ХАБАРОВСКИЙ ИНСТИТУТ ИНФОКОММУНИКАЦИЙ (филиал)  
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО

ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ

ВЫСШЕГО ОБРАЗОВАНИЯ  
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Лабораторные работы

По теме: «Разработка простых графических приложений»

|  |
| --- |
| Выполнил:  студент группы ПКС-320 Медведев Андрей  Проверил преподаватель:  Поляков А.Н. |

Хабаровск

2022

**Я люблю Python 🔥  🔥   🔥**

[1. БД с уникальным полем через запрос в питоне. 3](#_Toc119704928)

[2. Окно с ФИО. 4](#_Toc119704929)

[3. Вывод картинки через PyQt 5](#_Toc119704930)

[4. Листание картинок вперед/назад 7](#_Toc119704931)

[5. Окно с ФИО в .pyw 9](#_Toc119704932)

[6. Итерирование в потоке 10](#_Toc119704933)

[7. Активность потоков, принудительное завершение и изменение потока 12](#_Toc119704934)

[8. Числа Фибоначчи с распределением потоков 14](#_Toc119704935)

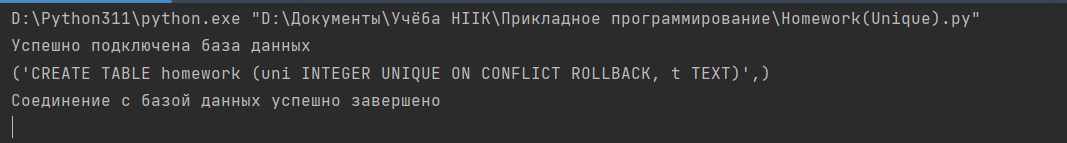
[9. Multiprocessing, threading, QSemaphore, QWaitCondition. 20](#_Toc119704936)

[10. Есть заставка, для продвижения прогресса загрузки - нужно решать задачи, при каждой решенной задаче меняется заставка 28](#_Toc119704937)

1. БД с уникальным полем через запрос в питоне.

Листинг:

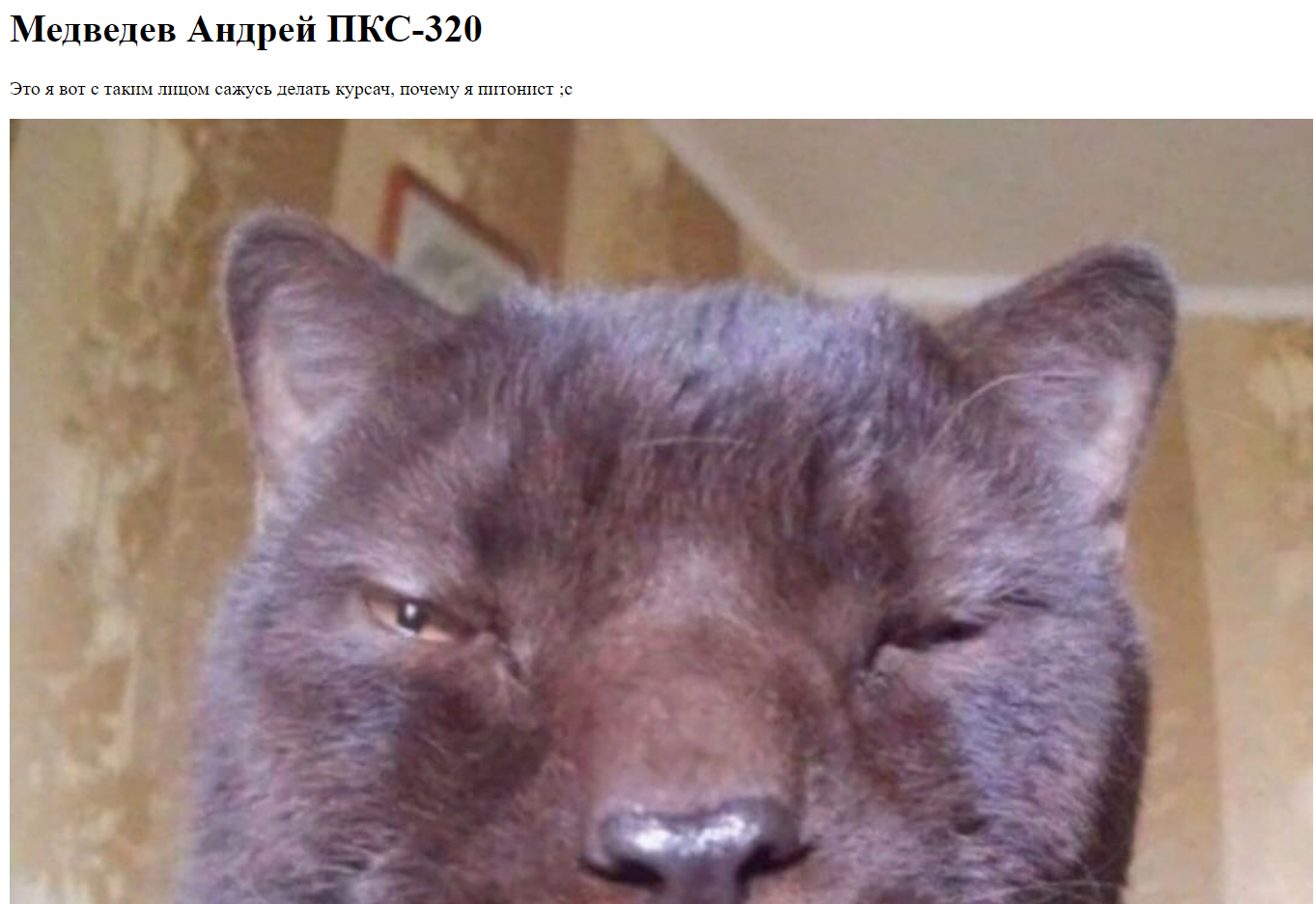
import sqlite3  
  
def sqlite\_output(cur, sql\_command):  
 for row in cur.execute(sql\_command).fetchall():  
 print(row)  
  
  
try:  
 con = sqlite3.connect(":memory:")  
except sqlite3.DatabaseError as err:  
 print(f"Ошибка подключения к БД : {err}")  
else:  
 print("Успешно подключена база данных")  
  
cur = con.cursor()  
sql\_create = """   
 CREATE TABLE IF NOT EXISTS homework (uni INTEGER UNIQUE ON CONFLICT ROLLBACK, t TEXT);  
 """  
cur.executescript(sql\_create)  
  
sql\_uniq\_check = """  
 SELECT sql FROM sqlite\_master WHERE TYPE = "table" and NAME = "homework";  
 """  
sqlite\_output(cur, sql\_uniq\_check)  
  
cur.close()  
con.close()  
print("Соединение с базой данных успешно завершено")  
input()



1. Окно с ФИО.

Листинг:

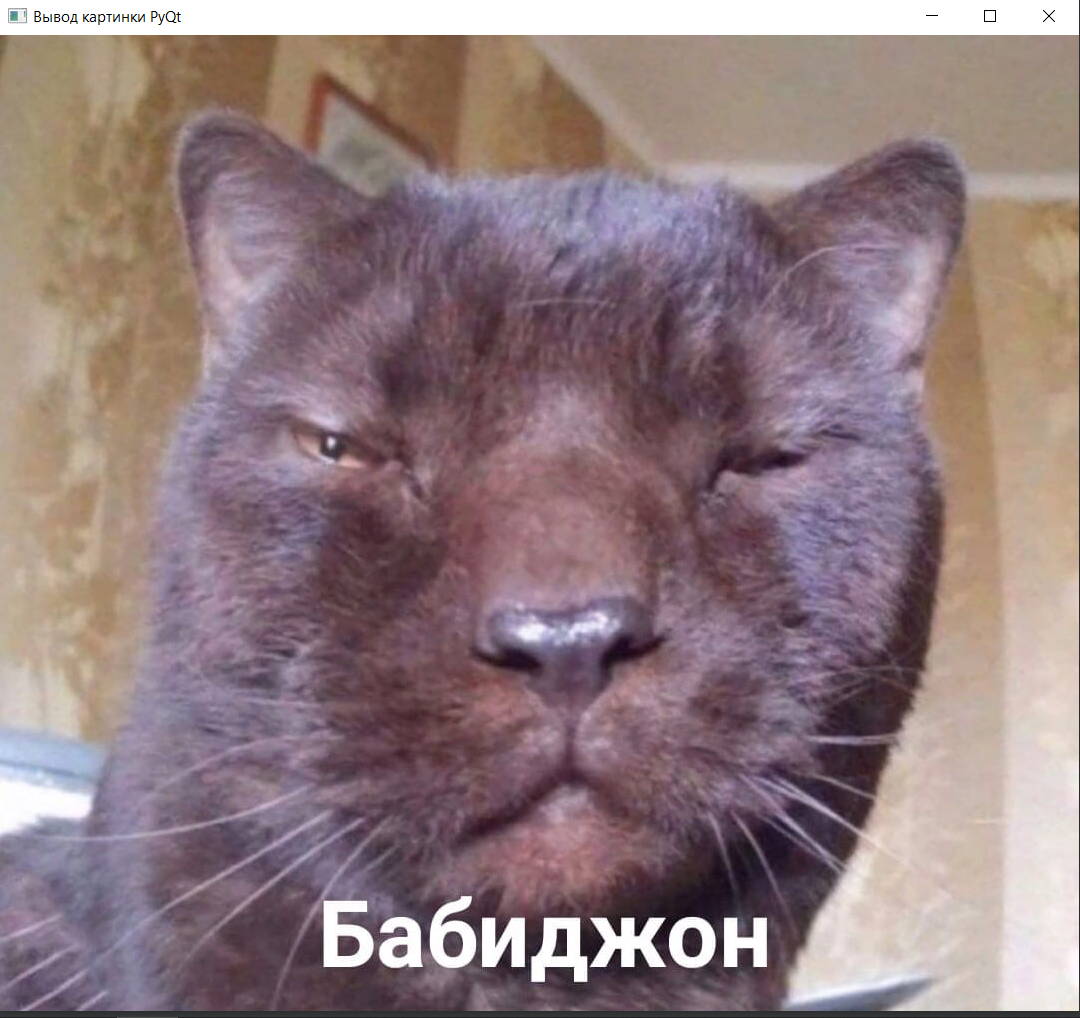
<html>  
 <head>  
 <title>Медведев</title>  
 <meta charset="utf-8">  
 </head>  
 <body>  
 <h1>Медведев Андрей ПКС-320</h1>  
 <p>Это я вот с таким лицом сажусь делать курсач, почему я питонист ;c</p>  
 <p><img src="C:\Users\Семья\Desktop\Андрея\SQLite\HTML+QT\lapot.jpg" alt="Произошла абоба"></p>  
 </body>  
</html>



