

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 №3**  
**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: работа с переменными окружениями в python3

Цель работы: приобретение навыков по работе с переменными окружениями с помощью языка программирования Python версии 3.x.

### Порядок выполнения работы:

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

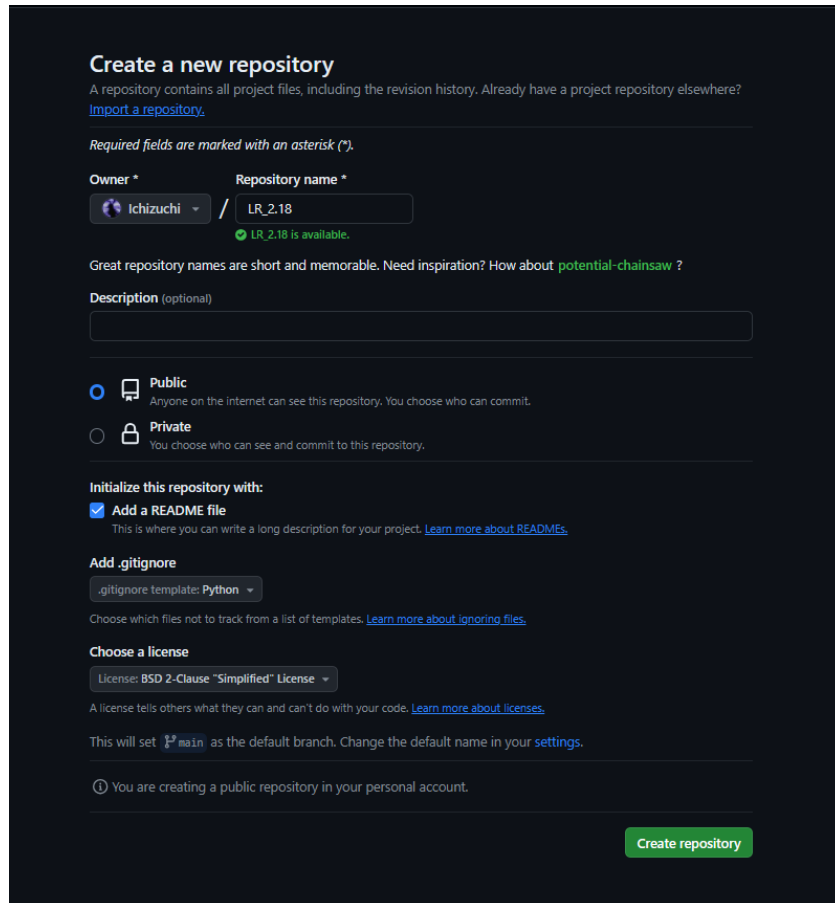


Рисунок 1. Создан новый репозиторий

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

```
@Ichizuchi →/workspaces/LR_2.18 (develop) $ git clone https://github.com/Ichizuchi/LR_2.18
Cloning into 'LR_2.18'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
@Ichizuchi →/workspaces/LR_2.18 (develop) $ git checkout develop
Already on 'develop'
```

