

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 ЛАБОРАТОРНОЙ РАБОТЕ №11**  
**дисциплины «Анализ данных»**

Выполнил:  
Гайчук Дарья Дмитриевна  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика  
и вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А.-доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

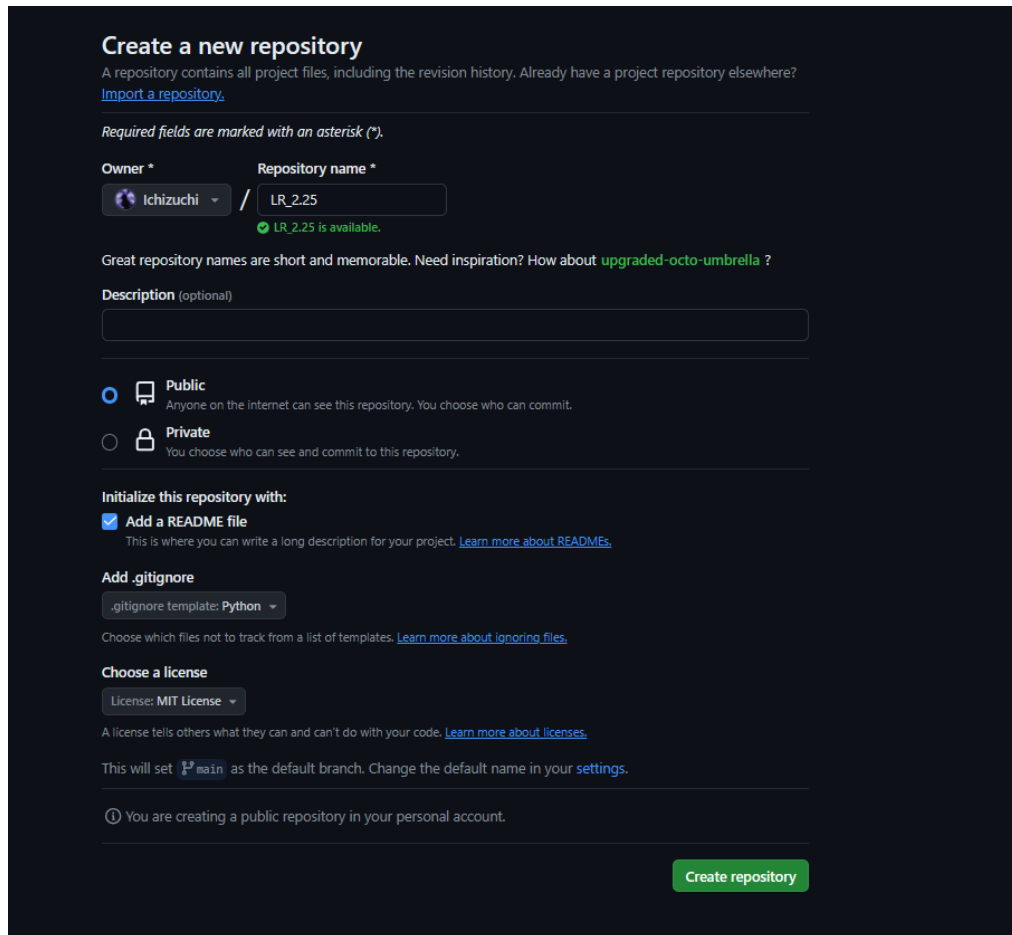
Ставрополь, 2024 г.

**Тема:** «Управление процессами в Python».

**Цель работы:** приобретение навыков написания многозадачных приложений на языке программирования Python версии 3.x.

**Порядок выполнения работы:**

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



**Create a new repository**  
A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \* Ichizuchi / Repository name \* LR\_2.25  
✔ LR\_2.25 is available.

Great repository names are short and memorable. Need inspiration? How about [upgraded-octo-umbrella](#)?

Description (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:  
☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore  
.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license  
License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your settings.

① You are creating a public repository in your personal account.

