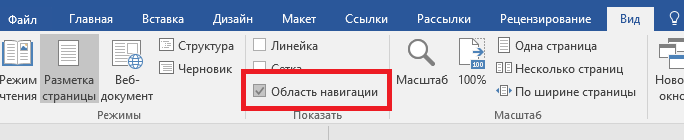
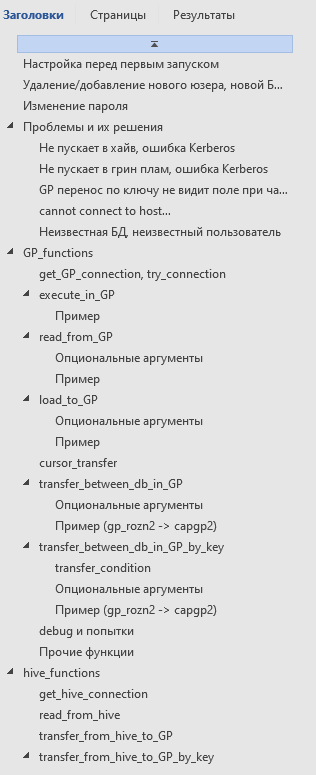
**Лучше включить навигацию**





# Настройка перед первым запуском

####################################################################

#эта часть нужна если файлы с модулями лежат в отдельной папке

#в sys.path.insert должен быть указан путь к этой папке

#если запускаем скрипт/ноутбук из той же папки где лежат модули, эту часть

#можно пропустить

import sys

sys.path.insert(0, '/home/11111111\_omega-sbrf-ru/notebooks/scripts/my\_modules')

####################################################################

from settings\_funs import \*

**#создаём переменные**

password\_file='file\_name' #псевдоним пароля для GP и почты

hive\_password\_file='hive\_file\_name' #псевдоним пароля для hive

user\_name='1111' #логин

default\_list=[

"mail@omega.sbrf.ru", #дефолтная почта с которой летят письма

user\_name, #дефолтный логин для хайва и почты

]

gp\_pass="пароль от омеги"

hive\_pass="пароль от хайва"

**#заполняем файлы с настройками**

clear\_users\_settings()

create\_password\_file(password\_file, gp\_pass)

create\_password\_file(hive\_password\_file, hive\_pass)

add\_new\_GP\_user(user\_name, password\_file)

add\_new\_hive\_user(user\_name, hive\_password\_file)

set\_defaults\_for\_mail(default\_list)

# Удаление/добавление нового юзера, новой БД в GP

####################################################################

#эта часть нужна если файлы с модулями лежат в отдельной папке

#в sys.path.insert должен быть указан путь к этой папке

#если запускаем скрипт/ноутбук из той же папки где лежат модули, эту часть

#можно пропустить

import sys

sys.path.insert(0, '/home/11111111\_omega-sbrf-ru/notebooks/scripts/my\_modules')

####################################################################

from settings\_funs import \*

**#Добавление нового юзера для GP**

password\_file='user\_file\_name' #псевдоним пароля для GP и почты

user\_name='11111111' #логин

gp\_pass="тут пароль от омеги"

create\_password\_file(password\_file, gp\_pass)

add\_new\_GP\_user(user\_name, password\_file)

**#Удаление юзера GP**

remove\_GP\_user('11111111')

#####################################################################

**#Добавление нового юзера для hive**

password\_file='user\_file\_name' #псевдоним пароля для hive

user\_name='11111111' #логин

hive\_pass="тут пароль от hive"

add\_new\_hive\_user(user\_name, password\_file)

**#Удаление юзера hive**

remove\_hive\_user('11111111')

#####################################################################

**#Добавление новой БД в GP**

add\_new\_GP\_db('mis', #псевдоним по которому обращаемся к БД в функциях из GP\_functions

'gp\_mis2', #реальное имя БД

'host' #хост БД

)

**#Удаление БД**

remove\_GP\_db('mis') #псевдоним указанный раннее

#можно использовать разные псевдонимы для одной и той же БД

#тогда функции из GP\_functions будут открывать разные коннекшены

#для разных псевдонимов

#(но эту фишку я не проверял)

