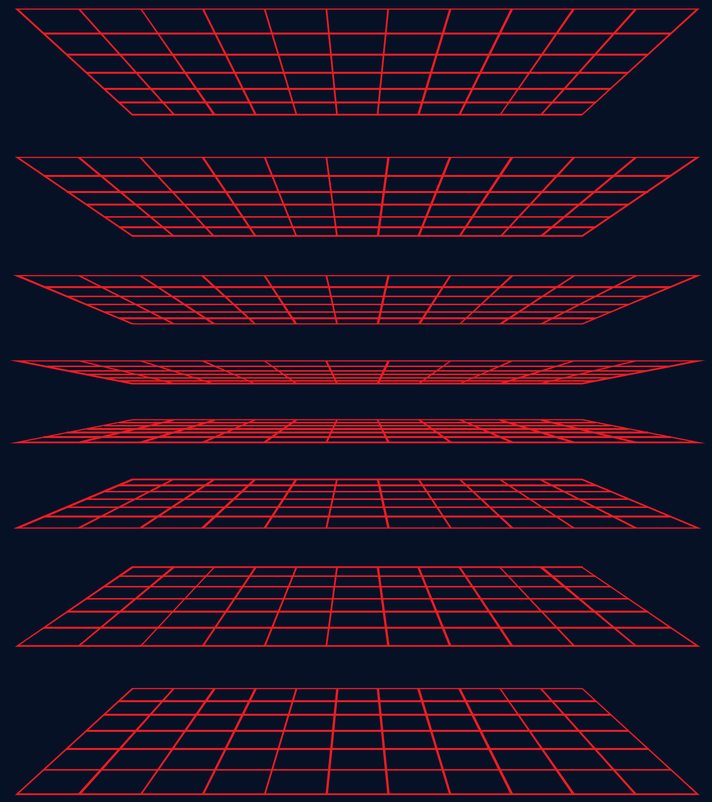
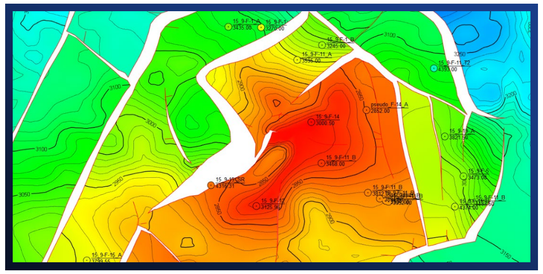
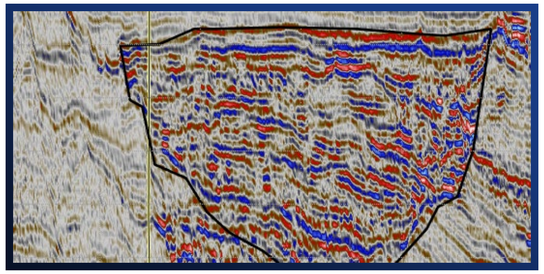
Управление проектами экосистемы тНавигатор с помощью API



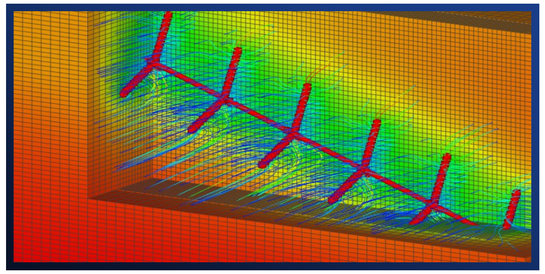
Геологическое моделирование



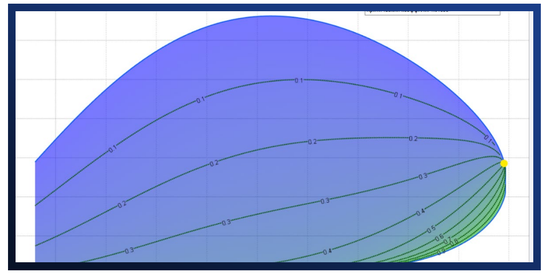
Интерпретация сейсмики



Гидродинамическое моделирование



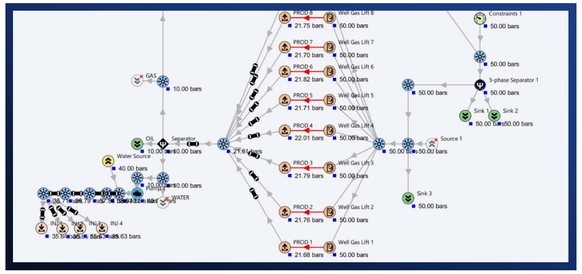
Моделирование PVT



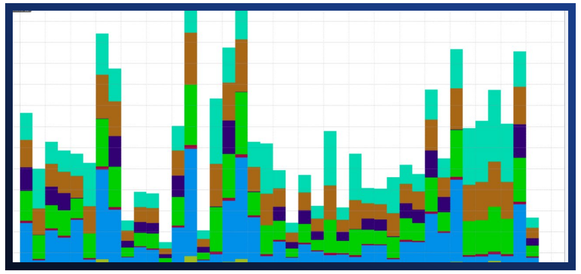
ТНАВИГАТОР

Полный цикл в одном приложении:

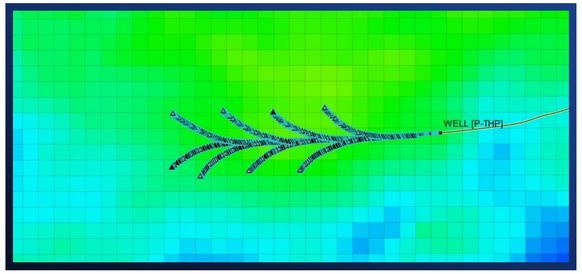
от интерпретации сейсмики до поверхностной сети сбора



Поверхностные сети



Оценка неопределённостей



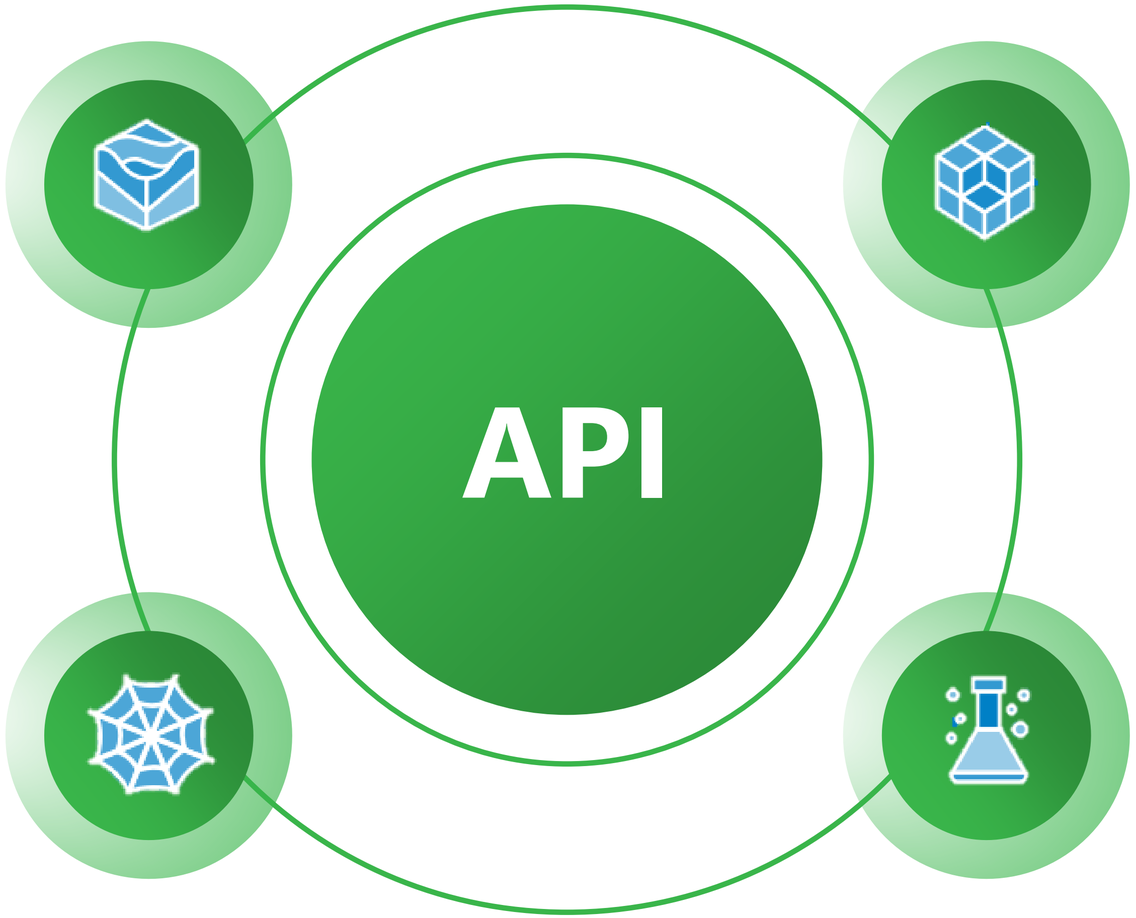
Моделирование скважин

API – Доступ ко всей экосистеме тНавигатор

Дизайнер Геологии

Сейсмика

Геостиринг



Дизайнер Моделей

МатБаланс

Дизайнер Сетей

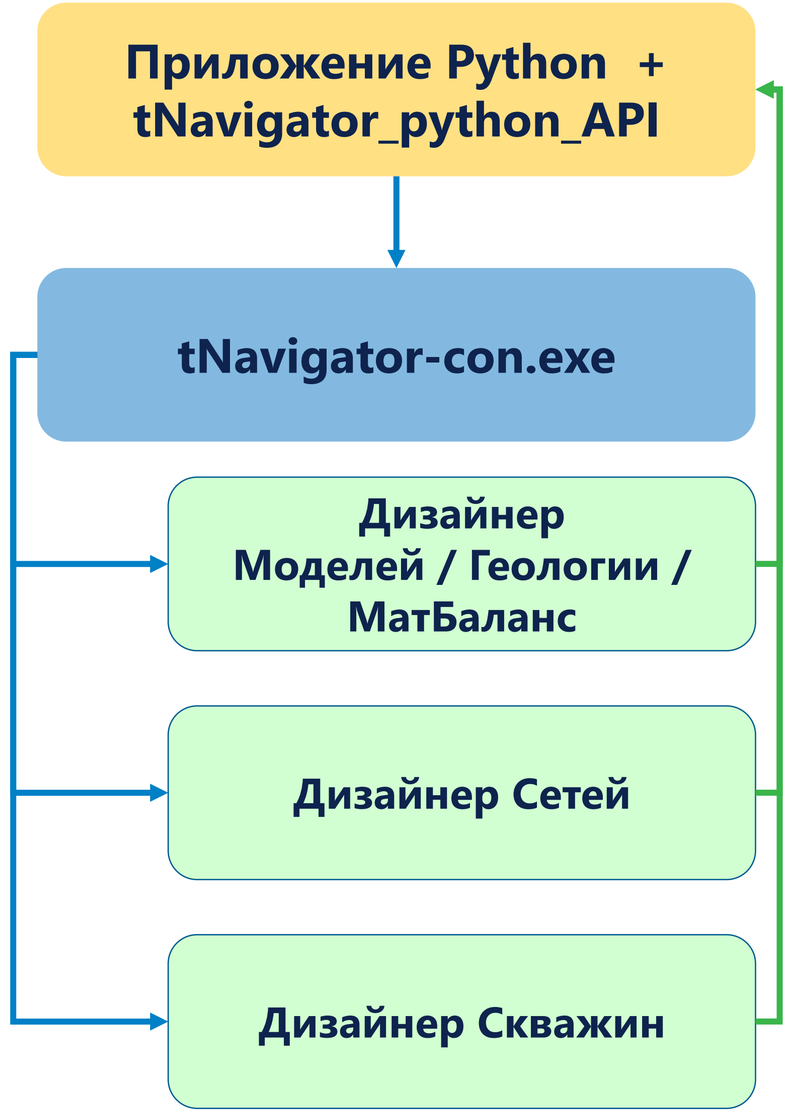
Дизайнер Скважин

Дизайнер PVT

Дизайнер ОФП

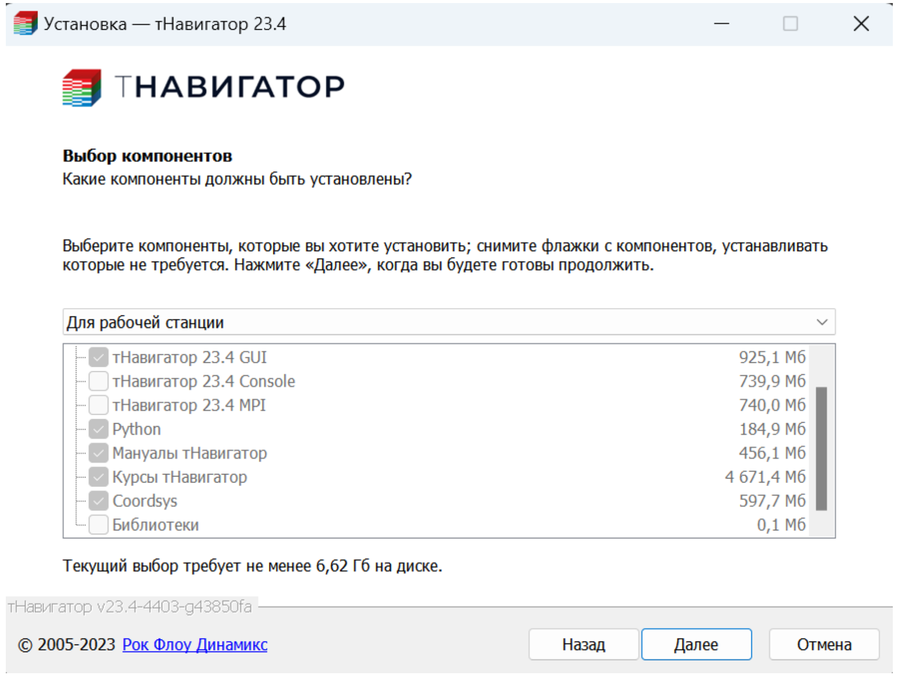
API - общая схема

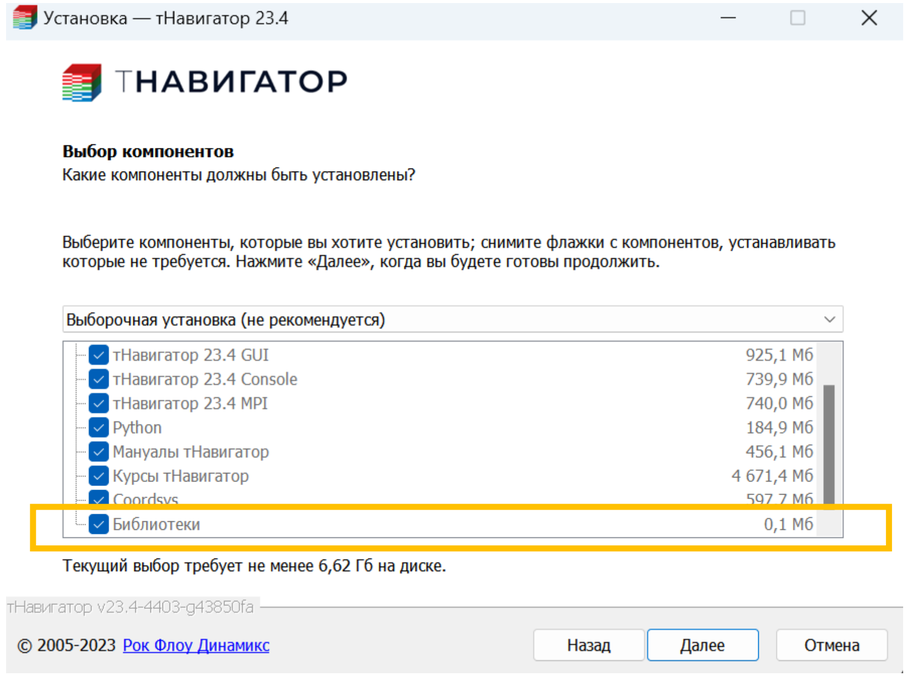
- Предназначен для управления проектом тНавигатор с помощью внешних управляющих систем
- Использует проприетарную python-библиотеку tNavigator_python_API
- Общая схема работы:
  - Приложение на Python обращается к тНавигатор
  - Подключается к проекту – поддержана вся экосистема тНавигатор
  - Передает на выполнение в проект команды, аналогичные расчетам Графа Моделирования
  - Получает результаты выполнения команд.



API – Установка tNavigator_python_API