1. Вывод картинки через PyQt

Листинг

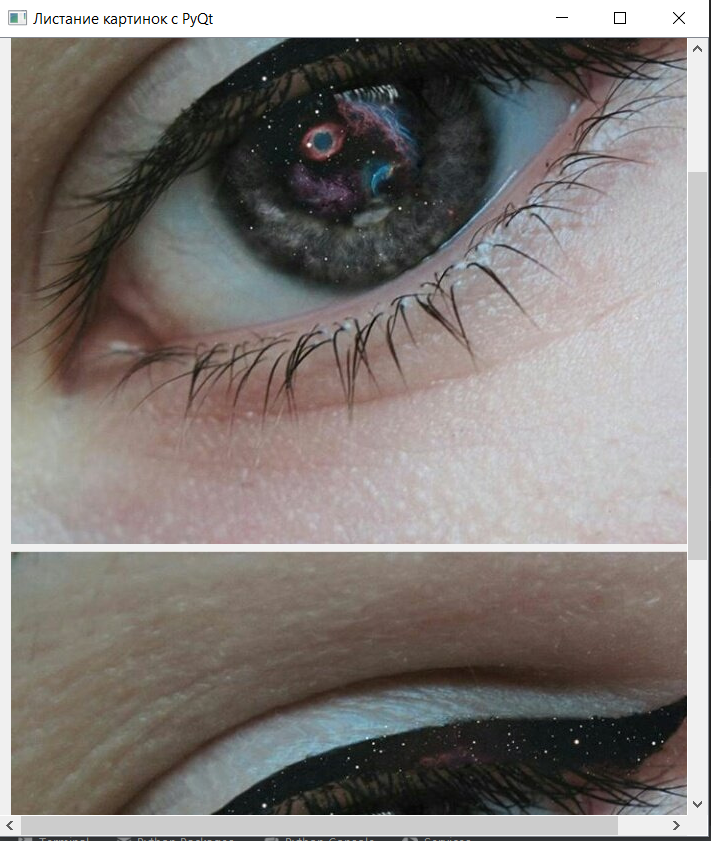
from PyQt5.QtWidgets import QMainWindow, QApplication, QLabel  
from PyQt5.QtGui import QPixmap  
  
  
class App(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setWindowTitle("Вывод картинки PyQt")  
 self.setGeometry(200, 200, 300, 300)  
  
 def load\_image(self, file\_name):  
 pixmap = QPixmap(file\_name)  
 self.label = QLabel(self)  
 self.label.setPixmap(pixmap)  
 self.label.resize(pixmap.width(), pixmap.height())  
 self.resize(pixmap.width(), pixmap.height())  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 import sys  
 app = QApplication(sys.argv)  
 res = App()  
 res.load\_image('lapot.jpg')  
 res.show()  
 sys.exit(app.exec\_())



1. Листание картинок вперед/назад

Листинг

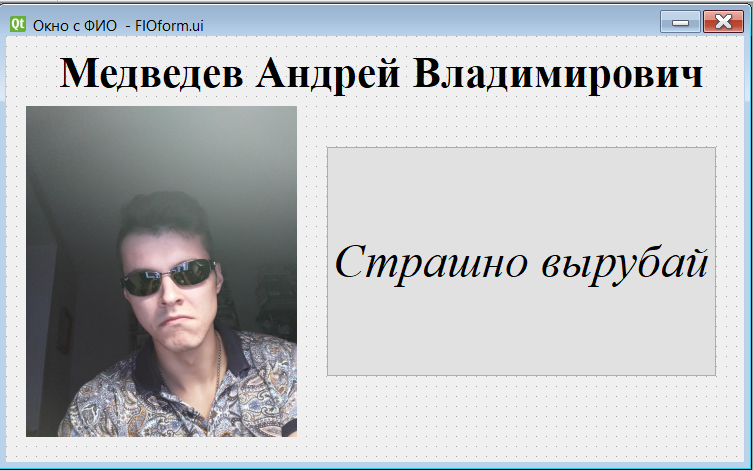
import sys  
from PyQt5.QtWidgets import QApplication, QLabel, QScrollArea, QVBoxLayout, QWidget, QMainWindow  
from PyQt5.QtGui import QPixmap  
  
  
class App(QMainWindow):  
 def \_\_init\_\_(self):  
 QMainWindow.\_\_init\_\_(self)  
 self.initUI()  
  
 def initUI(self):  
 self.setGeometry(500, 500, 400, 600)  
 self.setWindowTitle('Листание картинок с PyQt')  
 self.img = QPixmap("glaz.jpg")  
 scr = QScrollArea(self)  
 pnl = QWidget(self)  
  
 vbox = QVBoxLayout(self)  
 for i in range(2):  
 lbl = QLabel()  
 lbl.setPixmap(self.img)  
 vbox.addWidget(lbl)  
  
 pnl.setLayout(vbox)  
 scr.setWidget(pnl)  
 self.setCentralWidget(scr)  
 self.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = App()  
 sys.exit(app.exec\_())