#если использовать один и тот же псевдоним – раннее созданный коннекшн

#переиспользуется (не создаётся заново), что экономит время

# Изменение пароля

####################################################################

#эта часть нужна если файлы с модулями лежат в отдельной папке

#в sys.path.insert должен быть указан путь к этой папке

#если запускаем скрипт/ноутбук из той же папки где лежат модули, эту часть

#можно пропустить

import sys

sys.path.insert(0, '/home/11111111\_omega-sbrf-ru/notebooks/scripts/my\_modules')

####################################################################

from settings\_funs import \*

**#если поменяли пароль**

create\_password\_file('псевдоним пароля', 'пароль')

#тот же псевдоним пароля что указывали при настройке перед первым запуском

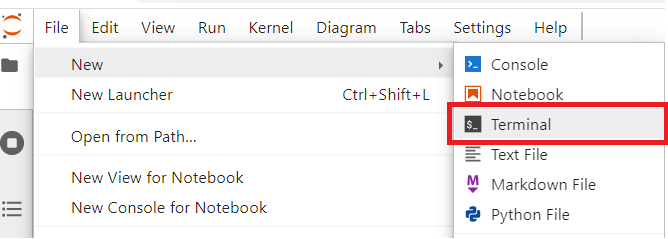
#(в разделе «Настройка перед первым запуском» см. переменную password\_file если меняем #пароль для грин плама или hive\_password\_file если меняем пароль для хайва)

# **Проблемы и их решения**

## **Не пускает в хайв, ошибка Kerberos**

**Решение:**

1. Зайти в терминал (в юпитер лабе: левый верхний угол – file – new –Terminal)



1. Вбить kinit {омеговский\_логин}@DF.SBRF.RU (пример 11111111@OMEGA.SBRF.RU), нажать Enter
2. Вбить хайвовский пароль, нажать Enter **(пароль при вводе не отображается, это нормально)**
3. Если выдаст что пароль устарел, придумать новый
4. Сохранить новый пароль (см. раздел «Изменение пароля»), псевдоним должен тот же что был указан при первой настройке (см. переменную hive\_password\_file в разделе «Настройка перед первым запуском)
5. Если запускаем скрипт из ноутбука – рестартнуть kernel

(кнопка выделена красным)



1. Запустить скрипт ещё раз

## **Не пускает в грин плам, ошибка Kerberos**

**Решение:** см. раздел «Изменение пароля», после изменения нужно рестартнуть kernel (если запускаем через ноутбук)



## **GP перенос по ключу не видит поле при частичной перекладке (7-ой элемент листа False)**

**Решение:**

Функция при частичной перекладке проверяет уже переложенные значения

(это было ошибочное решение, в GP\_functions 2.0 я от него откажусь)

Например, если переносим по значения поля date, то она попытается найти уже переложенные значения поля date в целевой табличке.

Но если в целевой табличке поле date названо по-другому (например, day), то вылетит ошибка.

Костыль:

использовать аргумент target\_table\_keys\_alias\_list=[‘day’]

## **cannot connect to host…**

**Решение:** проблема на стороне GP, ждём, коннектимся снова

## **Неизвестная БД, неизвестный пользователь**

**Решение:** см. раздел «Добавление/удаление нового юзера, новой БД в GP», после добавления нужно рестартнуть kernel (если запускаем через ноутбук)



# **GP\_functions**

У большинства функций 1 обязательный аргумент – лист с параметрами. Изначально у такого подхода были причины, сейчас это атавизм, но переписывать все мои скрипты по-новому лень, возможно когда-нибудь появится GP\_functions 2.0 без обратной совместимости.

## **get\_GP\_connection, try\_connection**

Сами по себе эти функции не используются, но их используют другие, более продвинутые функции.

**get\_GP\_connection** в качестве на вход принимает лист где первый элемент – псевдоним БД, второй – логин юзера от имени, которого к ней подключаемся. Пример: [‘1111111’, ‘gp\_mis’]

Если:

* в глобальном нэймспейсе не существует коннекшена для такого псевдонима БД
* или он закрыт
* или он существует, но для другого юзера,

То функция возвращает в глобальный неймспейс коннекшн для указанного псевдонима от имени указанного юзера.