- **Библиотека является частью инсталлятора тНавигатор**
- **В процессе установки необходимо выбрать режим «Выборочная установка» и отметить пункт «Библиотеки» и консольную версию «тНавигатор Console»**

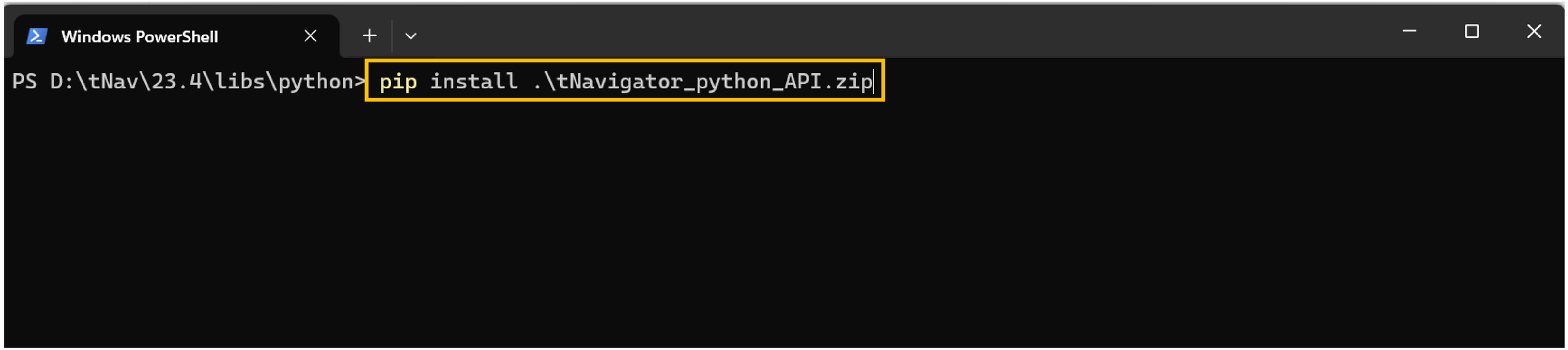




API – Установка tNavigator_python_API

- После завершения установки тНавигатор, библиотека будет расположена в директории установки ПО
- Установка производится с помощью стандартного метода установки библиотек Python:

  $pip install <package name>



API – Шаблон python-приложения

1. Подключение библиотеки

```
# Подключение необходимых библиотек
import tNavigator_python_API as tnav
from tNavigator_python_API import ProjectType
```

2. Подключение к Проекту

```
# Создание объекта Connection для запуска исполняемого файла tNavigator-con
conn = tnav.Connection(path_to_exe='d:/tNav/24.1/tNavigator-con.exe')
# Подключение к проекту mНавигатор
snp_project = conn.open_project(path='E:/Projects/API/tnav_api/api_calc.snp’)
```

3. Запрос лицензии

```
# Получение списка проектов Дизайнера Скважин, которые присутствуют в snp-проекте и запись в переменную "name" имени первого проекта
wd_proj_list = snp_project.get_list_of_subprojects(type=ProjectType.WD)
name = wd_proj_list[0]

# Подключение к подпроекту mНавигатор (Дизайнер Скважин, Дизайнер Сетей итд)
well_designer_project = snp_project.get_subproject_by_name(name=name, type=ProjectType.WD)
# Запрос лицензии модуля, которым необходимо управлять
well_designer_project.run_py_code(code="request_license_feature (feature='FEAT_WELL_DESIGNER')")
```

4. Управление проектом

```
# Выполнение команды графа моделирования
vfp_param = well_designer_project.run_py_code(code=
"""vfp = get_vfp_table_by_name (name='VFP 1')
fr = vfp.get_friction ()
hc = vfp.get_hydrostatic ()
a = [fr, hc]
return a""")
# Печать данных из переменной vfp_param в лог
print(vfp_param)
well_designer_project.close_project() # закрытие проекта при завершении работы с ним
```

