

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО РАБОТЕ №2.24
дисциплины «Основы кроссплатформенного программирования»

Выполнил:

Баканов Артем Вадимович

2 курс, группа ИТС-б-о-22-1,

11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Цель работы: приобретение навыков использования примитивов синхронизации в языке программирования Python версии 3.x.

Порядок выполнения работы:

1. Создал новый репозиторий и клонировала его на свой компьютер.

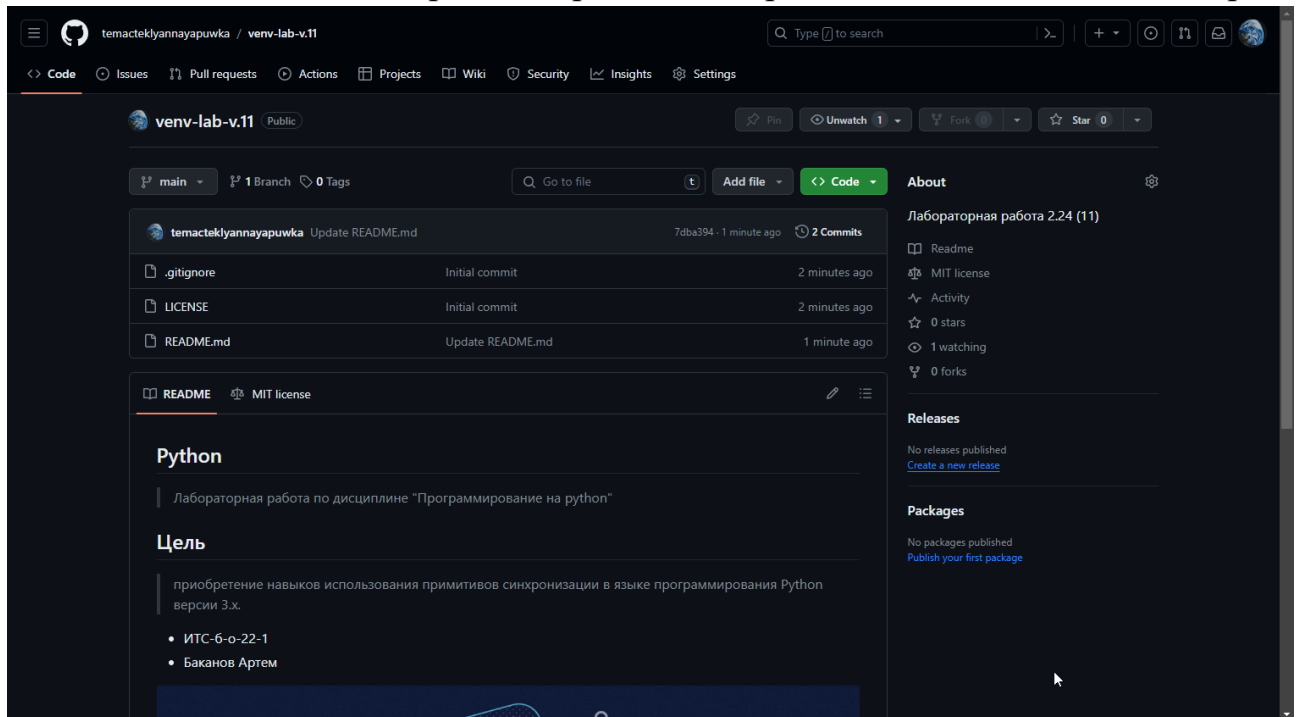


Рисунок 1 – Создан новый репозиторий

2. Клонировал репозиторий на свой компьютер. В ходе данной лабораторной работы работала с моделью ветвления git-flow.

```
PS D:\study\2course\pythonUnic\venv-lab> git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/study/2course/pythonUnic/venv-lab/.git/hooks]
PS D:\study\2course\pythonUnic\venv-lab>
```

Рисунок 2 – Клонирование и модель ветвления git-flow

3. Создал виртуальное окружение Anaconda с именем репозитория.

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate venv
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#

(base) PS D:\study\2course\pythonUnic\venv-lab\venv> conda activate venv
(venv) PS D:\study\2course\pythonUnic\venv-lab\venv> conda install django, pandas
Collecting package metadata (current_repodata.json): \ DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): r
repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.com:443
/ DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/current_repodata.json HTTP/1.1" 304 0
0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/win-64/current_repodata.json HTTP/1.1" 304 0
- DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/win-64/current_repodata.json HTTP/1.1" 304 0
done
Solving environment: \ _
```

Рисунок 3 – Создание виртуального окружения

Выполнение индивидуального задания

Разработать приложение, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) с помощью паттерна “Производитель-Потребитель”, условие которой предварительно необходимо согласовать с преподавателем.