Иначе функция ничего не делает.

**try\_connection** – На вход получает тот же лист что и get\_GP\_connection, проверяет работает ли коннекшн через считывание select 1. Это нужно на случай если коннекшн отрубили со стороны GP, но в созданном функцией get\_GP\_connection объекте прометка о закрытии коннекшена не проставилась (а такое бывает часто)

## **execute\_in\_GP**

**execute\_in\_GP –** исполняет запрос в GP.На вход принимает лист, где первый элемент которого – комментарий, второй – лист отправляемый в try\_connection, третий – sql.

Функция исполняет запрос в указанной БД и не возвращает никаких объектов.

### **Пример**

execute\_in\_GP(

[

‘Создаём временные таблички’,

[‘1111111’, ‘gp\_mis’],

"""

create temporary table my\_test as(

select \*

from table

);

"""

]

)

## **read\_from\_GP**

**read\_from\_GP–** считывает данные из GP в df.На вход принимает лист, где первый элемент которого – комментарий, второй – лист отправляемый в try\_connection, третий – sql.

Функция проверяет есть ли в данных bigint или json, если есть, кастит их в text, затем возвращает df с считанными данными.

### **Опциональные аргументы**

**order\_list** - лист с указанием как сортировать данные в возвращаемом df. Без указания этого аргумента данные в возвращаемом df сортируются случайно, независимо от того какая сортировка указана в sql запросе.

Это связано с тем, что для определения типа данных в процессе проверки на bigint и json создаётся временная табличка distributed by randomly, что сбивает сортировку. Из этой таблички данные и считываются в возвращаемый df.

Distriubuted by randomly необходим, т. к. без этого для дистрибуции могут быть использованы поля типа bigint или json, тогда изменить их тип на text через alter table не получится.

### **Пример**

df=read\_from\_GP(

[

‘Cчитывание данных’,

[‘1111111’, ‘gp\_mis’],

"""

Select \*

From table

"""

],

order\_list=[‘asc’, ‘desc’] #опциональный аргумент

#в данном случае df будет #отсортирован по возрастание

#по первой колонке и

#по убыванию по второй

)

## **load\_to\_GP**

**load\_to\_GP–** загружает данные в GP из df. На вход принимает лист,

где:

* 1-ый элемент – комментарий
* 2-ой – лист, отправляемый в try\_connection,
* 3-ий - название таблички. Если грузим в обычную табличку, то название должно быть в виде schema.table **(без указания БД!).** Если грузим во временную, просто table.
* 4-ый - df который загружаем в GP
* 5-ый – True/False, сбрасывать ли уже существующую табличку с таким именем

Никакой объект функция не возвращает.

Данные сперва прогружаются в промежуточную табличку, затем в финальную (указанную в третьем элементе листа). Это позволяет не лочить финальную табличку если в неё периодически доливаются данные.

### **Опциональные аргументы**

* **to\_temporary\_table** - загружает данные во временную табличку (указать to\_temporary\_table=True).

Эту временную табличку в дальнейшем можно использовать в функциях execute\_in\_GP, read\_from\_GP, transfer\_between\_db\_in\_GP и transfer\_between\_db\_in\_GP\_by\_key

* **distributed\_by** – устанавливает distributed by для таблички после загрузки данных.
* **Прочие** аргументы используются для перекладки данных функциями transfer\_between\_db\_in\_GP и transfer\_between\_db\_in\_GP\_by\_key

### **Пример**

load\_to\_GP(

[

‘Загрузка курсов’,

[‘capgp’, ‘1111111’],

‘schema.table’,

courses\_df,

True

]

)

## **cursor\_transfer**

**cursor\_transfer** – считывает данные из БД в GP в другую БД в GP.

При этом на промежуточном этапе данные не записываются в df, нет кастов bigint::text, json::text, что ускоряет перенос данных.

**Обязательные аргументы:**

* **start\_db** – лист для try\_connection, откуда тянем данные
* **sql** – sql запрос для считывания данных
* **end\_db** – лист для try\_connection, куда складываем данные,
* **target\_table** – название таблички. Если грузим в обычную табличку, то название должно быть в виде schema.table **(без указания БД!).** Если грузим во временную, просто table.