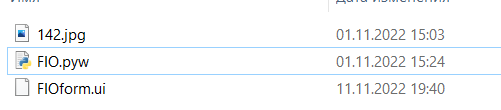


1. Окно с ФИО в .pyw

Листинг:

from PyQt5 import QtWidgets, uic  
import sys  
  
app = QtWidgets.QApplication(sys.argv)  
window = uic.loadUi("FIOform.ui")  
window.btnquit.clicked.connect(app.quit)  
window.show()  
sys.exit(app.exec\_())

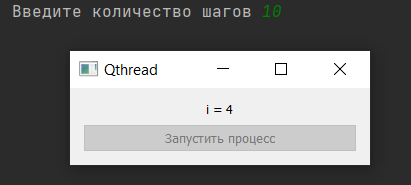




1. Итерирование в потоке

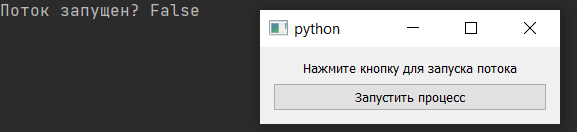
Листинг:

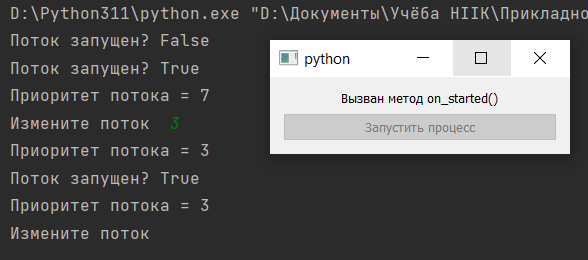
from PyQt5 import QtCore, QtWidgets  
  
class MyThread(QtCore.QThread):  
 mysignal = QtCore.pyqtSignal(str)  
 def \_\_init\_\_(self,parent=None):  
 QtCore.QThread.\_\_init\_\_(self,parent)  
 def run(self):  
 try:  
 k = int(input("Введите количество шагов "))  
 except (TypeError, ValueError) as err:  
 print("Ошибка", err)  
 for i in range(1,k+1):  
 self.sleep(1)  
 self.mysignal.emit("i = %s" % i)  
  
class MyWindow(QtWidgets.QWidget):  
 def \_\_init\_\_(self,parent=None):  
 QtWidgets.QWidget.\_\_init\_\_(self,parent)  
 self.label = QtWidgets.QLabel("Нажмите кнопку для запуска потока")  
 self.label.setAlignment(QtCore.Qt.AlignHCenter)  
 self.button = QtWidgets.QPushButton("Запустить процесс")  
 self.vbox = QtWidgets.QVBoxLayout()  
 self.vbox.addWidget(self.label)  
 self.vbox.addWidget(self.button)  
 self.setLayout(self.vbox)  
 self.mythread = MyThread()  
 self.button.clicked.connect(self.on\_clicked)  
 self.mythread.started.connect(self.on\_started)  
 self.mythread.finished.connect(self.on\_finished)  
 self.mythread.mysignal.connect(self.on\_change, QtCore.Qt.QueuedConnection)  
 def on\_clicked(self):  
 self.button.setDisabled(True)  
 self.mythread.start()  
 def on\_started(self):  
 self.label.setText("Вызван метод on\_started()")  
 def on\_finished(self):  
 self.label.setText("Вызвван метод on\_finished()")  
 self.button.setDisabled(False)  
 def on\_change(self,s):  
 self.label.setText(s)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
 app = QtWidgets.QApplication(sys.argv)  
 window = MyWindow()  
 window.setWindowTitle("Qthread")  
 window.resize(300,70)  
 window.show()  
 sys.exit(app.exec\_())



1. Активность потоков, принудительное завершение и изменение потока

Листинг

from PyQt5 import QtCore, QtWidgets  
  
  
class MyThread(QtCore.QThread):  
 mysignal = QtCore.pyqtSignal(str)  
  
 def \_\_init\_\_(self, parent=None):  
 QtCore.QThread.\_\_init\_\_(self, parent)  
 print(f"Поток запущен? {QtCore.QThread.isRunning(self)}")  
  
 def run(self):  
 while True:  
 print(f"Поток запущен? {QtCore.QThread.isRunning(self)}")  
 print(f"Приоритет потока = {QtCore.QThread.priority(self)} ")  
 s = int(input("Измените поток\t"))  
 if s == -1:  
 print("Поток принудительно завершён")  
 return -1  
 else:  
 self.setPriority(s)  
 print(f"Приоритет потока = {QtCore.QThread.priority(self)} ")  
  
