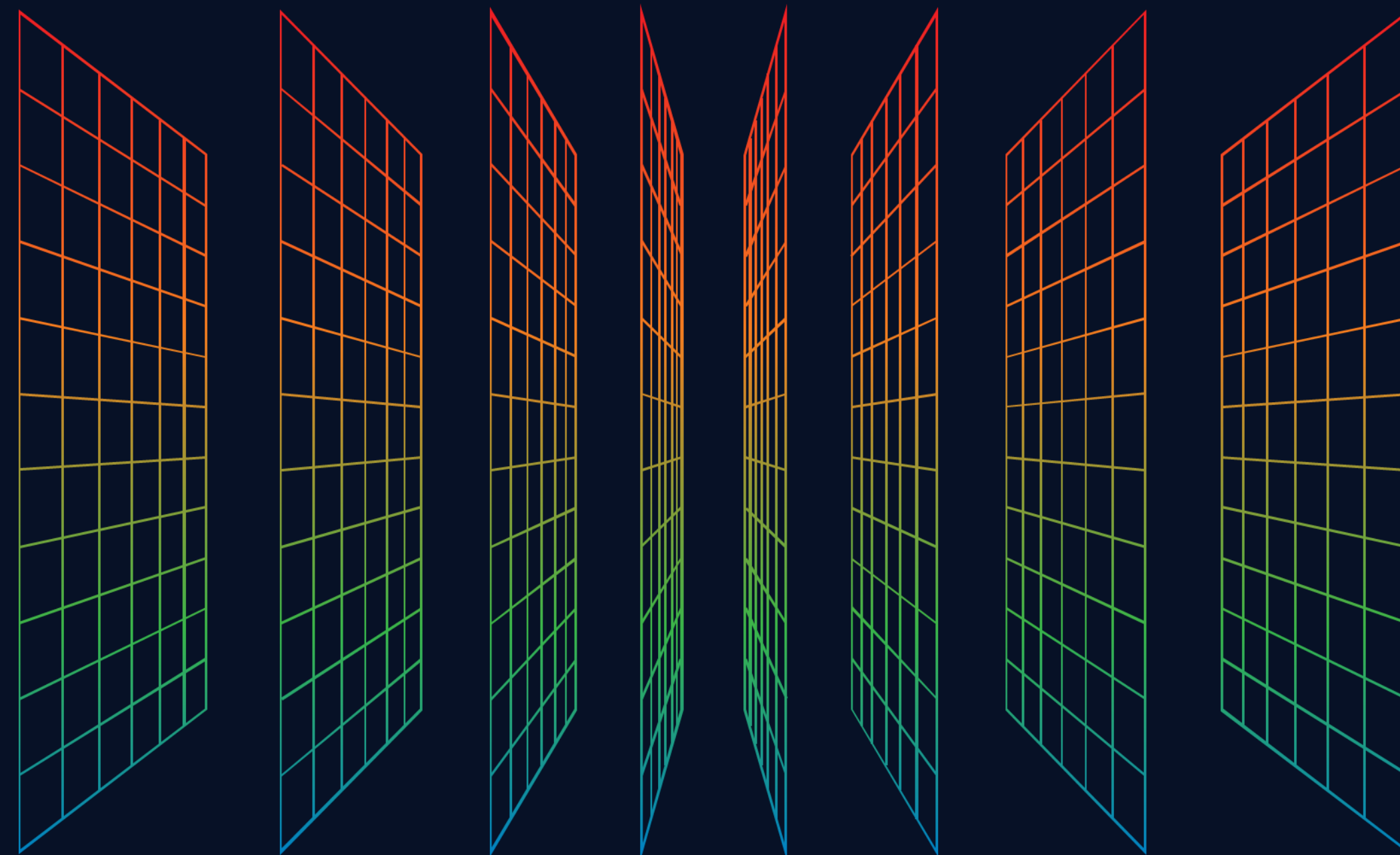


Использование языка Python в Schedule



Внимание! При прохождении данного курса следует помнить, что методики, описанные в рамках урока, носят рекомендательный характер и не являются единственно верными. Основной целью данного курса является рассмотрение всех основных функций, доступных в тНавигатор. В реальных проектах применяемые методики могут отличаться от описанных в данном курсе. Все данные, используемые в курсе, не являются реальными.

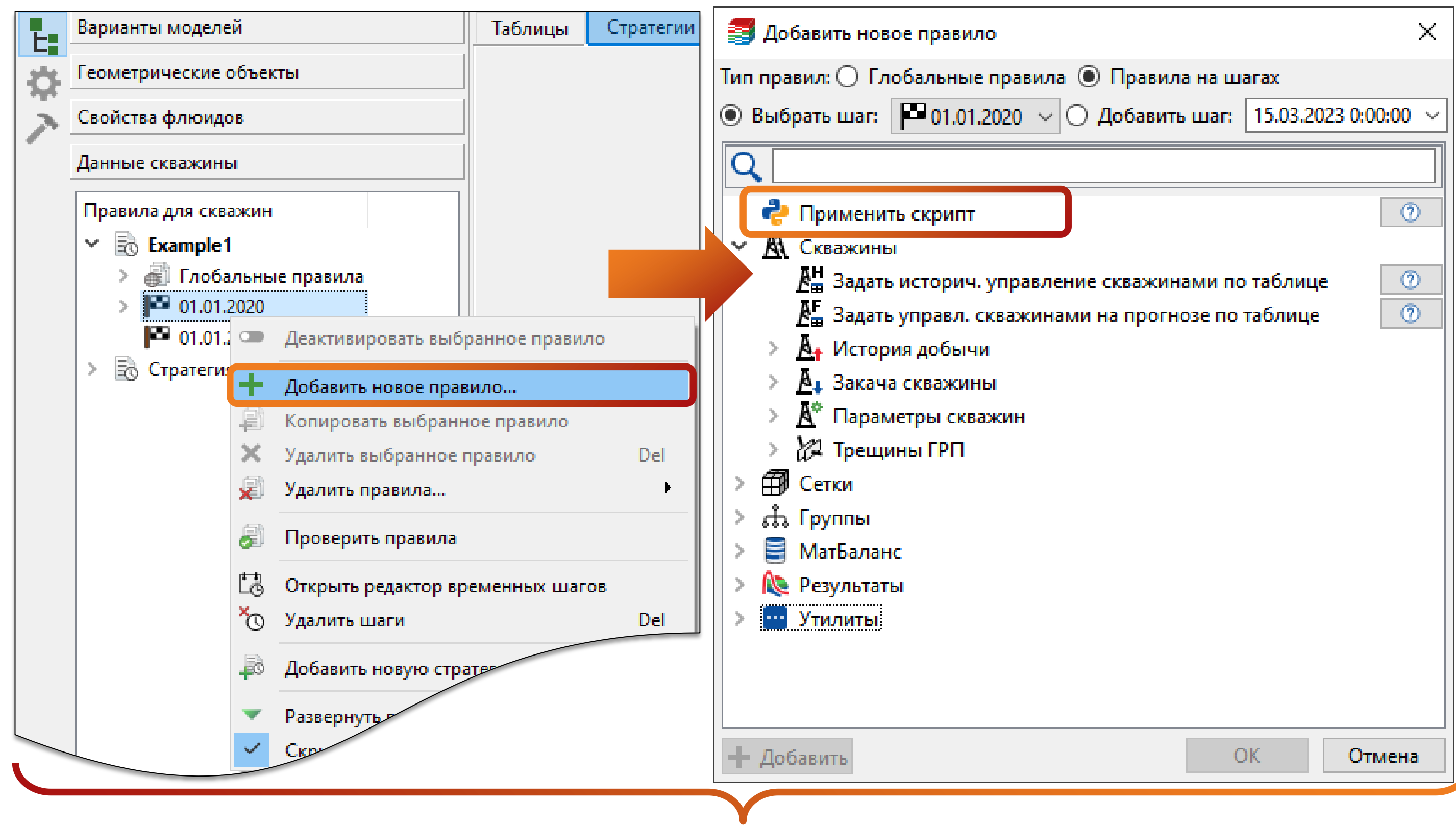
Python в SCHEDULE

Действия над скважинами, выполняемые при определённых условиях

- Ключевые слова **ACTION*** + **UDQ**
 - Неудобный и непонятный синтаксис
 - Ограниченная функциональность
- Ключевое слово **APPLYSCRIPT** + скрипты Python
 - Синтаксис и логика языка Python
 - Полноценный язык программирования
 - Стандартные графические мнемоники
 - Создание и импорт дополнительных графиков
 - Задание стандартных правил как действий в SCHEDULE
 - API для доступа к параметрам модели и управления объектами
 - Создание пользовательских сообщений в log файле
 - Запись сообщений и данных во внешние файлы
 - Импорт сторонних библиотек

Работа с Python в тНавигатор

- Запуск Python выполняется с помощью ключевого слова **APPLYSCRIPT**
- Может быть добавлен в Дизайнер Моделей как любое другое правило для скважин
- Скрипт запускается на каждом расчетном шаге



Дизайнер Моделей

```
DynamicModel_SCHEDULE.inc x
77
78 APPLYSCRIPT
79 'INCLUDE/EOS_rein.py' 'eos_rein' /
80 /
81
82 DATES
83 01 FEB 2020 /
84 /
85
```

Симулятор

Интерфейс в Дизайнере Моделей

Таблицы x

Стратегии x

Данные по скважинам x

Дизайнер Скважин x

+

Применить скрипт

Файл скрипта: script 1 + -

Функция: Ничего Выполнить синтаксич. анализ параметров

Функция, которая будет вызываться на расчётных шагах

Типы объектов

Параметры объектов

Объекты:

Скважина

Группа

Инт. перфорации

Группа перфораций

Отч. рег.

Аквифер

Численный аквифер

Сегмент

Месторождение

Блоки

Свойства

Параметры:

Анализ

Давление

Дебиты

Накопл. показатели

Редактор кода:

1 #пишите здесь ваш код

Редактор кода

Функции для работы с объектами tНавигатор

Функции:

Скважина

Группа

Сегмент

Группа перфораций

Перфорация

Стадия ГРП

Блок

Отч. рег.