```
Стреляет Миша - 2 раз
Вероятность попадания 2 выстрелов подряд - 0.28749600000000003

Стреляет Миша - 3 раз
Вероятность попадания 3 выстрелов подряд - 0.18974736000000003

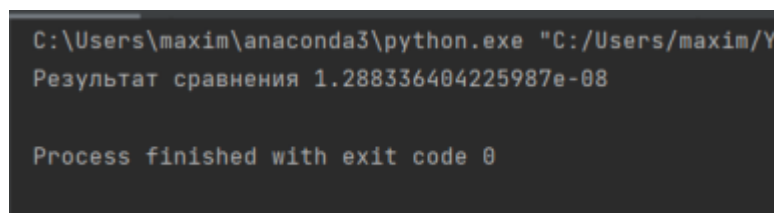
Стреляет Егор - 5 раз
Вероятность попадания 5 выстрелов подряд - 0.531441

Стреляет Павел - 3 раз
Вероятность попадания 3 выстрелов подряд - 0.04100625

Стреляет Миша - 4 раз
Вероятность попадания 4 выстрелов подряд - 0.12523325760000004
```

Рисунок 4 – Результат выполнения

Для своего индивидуального задания лабораторной работы 2.23 необходимо организовать конвейер, в котором сначала в отдельном потоке вычисляется значение первой функции, после чего результаты вычисления должны передаваться второй функции, вычисляемой в отдельном потоке. Потоки для вычисления значений двух функций должны запускаться одновременно.



```
C:\Users\maxim\anaconda3\python.exe "C:/Users/maxim/Y...
Результат сравнения 1.288336404225987e-08

Process finished with exit code 0
```

Рисунок 5 – Результат выполнения

Ответы на контрольные вопросы

1. Lock-объект может находиться в двух состояниях: захваченное (заблокированное) и не захваченное (не заблокированное, свободное). После создания он находится в свободном состоянии. Для работы с Lock-объектом используются методы `acquire()` и `release()`. Если Lock свободен, то вызов метода `acquire()` переводит его в заблокированное состояние. Повторный вызов `acquire()` приведет к блокировке инициировавшего это действие потока до тех пор, пока Lock не будет разблокирован каким-то другим потоком с помощью метода `release()`. Вывоз метода `release()` на свободном Lock-объекте приведет к выбросу исключения `RuntimeError`.

2. В отличие от рассмотренного выше Lock-объекта `RLock` может освободить только тот поток, который его захватил. Повторный захват потоком уже захваченного `RLock`-объекта не блокирует его. `RLock`-объекты поддерживают возможность вложенного захвата, при этом освобождение происходит только после того, как был выполнен `release()` для внешнего `acquire()`. Сигнатуры и назначение методов `release()` и `acquire()` `RLock`-объектов совпадают с приведенными для `Lock`, но в отличие от него у `RLock` нет метода `locked()`. `RLock`-объекты поддерживают протокол менеджера контекста. С помощью команды закрытия `close()`.

3. Порядок работы с условными переменными выглядит так: • На стороне Consumer'a: проверить доступен ли ресурс, если нет, то перейти в режим ожидания с помощью метода `wait()`, и ожидать оповещение от Producer'a о том, что ресурс готов и с ним можно работать. Метод `wait()` может быть вызван с таймаутом, по истечении которого поток выйдет из состояния блокировки и продолжит работу. • На стороне Producer'a: произвести работы по подготовке ресурса, после того, как ресурс готов оповестить об этом ожидающие потоки с помощью методов `notify()` или `notify_all()`. Разница между ними в том, что `notify()` разблокирует только один поток (если он вызван без параметров), а `notify_all()`

все потоки, которые находятся в режиме ожидания. Чтобы обновить данные в таблице, просто создайте соединение, затем создайте объект курсора с помощью соединения и, наконец, используйте оператор UPDATE.

4. При создании объекта Condition вы можете передать в конструктор объект Lock или RLock, с которым хотите работать. Перечислим методы объекта Condition с кратким описанием: `acquire(*args)` – захват объекта-блокировки. `release()` – освобождение объекта-блокировки. `wait(timeout=None)` – блокировка выполнения потока до оповещения о снятии блокировки. Через параметр `timeout` можно задать время ожидания оповещения о снятии блокировки. Если вызвать `wait()` на Условной переменной, у которой предварительно не был вызван `acquire()`, то будет выброшено исключение `RuntimeError`.

5. Чтобы перечислить все таблицы в базе данных SQLite3, вы должны запросить данные из таблицы `sqlite_master`, а затем использовать `fetchall()` для получения результатов из инструкции SELECT

6. При создании таблицы мы должны убедиться, что она еще не существует. Аналогично, при удалении/удалении таблицы она должна существовать. Чтобы проверить, не существует ли таблица уже, мы используем `IF NOT EXISTS` с оператором `CREATE TABLE` следующим образом.

7. Метод `executemany` можно использовать для вставки нескольких строк одновременно.

8. В базе данных Python SQLite3 мы можем легко хранить дату или время, импортируя модуль `datetime`. Следующие форматы являются наиболее часто используемыми форматами для `datetime`:

Вывод: в результате выполнения работы были приобретены навыки использования примитивов синхронизации в языке программирования Python версии 3.x.