  
class MyWindow(QtWidgets.QWidget):  
 def \_\_init\_\_(self, parent=None):  
 QtWidgets.QWidget.\_\_init\_\_(self, parent)  
 self.label = QtWidgets.QLabel("Нажмите кнопку для запуска потока")  
 self.label.setAlignment(QtCore.Qt.AlignHCenter)  
 self.button = QtWidgets.QPushButton("Запустить процесс")  
 self.vbox = QtWidgets.QVBoxLayout()  
 self.vbox.addWidget(self.label)  
 self.vbox.addWidget(self.button)  
 self.setLayout(self.vbox)  
 self.MyThread = MyThread()  
 self.button.clicked.connect(self.on\_clicked)  
 self.MyThread.started.connect(self.on\_started)  
 self.MyThread.finished.connect(self.on\_finished)  
 self.MyThread.mysignal.connect(self.on\_change, QtCore.Qt.QueuedConnection)  
  
 def on\_clicked(self):  
 self.button.setDisabled(True)  
 self.MyThread.start()  
  
 def on\_started(self):  
 self.label.setText("Вызван метод on\_started()")  
  
 def on\_finished(self):  
 self.label.setText("Вызван метод on\_finished()")  
 self.button.setDisabled(False)  
  
 def on\_change(self):  
 self.label.setText(s)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
  
 app = QtWidgets.QApplication(sys.argv)  
 window = MyWindow()  
 window.resize(300, 70)  
 window.show()  
 sys.exit(app.exec\_())





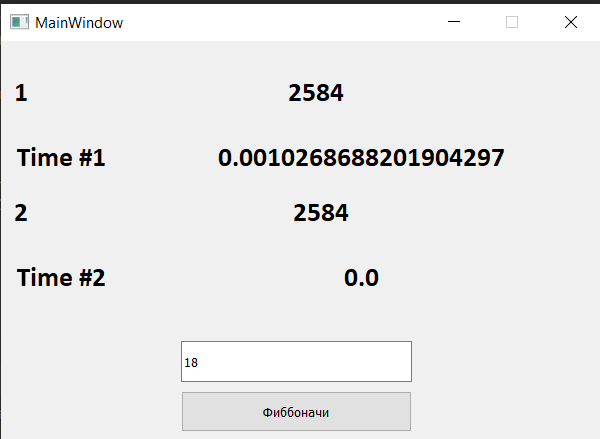
1. Числа Фибоначчи с распределением потоков

Листинг:

from des import \*  
from PyQt5 import QtCore, QtWidgets  
import sys  
import time  
  
  
def fib\_recursia(n):  
 if n in [1, 2]:  
 return 1  
 return fib\_recursia(n - 1) + fib\_recursia(n - 2)  
  
  
def fib\_chislo(n):  
 fib1 = fib2 = 1  
  
 for i in range(2, n):  
 fib1, fib2 = fib2, fib1 + fib2  
  
 return fib2  
  
  
class FirstThread(QtCore.QThread):  
 def \_\_init\_\_(self, changewindow=None):  
 super().\_\_init\_\_()  
 self.changewindow = changewindow  
 self.number = 0  
  
 def run(self):  
 self.number = int(self.changewindow.lineEdit.text())  
 now = time.time()  
 fib = fib\_recursia(self.number)  
 self.changewindow.first\_value.setText(str(fib))  
 self.changewindow.first\_time.setText(str(time.time() - now))  
  
  
class SecondThread(QtCore.QThread):  
 def \_\_init\_\_(self, changewindow=None):  
 super().\_\_init\_\_()  
 self.changewindow = changewindow  
 self.number = 0  
  
 def run(self):  
 self.number = int(self.changewindow.lineEdit.text())  
 print('Поток #2 - 1')  
 now = time.time()  
 self.number = fib\_chislo(self.number)  
 print('Поток #2 - 2')  
 self.changewindow.second\_value.setText(str(self.number))  
 self.changewindow.second\_time.setText(str(time.time() - now))  
  
  
class Ui(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
  
 self.ui.pushButton.clicked.connect(self.start)  
  
 def start(self):  
 self.firstthread = FirstThread(self.ui)  
 self.secondthread = SecondThread(self.ui)  
 self.firstthread.start()  
 self.secondthread.start()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QtWidgets.QApplication(sys.argv)  
 window = Ui()  
 window.show()  
 sys.exit(app.exec\_())

UI-файл.