API – Шаблон python-приложения

```python
# Подключение необходимых библиотек
import tNavigator_python_API as tnav
from tNavigator_python_API import ProjectType

# Создание объекта Connection для запуска исполняемого файла tNavigator-con
conn = tnav.Connection(path_to_exe='d:/tNav/24.1/tNavigator-con.exe')

# Подключение к проекту тНавигатор
snp_project = conn.open_project(path='E:/Projects/API/tnav_api/api_calc.snp')

# Получение списка проектов Дизайнера Скважин, которые присутствуют в snp-проекте и запись в переменную "name" имени первого проекта
wd_proj_list = snp_project.get_list_of_subprojects(type=ProjectType.WD)
name = wd_proj_list[0]

# Подключение к подпроекту тНавигатор (Дизайнер Скважин, Дизайнер Сетей итд)
well_designer_project = snp_project.get_subproject_by_name(name=name, type=ProjectType.WD)

# Запрос лицензии модуля, которым необходимо управлять
well_designer_project.run_py_code(code="request_license_feature (feature='FEAT_WELL_DESIGNER')")

# Выполнение команды графа моделирования
vfp_param = well_designer_project.run_py_code(code=
"""vfp = get_vfp_table_by_name (name='VFP 1')
fr = vfp.get_friction ()
hc = vfp.get_hydrostatic ()
a = [fr, hc]
return a""")
# Печать данных из переменной vfp_param в лог
print(vfp_param)
well_designer_project.close_project()
```

Результат выполнения кода:

```
E:\Projects\API\tnav_api\.venv\Scripts\python.exe E:\Projects\API\tnav_api\API_.py
[0.86, 1.0]

Process finished with exit code 0
```

API – Шаблон python-приложения с “with…”

```
# Подключение необходимых библиотек
import tNavigator_python_API as tnav
from tNavigator_python_API import ProjectType

# Создание объекта Connection для запуска исполняемого файла tNavigator-con
conn = tnav.Connection(path_to_exe='d:/tNav/24.1/tNavigator-con.exe')

# Подключение к проекту тНавигатор
with conn.open_project(path='E:/Projects/API/tnav_api/api_calc.snp', save_on_close=False) as snp_project:
    # Получение списка проектов Дизайнера Скважин, которые присутствуют в snp-проекте и запись в переменную "name" имени первого проекта
    wd_proj_list = snp_project.get_list_of_subprojects(type=ProjectType.WD)
    name = wd_proj_list[0]

    # Подключение к подпроекту тНавигатор (Дизайнер Скважин, Дизайнер Сетей итд)
    well_designer_project = snp_project.get_subproject_by_name(name=name, type=ProjectType.WD)

    # Запрос лицензии модуля, которым необходимо управлять
    well_designer_project.run_py_code(code="""request_license_feature (feature='FEAT_WELL_DESIGNER')""")

    # Выполнение команды графа моделирования
    vfp_param = well_designer_project.run_py_code(code="""
vfp = get_vfp_table_by_name (name='VFP 1')
fr = vfp.get_friction ()
hc = vfp.get_hydrostatic ()
a = [fr, hc]
return a
""")
    # Печать данных из переменной vfp_param в лог
    print(vfp_param)
```

Результат выполнения кода:

```
E:\Projects\API\tnav_api\.venv\Scripts\python.exe E:\Projects\API\tnav_api\API_.py
[0.86, 1.0]

Process finished with exit code 0
```

Методы и функции API

- **connection() – функция для соединения python-приложения с тНавигатор**

- **Аргументы функции:**

```
connection(path_to_exe: str,
           minimum_required_version: Any = None,
           license_wait_time_limit_secs: int = None)
```

- **Пример задания функции:**

```
# Подключение необходимых библиотек
import tNavigator_python_API as tnav
# Создание объекта Connection для запуска исполняемого файла tNavigator
conn = tnav.Connection(path_to_exe='d:/tNav/24.1/tNavigator-con.exe')
```

Методы и функции API

- **open_project() – функция для соединения python-приложения с Проектом тНавигатор**
- **Аргументы функции:**

```
open_project(path: str,
             save_on_close: bool = True/False)
```

- **Пример задания функции:**
  - Выполнить подключение к проекту без сохранения при выходе:

```
with conn.open_project(path='E:/Projects/API/tnav_api/api_calc.snp', save_on_close=False) as snp_project:
    wd_proj_list = snp_project.get_list_of_subprojects(type=ProjectType.WD)
    print(wd_proj_list)
```

  - Выполнить подключение к проекту с сохранением при выходе:

```
with conn.open_project(path='E:/Projects/API/tnav_api/api_calc.snp', save_on_close=True) as snp_project:
    wd_proj_list = snp_project.get_list_of_subprojects(type=ProjectType.WD)
    print(wd_proj_list)
```