Рисунок 2. Клонирование и модель ветвления git-flow

3. Создала виртуальное окружение Anaconda с именем репозитория.

```
Switched to branch 'develop'
@Ichizuchi →/workspaces/LR_2.18 (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	size
_libgcc_mutex-0.1	main	3 KB
_openmp_mutex-5.1	1_gnu	21 KB
bzip2-1.0.8	h5eee18b_5	262 KB
ca-certificates-2023.12.12	h06a4308_0	126 KB
ld_impl_linux-64-2.38	h1181459_1	654 KB
libffi-3.4.4	h6a678d5_0	142 KB

Рисунок 3. Создание виртуального окружения

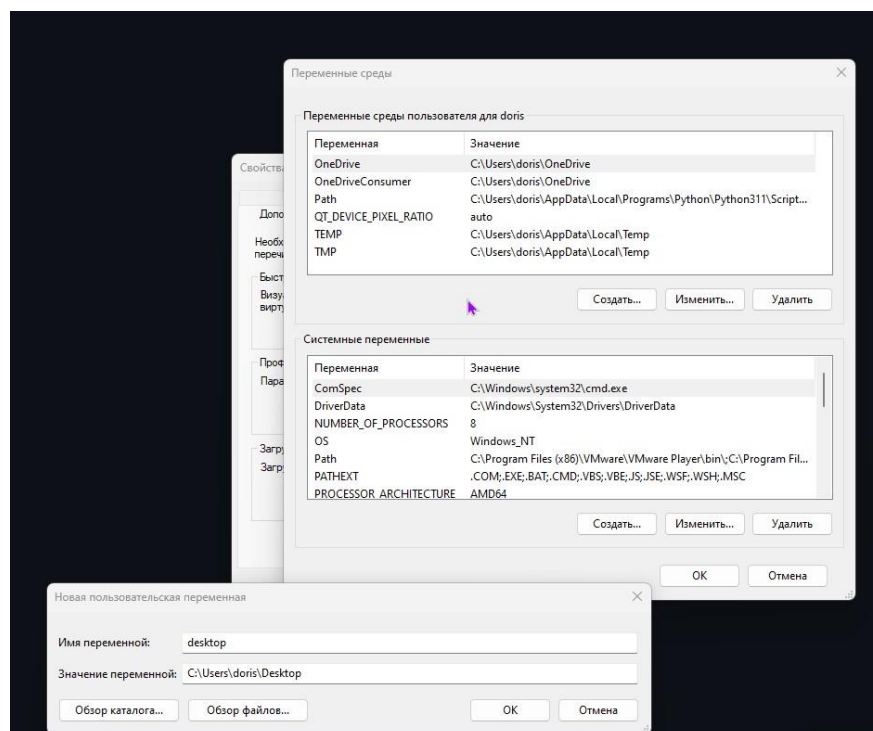
