**Функции и методы строк**

| Функция или метод | Назначение |
| --- | --- |
| S = 'str'; S = "str"; S = '''str'''; S = """str""" | Литералы строк |
| S = "s\np\ta\nbbb" | Экранированные последовательности |
| S = r"C:\temp\new" | Неформатированные строки (подавляют экранирование) |
| S = b"byte" | Строка байтов |
| S1 + S2 | Конкатенация (сложение строк) |
| S1 \* 3 | Повторение строки |
| S[i] | Обращение по индексу |
| S[i:j:step] | Извлечение среза |
| len(S) | Длина строки |
| S.find(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер первого вхождения или -1 |
| S.rfind(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер последнего вхождения или -1 |
| S.index(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError |
| S.rindex(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError |
| S.replace(шаблон, замена[, maxcount]) | Замена шаблона на замену. maxcount ограничивает количество замен |
| S.split(символ) | Разбиение строки по разделителю |
| S.isdigit() | Состоит ли строка из цифр |
| S.isalpha() | Состоит ли строка из букв |
| S.isalnum() | Состоит ли строка из цифр или букв |
| S.islower() | Состоит ли строка из символов в нижнем регистре |
| S.isupper() | Состоит ли строка из символов в верхнем регистре |
| S.isspace() | Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая строка" ('\n'), "перевод каретки" ('\r'), "горизонтальная табуляция" ('\t') и "вертикальная табуляция" ('\v')) |
| S.istitle() | Начинаются ли слова в строке с заглавной буквы |
| S.upper() | Преобразование строки к верхнему регистру |
| S.lower() | Преобразование строки к нижнему регистру |
| S.startswith(str) | Начинается ли строка S с шаблона str |
| S.endswith(str) | Заканчивается ли строка S шаблоном str |
| S.join(список) | Сборка строки из списка с разделителем S |
| ord(символ) | Символ в его код ASCII |
| chr(число) | Код ASCII в символ |
| S.capitalize() | Переводит первый символ строки в верхний регистр, а все остальные в нижний |
| S.center(width, [fill]) | Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию) |
| S.count(str, [start],[end]) | Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию) |
| S.expandtabs([tabsize]) | Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам |
| S.lstrip([chars]) | Удаление пробельных символов в начале строки |
| S.rstrip([chars]) | Удаление пробельных символов в конце строки |
| S.strip([chars]) | Удаление пробельных символов в начале и в конце строки |
| S.partition(шаблон) | Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки |
| S.rpartition(sep) | Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку |
| S.swapcase() | Переводит символы нижнего регистра в верхний, а верхнего – в нижний |
| S.title() | Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний |
| S.zfill(width) | Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями |
| S.ljust(width, fillchar=" ") | Делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar |
| S.rjust(width, fillchar=" ") | Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar |
| S.format(\*args, \*\*kwargs) | Форматирование строки |

**Символы форматирования**

| **Формат** | **Тип данных** |
| --- | --- |
| %d, %i | Целое число |
| %5d, %12d | Выделяет пространство 5 (или любое другое число) символов под это число. Выравнивание вправо, остальное пространство остаётся пустым |
| %05d | Также выделяется пространство в 5 символов, но свободное пространство слева заполняется нулями |
| %o | Число в восьмеричной системе счисления |
| %x | Число в шестнадцатеричной системе счисления |
| %f | Число с плавающей точкой |
| %10.2f | Число с плавающей точкой, для которого зарезервировано пространство из 10 символов и стоит ограничение на количество знаков после запятой — 2 |
| %e | Также число с плавающей точкой, но в экспоненциальной записи |
| %c | Код символа |
| %s | Другая строка |
| %% | Знак процента, если его необходимо использовать просто как символ в строке |