[Create repository](#)

Рисунок 1. Создан новый репозиторий

2. Клонировала репозиторий на свой компьютер.

```
@Ichizuchi →/workspaces/LR_2.25 (main) $ git branch develop
@Ichizuchi →/workspaces/LR_2.25 (main) $ git chekout develop
git: 'chekout' is not a git command. See 'git --help'.

The most similar command is
  checkout
@Ichizuchi →/workspaces/LR_2.25 (main) $ git checkout develop
Switched to branch 'develop'
@Ichizuchi →/workspaces/LR_2.25 (develop) $
```

Рисунок 2. Клонирование и модель ветвления git-flow

3. Создала виртуальное окружение Anaconda с именем репозитория.

```
@Ichizuchi →/workspaces/LR_2.25 (develop) $ conda create -n myenv python=3.10
Retrieving notices: ...working... done
Channels:
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /opt/conda/envs/myenv

added / updated specs:
- python=3.10

The following packages will be downloaded:
```

package	build	
-----	-----	
_libgcc_mutex-0.1	main	3 KB
_openmp_mutex-5.1	1_gnu	21 KB
bzip2-1.0.8	h5eee18b_6	262 KB
ca-certificates-2024.3.11	h06a4308_0	127 KB
ld_impl_linux-64-2.38	h1181459_1	654 KB
libffi-3.4.4	h6a678d5_1	141 KB

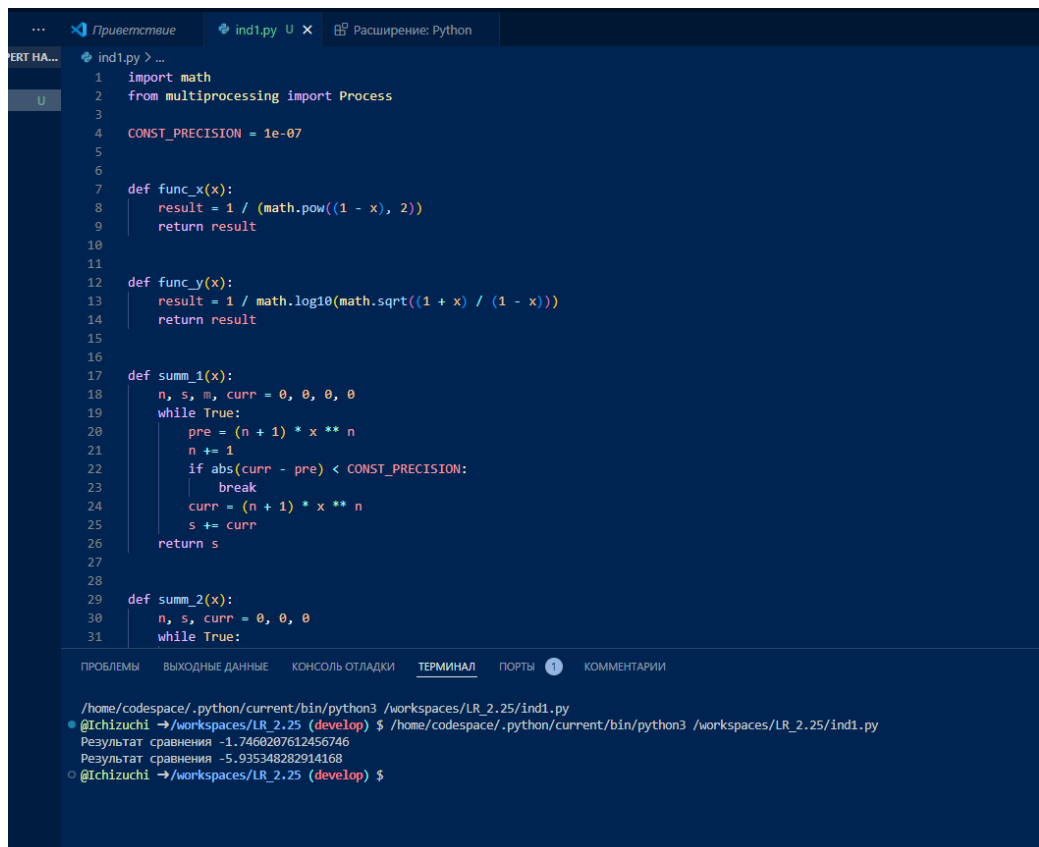
Рисунок 3. Создание виртуального окружения

### Выполнение индивидуального задания

Для своего индивидуального задания лабораторной работы 2.23 необходимо реализовать вычисление значений в двух функций в отдельных процессах.