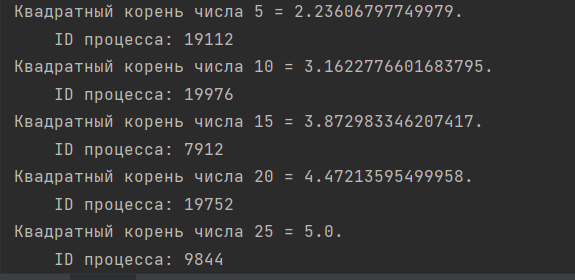
from PyQt5 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("Фиббоначи")  
 MainWindow.resize(1000, 400)  
 MainWindow.setMinimumSize(QtCore.QSize(400, 400))  
 MainWindow.setMaximumSize(QtCore.QSize(600, 400))  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.first\_value = QtWidgets.QLabel(self.centralwidget)  
 self.first\_value.setGeometry(QtCore.QRect(40, 20, 551, 61))  
 font = QtGui.QFont()  
 font.setFamily("Calibri")  
 font.setPointSize(16)  
 font.setBold(True)  
 font.setWeight(75)  
 self.first\_value.setFont(font)  
 self.first\_value.setAlignment(QtCore.Qt.AlignCenter)  
 self.first\_value.setObjectName("first\_value")  
 self.pushButton = QtWidgets.QPushButton(self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(180, 350, 231, 41))  
 self.pushButton.setObjectName("pushButton")  
 self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)  
 self.lineEdit.setGeometry(QtCore.QRect(180, 300, 231, 41))  
 self.lineEdit.setInputMask("")  
 self.lineEdit.setObjectName("lineEdit")  
 self.label\_2 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_2.setGeometry(QtCore.QRect(0, 30, 41, 41))  
 font = QtGui.QFont()  
 font.setFamily("Calibri")  
 font.setPointSize(16)  
 font.setBold(True)  
 font.setWeight(75)  
 self.label\_2.setFont(font)  
 self.label\_2.setAlignment(QtCore.Qt.AlignCenter)  
 self.label\_2.setObjectName("label\_2")  
 self.label\_3 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_3.setGeometry(QtCore.QRect(0, 80, 121, 71))  
 font = QtGui.QFont()  
 font.setFamily("Calibri")  
 font.setPointSize(16)  
 font.setBold(True)  
 font.setWeight(75)  
 self.label\_3.setFont(font)  
 self.label\_3.setAlignment(QtCore.Qt.AlignCenter)  
 self.label\_3.setObjectName("label\_3")  
 self.first\_time = QtWidgets.QLabel(self.centralwidget)  
 self.first\_time.setGeometry(QtCore.QRect(120, 80, 481, 71))  
 font = QtGui.QFont()  
 font.setFamily("Calibri")  
 font.setPointSize(16)  
 font.setBold(True)  
 font.setWeight(75)  
 self.first\_time.setFont(font)  
 self.first\_time.setText("")  
 self.first\_time.setAlignment(QtCore.Qt.AlignCenter)  
 self.first\_time.setObjectName("first\_time")  
 self.label\_5 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_5.setGeometry(QtCore.QRect(0, 150, 41, 41))  
 font = QtGui.QFont()  
 font.setFamily("Calibri")  
 font.setPointSize(16)  
 font.setBold(True)  
 font.setWeight(75)  
 self.label\_5.setFont(font)  
 self.label\_5.setAlignment(QtCore.Qt.AlignCenter)  
 self.label\_5.setObjectName("label\_5")  
 self.second\_value = QtWidgets.QLabel(self.centralwidget)  
 self.second\_value.setGeometry(QtCore.QRect(40, 140, 561, 61))  
 font = QtGui.QFont()  
 font.setFamily("Calibri")  
 font.setPointSize(16)  
 font.setBold(True)  
 font.setWeight(75)  
 self.second\_value.setFont(font)  
 self.second\_value.setAlignment(QtCore.Qt.AlignCenter)  
 self.second\_value.setObjectName("second\_value")  
 self.label\_7 = QtWidgets.QLabel(self.centralwidget)  
 self.label\_7.setGeometry(QtCore.QRect(0, 200, 121, 71))  
 font = QtGui.QFont()  
 font.setFamily("Calibri")  
 font.setPointSize(16)  
 font.setBold(True)  
 font.setWeight(75)  
 self.label\_7.setFont(font)  
 self.label\_7.setAlignment(QtCore.Qt.AlignCenter)  
 self.label\_7.setObjectName("label\_7")  
 self.second\_time = QtWidgets.QLabel(self.centralwidget)  
 self.second\_time.setGeometry(QtCore.QRect(120, 200, 481, 71))  
 font = QtGui.QFont()  
 font.setFamily("Calibri")  
 font.setPointSize(16)  
 font.setBold(True)  
 font.setWeight(75)  
 self.second\_time.setFont(font)  
 self.second\_time.setText("")  
 self.second\_time.setAlignment(QtCore.Qt.AlignCenter)  
 self.second\_time.setObjectName("second\_time")  
 MainWindow.setCentralWidget(self.centralwidget)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))  
 self.first\_value.setText(\_translate("MainWindow", "0"))  
 self.pushButton.setText(\_translate("MainWindow", "Фиббоначи"))  
 self.lineEdit.setPlaceholderText(\_translate("MainWindow", "Введите число"))  
 self.label\_2.setText(\_translate("MainWindow", "1"))  
 self.label\_3.setText(\_translate("MainWindow", "Time #1"))  
 self.label\_5.setText(\_translate("MainWindow", "2"))  
 self.second\_value.setText(\_translate("MainWindow", "0"))  
 self.label\_7.setText(\_translate("MainWindow", "Time #2"))



1. Multiprocessing, threading, QSemaphore, QWaitCondition.

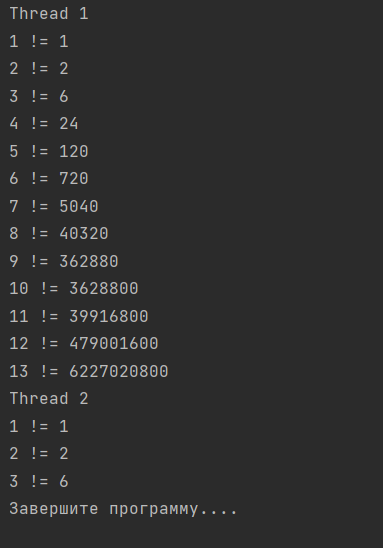
Multiprocessing:

import math  
import os  
from multiprocessing import Process  
  
  
def doubler(number):  
 # Квадратный корень числа  
 result = math.sqrt(number)  
 proc = os.getpid()  
 print(f"Квадратный корень числа {number} = {result}.\n\tID процесса: {proc}")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 numbers = [5, 10, 15, 20, 25]  
 procs = []  
  
 for index, number in enumerate(numbers):  
 proc = Process(target=doubler, args=(number,))  
 procs.append(proc)  
 proc.start()  
  
 for proc in procs:  
 proc.join()  
  
 input("Чтобы завершить программу введите что-нибудь")