Трассер

Аквифер

Численный аквифер

Свойство сетки

График

Глобальные функции

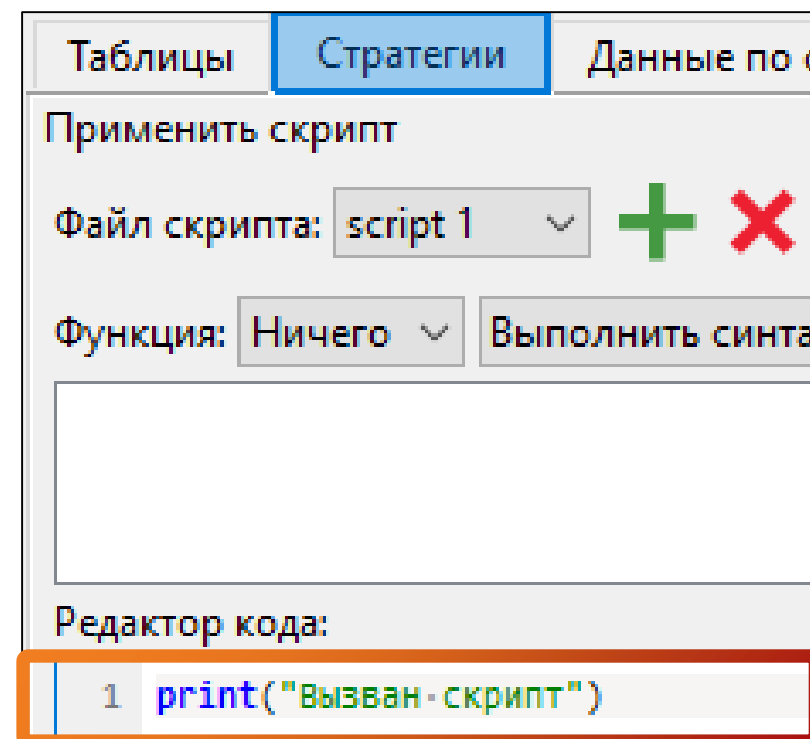
Глобальные функции гра

Глобальные функции SCI

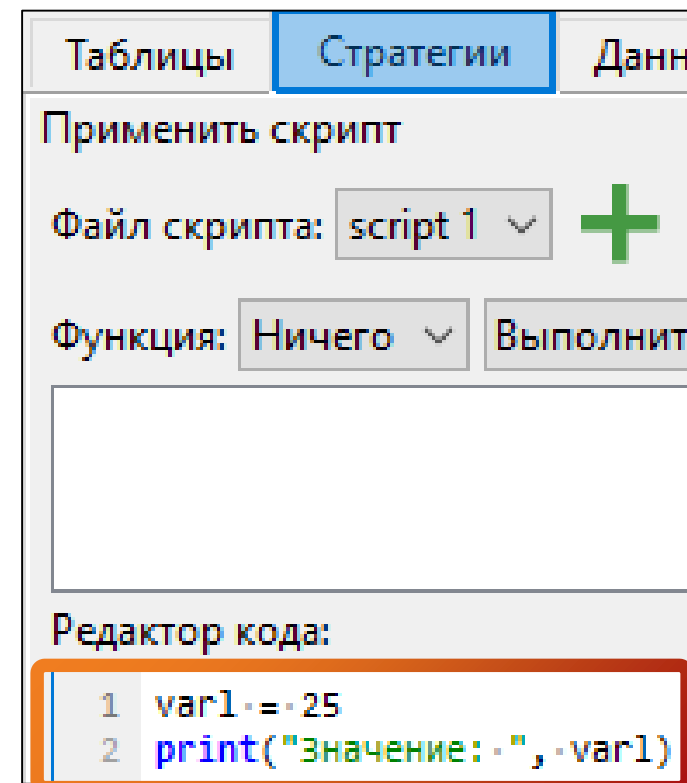
Глобальные функции скр

Функции языка Python – print

- Вывод сообщений в log файл
- Полезны для вывода информации при отладке скрипта или любой дополнительной информации в окне консоли
- **Пример 1:**



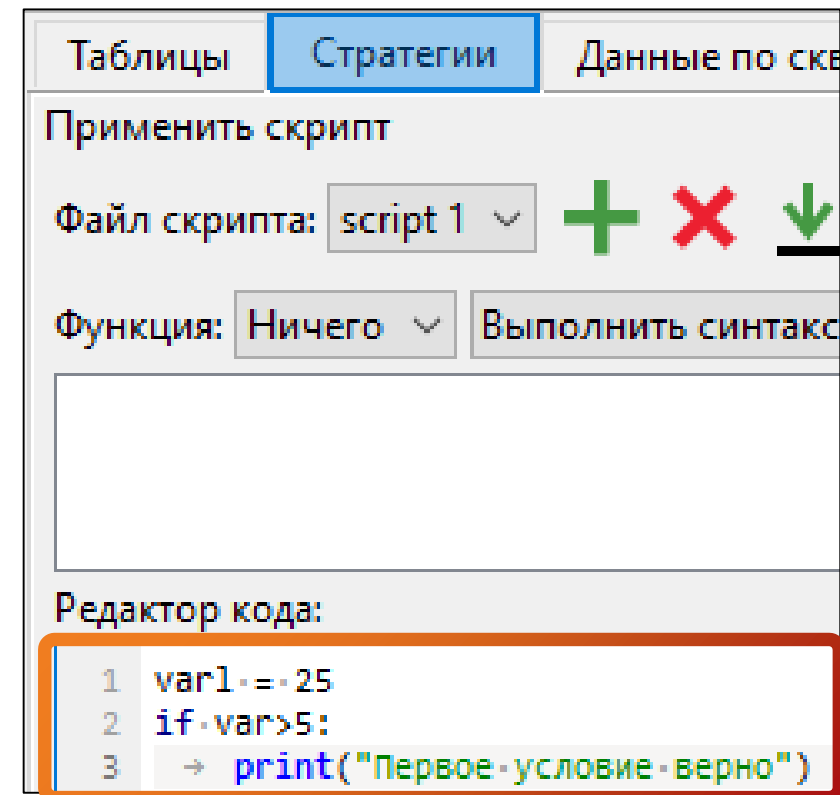
- Строки можно объединять
- **Пример 2:**



Функции языка Python – if, else

- Оператор **if** используется для запуска кода, если выполнены определенные условия
- В качестве условий используются сравнения величин (**>**, **<**, **==**, **!=**) или логические переменные (Истина (True) или Ложь (False))

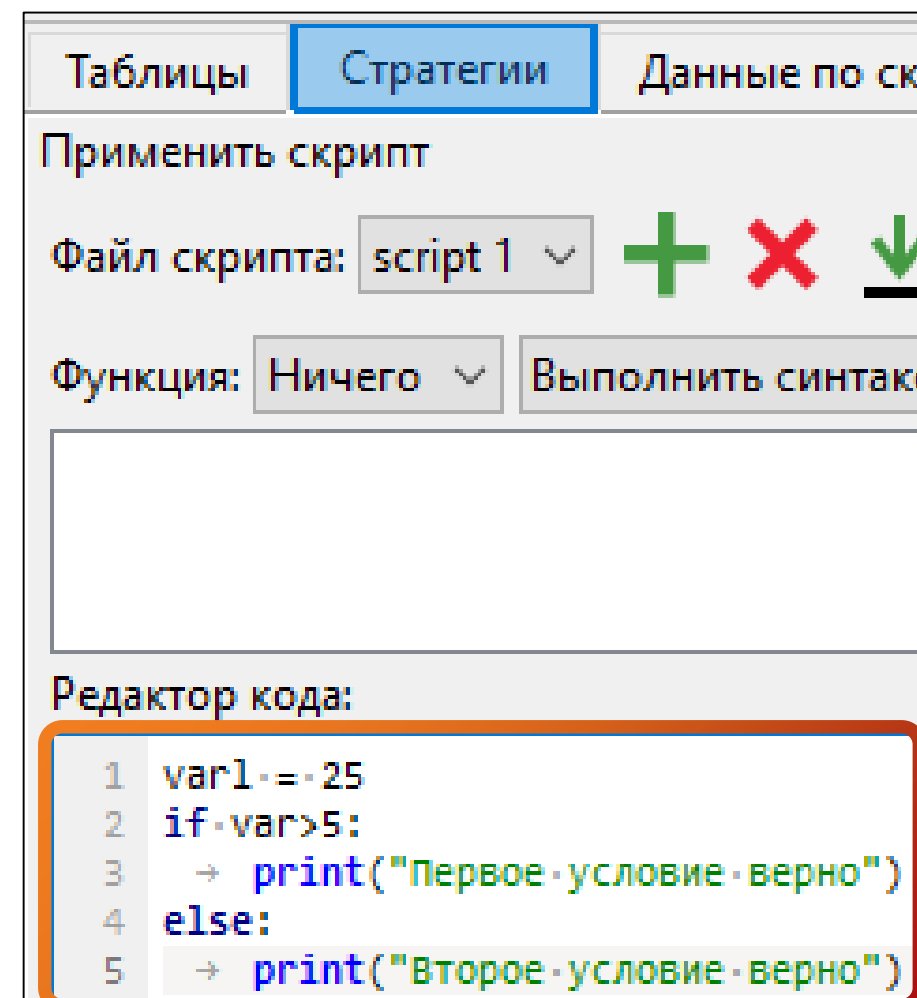
Пример 1:



```
1 var1 = -25
2 if var > 5:
3     print("Первое условие верно")
```

- Оператор **else** следует за оператором **if** и содержит код, который вызывается, когда оператор **if** принимает значение **False**

Пример 2:

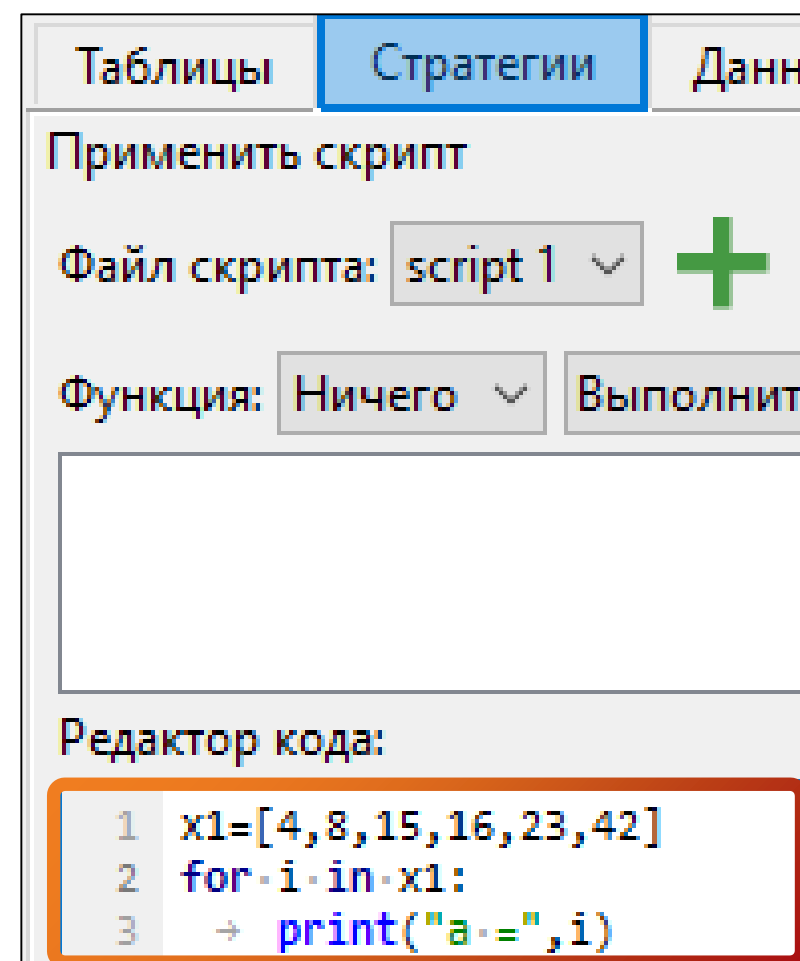


```
1 var1 = -25
2 if var > 5:
3     print("Первое условие верно")
4 else:
5     print("Второе условие верно")
```