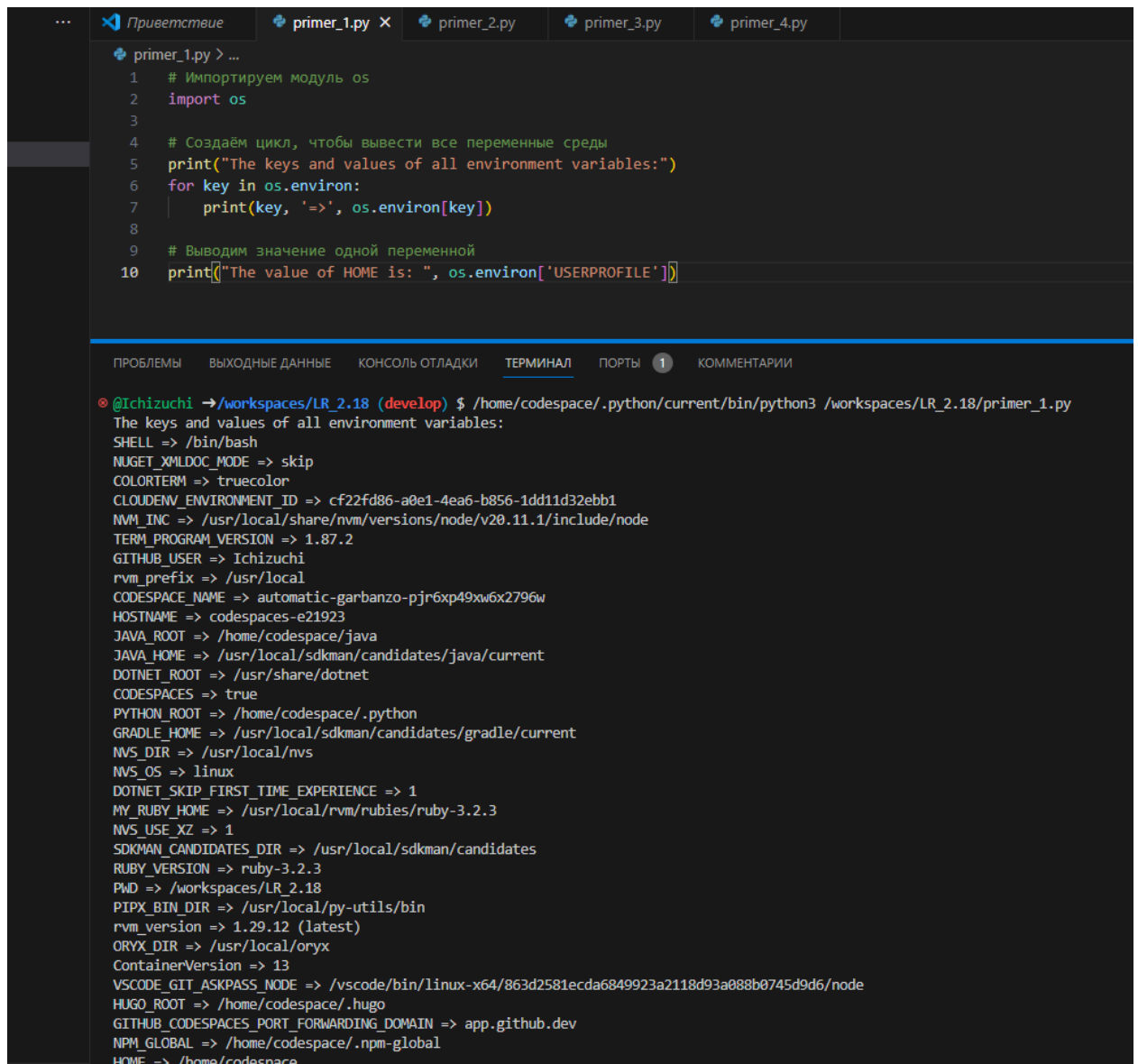


Рисунок 4. Отработка примера 1



The image shows a VS Code editor window with a dark theme. At the top, there are several tabs: 'Приветствие', 'primer\_1.py', 'primer\_2.py', 'primer\_3.py', and 'primer\_4.py'. The 'primer\_1.py' tab is active, displaying a Python script. Below the editor, a terminal window is open, showing the execution of the script. The script prints the keys and values of all environment variables, and the terminal output lists various system and workspace-related variables.