**Срезы в списках**

| **Срез** | **Как работает?** | **Пример** |
| --- | --- | --- |
| [:] | Возвращает элементы полностью | [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’] |
| [2:] | Возвращает элементы списка, начиная с элемента индекса 2 и до конца списка | [‘c’, ‘d’, ‘e’, ‘f’, ‘g’] |
| [:3] | Возвращает элементы списка от его начала до элемента с индексом 3, **не включая** его | [‘a’, ‘b’, ‘c’] |
| [1:4] | Объединяя предыдущие два способа, можно получить элементы из середины. В данном случае начиная с индекса 1 до индекса 4, не включительно. Иными словами, элементы с индексами 1,2 и 3 | [‘b’, ‘c’, ‘d’] |
| [::2] | Задает шаг, через который извлекаются элементы | [‘a’, ‘c’, ‘e’, ‘g’] |
| [::-1] | Используя отрицательный шаг, можно развернуть массив | [‘g’, ‘f’, ‘e’, ‘d’, ‘c’, ‘b’, ‘a’] |

**Приоритет операций**

| **Операция** | Описание |
| --- | --- |
| ( ) | Скобки – **высший приоритет** – первые |
| \*\* | Экспонента (возведение в степень) |
| +x, -x, ~x | Унарные плюс, минус и битовое отрицание |
| \*, /, //, % | Умножение, деления, взятие остатка |
| +, - | Сложение и вычитание |
| <<, >> | Битовые сдвиги |
| & | Битовое И |
| ^ | Битовое исключающее ИЛИ (XOR) |
| | | Битовое ИЛИ |
| ==, !=, >, >=, <, <=,  is, is not,  in, not in | Сравнение, проверка идентичности, проверка вхождения |
| not | Логическое НЕ |
| and | Логическое И |
| or | Логическое ИЛИ – **низший приоритет** – последние |

**Операции над множествами**

| **Операция** | **Название** | **Смысл** |
| --- | --- | --- |
| set.union(other) | Объединение | Возвращает множество, состоящее из элементов set и other. |
| set.intersection(other) | Пересечение | Возвращает множество, состоящее из элементов, которые встречаются и в set, и в other. |
| set.difference(other) | Разность | Возвращает множество элементов set, которые не встречаются в other. |
| set.symmetric\_difference(other) | Симметричная разность | Возвращает множество элементов, встречающиеся в одном из множеств, но не в обоих одновременно. |

**Логические операторы**

 and: если все операнды являются истинными (ненулевые или непустые), то возвращается последнее истинное значение.

**print**( **1** **and** "hello" **and** [False])

# [False]

Несмотря на то, что последний операнд похож на False, он является непустым списком, а значит, он истинный.

 and: если один из операндов является ложным, то возвращается первый такой операнд.

**print**(**42** **and** **0** **and** '' **and** False)

# 0

Первый операнд является ненулевым числом — значит, истинный, а все остальные — ноль, пустая строка, булево значение *False* — ложные. И согласно правилу возвращается первый ложный операнд (слева направо).

 or: если один из операндов является истинным, то возвращается первый такой операнд, а остальные игнорируются.

**print**([] **or** **3.14** **or** False)

# 3.14

Первый операнд (пустой список) является ложным, следующий  (ненулевое число) — истинным, а значит, возвращается именно он, а все остальные игнорируются (не вычисляются).

 or: если все операнды являются ложными, то возвращается последний.

**print**(**0** **or** '' **or** False)

# False

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**SQL**

***Создание объекта***

**CREATE** "тип\_объекта" "имя\_объекта" "условия";

**CREATE** **TABLE** products (

id BIGSERIAL PRIMARY KEY, -- первичный ключ

name TEXT **NOT** **NULL**, -- название товара

price INT **CHECK**(price > **0**), -- цена (гарантируется, что цена > 0)

vendor\_id BIGINT **REFERENCES** vendors(id), -- ссылка на продавца

manufacturer\_id BIGINT **REFERENCES** manufacturers(id) -- ссылка на производителя

);