**Опциональные аргументы:**

* **drop –** сбросить табличку с таким же названием (True) или добавить перекладываемые данные в уже существующую табличку (False)
* **batch\_size –** сколько строк обрабатываем за один раз
* **to\_temporary\_table** - загружает данные во временную табличку (указать to\_temporary\_table=True).

**Пример:**

cursor\_transfer(start\_db=['gp\_rozn', '11111111'],

                             sql="""

                                           SELECT ...

                                           FROM table

                                           join ...

                                          where ...

                                    """,

                             end\_db=['capgp', '11111111'],

                             target\_table='schema.table',

                             batch\_size=1000000

)

## **transfer\_between\_db\_in\_GP**

**рекомендуется использовать cursor\_transfer**

**transfer\_between\_db\_in\_GP–** перекладывает данные из одной БД в GP в другую, например gp\_rozn2 -> capgp2. На вход принимает лист,

где:

* 1-ый элемент – комментарий
* 2-ой – лист, отправляемый в try\_connection для чтения данных
* 3-ий – sql для считывания перекладываемых данных
* 4-ый - лист, отправляемый в try\_connection для загрузки данных
* 5-ый – название таблички куда перекладываем данные. Если грузим в обычную табличку, то название должно быть в виде schema.table **(без указания БД!).** Если грузим во временную, просто table.

### **Опциональные аргументы**

* **to\_temporary\_table** - загружает данные во временную табличку (указать to\_temporary\_table=True).

Эту временную табличку в дальнейшем можно использовать в функциях execute\_in\_GP, read\_from\_GP, transfer\_between\_db\_in\_GP и transfer\_between\_db\_in\_GP\_by\_key

* **distributed\_by** – устанавливает distributed by для таблички после загрузки данных.

### **Пример (gp\_rozn2 -> capgp2)**

transfer\_between\_db\_in\_GP(

[

'Перекладка сделок',

['gp\_rozn', '11111111'],

"""

Select

Day,

Name,

Sales

From table\_sales t1

Join table\_names t2

on t1.id=t2.id

""",

['capgp', '22222222'],

'schema.table'

]

)

## **transfer\_between\_db\_in\_GP\_by\_key**

**рекомендуется использовать cursor\_transfer**

**transfer\_between\_db\_in\_GP\_by\_key–** перекладывает данные из одной БД в GP в другую, например gp\_rozn2 -> capgp2 по одному или нескольким ключам. На вход принимает лист,

где:

* 1-ый элемент – комментарий
* 2-ой – лист, отправляемый в try\_connection для чтения данных
* 3-ий – sql для считывания перекладываемых данных
* 4-ый - лист, отправляемый в try\_connection для загрузки данных
* 5-ый – название таблички куда перекладываем данные. Если грузим в обычную табличку, то название должно быть в виде schema.table **(без указания БД!).** Если грузим во временную, просто table.
* 6-ой – лист с названиями колонок которые используем как ключи
* 7-ой – True/False сбрасывать ли уже существующую табличку с таким именем

### **transfer\_condition**

В 3-м элементе листа (sql запрос) нужно указывать место куда функция будет подставлять условие для переноса.

Например, если перекладываем по полю id с значениями 1, 2, 3, то часть запроса должна выглядеть так:

Where 1=1

And {transfer\_condition}

And …(другие условия)…

В этом примере перекладка пройдёт в 3 шага. На первом функция вместо {transfer\_condition} подставит id=1, на втором id=2 и т.д.

Ключей для перекладки может быть несколько. Например, будем перекладывать по полям id и day с значениями 1, 2, 3 и 2023-01-01, 2023-01-02.

**Пример:**

Where 1=1

And {transfer\_condition\_1}

And {transfer\_condition\_2}

And …(другие условия)…

В данном примере функция сгенерирует декартово произведение из множеств значений id и day (1 и 2023-01-01, 2 и 2023-01-01, 3 и 2023-01-01, 1 и 2023-01-02, …)

Перекладка будет состоять из шести этапов, на первом шаге функция подставит id=1 и day=2023-01-01, на втором id=1 и day=2023-01-02 и т.д.

