Занятие 25

PyQt6

Какая из этих строчек неправильная?

1.xyz = 1,000,000

2.x y z = 1000 2000 3000

3.x,y,z = 1000, 2000, 3000

4.x_y_z = 1,000,000

Задача 24-1

Напишите функцию, которая сортирует числовой список, не используя никаких функций, вроде sort, sorted и т.д.

Задача 24-2

Создайте запрос, который находит авторов, у которых только минимальное количество книг на складе (в таблице book). Используйте для этого View.

Задача 24-3

Создайте функцию, которая принимает на входе строку из круглых скобок, открывающих и закрывающих, и возвращает True или False, если строка является «правильной», т.е. никогда количество закрывающих не больше, чем количество открывающих, если двигаться по строке слева направо.

Примеры:

"()" => true

")(()))" => false

"(" => false

"(()) ((() ()) ())" => true

Решение задачи 24-3

```
def valid_parentheses(x):
    while "()" in x:
        x = x.replace("()", "")
    return x == ""
```

View

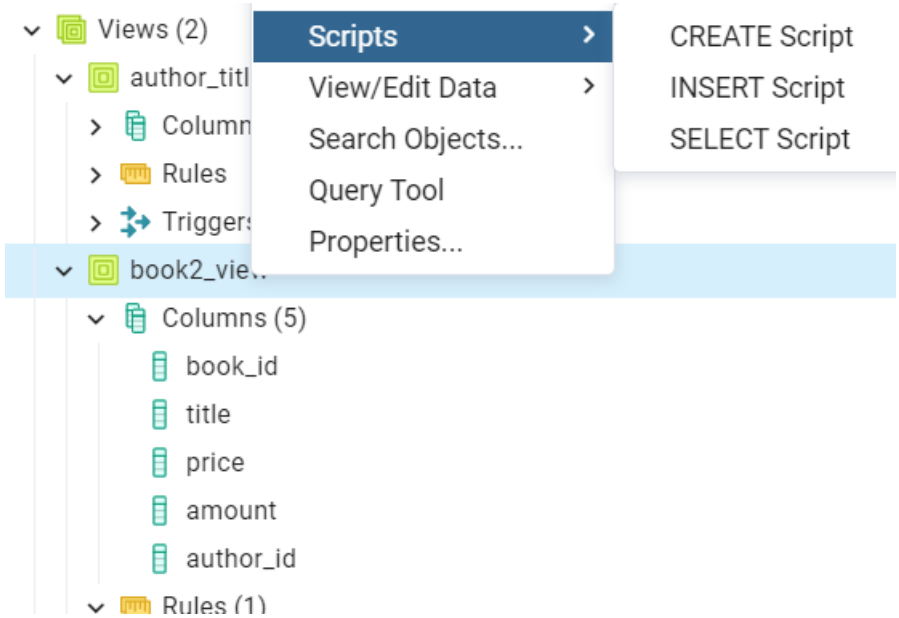
CREATE VIEW view_name AS

SELECT select_statement

SELECT * FROM view_name – можем использовать в запросах

CREATE OR REPLACE VIEW – есть ограничения

Как посмотреть View – через Create script



Запрос в несколько этапов

Вывести авторов, общее количество книг которых на складе максимально.

Шаг 1. Найдем суммарное количество книг на складе по каждому автору. Поскольку фамилии автора в этой таблице нет, то группировку будем осуществлять по author_id.

SELECT author_id, SUM(amount) AS sum_amount FROM book GROUP BY author_id

Давайте перепишем сложный запрос с помощью создания view

Шаг 2

Шаг 2. В результирующей таблице предыдущего запроса необходимо найти максимальное значение. Для этого запросу, созданному на шаге 1, необходимо присвоить имя (например, query_in) и использовать его в качестве таблицы-источника после FROM. Затем уже находить максимум по столбцу sum_amount.

```
SELECT MAX(sum_amount) AS max_sum_amount
FROM
    (
     SELECT author_id, SUM(amount) AS sum_amount
     FROM book
     GROUP BY author_id
    ) query_in
```

Шаг 3

Шаг 3. Выведем фамилию автора и общее количество книг для него.

```
SELECT name_author, SUM(amount) as Количество
FROM
    author INNER JOIN book
    on author.author_id = book.author_id
GROUP BY name_author
```

Шаг 4

Шаг 4. Включим запрос с шага 2 в условие отбора запроса с шага 3. И получим всех авторов, общее количество книг которых максимально.