Методы и функции API

- **ProjectType** – class для определения типа подпроекта, к которому идет подключение
- **Доступны следующие типы:**
  - ProjectType.MD – подключение проекта Дизайнера Моделей
  - ProjectType.GD – подключение проекта Дизайнера Геологии
  - ProjectType.ND – подключение проекта Дизайнера Сетей
  - ProjectType.WD – подключение проекта Дизайнера Скважин
  - ProjectType.RP – подключение проекта Дизайнера ОФП
  - ProjectType.PVT – подключение проекта Дизайнера PVT
- **Пример использования:**

```
with conn.open_project(path='E:/Projects/api_calc.snp', save_on_close=True) as snp_project:
    wd_proj_list = snp_project.get_list_of_subprojects(type=ProjectType.WD)
    print(wd_proj_list)
```

Методы и функции API

- **get_list_of_subprojects() – функция возвращает список подпроектов указанного типа**

- **Аргументы функции:**

  get_list_of_subprojects(type=ProjectType.MD/GD/ND/WD/PVT/RP)

- **Пример задания функции:**

```
with conn.open_project(path='E:/Projects/API/tnav_api/api_calc.snp', save_on_close=False) as snp_project:
    wd_proj_list = snp_project.get_list_of_subprojects(type=ProjectType.WD)
    print(wd_proj_list)
```

Методы и функции API

- **get_subproject_by_name() – функция возвращает python-объект, с выбранным подпроектом**

- **Аргументы функции:**

  get_subproject_by_name(name=<Имя проекта>,
  type= ProjectType.MD/GD/ND/WD/PVT/RP)

- **Пример задания функции:**

  well_designer_project = snp_project.**get_subproject_by_name**(name="Well_1", type=ProjectType.WD)

Методы и функции API

- **run_py_code() – функция отвечает за передачу и выполнение проекте тНавигатор указанного python-кода**

run_py_code(code: str = None, file: str = None, files: list = None, save: bool = False)

code	Код на языке Python, который будет передан в проект тНавигатор и выполнен
file	Путь к файлу ***.py**, в котором могут быть сохранены функции. Функции могут быть вызваны в аргументе “code”
files	Путь к списку файлов с функциями, которые могут быть вызваны в аргументе “code”
save	Логический флаг, определяющий необходимость сохранения проекта после выполнения функции

API – Шаблон python-приложения

1. Подключение библиотеки

```
# Подключение необходимых библиотек
import tNavigator_python_API as tnav
from tNavigator_python_API import ProjectType
```

2. Подключение к Проекту

```
# Создание объекта Connection для запуска исполняемого файла tNavigator-con
conn = tnav.Connection(path_to_exe='d:/tNav/24.1/tNavigator-con.exe')
# Подключение к проекту mНавигатор
snp_project = conn.open_project(path='E:/Projects/API/tnav_api/api_calc.snp’)
```

3. Запрос лицензии

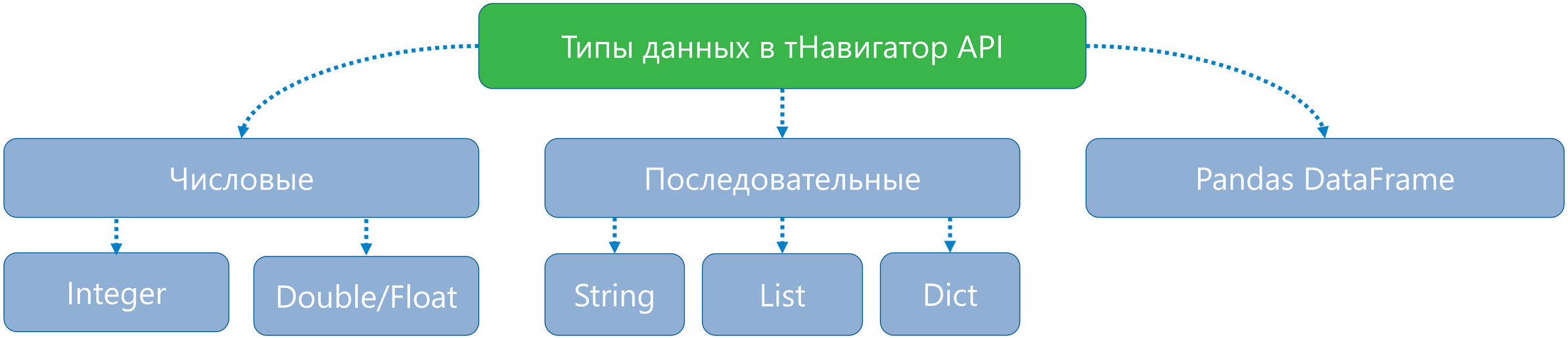
```
# Получение списка проектов Дизайнера Скважин, которые присутствуют в snp-проекте и запись в переменную "name" имени первого проекта
wd_proj_list = snp_project.get_list_of_subprojects(type=ProjectType.WD)
name = wd_proj_list[0]

# Подключение к подпроекту mНавигатор (Дизайнер Скважин, Дизайнер Сетей итд)
well_designer_project = snp_project.get_subproject_by_name(name=name, type=ProjectType.WD)
# Запрос лицензии модуля, которым необходимо управлять
well_designer_project.run_py_code(code="request_license_feature (feature='FEAT_WELL_DESIGNER')")
```