```
primer_1.py > ...
1  # Импортируем модуль os
2  import os
3
4  # Создаём цикл, чтобы вывести все переменные среды
5  print("The keys and values of all environment variables:")
6  for key in os.environ:
7      print(key, '=>', os.environ[key])
8
9  # Выводим значение одной переменной
10 print("The value of HOME is: ", os.environ['USERPROFILE'])
```

PROБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ    ПОРТЫ 1    КОММЕНТАРИИ

@Ichizuchi →/workspaces/LR\_2.18 (develop) \$ /home/codespace/.python/current/bin/python3 /workspaces/LR\_2.18/primer\_1.py

The keys and values of all environment variables:

SHELL => /bin/bash

NUGET\_XMLDOC\_MODE => skip

COLORTERM => truecolor

CLOUDENV\_ENVIRONMENT\_ID => cf22fd86-a0e1-4ea6-b856-1dd11d32ebb1

NVM\_INC => /usr/local/share/nvm/versions/node/v20.11.1/include/node

TERM\_PROGRAM\_VERSION => 1.87.2

GITHUB\_USER => Ichizuchi

rvm\_prefix => /usr/local

CODESPACE\_NAME => automatic-garbanzo-pjr6xp49xw6x2796w

HOSTNAME => codespaces-e21923

JAVA\_ROOT => /home/codespace/java

JAVA\_HOME => /usr/local/sdkman/candidates/java/current

DOTNET\_ROOT => /usr/share/dotnet

CODESPACES => true

PYTHON\_ROOT => /home/codespace/.python

GRADLE\_HOME => /usr/local/sdkman/candidates/gradle/current

NVS\_DIR => /usr/local/nvs

NVS\_OS => linux

DOTNET\_SKIP\_FIRST\_TIME\_EXPERIENCE => 1

MY\_RUBY\_HOME => /usr/local/rvm/rubies/ruby-3.2.3

NVS\_USE\_XZ => 1

SDKMAN\_CANDIDATES\_DIR => /usr/local/sdkman/candidates

RUBY\_VERSION => ruby-3.2.3

PWD => /workspaces/LR\_2.18

PIPX\_BIN\_DIR => /usr/local/py-utils/bin

rvm\_version => 1.29.12 (latest)

ORYX\_DIR => /usr/local/oryx

ContainerVersion => 13

VSCODE\_GIT\_ASKPASS\_NODE => /vscode/bin/linux-x64/863d2581ecda6849923a2118d93a088b0745d9d6/node

HUGO\_ROOT => /home/codespace/.hugo

GITHUB\_CODESPACES\_PORT\_FORWARDING\_DOMAIN => app.github.dev

NPM\_GLOBAL => /home/codespace/.npm-global

HOME => /home/codespace

Рисунок 5. Обработка примера 2

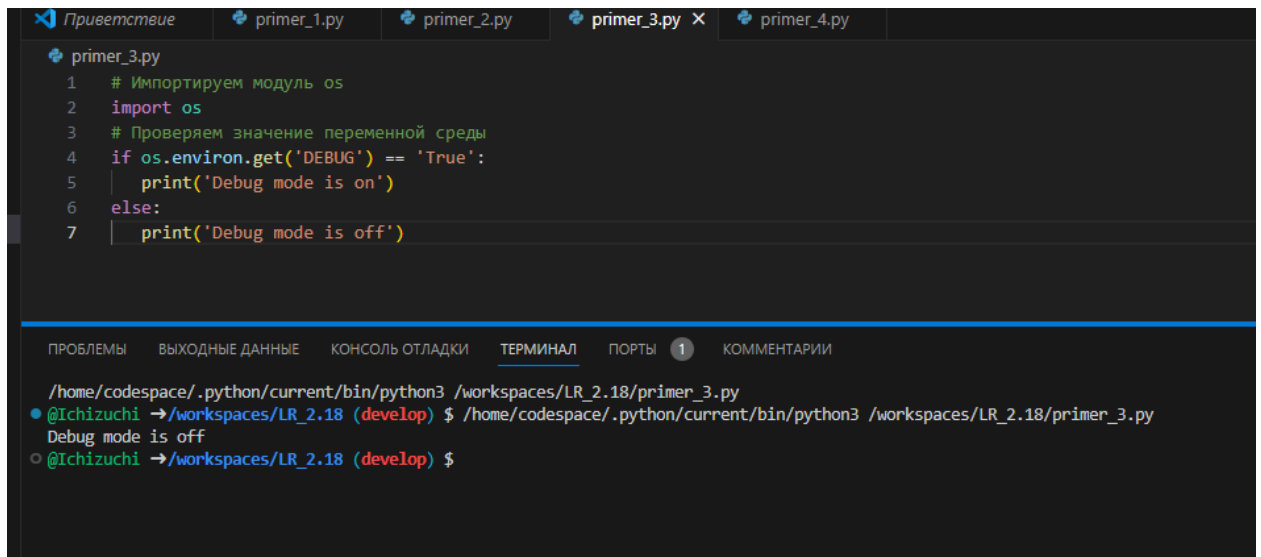
The image shows a VS Code editor window with a dark theme. The top bar displays several open files: 'Приветствие', 'primer\_1.py', 'primer\_2.py' (which is the active file and has a close button), 'primer\_3.py', and 'primer\_4.py'. The editor area shows the code for 'primer\_2.py' with line numbers 4 through 23. The code is a Python script that uses a while loop to repeatedly ask the user for an environment variable key. If the key exists in the environment, it prints its value; otherwise, it prints an error message and exits with a status code of 1.

