Как использовать класс Cardiobase

1. Установить python 3.5 (x64). Один из способов сделать это – установить anaconda
2. Находиться в сети университета или подключиться к ней по VPN:
   IP адрес для подключения: 85.143.3.50
   Имя пользователя: ccam\%ВАШ_ЛОГИН_В_УНИВЕРСИТЕТЕ% (например, ccam\ivanov.i)
   Пароль: %ВАШ_ПАРОЛЬ_В_УНИВЕРСИТЕТЕ%
3. Для проверки запустить test.py
4. Для ипользования класса Cardiobase в своем скрипте необходимо указать путь до cardiobase.py. Сделать это можно, например, так:

```
import sys
sys.path.append(RELATIVE_PATH)
from cardiobase import Cardiobase
```

RELATIVE_PATH – относительный путь от скрипта до директории с cardiobase.py

Функции класса Cardiobase

--- Подключение к базе ---
connect()
необходимо вызвать перед выполнением любой другой функции

--- Отключение от базы ---
disconnect()

--- Зафиксировать изменения в базе ---
commit()

Функции пациента

--- Создать пациента ---
create_patient(fio, birthday, gender) return id_patient
id_patient(int) – уникальный идентификатор созданного пациента.
fio(string) – Фамилия И.О.
birthday(string) – дата рождения в формате DD.MM.YYYY
gender(string, 1 символ) – пол, символ в таблице должен иметь номер от 0 до 99.

--- Взять информацию о пациенте по его id ---
get_patient(id_patient) return (fio, birthday, gender)

--- Есть ли пациент с таким id? ---
check_patient_exist(id_patient) return is_exist
is_exist(bool) – True, если существует

--- Возвращает список id всех пациентов ---
get_patient_list() return patient_list

Функции измерения

--- Создать файл измерения ---
create_file(id_patient, fname, type_id_str="default") return id_file
id_patient(int) – id пациента, к которому относится измерение

fname(string) – имя файла (должно быть уникальным среди всех файлов)
type_id_str(string) – тип измерения (по умолчанию 'default')

--- Закрыть созданный файл (без вызова этой функции файл не создастся) ---
close_file(id_file)

--- Возвращает список имен файлов заданного типа ---
get_files(type_id, begin_date, end_date) return fname_list
type_id(int) – тип файлов
begin_date(string) и end_date(string) – диапазон поиска по времени создания файлов, формат строки “DD.MM.YYYY, HH24:MI:SS“

--- Возвращает id файла по его имени ---
get_file_id(fname) return id_file

Функции данных(столбцов) измерения

--- Вернуть информацию о столбцах ---
get_columns() return columns

--- Записать данные в базу ---
bulk_data_set(data)
Например, вызов **bulk_data_set(**
{
“diagnosis”: [(1091, “Hello!”), (1093, “Hello2!”)],
“sampling_rate”: [(1092, 250.0), (1093, 500.0)]
})
по стобцу ”diagnosis“ измерению 1091 запишет “Hello!”, измерению 1093 – “Hello2!”. По столбцу “sampling_rate” измерению 1092 запишет 250.0, а измерению 1093 – 500.0

--- Получить данные из базы ---
bulk_data_get(columns, condition) return data
columns(list<string>) – список запрашиваемых столбцов
condition(string) – дополнительные условия для запроса

--- Удалить все данные по измерению ---
delete(id_file)
id_file(int) – id файла измерения, по которому удаляются все данные

--- Удалить конкретный столбец у измерения ---
delete(id_file, column)
id_file(int) – id файла измерения, по которому удаляется столбец column(string)

Функции хеш-таблицы

--- Создать хеш-таблицу ---
create_type_hash(name) return id_type_hash
name(string) – имя хеш-таблицы
id_type_hash(int) – идентификатор хеш-таблицы

--- Вставить строку в хеш-таблицу ---
insert_hash_row(id_type_hash, id, name, data) return number_rows
id_type_hash(int) – идентификатор хеш-таблицы
id(int) – идентификатор записи в хеш-таблице (уникальный для данной таблицы)

name(string) – имя записи в хеш-таблице (уникальное для данной таблицы)
data(object) – произвольный объект в python
number_rows(int) – число вставленных строк (при успехе равно 1)

--- Удалить строку в хеш-таблице ---

delete_hash_row(id_type_hash, id) return number_rows
id_type_hash(int) – идентификатор хеш-таблицы
id(int) – идентификатор записи в хеш-таблице
number_rows(int) – число удаленных строк (при успехе равно 1)

--- Получить все записи хеш-таблицы ---

get_hash(id_type_hash) return table
id_type_hash(int) – идентификатор хеш-таблицы
table(dict<list, list, list>) – словарь из трех списков: id, name, data

--- Изменить поле name строки хеш-таблицы ---

update_hash_row_name(id_type_hash, id, name) return number_rows
id_type_hash(int) – идентификатор хеш-таблицы
id(int) – идентификатор записи в хеш-таблице
name(string) – имя записи в хеш-таблице
number_rows(int) – число обновленных строк (при успехе равно 1)

--- Изменить поле data строки хеш-таблицы ---

update_hash_row_data(id_type_hash, id, data) return number_rows
id_type_hash(int) – идентификатор хеш-таблицы
id(int) – идентификатор записи в хеш-таблице
data(object) – произвольный объект в python
number_rows(int) – число обновленных строк (при успехе равно 1)