### **Опциональные аргументы**

* **to\_temporary\_table** - загружает данные во временную табличку (указать to\_temporary\_table=True).

Эту временную табличку в дальнейшем можно использовать в функциях execute\_in\_GP, read\_from\_GP, transfer\_between\_db\_in\_GP и transfer\_between\_db\_in\_GP\_by\_key

* **distributed\_by** – устанавливает distributed by для таблички после загрузки данных.
* **transfer\_values** – лист с листами, каждый из которых содержит все возможные значения для соответствующего ключа используемого в перекладке. Если не заполнять – эти значения будут получены через select, этот процесс может занимать немало времени.
* **vacuum\_every\_step**– True/False выполнять ли vacuum после каждого шага

### **Пример (gp\_rozn2 -> capgp2)**

transfer\_between\_db\_in\_GP\_by\_key(

                                    [

                                     'перенос данных по cегментам',

                                      ['gp\_rozn', '11111111111'],

                                      """

Select \*

From table

                                       where 1=1

                                             and {transfer\_condition\_1}

                                             and {transfer\_condition\_2}

                                      """,

                                      ['capgp', '2222222222'],

                                      'schema.table',

                                      ["date\_trunc('month', report\_dt)::date", '((right(epk\_id::text, 1)::int)/4)'],

                                      False

                                    ],

                                    transfer\_values=[list(map(lambda x: first\_day(x),

                                                     generate\_date\_list('2023-01-01',

                                                                         get\_current\_date(),

                                                                        'month'

                                                    )

                                                    )),

                                                     [0, 1, 2]

                                                    ],

                                    vacuum\_every\_step=True

)

## **debug и попытки**

Ко всем описанным выше функциям можно применять эти аргументы:

**debug** – True/False принтить ли отсылаемые sql запросы

**try\_cnt** – число попыток выполнения функции (в transfer\_between\_db\_in\_GP\_ by\_key – число попыток выполнения шага)

**cooldown** – перерыв между попытками в секундах (удобно указывать в виде 60\*30 (30 минут) или 60\*60\*3 (3 часа))

## **Прочие функции**

**do\_vacuum –** vacuum указанной таблички. На вход принимает лист для try\_connection и название таблички.

do\_vacuum([‘capgp’, ‘1111111’], ‘schema.table’)

**get\_query\_info –** получить df с информацией о запущенных запросах. На вход принимает псевдоним БД и логин юзера.

df=get\_query\_info(‘capgp’, ‘1111111’)

**cancel\_query –** отменить запрос с указанным pid. На вход принимает псевдоним БД, логин юзера и pid запроса (можно посмотреть в df из get\_query\_info)

cancel\_query(‘capgp’, ‘1111111’, ‘22346’)

# **hive\_functions**

## **get\_hive\_connection**

Сама по себе функция не используется, она нужна для работы других функций

**get\_hive\_connection** – создаёт коннекшн к хайву в глобальном неймспейсе, позже его могут использовать другие функции. На вход принимает омеговский логин.

get\_hive\_connection(‘111111111’)

## **read\_from\_hive**

**read\_from\_hive** – считывает данные из хайва в df.

**Обязательные аргументы:**

* **comment** – комментарий, описание того чего хотим достичь
* **hive\_sql\_query** – sql запрос для считывания данных

**Опциональные аргументы:**

* **my\_login** – омеговский логин от имени которого подключаемся к хайву. По умолчанию используется указанный в настройке перед первым запуском.

**Пример:**

df=read\_from\_hive(comment=’прогрузка отделений’,

hive\_sql\_query=”””

select

vsp,

gosb

from table

“””

)

## **transfer\_from\_hive\_to\_GP**

**transfer\_from\_hive\_to\_GP** – перекладка данных hive->GP.

**Обязательные аргументы:**

* **comment** – комментарий, описание того чего хотим достичь
* **hive\_sql\_query** – sql запрос для считывания данных
* **gp\_connection** – лист для try\_connection (GP\_functions)
* **gp\_table** – название таблички куда перекладываем данные. Если грузим в обычную табличку, то название должно быть в виде schema.table **(без указания БД!).** Если грузим во временную, просто table.