```
4 import sys
5
6 while True:
7     # Принимаем имя переменной среды
8     key_value = input("Enter the key of the environment variable:")
9
10    # Проверяем, инициализирована ли переменная
11    try:
12        if os.environ[key_value]:
13            print(
14                "The value of",
15                key_value,
16                " is ",
17                os.environ[key_value]
18            )
19    # Если переменной не присвоено значение, то ошибка
20    except KeyError:
21        print(key_value, 'environment variable is not set.')
22    # Завершаем процесс выполнения скрипта
23    sys.exit(1)
```

Below the editor, the 'TERMINAL' tab is active, showing the command to run the script and its output. The user has entered 'PWD' and 'SHELL' as keys, and the script correctly outputs their values. The prompt is now waiting for another input.

```
@Ichizuchi →/workspaces/LR_2.18 (develop) $ /home/codespace/.python/current/bin/python3 /workspaces/LR_2.18/primer_2.py
Enter the key of the environment variable:PWD
The value of PWD is /workspaces/LR_2.18
Enter the key of the environment variable:SHELL
The value of SHELL is /bin/bash
Enter the key of the environment variable:
```

Рисунок 6. Отработка примера 3

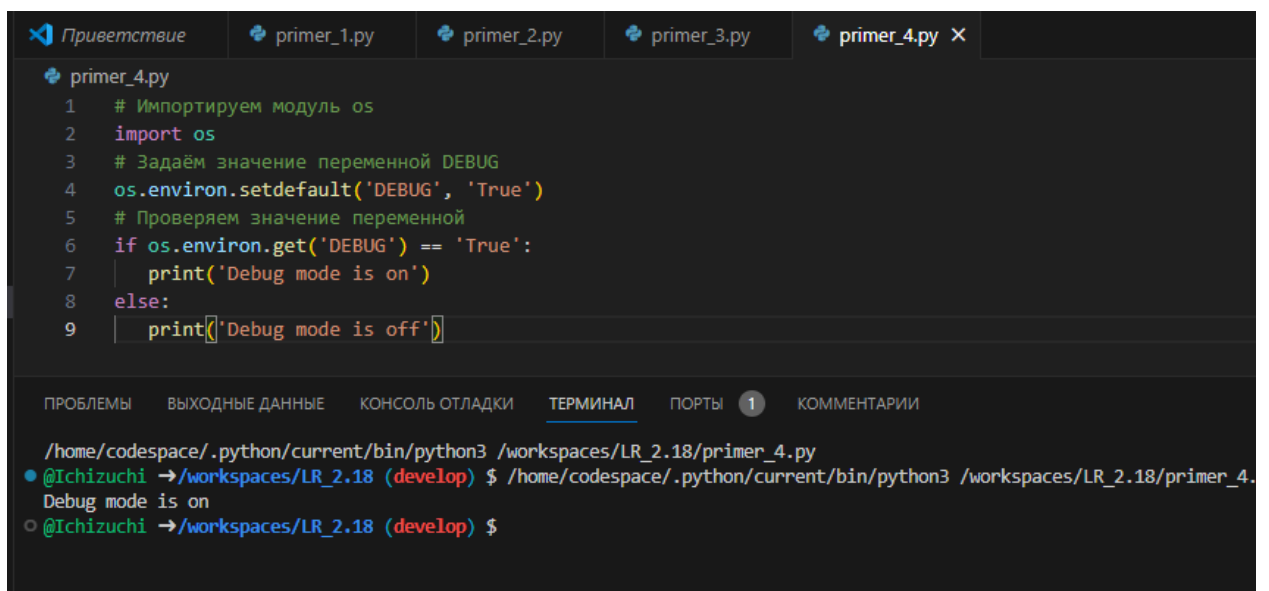


```
Приветствие primer_1.py primer_2.py primer_3.py X primer_4.py
primer_3.py
1 # Имportsируем модуль os
2 import os
3 # Проверяем значение переменной среды
4 if os.environ.get('DEBUG') == 'True':
5     print('Debug mode is on')
6 else:
7     print('Debug mode is off')