Функции языка Python – for

- Цикл **for** позволяет запускать один и тот же код несколько раз для разных значений переменных разных объектов. Это называется итерациями
- Аргументы для цикла **for**:
 - Переменные цикла
 - Список объектов или массив значений переменных

- **Пример:**



Функции языка Python – import

- **import** используется для импорта predefined модулей, как стандартных, так и сторонних

- **Пример:**

```
Редактор кода:
5 def convert_with_limits():
6     from datetime import timedelta, datetime
7     wct_lim=0.95
8     group_limit=3
9     all_limit=10
10    month_delta=6
11    groupnames=['G1','G2','G3','G4']
12    gc=get_global_graph(name='group_count')
13    ac=get_global_graph(name='all_count')
14    wells=[]
15    wcts=[]
16    if ac<all_limit:
17        for gname in groupnames:
18            g=get_group_by_name(gname)
19            if gc[g]<group_limit:
20                w=get_well_by_criterion(group=g, method='max', crit=wwct)
21                if wwct[w]>wct_lim:
22                    wells.append(w)
23                    wcts.append(float(wwct[w]))
24        if len(wells)>0:
25            w=wells[wcts.index(max(wcts))]
26            add_keyword(
27                """
28                WCONINJE
29                """+w.name+""" WATER OPEN BHP 2*210 /
30                /
31                """
32            )
33            print("At",get_current_date().strftime("%d.%m.%Y%H:%M:%S"),"well",w.name,"was converted to injector")
```

```
##### FINISH REPORT N 0197 FOR DAY 5996 ON 01.06.2036 #####
At 5996.0 (01 JUN 2036)
At 01.06.2036 00:00:00 well P 19 was converted to injector
At 5996.0 (01 JUN 2036)
At 5996.0 (01 JUN 2036)

Running script 'convert_with_limits' from file 'E:/DATA/19.4/APP
Finishing script 'convert_with_limits' from file 'E:/DATA/19.4/A
Action 'convert_with_limits' triggered on step 197!
```

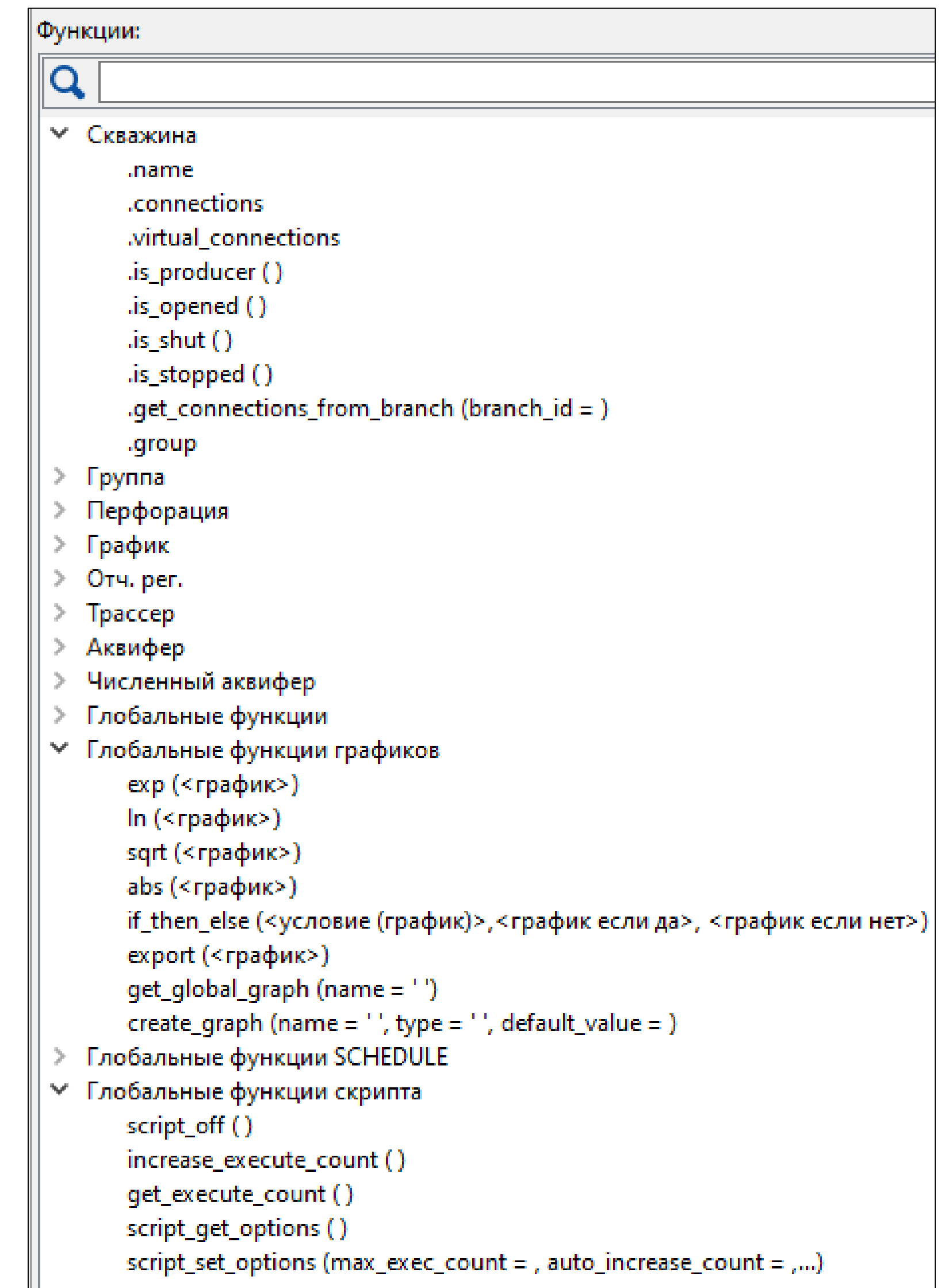
Объекты в Schedule

● Типы объектов

- График
- Скважина
- Группа
- Перфорация
- Отч. Рег.
- Месторождение
- и т.д.

● Каждый объект имеет собственный набор свойств и методов

● Существуют функции для работы с объектом и доступа к его свойствам



Скважины

● Простые свойства

.name – содержит имя скважины

.group – содержит группу, к которой принадлежит скважина

● Свойства-массивы

.connections – массив, содержащий перфорации скважины

.get_connections_from_branch (branch_id =) – возвращает массив, содержащий перфорации для выбранных ветвей

● Логические функции

.is_producer () – Истина, если скважина является добывающей

.is_opened () – Истина, если скважина является открытой

.is_shut () – Истина, если скважина закрыта

.is_stopped () – Истина, если скважина остановлена

▼ Скважина

.name

.connections

.virtual_connections

.is_producer ()

.is_opened ()

.is_shut ()

.is_stopped ()

.get_connections_from_branch (branch_id =)

.group

Группы

● Простые свойства

.name – содержит имя группы

.parent_group – содержит родительскую группу

● Свойства-массивы

.wells – содержит массив скважин группы

.all_wells – содержит массив скважин, входящих в эту группы и в ее дочерние группы

.child_groups – содержит массив дочерних групп

▼ Группа

- .name
- .wells
- .all_wells
- .parent_group
- .child_groups

Перфорации

● Простые свойства

.name – содержит имя перфорации

.i – содержит i-ю координату перфорации

.j – содержит j-ю координату перфорации

.k – содержит k-ю координату перфорации

.well – содержит скважину, которой принадлежит данная перфорация

.branch_id – содержит номер ветви, которой принадлежит данная перфорация

.lgr_name – содержит имя LGR (или GLOBAL для глобальной сетки)

● Методы

.is_opened() – возвращает True, когда перфорация открыта, и False в противном случае

.is_shut() – возвращает True, когда перфорация закрыта, и False в противном случае

▼ Перфорация

- .name
- .i
- .j
- .k
- .well
- .branch_id
- .lgr_name
- .is_opened ()
- .is_shut ()

Графики

● Функции извлечения и агрегирования данных

.fix (object = "") – возвращает значение выбранного графика для данного объекта

.min (objects = "")

.max (objects = "")

.avg (objects = "")

.sum (objects = "")

извлекает подмножество значений для заданных объектов, а затем возвращает минимальное, максимальное, среднее или сумму полученного массива

▼ График

.min (objects = "")

.max (objects = "")

.avg (objects = "")

.sum (objects = "")

.fix (object = "")

Отчетные регионы

● Свойства

.name – содержит имя региона

.family – содержит имя множества регионов

.number – содержит номер региона

▼ Отч. рег.

.name

.family

.number

Глобальные функции

● Функции массивов:

get_all_wells (type='prod') – возвращает массив, содержащий все добывающие скважины (без аргумента – все скважины; см. также type='inje', 'open', 'shut', 'stop')

get_all_groups () – возвращает массив, содержащий все группы

get_all_connections () – возвращает массив, содержащий все перфорации всех скважин

get_wells_by_mask (") – возвращает массив скважин с именем, соответствующим заданной маске

get_all_fip_regions () – возвращает массив, содержащий все отчетные регионы для всех семейств

get_fip_regions_from_family (<family name>) – возвращает массив отчетных регионов для заданного семейства

get_well_by_criterion (method, crit, group, recursive) – возвращает скважину с максимальным или минимальным значением графика из данной группы

● Логические функции:

if_then_else (<graph condition>, <graph if yes>, <graph if no>) – Оператор IF применяется последовательно для каждого элемента массива

Глобальные функции

- Математические операции над графиками (graph)

`exp (<graph>)` – экспонента

`ln (<graph>)` – натуральный логарифм

`sqrt (<graph>)` – квадратный корень

`abs (<graph>)` – абсолютное значение

- Простые функции

`get_project_folder ()` – возвращает полный путь к папке, содержащей текущую модель

`get_project_name ()` – возвращает имя файла текущей модели без расширения

Графики и мнемоники

- Графики и мнемоники являются массивами, которые индексируются по объектам соответствующего типа (скважина, группа, перфорации и т.д.) График по месторождению – это всего лишь одно число.
- Примеры:
 - **wopr[w]** – дебит нефти скважины w на текущем временном шаге
 - **clpt[c]** – накопленная жидкость для перфорации c на текущем временном шаге
 - **fwir** – приемистость воды для месторождения на текущем временном шаге
- Мнемоники сгруппированы по типу объекта, к которому они относятся:

Объекты:	WGVIT	WGVIR	WGPI	WGITH	WGIT	WGIRWEF
Скважины	WGIRT	WGIRH	WGIR	WECONWGR	WBULKP	WMCON
Группы	WOIRWEF	WBP	WSTATH	WOPRWEF	WBPO	WCPT_1
Инт.перфорации	WCPR_1	WCMIT_1	WCMIR_1	WCGMT_1	WCGMR_1	WWMPT_2
FIPNUM	WWMPR_2	WWMIT_2	WWMIR_2	WCPT_2	WCPR_2	WGPP2
Аквиферы	WBP9	WGCV	WBHPT	WBHPH	WBHP	WBHPU
Численные аквиферы	WAVGXI	WPIG	WGVPT	WGVPR	WGPTH	WGPT
Сегменты	WGPRWEF	WGPRT	WGPRH	WGPR	WGPP	WGMT
Месторождение	WGMR	WGLIR	WGDN	WEFF	WBPS	WBHRD
	WAMTI	WAMRI	WMPRWEF	WMPR		
	WPI	WLPTH	WLPT			

- Обратите внимание на отличие от Калькулятора графиков, в котором те же мнемоники рассматриваются как многомерные массивы, индексируемые также по моделям и временным шагам

Примеры использования APPLYSCRIPT

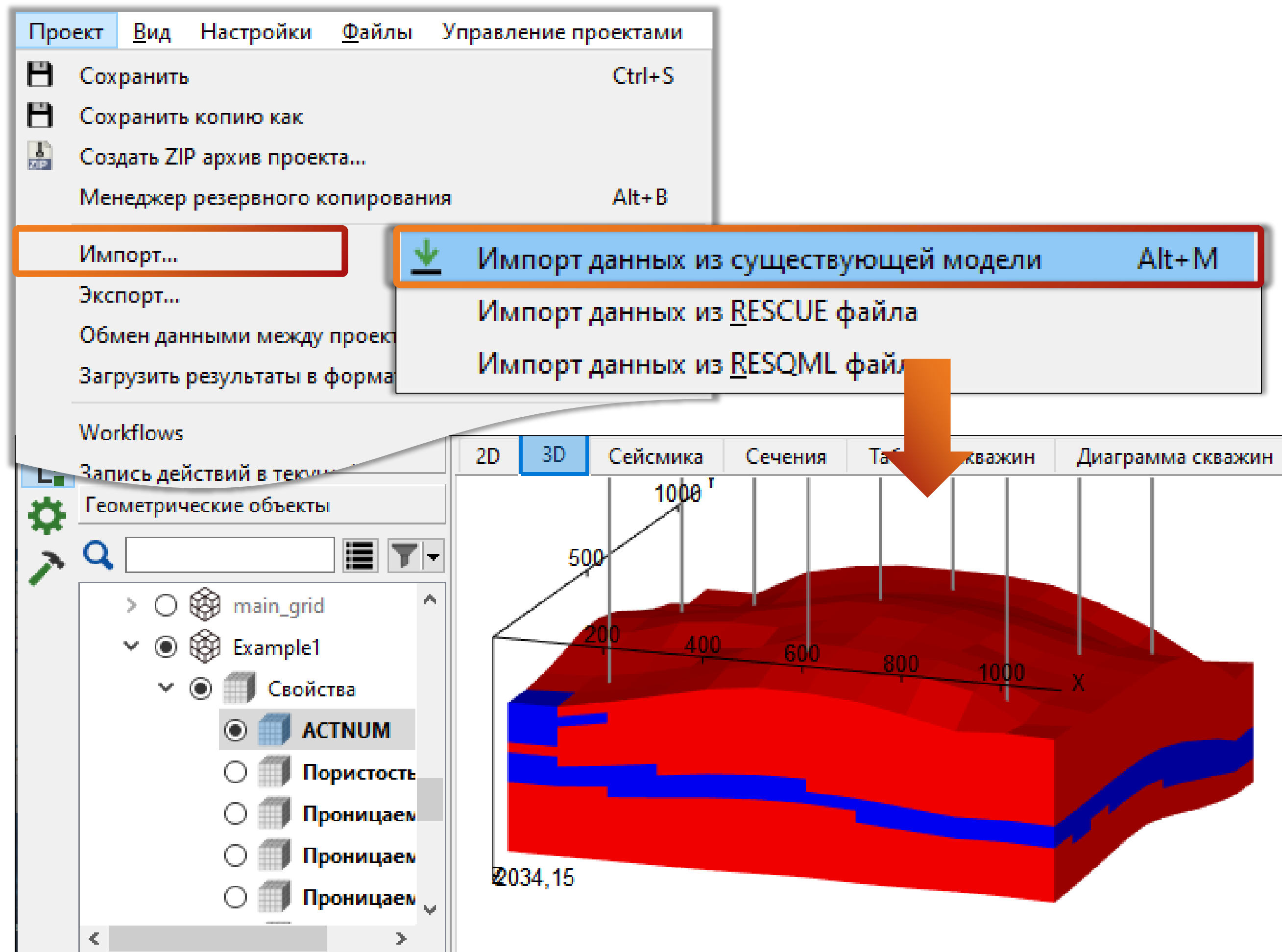
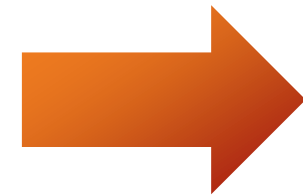
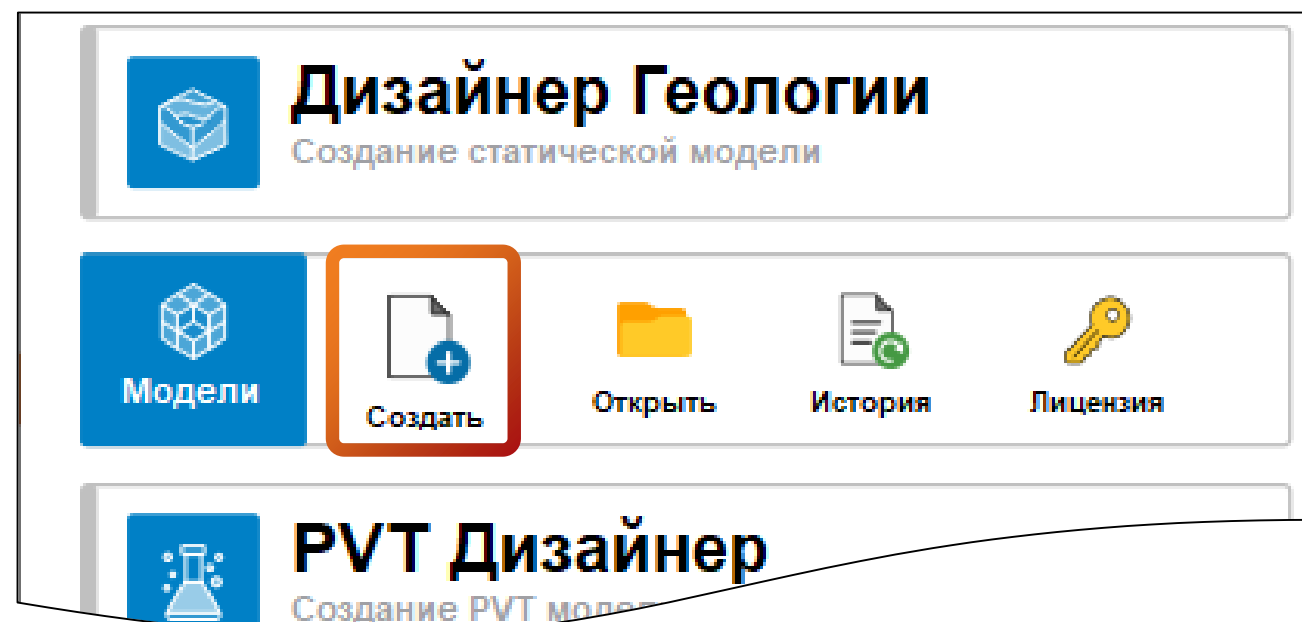
Содержание

- Пример 1
- Пример 2
- Пример 3
- Пример 4

Пример 1

Рассматривается упрощенная модель, состоящая из двух пластов. Один из пластов истощен и скважины начинают переключаться на другой пласт

1. Создайте новый проект **Дизайнера Моделей**
2. Импортируйте модель из **Example1/Example1.data**



Пример 1

1. Вкладка **Данные скважины** → **Стратегии**, откройте **Example1**
2. ПКМ на первую дату, выберите **Добавить новое правило**
3. Выберите **Другие. Применить скрипт**
4. Импортируйте скрипт **Example1/example1.py** и задайте значение параметра **Функция** как **change_layer**

The screenshot illustrates the process of applying a script to a well strategy in the TNavigator software. The interface is divided into several panels:

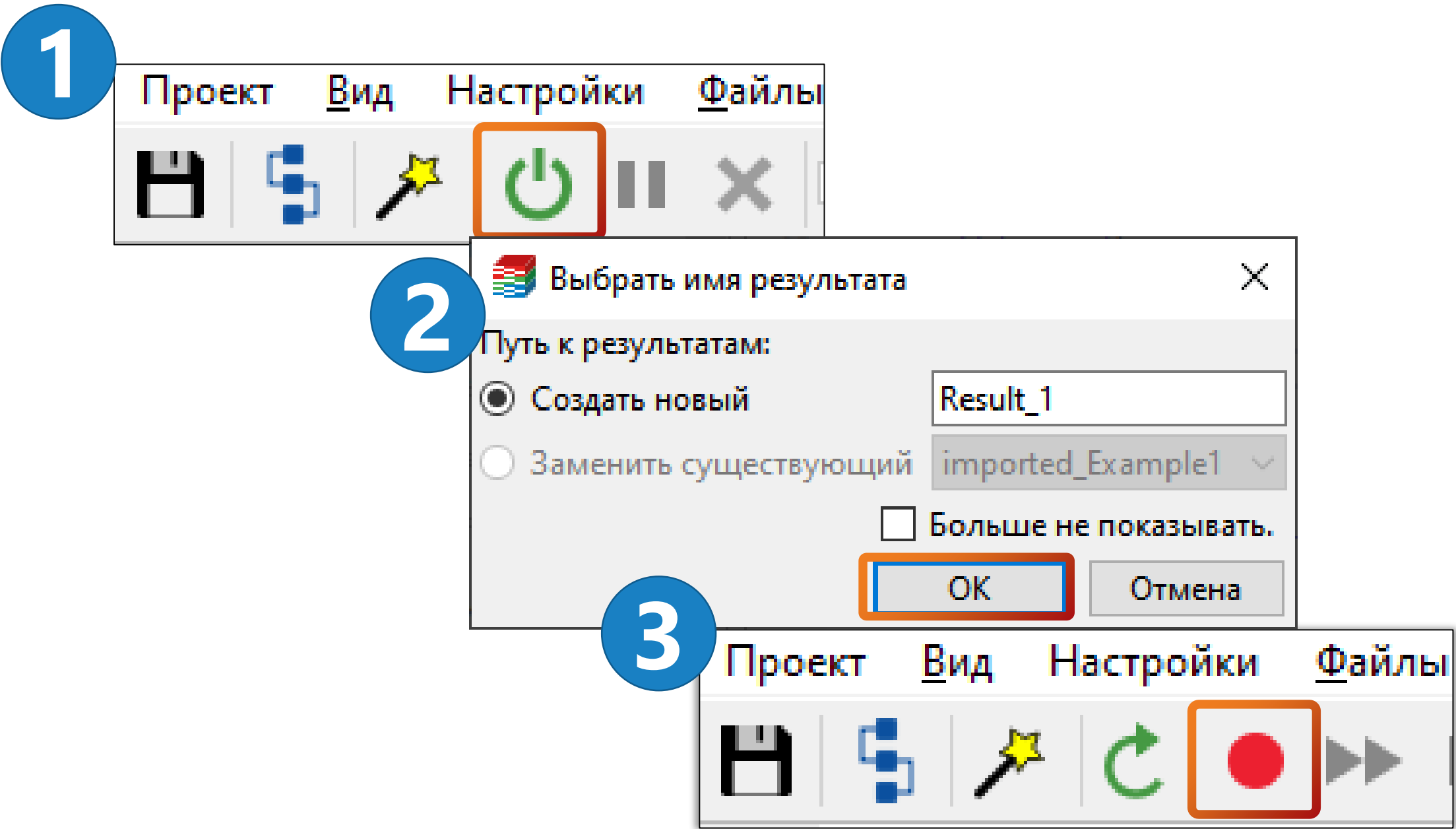
- Left Panel (1):** Shows the 'Данные скважины' (Well Data) tab selected under the 'Стратегии' (Strategies) section. The 'Example1' strategy is expanded, showing a list of dates. The first date, '01.01.2020', is selected.
- Right Panel (2):** A context menu is open over the selected date. The option 'Добавить новое правило...' (Add new rule...) is highlighted.
- Bottom Panel (3):** A sub-menu is open from 'Добавить новое правило...'. The option 'Применить скрипт' (Apply script) is highlighted.
- Top Panel (4):** The 'Применить скрипт' (Apply script) dialog box is open. The 'Файл скрипта' (Script file) is set to 'example1'. The 'Функция' (Function) is set to 'change_layer'. The 'Редактор кода' (Code editor) shows the Python script for 'change_layer'.