- SELECT name_author, SUM(amount) as Количество
- FROM author INNER JOIN book on author.author_id = book.author_id GROUP BY name_author
- HAVING SUM(amount) =
- (/* вычисляем максимальное из общего количества книг каждого автора */
  SELECT MAX(sum_amount) AS max_sum_amount
  FROM (/* считаем количество книг каждого автора */
  SELECT author_id, SUM(amount) AS sum_amount
  FROM book GROUP BY author_id)
  query_in);

Типы данных

INTEGER — целое число (smallint, bigint, int2, int4, int8)
NUMERIC — число с плавающей точкой (decimal, real, double precision, float)
VARCHAR() — текстовые данные (char)
TEXT — набор с максимальной длиной 65535
DATE — дата (SELECT current_time)
TIMESTAMP — дата и время (SELECT current_timestamp)
BOOLEAN — логический тип
JSON — текстовый json

Массивы:

```
CREATE TABLE arr (id int, x int[]);
INSERT INTO arr VALUES
(1, '{1,2,3}'::int[]);
SELECT * FROM arr;
SELECT id, x[1] as the_first FROM arr
```

INNER JOIN

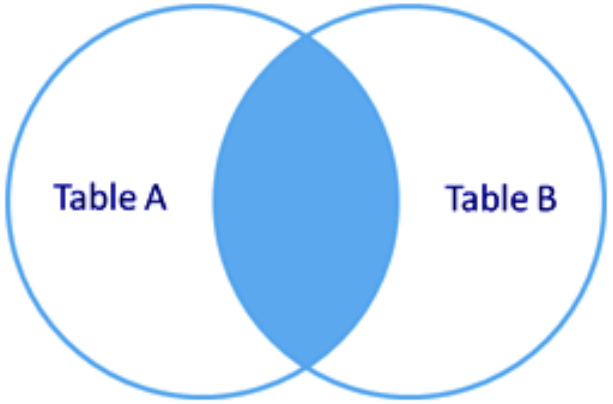


рис. Inner join

Шаблон запроса:

```
SELECT  a.name , b.value
  FROM  table1 a, table2 b
 WHERE  a.id = b.id
```

Объединение, пересечение, «разность»

Союзы

```
SELECT n from numders1;
 UNION
SELECT n from mumbers2;
```

UNION — Объединяет в одну таблицу результаты 2-х и более запросов.

UNION ALL — Для получения списка со всеми дубликатами.

INTERSECT — Возвращает пересечение результатов нескольких запросов.

EXCEPT — Возвращает исключение результатов второго запроса из первого.

Index

Индекс – это дополнительная структура данных для ускорения работы запросов.

Результат запроса с индексом и без должны быть одинаковыми.

Индексы важны для поиска, для сортировки, группировки, соединения таблиц.

Индексы не всегда полезны!!!

Затраты на содержание, могут замедлять работу при больших объемах при вводе и коррекции данных.

CREATE INDEX

CREATE INDEX имя-индекса

ON имя-таблицы (имя-столбца, …)

CREATE INDEX book2_author_id

ON book2 (author_id)


```
1  -- Index: book2_author_id
2  Loading...
3  -- DROP INDEX IF EXISTS public.book2_author_id;
4
5  CREATE INDEX IF NOT EXISTS book2_author_id
6      ON public.book2 USING btree
7      (author_id ASC NULLS LAST)
8      TABLESPACE pg_default;
9
```

PyQt6

PyQt — это библиотека Python для создания приложений с графическим интерфейсом с помощью инструментария Qt.

Созданная в Riverbank Computing, PyQt является свободным ПО (по лицензии GPL) и разрабатывается с 1999 года.

Последняя версия PyQt6 — на основе Qt 6 — выпущена в 2021 году, и библиотека продолжает обновляться.