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ 1 КОММЕНТАРИИ

/home/codespace/.python/current/bin/python3 /workspaces/LR_2.18/primer_3.py
● @Ichizuchi →/workspaces/LR_2.18 (develop) $ /home/codespace/.python/current/bin/python3 /workspaces/LR_2.18/primer_3.py
Debug mode is off
○ @Ichizuchi →/workspaces/LR_2.18 (develop) $
```

Рисунок 7. Отработка примера 4



```
Приветствие primer_1.py primer_2.py primer_3.py primer_4.py X
primer_4.py
1 # Имportsируем модуль os
2 import os
3 # Задаём значение переменной DEBUG
4 os.environ.setdefault('DEBUG', 'True')
5 # Проверяем значение переменной
6 if os.environ.get('DEBUG') == 'True':
7     print('Debug mode is on')
8 else:
9     print('Debug mode is off')

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ 1 КОММЕНТАРИИ

/home/codespace/.python/current/bin/python3 /workspaces/LR_2.18/primer_4.py
● @Ichizuchi →/workspaces/LR_2.18 (develop) $ /home/codespace/.python/current/bin/python3 /workspaces/LR_2.18/primer_4.py
Debug mode is on
○ @Ichizuchi →/workspaces/LR_2.18 (develop) $
```

Рисунок 8. Отработка примера 5

#### 4. Выполнение индивидуального задания



```
● @Ichizuchi →/workspaces/LR_2.18 (develop) $ conda env export > environment.yml
● @Ichizuchi →/workspaces/LR_2.18 (develop) $ pip freeze > requirements.txt
● @Ichizuchi →/workspaces/LR_2.18 (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
no change      /opt/conda/lib/python3.12/site-packages/xontrib/conda.xsh
no change      /opt/conda/etc/profile.d/conda.csh
modified       /home/codespace/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

○ @Ichizuchi →/workspaces/LR_2.18 (develop) $
```

Рисунок 10. Файлы environment.yml и requirements.txt

## 6. Слия ветку develop с веткой main/