The Python script in the code editor is as follows:

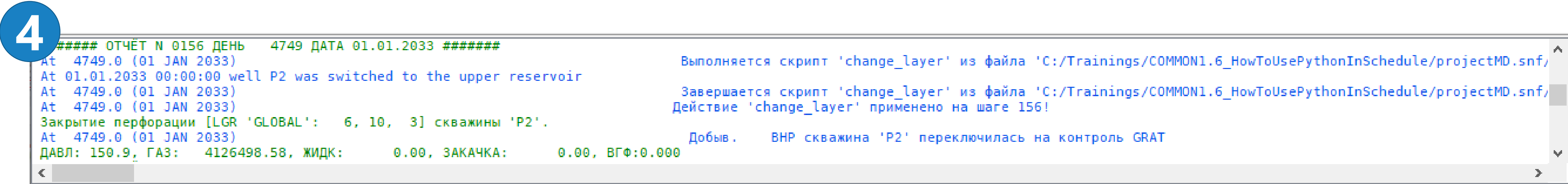
```
def change_layer():  
    ..script_set_options(max_exec_count=360, calculation_start=True)  
    ..increase_execute_count()  
    ..if get_execute_count()==1: return...#Don't start at the first step  
    ..#For all producing wells:  
    ..for w in get_all_wells(type="prod"):  
        ..#If the gas rate is below threshold,  
        ..if wgpr[w]<100:  
            ..#we close the lower connections and open the upper ones:  
            ..add_keyword(  
                ..""  
                ..COMPDATMD  
                ..""+w.name+""+1*.....2020.....2030.....MD.....SHUT...../  
                ..""+w.name+""+1*.....2000.....2010.....MD.....OPEN.....2*.....0.168.....1*.....0  
                ..1*.....1...../  
                ..""  
                ..""  
            ..print("At", get_current_date().strftime("%d.%m.%Y.%H:%M:%S"), "well", w.name, "was switched to the  
                upper reservoir")
```

Пример 1

- 1. Инициализируйте Динамическую Модель
- 2. Сохраните модель. Выберите имя результата Result_1
- 3. Запустите Расчёт
- 4. Обратите внимание на уведомления в логе

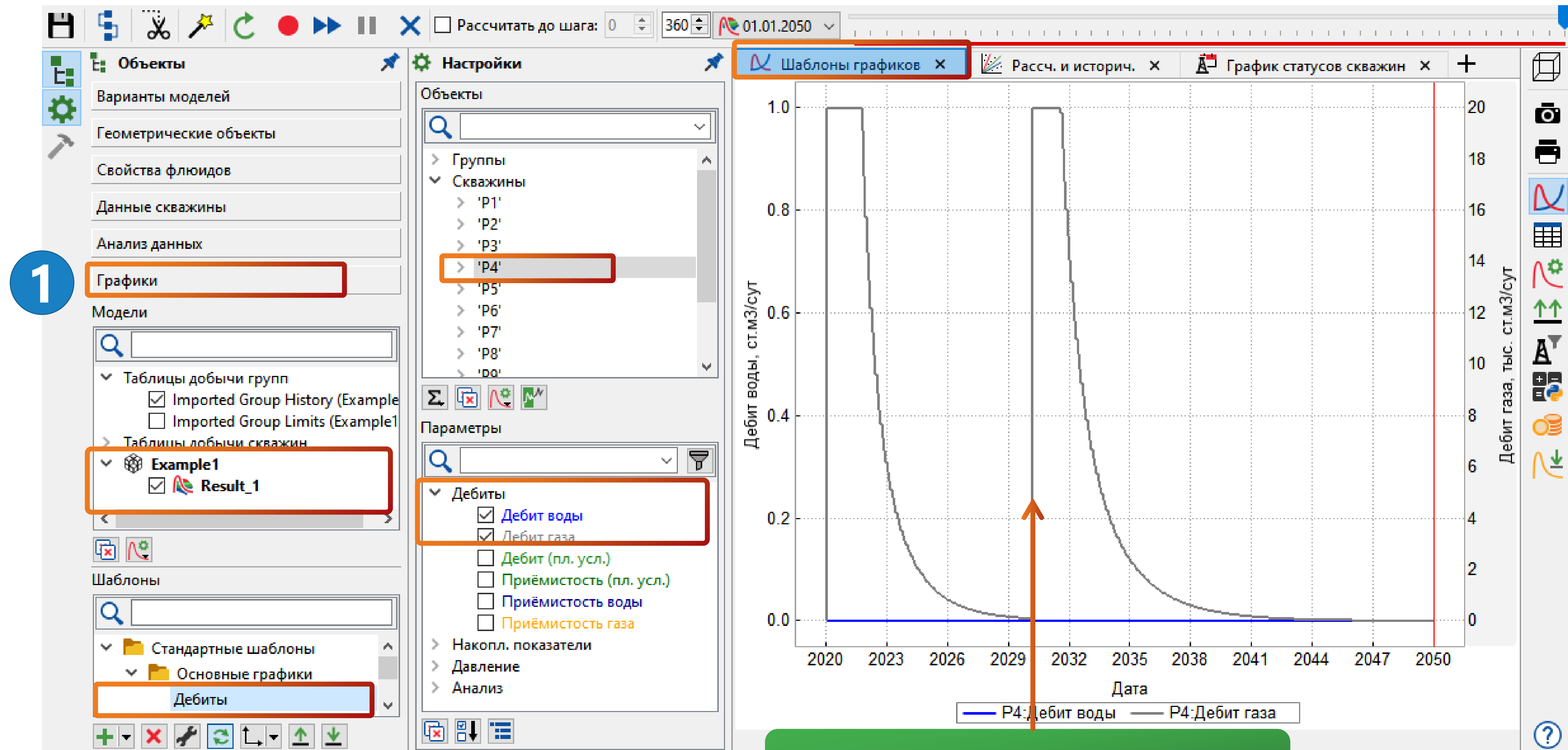


```
def change_layer():
    script_set_options(max_exec_count=360, calculation_start=True)
    increase_execute_count()
    if get_execute_count()==1: return #Don't start at the first step
    #For all producing wells:
    for w in get_all_wells(type="prod"):
        #If the gas rate is below threshold,
        if wgpr[w]<100:
            #we close the lower connections and open the upper ones:
            add_keyword(
                """
                COMPDATMD
                """+w.name+""""      1*      2020      2030      MD      SHUT      /
                """+w.name+""""      1*      2000      2010      MD      OPEN      2*      0.168      1*      0      1*      1
                /
                /
                """)
            print("At",get_current_date().strftime("%d.%m.%Y %H:%M:%S"), "well",w.name,"was switched to the upper reservoir")
```



Пример 1

1. **Графики. Example1. Result_1. Настройки.** Выберите скважину **P4**. Установите **Стандартные шаблоны – Основные графики – Дебиты**. График **Дебит газа** для скважины **P4**