Инсталлировать PyQt6

PyCharm:

File \ Settings \ Project \ Python Interpretator \ + \ PyQt6 \ Install Package

pip install pyqt6

Самое простое приложение

```
from PyQt6.QtWidgets import QApplication, QWidget
#import sys # Только для доступа к аргументам командной строки
# Приложению нужен один (и только один) экземпляр QApplication.
# Передаём sys.argv, чтобы разрешить аргументы командной строки для приложения.
# Если не будете использовать аргументы командной строки, QApplication([]) тоже работает
#app = QApplication(sys.argv)
app = QApplication([])
# Создаём виджет Qt — окно.
window = QWidget()
window.show() # Важно: окно по умолчанию скрыто.
# Запускаем цикл событий.
app.exec()
# Приложение не доберётся сюда, пока вы не выйдете и цикл  событий не остановится.
```

Кнопка Push Me

```
import sys
from PyQt6.QtWidgets import QApplication, QPushButton
app = QApplication(sys.argv)
window = QPushButton("Push Me")
window.show()
app.exec()
```

Окно

```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow
app = QApplication(sys.argv)
window = QMainWindow()
window.show()
# Запускаем цикл событий.
app.exec()
```

Цикл событий

Основной элемент всех приложений в Qt — класс QApplication.

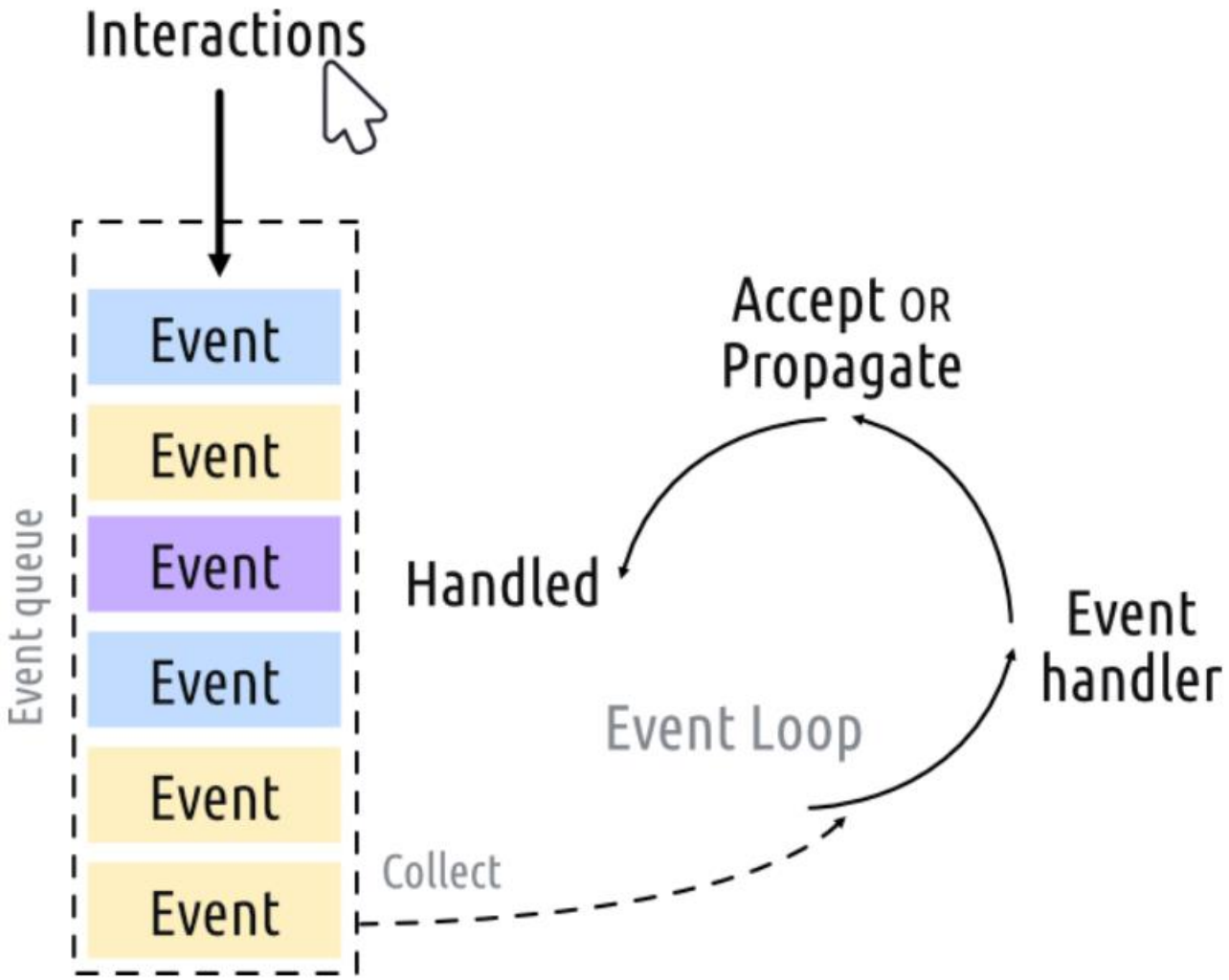
Для работы каждому приложению нужен один — и только один — объект QApplication, который содержит цикл событий приложения.