Для каждой функции compare создаются отдельные процессы с использованием Process. Каждый процесс запускается методом start(), после чего основной процесс блокируется при помощи join(), чтобы дождаться завершения всех созданных процессов.

Таким образом, функции compare, summ\_1 и summ\_2 будут выполняться параллельно в отдельных процессах.



```
1 import math
2 from multiprocessing import Process
3
4 CONST_PRECISION = 1e-07
5
6
7 def func_x(x):
8     result = 1 / (math.pow((1 - x), 2))
9     return result
10
11
12 def func_y(x):
13     result = 1 / math.log10(math.sqrt((1 + x) / (1 - x)))
14     return result
15
16
17 def summ_1(x):
18     n, s, m, curr = 0, 0, 0, 0
19     while True:
20         pre = (n + 1) * x ** n
21         n += 1
22         if abs(curr - pre) < CONST_PRECISION:
23             break
24         curr = (n + 1) * x ** n
25         s += curr
26     return s
27
28
29 def summ_2(x):
30     n, s, curr = 0, 0, 0
31     while True:
```

PROБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    **ТЕРМИНАЛ**    ПОРТЫ 1    КОММЕНТАРИИ

```
/home/codespace/.python/current/bin/python3 /workspaces/LR_2.25/ind1.py
@Ichizuchi →/workspaces/LR_2.25 (develop) $ /home/codespace/.python/current/bin/python3 /workspaces/LR_2.25/ind1.py
Результат сравнения -1.7460207612456746
Результат сравнения -5.935348282914108
@Ichizuchi →/workspaces/LR_2.25 (develop) $
```

Рисунок 8. Результат выполнения

#### 4. Сформировала файлы environment.yml и requirements.txt



```
Результат сравнения -5.935348282914108
@Ichizuchi →/workspaces/LR_2.25 (develop) $ conda env export > environment.yml
@Ichizuchi →/workspaces/LR_2.25 (develop) $ pip freeze > requirements.txt
@Ichizuchi →/workspaces/LR_2.25 (develop) $ conda init
no change      /opt/conda/condabin/conda
no change      /opt/conda/bin/conda
no change      /opt/conda/bin/conda-env
no change      /opt/conda/bin/activate
no change      /opt/conda/bin/deactivate
no change      /opt/conda/etc/profile.d/conda.sh
no change      /opt/conda/etc/fish/conf.d/conda.fish
no change      /opt/conda/shell/condabin/Conda.psm1
no change      /opt/conda/shell/condabin/conda-hook.ps1
```

Рисунок 10. Файлы environment.yml и requirements.txt

### Ответы на контрольные вопросы

#### 1. Создание и завершение процессов в Python:

В Python процессы создаются с помощью модуля `multiprocessing`. Для этого обычно используются классы `Process` или `Pool`. После создания процесса его выполнение запускается вызовом метода `start()`, а завершение

процесса происходит путем вызова метода `join()`, который блокирует выполнение основного потока до завершения дочернего процесса.

## 2. Особенность создания классов-наследников от `Process`:

Когда вы создаете класс-наследник от `multiprocessing.Process`, вы должны определить метод `run()`, который будет содержать код, выполняемый в новом процессе. Это позволяет вам легко организовывать и управлять параллельным выполнением различных задач в вашем приложении.

## 3. Выполнение принудительного завершения процесса:

Принудительное завершение процесса в Python можно выполнить вызовом метода `terminate()` для объекта процесса. Этот метод посылает процессу сигнал завершения, что приводит к прерыванию его выполнения.

## 4. Процессы-демоны и их запуск:

Процессы-демоны (daemon processes) в Python - это процессы, которые работают в фоновом режиме и завершаются, когда основной процесс завершается. Для того чтобы запустить процесс как демона, нужно установить флаг `daemon` в `True` перед запуском процесса. Это можно сделать либо при создании экземпляра класса `Process`, либо установив атрибут `daemon` после создания, но до запуска процесса.

Вывод: в результате выполнения работы были приобретены навыки написания многозадачных приложений на языке программирования Python версии 3.x