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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЁТ

по лабораторной работе №2.18

**Дисциплина: «Анализ данных»**
**Тема: «Работа с переменными окружения в Python3»**

Выполнил:
Епифанов Алексей Александрович
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 Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

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

Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.

2. Проработал пример лабораторной работы:

```
❯ program/prim.py -h
usage: workers [-h] [--version] {add,display,select} ...

positional arguments:
  {add,display,select}
    add                 Add a new worker
    display             Display all workers
    select              Select the workers

options:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
❯ program/prim.py display -h
usage: workers display [-h] [-d DATA]

options:
  -h, --help            show this help message and exit
  -d DATA, --data DATA  The data file name
❯ program/prim.py display
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Алексей                          | Студент              |     2022 |
+------+----------------------------------+----------------------+----------+
❯ program/prim.py add -n Виктор -p Студент -y 2012
❯ program/prim.py display -d prim.json
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Алексей                          | Студент              |     2022 |
+------+----------------------------------+----------------------+----------+
|    2 | Виктор                           | Студент              |     2012 |
+------+----------------------------------+----------------------+----------+
❯ program/prim.py select -P 8
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Виктор                           | Студент              |     2012 |
+------+----------------------------------+----------------------+----------+
❯ printenv WORKERS_DATA
./prim.json

 ~/Desktop/./../gits/Lab_2.18  lab-2-18-py3.12
```

Рисунок 1. Ввод, вывод и выбор работников в консоли

3. Выполнил индивидуальное задание 1: Для своего варианта лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import argparse
import bisect
import json
import os
import sys
from jsonschema import ValidationError, validate
def add_route(routes, start, end, count):
    """
    Добавить данные о маршруте.
    """
    route = {
        "начальный пункт": start.lower(),
        "конечный пункт": end.lower(),
        "номер маршрута": count,
    }
    if route not in routes:
        bisect.insort(
            routes,
            route,
            key=lambda item: item.get("номер маршрута"),
        )
    else:
        print("Данный маршрут уже добавлен.")
    return routes
def display_routes(routes):
    """
    Отобразить список маршрутов.
    """
    if routes:
        line = "+-{}-+-{}-+-{}-+".format("-" * 30, "-" * 20, "-" * 8)
        print(line)
        print("| {:^30} | {:^20} | {:^8} |".format("Начало", "Конец", "Номер"))
        print(line)
        for route in routes:
            print(
                "| {:<30} | {:<20} | {:>8} |".format(
                    route.get("начальный пункт", ""),
                    route.get("конечный пункт", ""),
```

```
                route.get("номер маршрута", ""),
            )
        )
        print(line)
    else:
        print("Список маршрутов пуст.")
def select_routes(routes, name_point):
    """
    Выбрать маршруты с заданным пунктом отправления или прибытия.
    """
    selected = []
    for route in routes:
        if (
            route["начальный пункт"] == name_point
            or route["конечный пункт"] == name_point
        ):
            selected.append(route)
    return selected
def save_routes(file_name, routes):
    """
    Сохранить все маршруты в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w") as file_out:
        # Записать данные из словаря в формат JSON и сохранить их
        # в открытый файл.
        json.dump(routes, file_out, ensure_ascii=False, indent=4)
def load_routes(file_name):
    """
    Загрузить все маршруты из файла JSON.
    """
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "начальный пункт": {"type": "string"},
                "конечный пункт": {"type": "string"},
                "номер маршрута": {"type": "integer"},
            },
            "required": [
                "начальный пункт",
                "конечный пункт",
                "номер маршрута",
            ],
```

```
        },
    }
    # Открыть файл с заданным именем и прочитать его содержимое.
    with open(file_name, "r") as file_in:
        data = json.load(file_in)  # Прочитать данные из файла
    try:
        # Валидация
        validate(instance=data, schema=schema)
        print("JSON валиден по схеме.")
        return data
    except ValidationError as e:
        print(f"Ошибка валидации: {e.message}")
        return []
def main(command_line=None):
    """
    Главная функция программы.
    """
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-d",
        "--data",
        action="store",
        required=False,
        help="The data file name",
    )
    parser = argparse.ArgumentParser("routes")
    parser.add_argument(
        "--version", action="version", version="%(prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")
    add = subparsers.add_parser(
        "add", parents=[file_parser], help="Add a new route"
    )
    add.add_argument(
        "-s", "--start", action="store", required=True, help="The route start"
    )
    add.add_argument(
        "-e", "--end", action="store", required=True, help="The route endpoint"
    )
    add.add_argument(
        "-n",
        "--number",
        action="store",
        type=int,
        required=True,
```