4. Управление проектом

```
# Выполнение команды графа моделирования
vfp_param = well_designer_project.run_py_code(code=
"""vfp = get_vfp_table_by_name (name='VFP 1')
fr = vfp.get_friction ()
hc = vfp.get_hydrostatic ()
a = [fr, hc]
return a""")
# Печать данных из переменной vfp_param в лог
print(vfp_param)
well_designer_project.close_project() # закрытие проекта при завершении работы с ним
```

API – типы возвращаемых объектов



- **API поддерживает получение данных из проектов экосистемы тНавигатор**
- **Поддержаны основные типы данных, необходимые для эффективной работы с API**
- **Использование:**
    1. **В рамках кода функции необходимо подготовить данные в нужном формате**
    2. **Оператор «return» возвращает результат работы функции Python**

API – Пример получения объектов

- **Задача** – выгрузить коэффициенты адаптации потерь давления по стволу скважины за счет гидростатики и трения
- Используем метод **run_py_code** для передачи в проект тНавигатор python-кода, который:
  1. Записывает в переменную «vfp» объект VFP-корреляции с помощью:
     get_vfp_table_by_name (name='VFP 1’)
  2. Записывает в переменную «fr» значение коэффициента за трение с помощью:
     vfp.get_friction ()
  3. Записывает в переменную «hc» значение коэффициента за гидростатику с помощью:
     vfp.get_hydrostatic ()
  4. Записывает в переменную «a» список [fr, hc]
  5. С помощью оператора «return» функция возвращает объект, записанный в переменную «a»

Пример кода:

```
# Выполнение команды графа моделирования
vfp_param = well_designer_project.run_py_code(code=
"""vfp = get_vfp_table_by_name (name='VFP 1’)
      fr = vfp.get_friction ()
      hc = vfp.get_hydrostatic ()
      a = [fr, hc]
      return a""")
# Печать данных из переменной vfp_param в лог
print(vfp_param)
```

Результат выполнения кода:

```
E:\Projects\API\tnav_api\.venv\Scripts\python.exe E:\Projects\API\tnav_api\API_.py
[0.86, 1.0]

Process finished with exit code 0
```

API – Использование переменных, циклы

Передача переменных в строковую команду производится с помощью стандартного функционала python:

- **f-строки:**

  vfp_name = ‘VFP 1’
  vfp_param = well_designer_project.run_py_code(code=
  f”””vfp = get_vfp_table_by_name (name='{**vfp_name**}’)”””)

- **Метод форматирования строк format():**

  vfp_name = ‘VFP 1’
  vfp_param = well_designer_project.run_py_code(code=
  """vfp = get_vfp_table_by_name (name='{}’)”””.format(**vfp_name**))

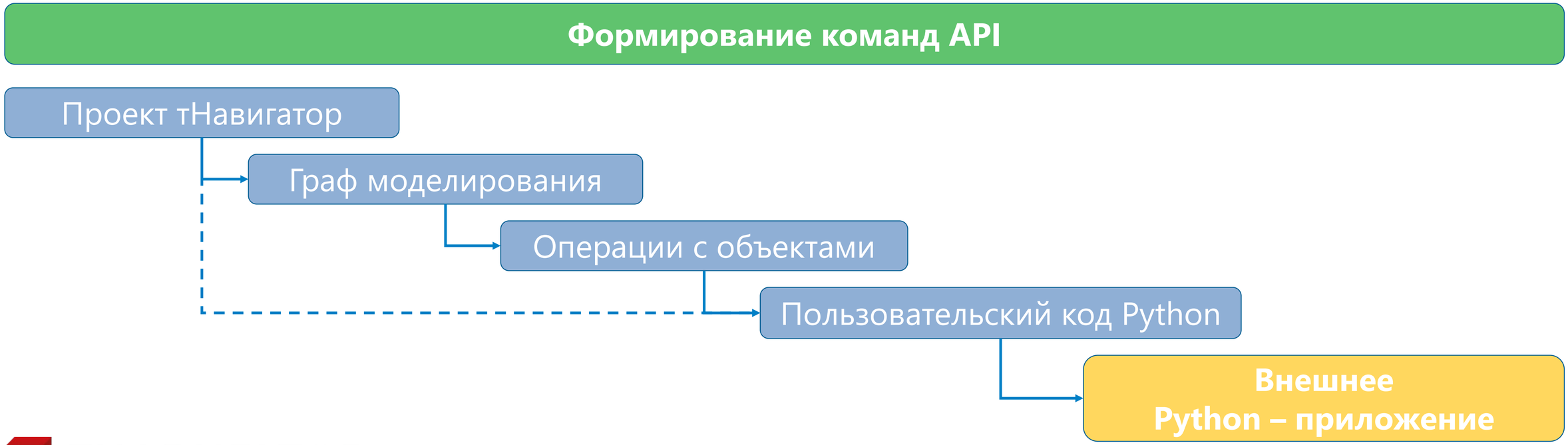
Пример кода:

```
vfp_list = ['VFP_1', 'VFP_2', 'VFP_3']

for i in vfp_list:
    vfp_name = i
    vfp_param = well_designer_project.run_py_code(code=f"""
vfp = get_vfp_table_by_name (name='{vfp_name}')
fr = vfp.get_friction ()
hc = vfp.get_hydrostatic ()
a = [fr, hc]
return a
""")
    print(vfp_param)
```