**Опциональные аргументы:**

* **my\_login** – омеговский логин от имени которого подключаемся к хайву. По умолчанию используется указанный в настройке перед первым запуском.
* **drop** – True/False, сбрасывать ли табличку с таким же названием.
* **data\_types** – лист с типами данных, пример: [‘bigint’, ‘text’, ‘date’]
* **distributed\_by** – устанавливает distributed by для таблички после загрузки данных.

**Пример:**

transfer\_from\_hive\_to\_GP(comment=’вытягиваем данные IVR’,

hive\_sql\_query=”””

select \*

from table t1

join table2 t2 on t1.id=t2.id

“””,

gp\_connection=[‘capgp’, ‘1111111’],

gp\_table=’schema.table’

)

## **transfer\_from\_hive\_to\_GP\_by\_key**

**transfer\_from\_hive\_to\_GP** – перекладка данных hive->GP в несколько шагов (например, по отдельным timestampcolumn).

**Обязательные аргументы:**

* **comment** – комментарий, описание того чего хотим достичь
* **hive\_sql\_query** – sql запрос для считывания данных
* **gp\_connection** – лист для try\_connection (GP\_functions)
* **gp\_table** – название таблички куда перекладываем данные. Если грузим в обычную табличку, то название должно быть в виде schema.table **(без указания БД!).** Если грузим во временную, просто table.
* **key\_field** – поле по значениям которого переносим
* **key\_values** – значения key\_filed которые переносим

**Опциональные аргументы:**

* **my\_login** – омеговский логин от имени которого подключаемся к хайву. По умолчанию используется указанный в настройке перед первым запуском.
* **drop** – True/False, сбрасывать ли табличку с таким же названием.
* **data\_types** – лист с типами данных, пример: [‘bigint’, ‘text’, ‘date’]
* **distributed\_by** – устанавливает distributed by для таблички после загрузки данных.

В SQL запросе необходимо указать transfer\_condition (пример см. в transfer\_betweeen\_db\_in\_GP\_by\_key), поддерживается только 1 transfer\_condition.

### **Пример:**

transfer\_from\_hive\_to\_GP\_by\_key(

comment=’вытягиваем данные IVR’,

hive\_sql\_query=”””

select \*

from table t1

join table2 t2 on t1.id=t2.id

where 1=1

and {transfer\_condition}

“””,

gp\_connection=[‘capgp’, ‘111111111’],

gp\_table=’schema.table’,

key\_field=’timestampcolumn’,

key\_values=[‘2023-01-01’, ‘2023-01-02’],

drop\_old=True

)

## **Прочие функции**

**do\_vacuum –** vacuum указанной таблички. На вход принимает лист для try\_connection и название таблички.

do\_vacuum([‘capgp’, ‘1111111’], ‘schema.table’)

**get\_query\_info –** получить df с информацией о запущенных запросах. На вход принимает псевдоним БД и логин юзера.

df=get\_query\_info(‘capgp’, ‘1111111’)

**cancel\_query –** отменить запрос с указанным pid. На вход принимает псевдоним БД, логин юзера и pid запроса (можно посмотреть в df из get\_query\_info)

cancel\_query(‘capgp’, ‘1111111’, ‘22346’)

# **email\_funs**

## **send\_mail**

**send\_mail** – отправка письма.

**Обязательные аргументы:**

* **receiver\_email** – лист с получателями письма, например [‘mail1@sber’, ‘mail2@sber’]
* **msg\_title** – титульник письма
* **msg\_text** – текст письма. Поддерживает HTML для жирного шрифта, увеличения размера шрифта и т.д. Можно добавлять в текст картинки, таблички и т.д., подробнее об этом «картинки и таблички в теле письма»