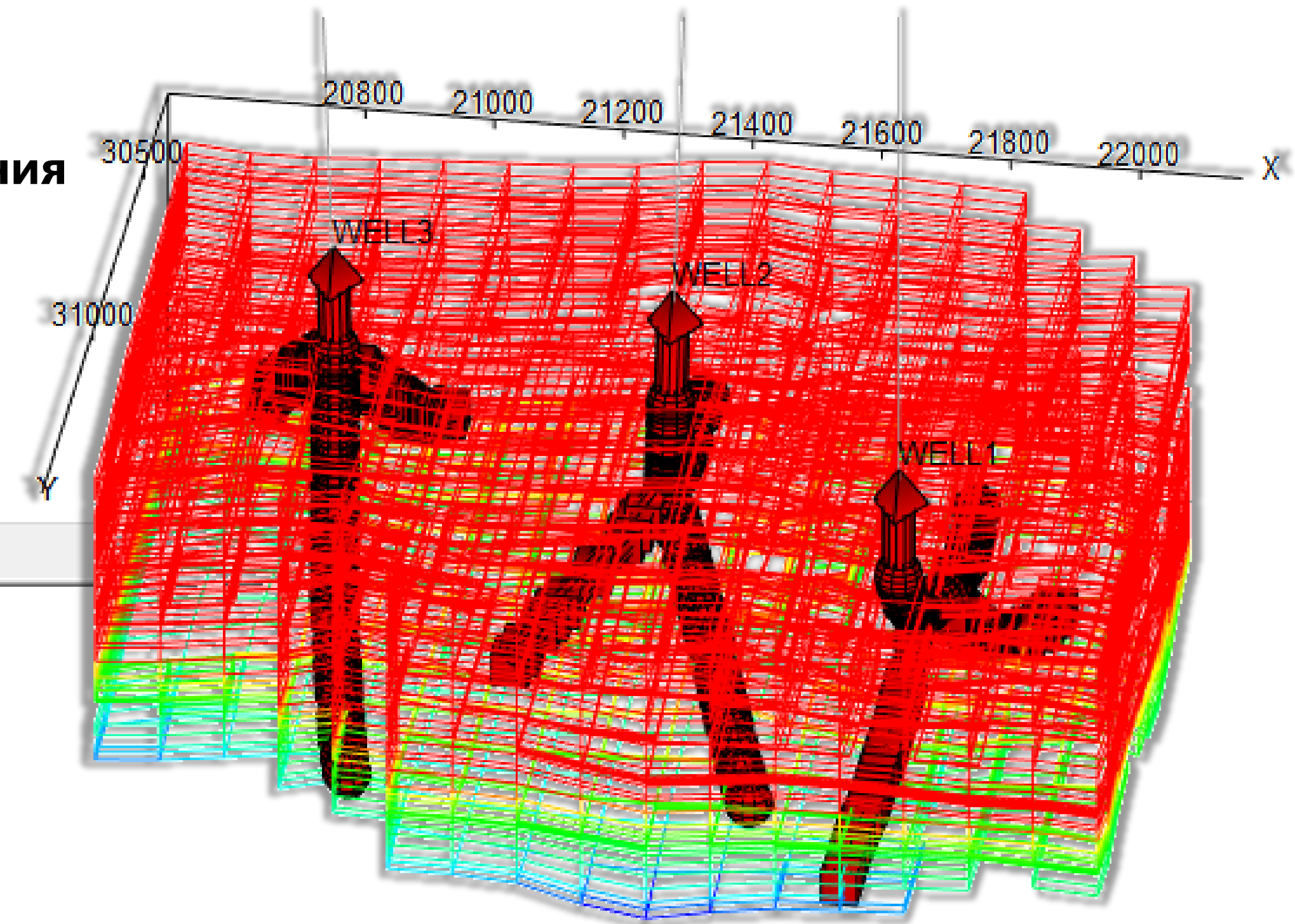
Пример 2

1. Многоствольная скважина

- Автоматическое закрытие ветви скважины (не существует аналога в виде ключевого слова)

2. Модель в примере

- 16x26x46 ячеек
- 3 добывающие скважины, каждая из которых имеет 3 ветви
- Ветвь закрывается, когда ее добыча падает ниже порогового значения



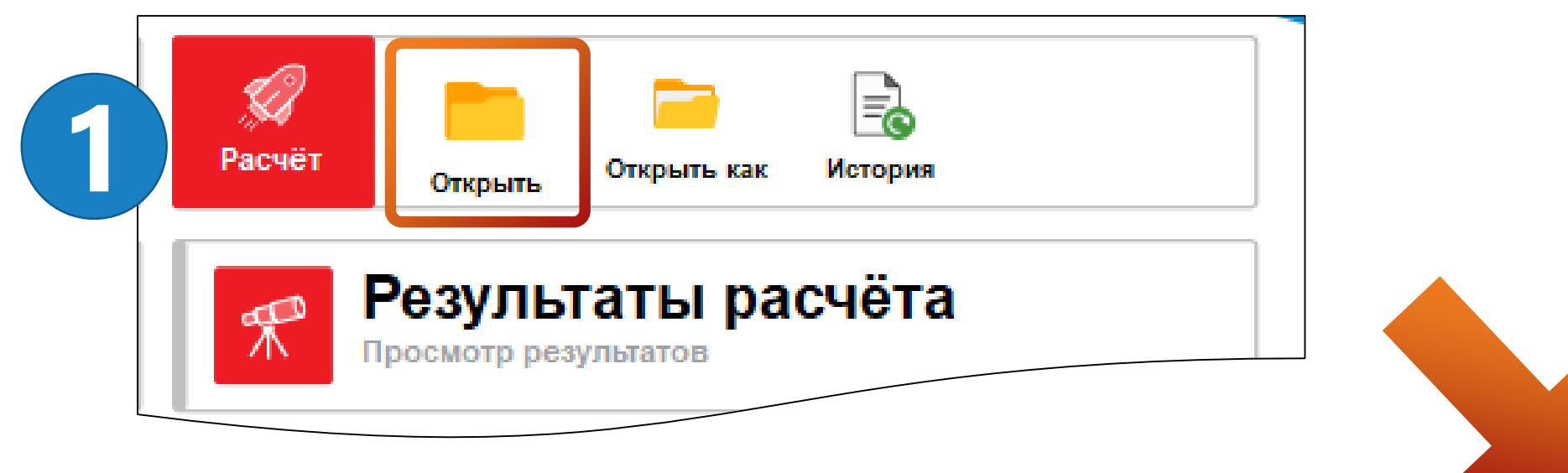
off_branch_by_ORAT.py

```
1 def off_branch_orat():
2     branches=[0,1,2]          #номера стволов
3     for w in get_all_wells():  #цикл по всем скважинам
4         for branch in branches: #цикл по стволам одной скважины
5             result=0
6             for c in w.get_connections_from_branch (branch_id = branch):
7                 if not c.is_opened(): continue
8                 result+=copr[c]          #Суммируется дебит для ствола
9             print("Oil rate on branch ",branch,result)
10            if result<75:              #Если дебит меньше порогового значения, закрыть ствол
11                print("Closing branch",branch,"with rate",result)
12                change_well_perforation(well=w.name, branch=branch, mdu=1600, mdl=2500, depth='MD', status='shut')
13
```



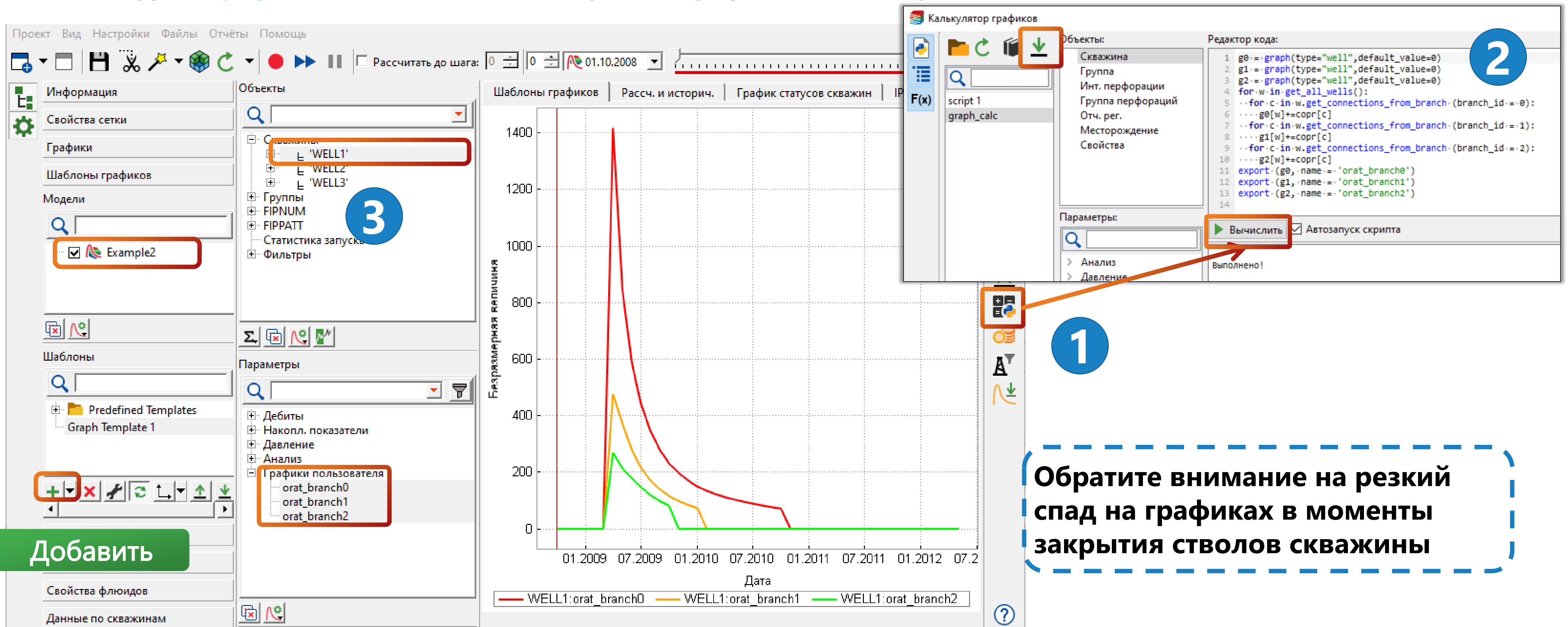
Пример 2

1. Откройте [Example2/Example2.data](#) в Симуляторе
2. Запустите [Расчёт](#)



Пример 2

1. Перейдите на вкладку **Шаблоны графиков** → **Калькулятор графиков**
2. Запустите скрипт **Example2/graph_calc.py** в **Графическом калькуляторе** для получения дебитов отдельных ветвей
3. Выберите скважину **WELL1**, создайте новый Шаблон, нажав кнопку **Добавить**. Выберите новый **Шаблон Графиков 1**, активируйте **Графики пользователя**. Посмотрите на графики дебита нефти для каждой ветви



Пример 3

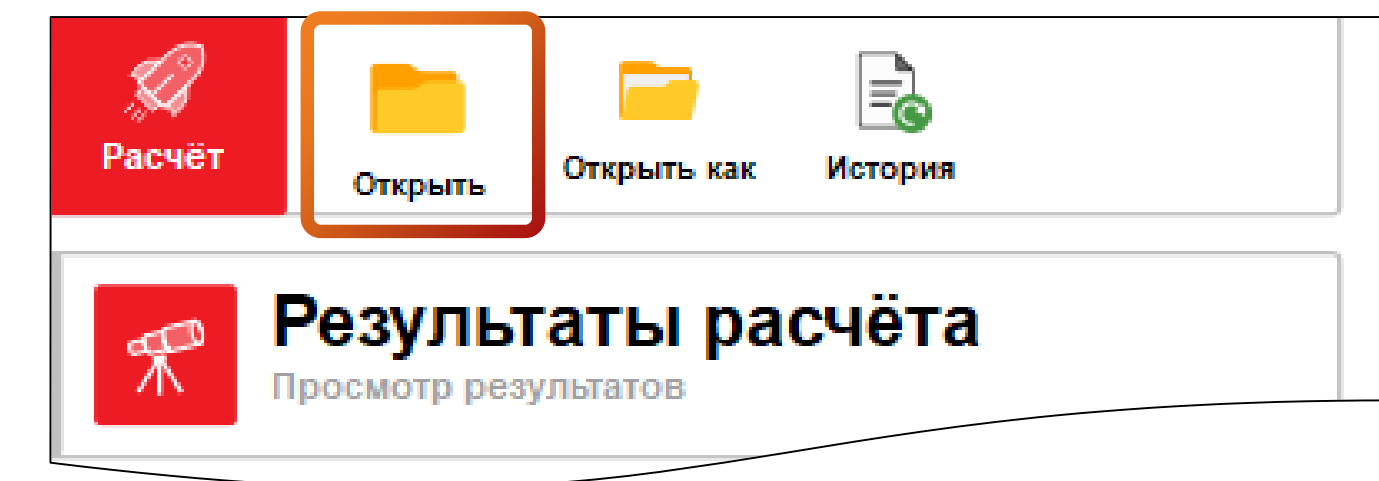
1. Автоматическое переключение скважины на закачку:

- Обводненность > 95%
- Макс. число скважины в каждой группе – 3
- Макс. число скважин 10

2. Откройте [Example3/Example3.data](#) в Симуляторе и запустите расчет

At 01.06.2036 00:00:00 well P_19 was converted to injector
At 01.07.2037 00:00:00 well P_20 was converted to injector
At 01.09.2037 00:00:00 well P_23 was converted to injector
At 01.12.2038 00:00:00 well P_32 was converted to injector
At 01.07.2039 00:00:00 well P_14 was converted to injector
At 01.11.2041 00:00:00 well P_18 was converted to injector
At 01.03.2042 00:00:00 well P_34 was converted to injector
Maximum number of wells in group G4 was converted to injector
At 01.05.2042 00:00:00 well P_21 was converted to injector
Maximum number of wells in group G2 was converted to injector
At 01.12.2042 00:00:00 well P_2 was converted to injector
At 01.08.2043 00:00:00 well P_1 was converted to injector
Maximum number of wells in group G1 was converted to injector
Maximum number of wells in the model was converted to injector

1



Пример 3: скрипт

1. Скрипт работает следующим образом:

- Перед расчетом создаются графики для хранения числа переключаемых скважин для группы и месторождения (`__init_script__`)

2. Остальная часть скрипта выполняется на каждом шаге:

- Извлечение числа переключенных скважин (`get_global_graph`)
- Для каждой группы, если число переключенных скважин для этой группы и месторождения еще не исчерпано, выбирается скважина с максимальным значением обводненности (`get_well_by_criterion`)
- Из выбранных скважин (если таковые имеются) выбирается скважина, имеющая максимальное значение обводненности, и переключается на закачку (`add_keyword`) с контролем по ВНР
- Обновление и сохранение числа переключенных скважин (`export`)

Пример 3: скрипт

```
from datetime import datetime

def __init_script__ ():
    #Создание графиков для хранения числа переключаемых скважин для группы и месторождения
    create_graph (name = 'group_count', type = 'group', default_value = 0)
    create_graph (name = 'all_count', type = 'field', default_value = 0)

def convert_with_limits():
    wct_lim=0.95
    group_limit=3
    all_limit=10
    groupnames=['G1','G2','G3','G4']
    #Извлечение числа переключенных скважин
    gc=get_global_graph(name = 'group_count')
    ac=get_global_graph(name = 'all_count')
    wells=[]
    wcts=[]
    #Если число переключенных скважин для месторождения еще не исчерпано
    if ac<all_limit:
        #Для каждой группы
        for gname in groupnames:
            g=get_group_by_name(gname)
            #Если число переключенных скважин для этой группы еще не исчерпано
            if gc[g]<group_limit:
                #выбирается скважина с максимальным значением обводненности
                w=get_well_by_criterion(group=g, method='max',crit=wwct)
                if wwct[w]>wct_lim:
                    wells.append(w)
                    wcts.append(float(wwct[w]))
```

```
APPLYSCRIPT
'INCLUDE/CONV.py' 'convert_with_limits' /
/
```

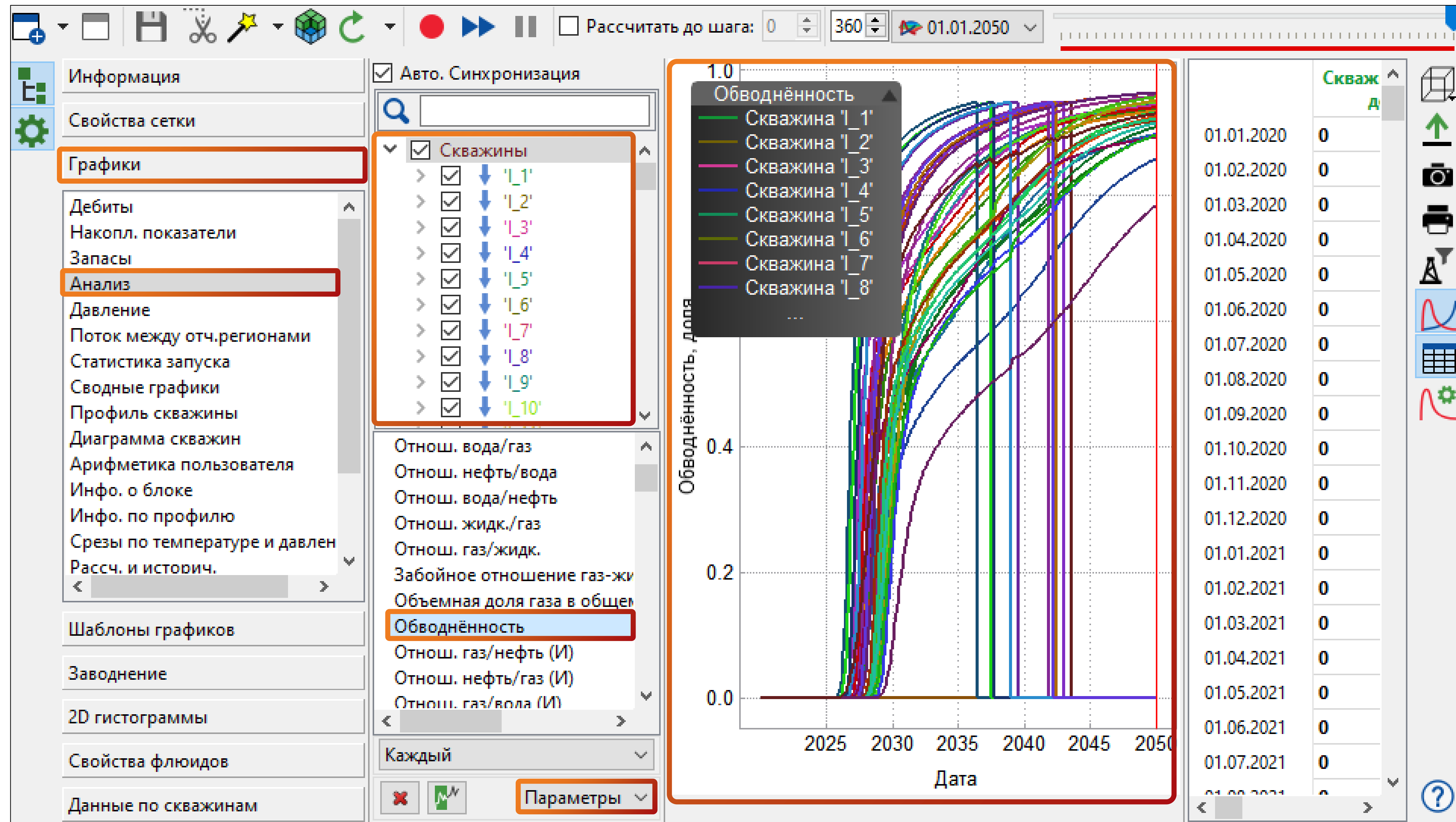


Пример 3: скрипт

```
#Из выбранных скважин (если таковые имеются)
if len(wells)>0:
    #выбирается скважина, имеющая максимальное значение обводненности
    w=wells[wcts.index(max(wcts)) ]
    #и переключается на закачку с контролем по ВНР
    add_keyword(
        """
        WCONINJE
        """+w.name+""" WATER OPEN BHP 2* 210 /
        /
        """
    )
    print("В",get_current_date ().strftime("%d.%m.%Y %H:%M:%S"), "скважина",w.name, "переключилась на закачку")
    gc[w.group]+=1
    if gc[w.group]==group_limit:
        print("В группе",w.group.name, "переключено на закачку предельное число скважин")
        ac+=1
    if ac==all_limit:
        print("В модели переключено на закачку предельное число скважин")
#Обновление и сохранение числа переключенных скважин
export(gc, name = 'group_count')
export(ac, name = 'all_count')
```

Пример 3: результаты

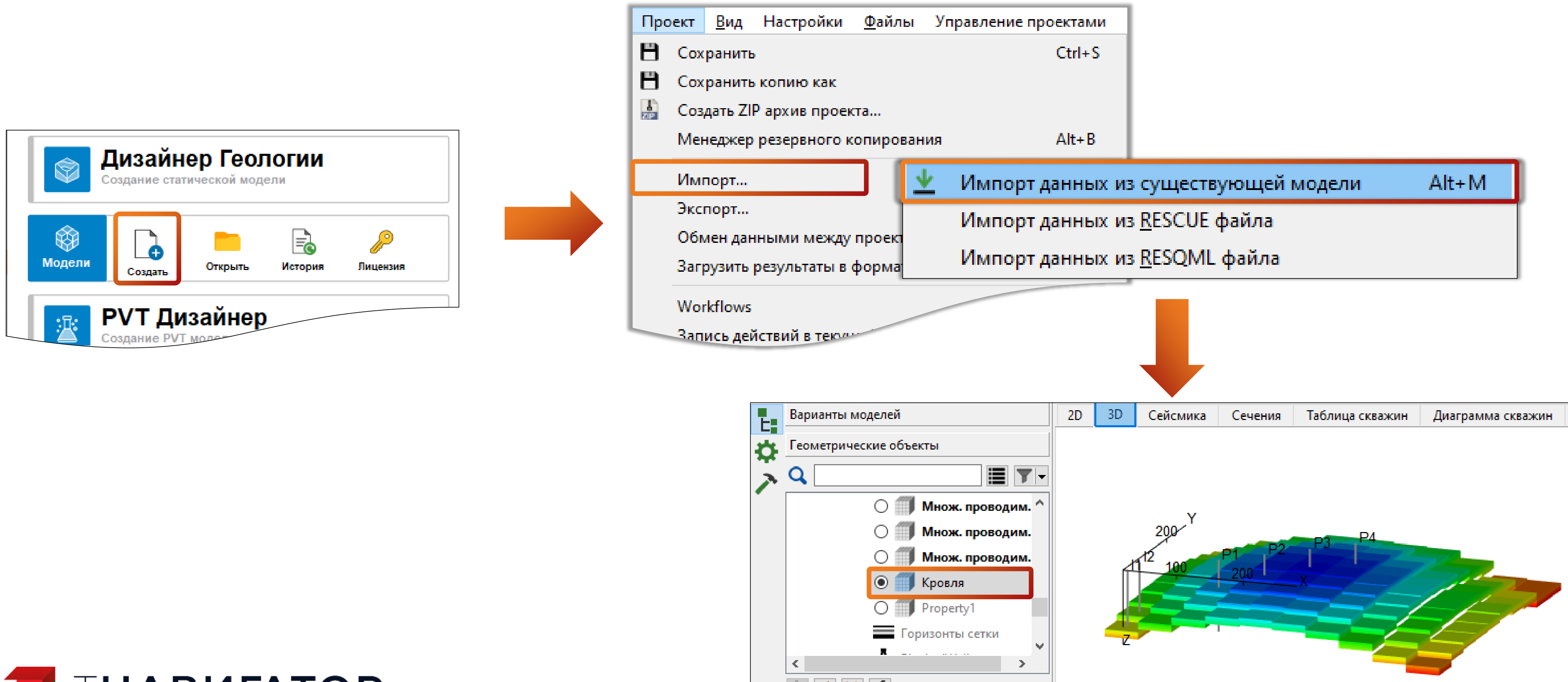
1. **Графики. Анализ.** Установите **Параметры**, выберите **Обводнённость**, активируйте все скважины. График обводнённости для скважин



Пример 4

Пласт содержит высокий процент CO₂. Продаваемый газ должен содержать не более 5% CO₂. Избыток CO₂ сепарируется и закачивается обратно в пласт.

