NeuroNetLibrary - python-библиотека, содержит функции, используемые до, во время и/или после звонка.

Доступна внутри звонка в виде объекта **nn** (nn = NeuroNetLibrary())

Метод	Описание	Пример
nn.call(msisdn: str, entry_point: str)	Функция создания звонка. msisdn - номер абонента, кому звонить entry_point - имя функции (точки входа), с которой будет запущен скрипт	nn.call(msisdn='1234567890', entry_point='main')
nn.has_records(lst:list)→ list	Проверяет существование промптов в базе. Возвращает список **отсутствующих** промптов.	not_found = nn.has_records(['hello_main', 'recommend_main'])

nn.env(*args, **kwargs)	Установка / получение переменных диалога (внутренний аналог словаря)	// получить все переменные из env в виде словаря
		all_env = nn.env()
		
		// установить значения tag='positive', prompt='hello_main'
		nn.env(tag='positive', prompt='hello_main')
		
		// получить значение ключа tag
		tag = nn.env('tag') // 'positive'

nn.counter(name, op=None) → int	Получение/изменение внутреннего счетчика в логике.
	В качестве параметра op можно передать:
	'+' - увеличить значение счётчика на 1
	'-' - уменьшить значение счётчика на 1
	5 — установить значение счётчика равным 5 (любое целое число).
	Возвращает целочисленное значение.
		counter = nn.counter('some_counter') // 0
		
		counter = nn.counter('some_counter', '+') // 1
		
		counter = nn.counter('some_counter', 5) // 5
		
		counter = nn.counter('some_counter', '-') // 4

nn.log(name, data=None)	Функция для логгирования данных. Можно передать 1 или 2 аргумента.	nn.log('Проигрывается промпт hello_main') nn.log('tag', 'абонент взял трубку')
nn.dump()	Выгрузка переменных, которые были сохранены в nn.env, в статистику по обзвону	nn.env(tag='абонент взял трубку', call_result='абонент поставил плохую оценку') nn.dump()

NeuroVoiceLibrary - python-библиотека с методами, работающими во время звонка

Доступна внутри звонка в виде объекта **nv** = NeuroVoiceLibrary()

Метод	Описание	Пример
nv.say(name)	Проигрывание промпта по **названию** name.	nv.say('hello_main')
nv.listen(entities: list = None, entities_exclude: list = None)	Контекстный менеджер. Запускает распознавание ответа абонента. Параметры: **entities** — список сущностей для распознавания. Использование: entities=['entity1', 'entity2'] **entities_exclude** - исключённые сущности из распознавания. Имеет бОльший приоритет, чем *entities*. Использование: entities_exclude=['entity3', 'entity4'] Возвращает экземпляр класса NeuroNluRecognitionResult (см. ниже)	with nv.listen(entities=['some_entity_1', 'some_entity_2']) as result: nv.say('hello_main')
nv.bridge(msisdn)	Соединить абонента с другим номером	nv.bridge('1234567890')
nv.hangup()	Завершить звонок	nv.hangup()

Методы класса NeuroNluRecognitionResult

Представим, что во время проигрывания промпта 'hello_main', абонент ответил: «Здравствуйте, я сейчас занят, повторите, пожалуйста».

Методы	Описание	Пример
result.utterance() → str	Получить распознанный текст	utterance = result.utterance() // 'Здравствуйте, я сейчас занят, повторите, пожалуйста'
result.entities() → dict	Возвращает словарь распознанных сущностей, **ключи и значения которых имеют тип str**	all_entities = result.entities() // {'wrong_time': 'true', 'repeat': 'true'}
result.entity(entity: str) → str	Возвращает значение сущности entity. Если сущность entity не распознана, то вернет None.	value = result.entity('repeat') // 'true' value = result.entity('question') // None

result.has_entity(entity: str) → bool	Проверяет наличие ключа entity в словаре распознанных сущностей.
	Возвращаемое значение: True/False.
		check_wrong_time = result.has_entity('wrong_time') // True
		check_ question = result.has_entity('question') // False

result.has_entities() → bool	Проверяет, содержится ли в результате распознавания хотя бы одна сущность. Возвращаемое значение: True/False.	has_entities = result.has_entities() // True