Threading:

import threading  
  
threadId = 1  
thread1 = threading.Thread()  
thread2 = threading.Thread()  
  
  
def factorial(n):  
 global threadId  
 if n < 1:  
 print(f"Thread {threadId}")  
 threadId += 1  
 return 1  
 else:  
 returnNumber = n \* factorial(n - 1)  
 print(f"{n} != {returnNumber}")  
 return returnNumber  
  
  
try:  
 thread1.start(factorial(13))  
except (ValueError, TypeError) as err:  
 pass  
  
try:  
 thread2.start(factorial(3))  
except (ValueError, TypeError) as err:  
 pass  
  
c = input("Завершите программу....\n")



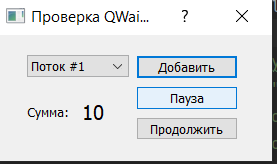
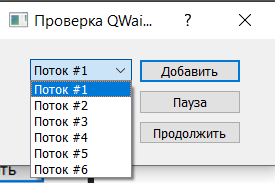
QSemaphore:

from PyQt5.QtCore import QSemaphore  
import sqlite3  
  
  
class Database:  
 # *TODO: База данных под создание* def \_\_init\_\_(self):  
 try:  
 self.name = None  
 self.dbsemaphore = QSemaphore(1) # Контроль БД  
 self.dbname = r"D:/SQLite/DataBases/" + str(self.name)  
  
 except (Exception) as err:  
 print("[-] Не получилось создать БД.", err)  
  
 def openDB(self, dbfilename):  
 try:  
 self.name = dbfilename  
 self.dbname = r"D:/SQLite/DataBases/" + str(dbfilename)  
 sql\_con = sqlite3.connect(self.dbname)  
 return sql\_con  
 except (TypeError) as err:  
 print("[-] Не получилось создать БД.", err)  
  
 def commit(self):  
 self.dbsemaphore.acquire()  
 self.dbsemaphore.release()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 db = Database()  
 db.name = "mydatabase142.db"  
 chkc = db.openDB(db.name)  
 db.commit()

Вывод: Программа создаёт в папке базу данных, можно доработать программу отдельными потоками под INSERT, SELECT, UPDATE запросы. Плюс в том, что потоки не будут обращаться одновременно к БД.

QWaitCondition

from PyQt5 import QtCore, QtGui, QtWidgets  
from PyQt5.QtCore import QThread  
from PyQt5.QtWidgets import \*  
from PyQt5.QtCore import \*  
import sys  
import time  
  
class Ui\_Dialog(object):  
 def setupUi(self, Dialog):  
 Dialog.setObjectName("QCondition")  
 Dialog.resize(275, 126)  
 self.widget = QtWidgets.QWidget(Dialog)  
 self.widget.setGeometry(QtCore.QRect(30, 20, 211, 85))  
 self.widget.setObjectName("widget")  
 self.horizontalLayout\_2 = QtWidgets.QHBoxLayout(self.widget)  
 self.horizontalLayout\_2.setContentsMargins(0, 0, 0, 0)  
 self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")  
 self.verticalLayout = QtWidgets.QVBoxLayout()  
 self.verticalLayout.setObjectName("verticalLayout")  
 self.comboBox = QtWidgets.QComboBox(self.widget)  
 self.comboBox.setObjectName("comboBox")  
 self.verticalLayout.addWidget(self.comboBox)  
 self.horizontalLayout = QtWidgets.QHBoxLayout()  
 self.horizontalLayout.setObjectName("horizontalLayout")  
 self.label = QtWidgets.QLabel(self.widget)  
 self.label.setObjectName("label")  
 self.horizontalLayout.addWidget(self.label)  
 self.label\_2 = QtWidgets.QLabel(self.widget)  
 font = QtGui.QFont()  
 font.setPointSize(12)  
 self.label\_2.setFont(font)  
 self.label\_2.setText("")  
 self.label\_2.setObjectName("label\_2")  
 self.horizontalLayout.addWidget(self.label\_2)  
 self.verticalLayout.addLayout(self.horizontalLayout)  
 self.horizontalLayout\_2.addLayout(self.verticalLayout)  
 self.verticalLayout\_2 = QtWidgets.QVBoxLayout()  
 self.verticalLayout\_2.setObjectName("verticalLayout\_2")  
 self.pushButton = QtWidgets.QPushButton(self.widget)  
 self.pushButton.setObjectName("pushButton")  
 self.verticalLayout\_2.addWidget(self.pushButton)  
 self.pushButton\_2 = QtWidgets.QPushButton(self.widget)  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 self.verticalLayout\_2.addWidget(self.pushButton\_2)  
 self.pushButton\_3 = QtWidgets.QPushButton(self.widget)  
 self.pushButton\_3.setObjectName("pushButton\_3")  
 self.verticalLayout\_2.addWidget(self.pushButton\_3)  
 self.horizontalLayout\_2.addLayout(self.verticalLayout\_2)  
  