**Опциональные аргументы:**

* **shown\_from\_mail**– отправитель, который отобразится у получателей письма, например [FX\_PM\_Analytics@omega.sbrf.ru](mailto:FX_PM_Analytics@omega.sbrf.ru)
* **my\_email** – email через который идёт авторизация, по умолчанию используется тот, который указан в настройке перед первым запуском. my\_email и shown\_from\_email могут различаться.
* **my\_login** – логин через который идёт авторизация.
* **df\_to\_html\_list** – лист с df которые можно вставить как таблички в теле письма, подробнее в разделе «картинки и таблички в теле письма»
* **html\_table\_style –** стиль для таблички в теле письма, все стили можно найти в документации к пакету pretty\_html\_table.
* **even\_bg\_color** – цвет чётных строк в табличке (по умолчанию ‘white’)
* **width\_dict\_list** – лист с листами каждый из которых содержит ширину для колонок таблички в теле письма, например [ [‘auto’, ‘100px’], [’120px’, ‘120px’]]
* **table\_thousand\_sep** – разделитель разрядов в числах в табличказ в теле письма (по умолчанию запятая)
* **text\_align** – центрирование текста в табличке в теле письма (по умолчанию ‘left’)
* **default\_table** – использовать дефолтное оформление для табличек в теле письма (аргументы html\_table\_style, even\_bg\_color, width\_dict\_list, table\_thousand\_sep, text\_align игнорируются)
* **files\_list** – лист с названиями файлов, которые будут прикреплены как вложения, например [‘file1.xlsx’, ‘pic.png’]. Если файл не в той же папке что и скрипт/ноутбук – нужно указывать полный путь до него.
* **img\_in\_body\_list** – лист с названиями файлов типа png, jpg и т.д., которые будут использованы в теле письма, подробнее в разделе «картинки и таблички в теле письма»

### **Картинки и таблички в теле письма**

**Пример:**

msg\_text=f"""\

<b>Актуальность источников данных:</b>

{table\_1}

<b>Показатели:</b>

{table\_2}

<b>Графики по доходу (млн. RUB):</b>

{image\_1}{image\_2}

{image\_3}

"""

В этом примере вместо {table\_1} функция подставит первый df из листа df\_to\_html\_list, вместо {table\_2} второй df из того же листа.

Вместо {image\_1} подставит первый элемент из листа img\_in\_body\_list и т.д. При этом, первая и вторая картинка будут расположены рядом друг с другом, а третья под ними.

# **Другие функции**

## **date\_funs**

Мне не очень нравится питоновский datetime, поэтому я написал эти функции. Все функции предназначены для работы с днями (timestamp вроде тоже поддерживается, но я почти не проверял).

На вход любой функции можно подавать как и string вроде ‘2023-01-01’, так и аналогичный datetime объект.

В каждой функции возвращающей дату есть параметр output, по умолчанию равный ‘text’, это значит что функция вернёт string вроде '2023-01-01', можно указать ‘datetime’, тогда будет возвращён соответствующий datetime.

### **Функции**

* **get\_current\_date()** – возвращает текущую дату.
* **get\_current\_datetime()** – возвращает текущую дату и время.
* **get\_current\_time()** – возвращает текущее время.
* **day\_of\_week\_num(input\_dt)** – возвращает номер дня недели для поданного на вход дня.
* **first\_day(input\_dt)** – возвращает первый день месяца для поданого дня.
* **last\_day(input\_dt)** – возвращает последний день месяца для поданого дня.
* **add\_n\_days(input\_dt, n)** – добавляет n дней к поданому дню (n может быть отрицательным)
* **go\_n\_months\_ahead(input\_dt, n)** – получить первый день месяца который настанет через n месяцев относительно поданого дня (n может быть отрицательным)
* **generate\_date\_list(start\_date, end\_date, date\_range)** – возвращает лист с датами от стартовой даты до конечной. Если date\_range = ‘day’, то это лист с днями, если date\_range=’month’, то это лист с последними днями месяцев.

### **Примеры использования**

day\_of\_week\_num(get\_current\_date()) – получить текущий день недели

получить лист с днями от 2021-01-01 до позавчерашнего дня (полезно для перекладки):

generate\_date\_list(‘2021-01-01’,

add\_n\_days(get\_current\_date(), -2),

‘day’

)

## **Разное**

* **time\_print(string) –** print с указанием времени
* **get\_str\_from\_txt(txt\_file) –** получить содержание текстового файла в виде string
* **get\_greet() –** сгенерировать приветствие в зависимости от текущего времени (полезно дял настройки рассылок)