Это основной цикл, управляющий всем взаимодействием пользователя с графическим интерфейсом.

При каждом взаимодействии с приложением — будь то нажатие клавиши, щелчок или движение мыши — генерируется событие, которое помещается в очередь событий.

В цикле событий очередь проверяется на каждой итерации: если найдено ожидающее событие, оно вместе с управлением передаётся определённому обработчику этого события.

Последний обрабатывает его, затем возвращает управление в цикл событий и ждёт новых событий.



app.py

```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton
class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.setWindowTitle("My App")
app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Окно фиксированного размера с кнопкой

```
import sys
from PyQt6.QtCore import QSize, Qt
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton
# Подкласс QMainWindow для настройки главного окна приложения
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        button = QPushButton("Press Me!")
        self.setFixedSize(QSize(400, 300))
        # Устанавливаем центральный виджет Window.
        self.setCentralWidget(button)
app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Нажатие кнопки

```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        button = QPushButton("Press Me!")
        button.setCheckable(True)
        button.clicked.connect(self.the_button_was_clicked)
        # Устанавливаем центральный виджет Window.
        self.setCentralWidget(button)
    def the_button_was_clicked(self):
        print("Clicked!")
app = QApplication(sys.argv)
window = MainWindow()
window.show() app.exec()
```

Clicked & Toggled

```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        button = QPushButton("Press Me!")
        button.setCheckable(True)
        button.clicked.connect(self.the_button_was_clicked)
        button.clicked.connect(self.the_button_was_toggled)
        self.setCentralWidget(button)
    def the_button_was_clicked(self):
        print("Clicked!")
    def the_button_was_toggled(self, checked):
        print("Checked?", checked)
app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Переменная хранит состояние

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.button_is_checked = True
        self.setWindowTitle("My App")
        button = QPushButton("Press Me!")
        button.setCheckable(True)
        button.clicked.connect(self.the_button_was_toggled)
        button.setChecked(self.button_is_checked)
        self.setCentralWidget(button)
    def the_button_was_toggled(self, checked):
        self.button_is_checked = checked
        print(self.button_is_checked)
```

Одноразовая кнопка

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        self.button = QPushButton("Press Me!")
        self.button.clicked.connect(self.the_button_was_clicked)
        self.setCentralWidget(self.button)
    def the_button_was_clicked(self):
        self.button.setText("You already clicked me.")
        self.button.setEnabled(False)
       # Также меняем заголовок окна.
        self.setWindowTitle("My Oneshot App")
       # self.setWindowTitle("A new window title")
```

Случайный выбор Window Title по нажатию кнопки

```
from PyQt6.QtWidgets import QApplication, QMainWindow, QPushButton
import sys
from random import choice
window_titles = [    'My App’,     'My App',
    'Still My App’,     'Still My App',    'What on earth',
    'What on earth',    'This is surprising',
    'This is surprising',    'Something went wrong']
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.n_times_clicked = 0
        self.setWindowTitle("My App")
        self.button = QPushButton("Press Me!")
        self.button.clicked.connect(self.the_button_was_clicked)
        self.windowTitleChanged.connect(self.the_window_title_changed)
        self.setCentralWidget(self.button)
    def the_button_was_clicked(self):
        print("Clicked.")
        new_window_title = choice(window_titles)
        print("Setting title:  %s" % new_window_title)
        self.setWindowTitle(new_window_title)
    def the_window_title_changed(self, window_title):
        print("Window title changed: %s" % window_title)
        if window_title == 'Something went wrong':
            self.button.setDisabled(True)
app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