 self.retranslateUi(Dialog)  
 QtCore.QMetaObject.connectSlotsByName(Dialog)  
 self.show()  
  
 def retranslateUi(self, Dialog):  
 \_translate = QtCore.QCoreApplication.translate  
 Dialog.setWindowTitle(\_translate("Dialog", "Проверка QWaitCondition"))  
 self.label.setText(\_translate("Dialog", "Сумма:"))  
 self.pushButton.setText(\_translate("Dialog", "Добавить"))  
 self.pushButton\_2.setText(\_translate("Dialog", "Пауза"))  
 self.pushButton\_3.setText(\_translate("Dialog", "Продолжить"))  
  
  
class Thread(QThread):  
 def \_\_init\_\_(self):  
 QThread.\_\_init\_\_(self)  
 self.\_isPause = False  
 self.condition = QWaitCondition()  
 self.mutex = QMutex()  
 self.sum = 0  
  
 def pause(self):  
 self.\_isPause = True  
  
 def resume(self):  
 self.\_isPause = False  
 self.condition.wakeAll()  
  
 def run(self):  
 while True:  
 self.mutex.lock()  
 if self.\_isPause:  
 self.condition.wait(self.mutex)  
 self.sum += 1  
 time.sleep(2)  
 self.mutex.unlock()  
  
class Window(QDialog, Ui\_Dialog):  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setupUi(self)  
 self.threads = []  
 self.sum = 0  
 self.count = 1  
 self.timer = QtCore.QTimer()  
 self.timer.timeout.connect(self.update)  
 self.timer.start(6000)  
  
 self.pushButton.clicked.connect(self.add)  
 self.pushButton\_2.clicked.connect(self.stop)  
 self.pushButton\_3.clicked.connect(self.resume)  
  
 def update(self):  
 self.sum = 0  
 for i in range(len(self.threads)):  
 self.sum += self.threads[i].sum  
 self.label\_2.setText(str(self.sum))  
  
 def add(self):  
 self.thread = Thread()  
 self.comboBox.addItem(f'Поток #{self.count}')  
 self.thread.start()  
 self.threads.append(self.thread)  
 self.count += 1  
  
 def stop(self):  
 id = self.comboBox.currentIndex()  
 self.threads[id].pause()  
  
 def resume(self):  
 id = self.comboBox.currentIndex()  
 self.threads[id].resume()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QtWidgets.QApplication(sys.argv)  
 window = Window()  
 sys.exit(app.exec\_())

Self.condition используется вместе с self.mutex, их задача в том, чтобы реагировать на события пользователя с кнопками Пауза и Продолжить. Пауза блокирует выполнение потоков, т.е срабатывает mutex. Condition в ответ на Продолжить заставляет “проснуться” всем потокам и продолжить работу.

1. Есть заставка, для продвижения прогресса загрузки - нужно решать задачи, при каждой решенной задаче меняется заставка

from PyQt5 import QtCore, QtGui, QtWidgets  
import sys  
  
  
class MyWindow(QtWidgets.QPushButton):  
 def \_\_init\_\_(self):  
 QtWidgets.QPushButton.\_\_init\_\_(self)  
 self.setText("Закрыть окно")  
 self.clicked.connect(QtWidgets.qApp.quit)  
  
 def load\_data(self, sp):  
 task = [2, 2, 6, 27, 6, 2, 15, 64, 3, 8]  
 exp = ["1+1", "3-1", "4+2", "3\*9", "3!", "ln(100)", "11+2\*2", "8^2", "13-10", "9-1"]  
 sp.setFont(QtGui.QFont("Times", 14, QtGui.QFont.Bold))  
 for i in range(10):  
 print(f"Решите уравнение\t{exp[i]}")  
 while True:  
 try:  
 k = int(input("Введите решение уравнения\t"))  
 except ValueError as err:  
 print("Введите число!")  
 else:  
 break  
  
 if k == task[i]:  
 sp.showMessage("Загрузка данных... {0}%".format(i \* 10),  
 QtCore.Qt.AlignHCenter | QtCore.Qt.AlignBottom, QtCore.Qt.red)  
 QtWidgets.qApp.processEvents()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QtWidgets.QApplication(sys.argv)  
 splash = QtWidgets.QSplashScreen(QtGui.QPixmap("image.png"))  
 splash.setFont(QtGui.QFont("Times", 14, QtGui.QFont.Bold))  
 splash.showMessage("Загрузка данных... 0%",  
 QtCore.Qt.AlignHCenter | QtCore.Qt.AlignBottom, QtCore.Qt.red)  
 splash.show()  
 QtWidgets.qApp.processEvents()  
 window = MyWindow()  
 window.setWindowTitle("Решение задачек")  
 window.resize(300, 30)  
 window.load\_data(splash)  
 window.show()  
 splash.finish(window)  
 sys.exit(app.exec\_())