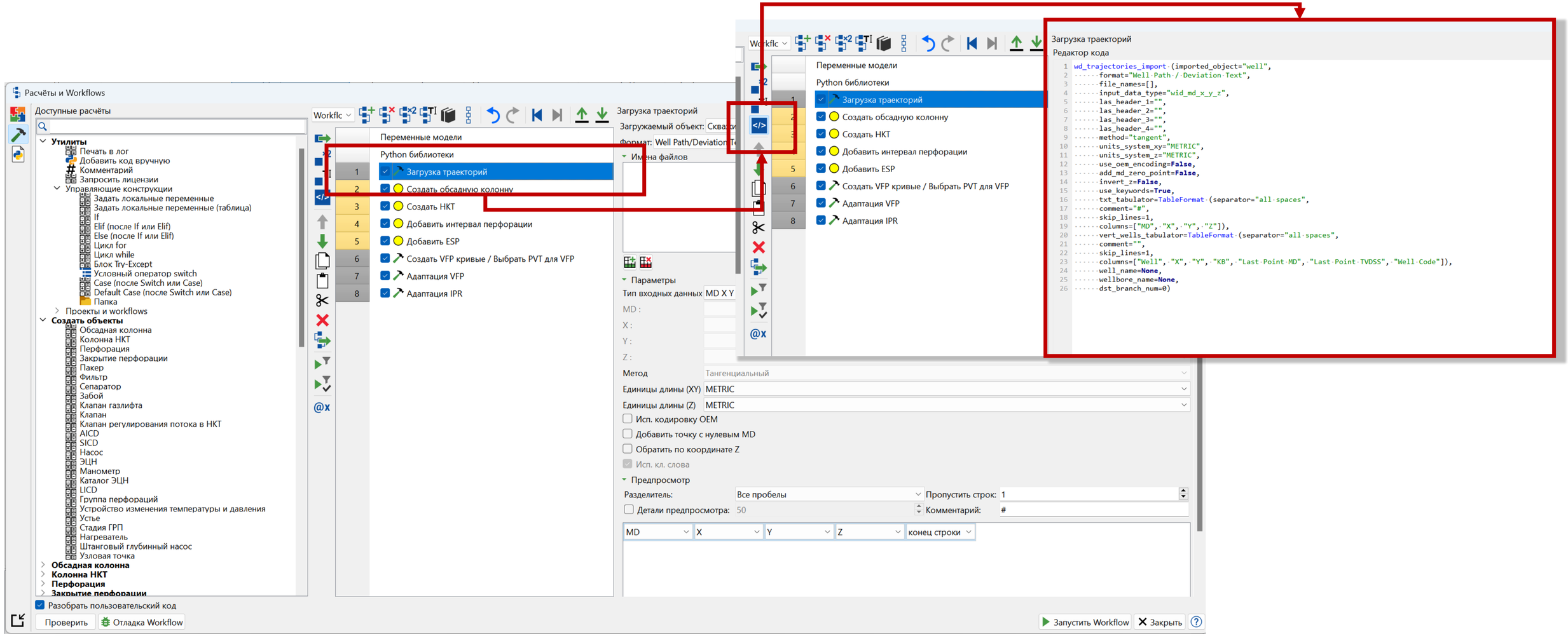
Формирование python-кода для API

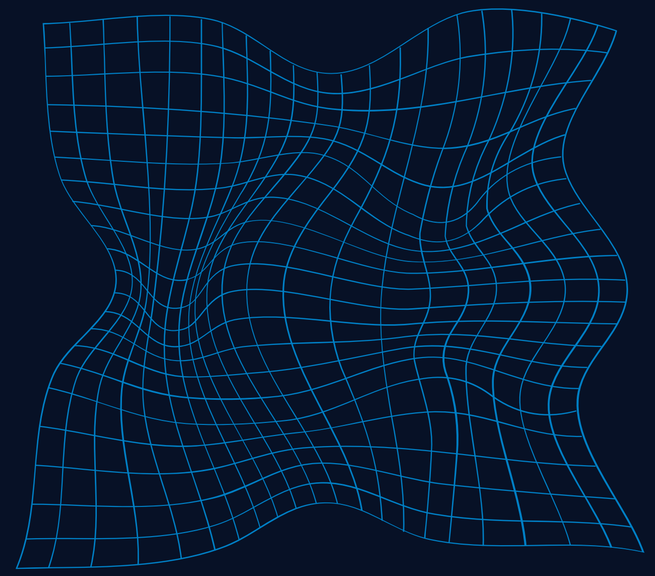
- **Команда API может быть сформирована автоматически, с помощью стандартного функционала графа моделирования тНавигатор**
- **Ключевые преимущества – Формирование команд в виде графических схем, объединение команд, использование внутренних циклов, логических выражений и проверка работы в графическом интерфейсе**
- **Граф моделирование поддерживает любые операции с объектами проекта – создание, редактирование и получение параметров**



Формирование python-кода с помощью Графа

- **Все расчеты Графа моделирования могут быть преобразованы в python-код**





Хотите узнать больше?

Описание функционала, учебные курсы и видеоуроки доступны на сайте:

www.rfdyn.ru

Остались вопросы?

Обратиться в техническую поддержку:

tnavigator@rfdyn.ru