QMouseEvent

QMouseEvent — одно из основных событий, получаемых виджетами. События QMouseEvent создаются для каждого отдельного нажатия кнопки мыши и её перемещения в виджете. Вот обработчики событий мыши:

Обработчик	Событие
`mouseMoveEvent`	Мышь переместилась
`mousePressEvent`	Кнопка мыши нажата
`mouseReleaseEvent`	Кнопка мыши отпущена
`mouseDoubleClickEvent`	Обнаружен двойной клик

Например, нажатие на виджет приведёт к отправке QMouseEvent в обработчик событий .mousePressEvent в этом виджете.

Mouse

```
import sys
from PyQt6.QtCore import Qt from PyQt6.QtWidgets
import QApplication, QLabel, QMainWindow, QTextEdit
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.label = QLabel("Click in this window")
        self.setCentralWidget(self.label)
    def mouseMoveEvent(self, e):
        self.label.setText("mouseMoveEvent")
    def mousePressEvent(self, e):
        self.label.setText("mousePressEvent")
    def mouseReleaseEvent(self, e):
        self.label.setText("mouseReleaseEvent")
    def mouseDoubleClickEvent(self, e):
        self.label.setText("mouseDoubleClickEvent")
app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Widgets

```
import sys
from PyQt6.QtWidgets import (
   QMainWindow, QApplication,
   QLabel, QCheckBox, QComboBox,  QLineEdit,
   QLineEdit, QSpinBox, QDoubleSpinBox, QSlider
)
from PyQt6.QtCore import Qt

class MainWindow(QMainWindow):

   def __init__(self):
      super(MainWindow, self).__init__()

      self.setWindowTitle("My App")



app = QApplication(sys.argv)
w = MainWindow()
w.show()
app.exec()
```

QLabel

```
widget = QLabel("Hello")
widget = QLabel("1") # Создана метка с текстом 1.
widget.setText("2") # Создана метка с текстом 2.
class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.setWindowTitle("My App")
        widget = QLabel("Hello")
        font = widget.font()
        font.setPointSize(30)
        widget.setFont(font)
        widget.setAlignment(Qt.AlignmentFlag.AlignHCenter | Qt.AlignmentFlag.AlignVCenter)
        self.setCentralWidget(widget)
```

QLineEdit

```
class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.setWindowTitle("My App")

        widget = QLineEdit()
        widget.setMaxLength(10)
        widget.setPlaceholderText("Enter your text")

        #widget.setReadOnly(True) # раскомментируйте, чтобы сделать доступным только для чтения

        widget.returnPressed.connect(self.return_pressed)
        widget.selectionChanged.connect(self.selection_changed)
        widget.textChanged.connect(self.text_changed)
        widget.textEdited.connect(self.text_edited)

        self.setCentralWidget(widget)

    def return_pressed(self):
        print("Return pressed!")

    def selection_changed(self):
        print("Selection changed")
    def text_changed(self, s):
        print("Text changed...")
        print(s)

    def text_edited(self, s):
        print("Text edited...")
        print(s)
```

QPushButton

```
button = QPushButton("Результат!")
button.setCheckable(True)
button.clicked.connect(self.the_button_was_clicked)

def the_button_was_clicked(self):
    print("Clicked!")
    self.label_result.setText(self.text)
    if self.tf:
        self.setWindowTitle('Result')
        self.tf = False
    else:
        self.tf = True
        self.setWindowTitle('MyApp')
```

Задача 25-1

Создайте приложение с горизонтальным расположением двух виджетов

Задача 25-2

Дана последовательность целых чисел.

Необходимо определить те из них, который являются «непоследовательными», т.е. не являются на 1 больше, чем предыдущее число.

Первое число всегда последовательное.

Т.е. если последовательность [1,5,6,7,9,10], то результатом должна быть список [5, 9]

Задача 25-3

Напишите функцию, которой на вход подается строка, содержащая последовательность слов (которые могут включать буквы верхнего и нижнего регистра). На выходе должна получиться строка в CamelStyle.

Например, "camel case word" => CamelCaseWord