Ограничения, доступные при описании столбцов таблиц. В *SQL* таких ограничений всего пять:

| **№** | **Ограничение** | **Описание** |
| --- | --- | --- |
| 1 | NOT NULL | Гарантирует, что в БД не попадут пустые значения. |
| 2 | UNIQUE | Гарантирует, что для данного столбца или группы столбцов все значения в таблице будут различаться. |
| 3 | PRIMARY KEY | Делает атрибут или группу атрибутов первичным ключом (то есть NOT NULL UNIQUE). |
| 4 | FOREIGN KEY | Обозначает внешний ключ: атрибут или группа атрибутов ссылаются на ключи в другой таблице и должны совпадать со значениями в ней. |
| 5 | CHECK | Произвольная проверка содержимого ячеек одного или нескольких столбцов. |

***Изменение объекта***

ИЗМЕНИТЬ таблицу ПЕРЕИМЕНОВАТЬ; ALTER TABLE posts RENAME TO posts1;

ИЗМЕНИТЬ таблицу ДОБАВИТЬ столбец; **ALTER** **TABLE** posts **ADD** **COLUMN** IF **NOT** **EXISTS** publication\_date BIGINT **NOT** **NULL**

ИЗМЕНИТЬ таблицу УДАЛИТЬ столбец; **ALTER TABLE posts DROP** **COLUMN** IF **EXISTS** content

ИЗМЕНИТЬ таблицу ИЗМЕНИТЬ ограничение;  **ALTER** **TABLE** posts **ADD** **CONSTRAINT** pub\_date\_unique **UNIQUE**(publication\_date)

*(Для ограничения в PostgreSQL нельзя использовать IF NOT EXISTS. Можно попытаться его удалить, используя IF EXISTS, после чего создать. Нелогично, но таков уж SQL!)*

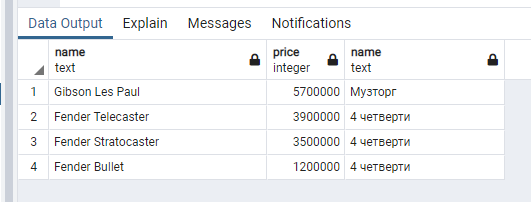
***Извлечение данных***

**SELECT** имя\_таблицы**1**.имя\_поля\_1, ..., имя\_таблицы**n**.имя\_поля\_n **SELECT** products.name, products.price, vendors.name

**FROM** имя\_таблицы**1**, ..., имя\_таблицы\_n  **FROM** products, vendors

**WHERE** условие\_объединения\_таблиц **AND** условие\_фильтрации  **WHERE** products.vendor\_id = vendors.id

**ORDER** **BY** имя\_поля\_сортировки;  **ORDER** **BY** price **DESC**



***Создание записи в таблице***

**INSERT** **INTO**

имя\_таблицы(имя\_столбца\_1, ..., имя\_столбца\_n)

**VALUES**

(значение\_столбца\_1, ..., значение\_столбца\_n),

...

(значение\_столбца\_1, ..., значение\_столбца\_n);

***Изменение записи в таблице***

**UPDATE** имя\_таблицы UPDATE manufacturers

**SET** атрибут\_1 = значение\_1, ... , атрибут\_n = значение\_n SET website = 'вебсайт завода №1'

**WHERE** условие\_поиска\_записи\_для\_обновления; WHERE id = 4;

***Удаление записи в таблице***

**DELETE** **FROM** имя\_таблицы **WHERE** условие\_отбора\_записей\_для\_удаления

***Объединение таблиц (общая формула)***

SELECT таблица1.атрибут1, таблица1.атрибут2,...   
FROM таблица1  
JOIN таблица2  
ON таблица1.фтрибут = таблица2.атрибут  
WHERE условие отбора

***Выполнение функций внутри оператора***

**SELECT** **min**(price), **max**(price), **avg**(price), **count**(\*) **FROM** PRODUCTS;

**Django**

**django-admin startproject project –** создание нового проекта джанго