```
your branch is up to date with 'origin/main'.
● (base) @Ichizuchi →/workspaces/LR_2.18 (main) $ git merge develop
Updating c1694b3..1929f9a
Fast-forward
 LR_2.18      | 1 +
 ind_1.py     | 65 +++++
 primer_1.py  | 10 +
 primer_2.py  | 23 +
 primer_3.py  | 7 +
 primer_4.py  | 9 +
 6 files changed, 115 insertions(+)
 create mode 160000 LR_2.18
 create mode 100644 ind_1.py
 create mode 100644 primer_1.py
 create mode 100644 primer_2.py
 create mode 100644 primer_3.py
 create mode 100644 primer_4.py
○ (base) @Ichizuchi →/workspaces/LR_2.18 (main) $
```

Рисунок 11. Слияние веток

## Ответы на контрольные вопросы

### 1. Каково назначение переменных окружения?

Назначение переменных окружения состоит в сохранении и обмене данными между операционной системой и запущенными процессами. Они предоставляют способ хранения конфигурационной информации, путей к



исполняемым Файлам, настройкам системы и других панелях, которые должны быть доступны во время выполнения программ

## 2. Какая информация может храниться в переменных окружения?

В переменных окружения можно хранить различные виды информации, такие как пути к директориям, пути к исполняемым файлам, значения настроек системы, ключи API и другие данные, необходимые программам для работы.

## 3. Как получить доступ к переменным окружения в ОС Windows?

В ОС Windows можно получить доступ к переменным окружения следующим образом: - Через Панель управления: Перейдите в "Система -> Дополнительные параметры системы -> Переменные среды". Здесь можно просмотреть и изменить переменные окружения для текущего пользователя или для системы. - Через командную строку: Используйте команду SET для просмотра переменных окружения текущего пользователя, а команду SETX для создания или изменения переменных окружения

## 4. Каково назначение переменных PATH и PATHEXT?

Переменная окружения PATH используется для определения списка директорий, в которых операционная система будет искать исполняемые файлы при запуске команд. Переменная окружения PATHEXT содержит список расширений файлов, которые считаются исполняемыми и могут быть запущены без указания полного пути.

## 5. Как создать или изменить переменную окружения в Windows?

Чтобы создать или изменить переменную окружения в Windows, выполните следующие действия: Откройте Панель управления и перейдите в "Система Дополнительные параметры системы -> Переменные среды". - В окне "Переменные среды" можно создать новую переменную, щелкнув кнопку "Создать", или изменить существующую переменную, выделив ее и нажав кнопку "Изменить"

## 6. Что представляют собой переменные окружения в ОС Linux?

В ОС Linux переменные окружения являются способом передачи информации между оболочкой и запущенными процессами. Они могут

содержать данные о текущей среде, заданные внешними программами или пользовательскими настройками. В Linux переменные окружения могут быть установлены в текущей оболочке или экспортированы для использования другими процессами.

#### 7. В чем отличие переменных окружения от переменных оболочки?

Переменные окружения представляют значения, которые передаются из операционной системы записанные процессы и доступны им для использования. Переменные • оболочки. с другой стороны, являются локальными переменными, определенными внутри оболочки, и они доступны только в рамках этой оболочки.

#### 8. Как вывести значение переменной окружения в Linux?

Чтобы вывести значение переменной окружения в Linux, можно воспользоваться командой `echo` и указать имя переменной окружения с символом `$`

#### 9. Какие переменные окружения Linux Вам известны?

Примеры переменных окружения в Linux: - `PATH`: содержит пути к директориям, в которых операционная система ищет исполняемые файлы. - `HOME`: указывает на домашнюю директорию текущего пользователя. - `USER`: содержит имя текущего пользователя. - `LANG`: определяет язык, используемый для локализации программного обеспечения и вывода сообщений

#### 10. Какие переменные оболочки Linux Вам известны?

Примеры переменных оболочки Linux: - `PS1`: определяет строку приглашения командной оболочки. - `PS2`: определяет вторичную строку приглашения для многострочных - `HOME`: указывает на домашнюю директорию текущего пользователя. - `PWD`: содержит путь к текущей рабочей директории

#### 11. Как установить переменные оболочки в Linux?

Переменные оболочки устанавливаются путем присваивания значений переменным.

## 12. Как установить переменные окружения в Linux?

Переменные окружения в Linux устанавливаются путем задания значений переменным в файле окружения.

13. Для чего необходимо делать переменные окружения Linux постоянными?

Постоянные переменные окружения нужны для того, чтобы значения переменных сохранялись между сеансами работы пользователя.

## 14. Для чего используется переменная окружения PYTHONHOME?

Переменная окружения PYTHONHOME используется для указания директории, в которой находится установленная версия Python.

## 15. Для чего используется переменная окружения PYTHONPATH?

Переменная окружения PYTHONPATH используется для указания директорий, в которых Python ищет модули при импорте.

16. Какие еще переменные окружения используются для управления работой интерпретатора Python?

Другие переменные окружения для управления работой интерпретатора Python: - PYTHONSTARTUP: путь к файлу, который будет выполняться при запуске интерпретатора. - PYTHONCASEOK: если установлено в любое значение, регистр букв в именах модулей не будет учитываться при импорте.

17. Как осуществляется чтение переменных окружения в программах на языке программирования Python?

В Python переменные окружения могут быть прочитаны с помощью модуля Os: `import os value = os.environ.get("VARIABLE_NAME")`

18. Как проверить, установлено или нет значение переменной окружения в программах на языке программирования Python?

Чтобы проверить, установлено ли значение переменной окружения, Можно использовать условное выражение. `if "VARIABLE NAME" in os.environ: value = os.environ["VARIABLE_NAME"]`

19. Как присвоить значение переменной окружения в программах на языке программирования Python?

Чтобы установить • значение переменной окружения, МОЖНО использовать os.environ: os.environ["VARIABLE NAME" = "value"]

**Вывод:** приобрела навыки по работе с переменными окружениями в python3.