1. Создайте новый проект **Дизайнера Моделей**
2. Импортируйте модель из **Example4/Example4.data**



Пример 4

1. Вкладка **Данные скважины** → **Стратегии**, откройте **Example4**
2. **ПКМ** на первую дату, выберите **Добавить новое правило**
3. Выберите **Другие. Применить скрипт**
4. Импортируйте скрипт **Example4/example4.py**
5. Установите **eos_rein** для **Функция**

The image shows a sequence of three screenshots from a software application, illustrating the steps to apply a script to a well strategy.

Скриншот 1: Показывает панель **Стратегии** (Strategies) в верхней части интерфейса. В левой панели под **Данные скважины** (Well Data) выделен файл **Example4**. В правой панели под **Правила для скважин** (Well Rules) выделена дата **01.01.1990**. В нижней части панели **Данные скважины** наведена курсором кнопка **Добавить новое правило...** (Add new rule...).

Скриншот 2: Показывает диалоговое окно **Добавить новое правило** (Add new rule). В нем выбран тип **Правила на шагах** (Rules on steps). В поле **Выбор шаг:** (Select step) выбрана дата **01.01.2020**. В поле **Добавить шаг:** (Add step) выбрана дата **15.03.2023 0:00:00**. В центре списка правил выделена кнопка **Применить скрипт** (Apply script).

Скриншот 3: Показывает панель **Стратегии** (Strategies) в верхней части интерфейса. В правой панели под **Правила для скважин** (Well Rules) выделена дата **01.01.1990**. В нижней части панели **Данные скважины** наведена курсором кнопка **Добавить новое правило...** (Add new rule...).

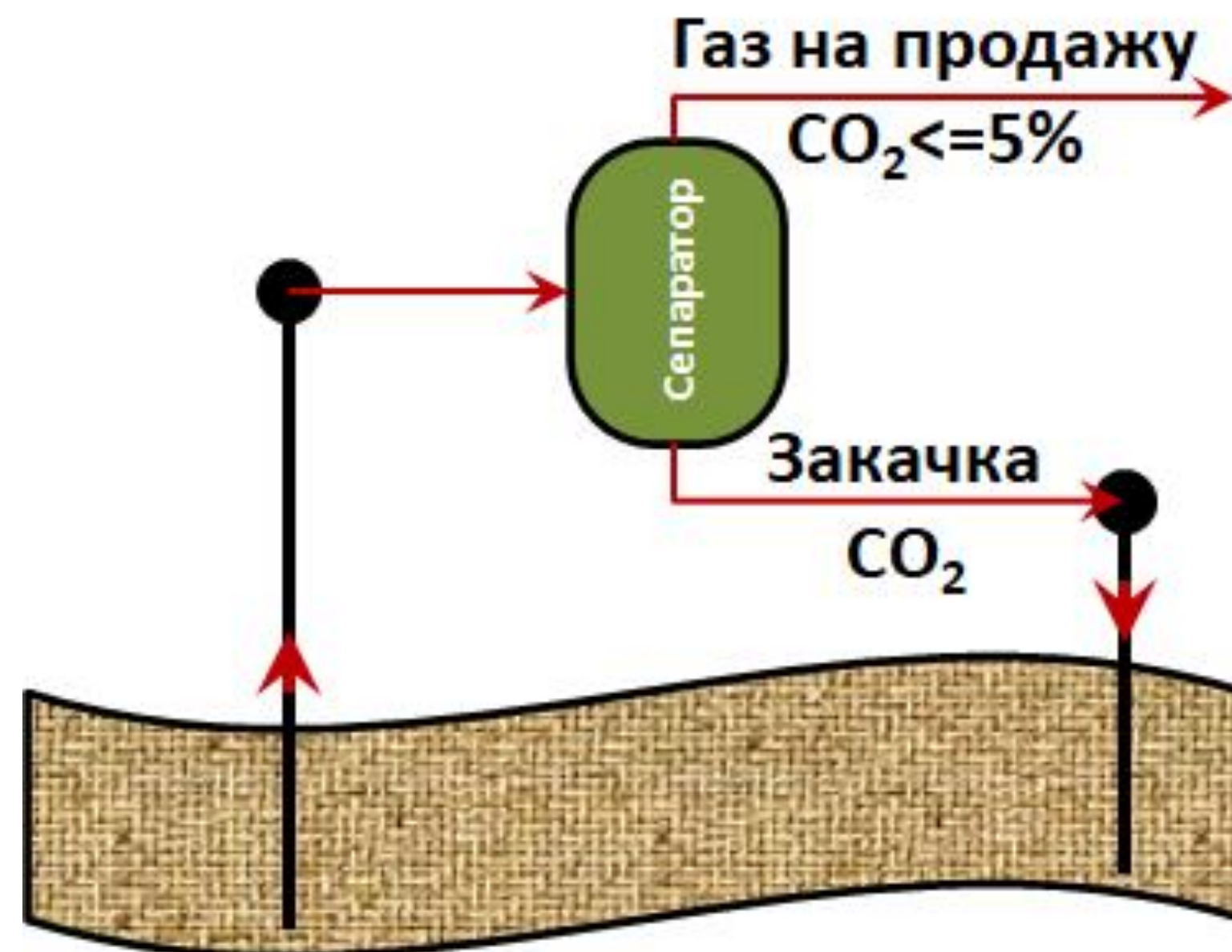
Скриншот 4: Показывает панель **Стратегии** (Strategies) в верхней части интерфейса. В правой панели под **Правила для скважин** (Well Rules) выделена дата **01.01.1990**. В нижней части панели **Данные скважины** наведена курсором кнопка **Добавить новое правило...** (Add new rule...).

Скриншот 5: Показывает панель **Стратегии** (Strategies) в верхней части интерфейса. В правой панели под **Правила для скважин** (Well Rules) выделена дата **01.01.1990**. В нижней части панели **Данные скважины** наведена курсором кнопка **Добавить новое правило...** (Add new rule...).

Пример 4: скрипт

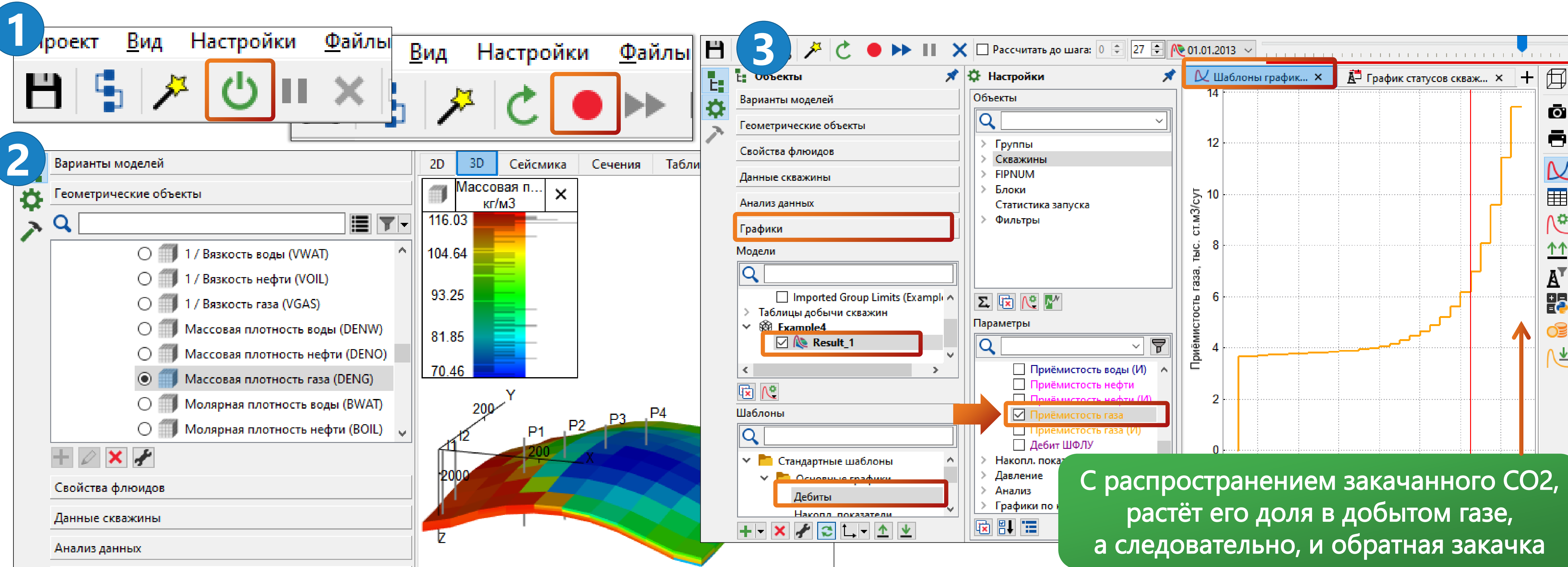
1. Скрипт работает следующим образом:

- Проверяет, является ли шаг отчетным (is_report_step). Если нет, то ничего не делает
- Вычисляет долю CO₂ в добытом газе (fcmpr_2/fmpr) и сравнивает ее с пороговой величиной (5%)
- Если доля CO₂ больше, чем пороговое значение, вычисляется количество CO₂, которое необходимо удалить из добытого газа (co2_to_inject)
- Вычисляет долю CO₂ на продажу (co2_sale_frac)
- Передает последнее значение в модель как аргумент ключевого слова **GRUPSALE** (add_keyword)



Пример 4: результаты

1. Инициализируйте и рассчитайте модель
2. Посмотрите свойства сетки
3. Перейдите в **Шаблоны графиков**. Выберите **Example4. Result_1**. Выберите **Скважины**. Установите **Стандартные шаблоны – Основные графики – Дебиты**. Выберите **Приёмистость газа** в списке **Параметров**. Посмотрите графики



Хотите узнать больше?

Описание функционала, учебные курсы и видеоуроки доступны на сайте:

www.rfdyn.ru

Остались вопросы?

Обратиться в техническую поддержку:

tnavigator@rfdyn.ru