```
        help="The number of route",
    )
    _ = subparsers.add_parser(
        "list", parents=[file_parser], help="Display all routes"
    )
    select = subparsers.add_parser(
        "select", parents=[file_parser], help="Select the routes"
    )
    select.add_argument(
        "-p",
        "--point",
        action="store",
        required=True,
        help="Routes starting or ending at this point",
    )
    args = parser.parse_args(command_line)
    data_file = args.data
    if not data_file:
        data_file = os.environ.get("WORKERS_DATA_IND")
    if not data_file:
        print("The data file name is absent", file=sys.stderr)
        sys.exit(1)
    # Загрузить всех работников из файла, если файл существует.
    is_dirty = False
    if os.path.exists(data_file):
        routes = load_routes(data_file)
    else:
        routes = []
    match args.command.lower():
        case "add":
            routes = add_route(routes, args.start, args.end, args.number)
            is_dirty = True
        case "list":
            display_routes(routes)
        case "select":
            name_point = args.point.lower()
            selected = select_routes(routes, name_point)
            display_routes(selected)
    if is_dirty:
        save_routes(data_file, routes)
if __name__ == "__main__":
    main()
```

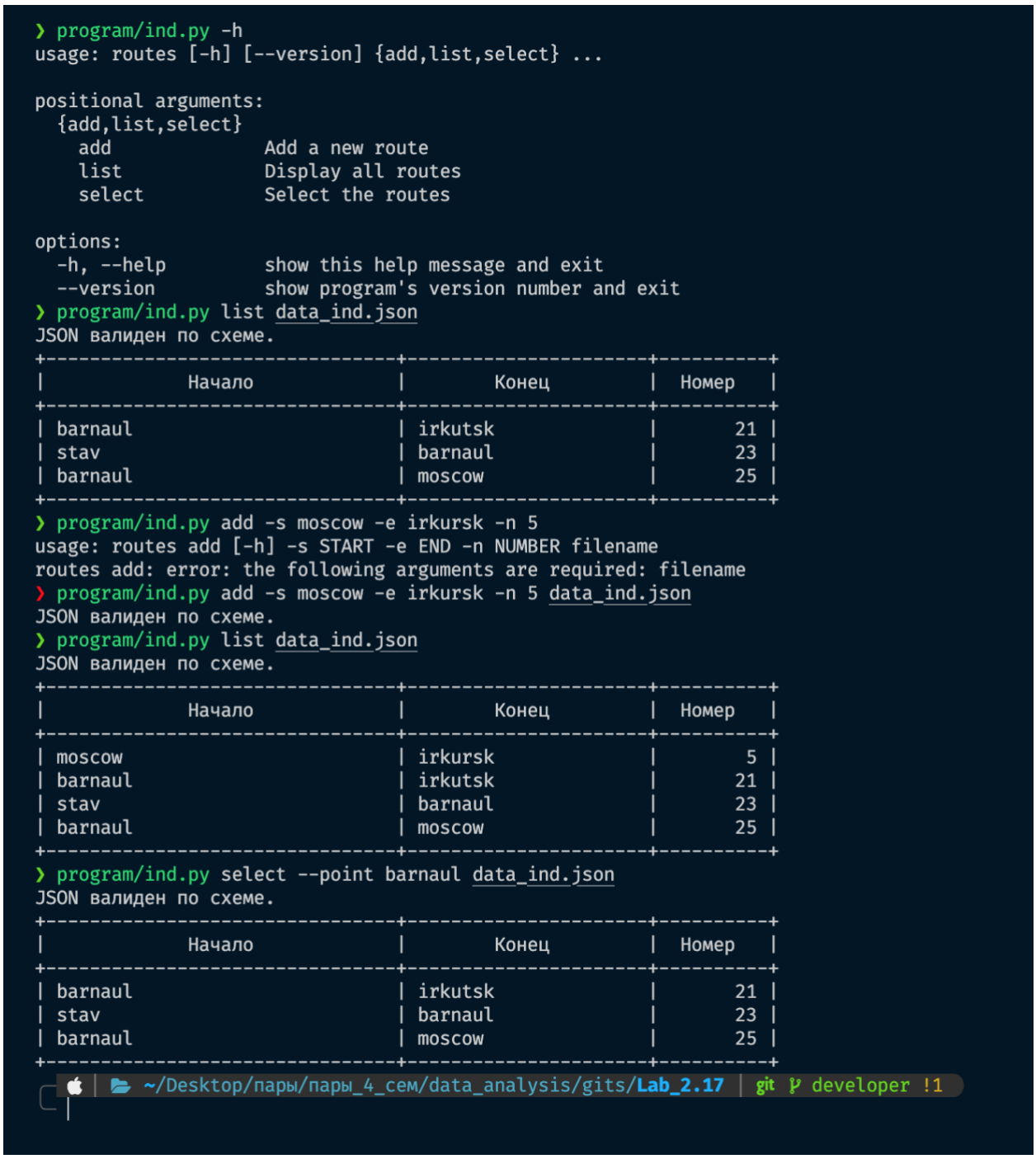


Рисунок 3. Ввод, вывод и выборка маршрутов

4. Выполнил задание повышенной сложности: Самостоятельно изучите работу с пакетом click для построения интерфейса командной строки (CLI). Для своего варианта лабораторной работы 2.16 необходимо реализовать интерфейс командной строки с использованием пакета click .

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import bisect
import json
```

```
import os
import click
from jsonschema import ValidationError, validate
def add_route(routes, start, end, number):
    """
    Добавить данные о маршруте.
    """
    is_dirty = False
    route = {
        "начальный пункт": start.lower(),
        "конечный пункт": end.lower(),
        "номер маршрута": number,
    }
    if route not in routes:
        bisect.insort(
            routes,
            route,
            key=lambda item: item.get("номер маршрута"),
        )
        is_dirty = True
    else:
        print("Данный маршрут уже добавлен.")
    return routes, is_dirty
def display_routes(routes):
    """
    Отобразить список маршрутов.
    """
    if routes:
        line = "+-{}-+-{}-+-{}-+".format("-" * 30, "-" * 20, "-" * 8)
        print(line)
        print("| {:^30} | {:^20} | {:^8} |".format("Начало", "Конец", "Номер"))
        print(line)
        for route in routes:
            print(
                "| {:<30} | {:<20} | {:>8} |".format(
                    route.get("начальный пункт", ""),
                    route.get("конечный пункт", ""),
                    route.get("номер маршрута", ""),
                )
            )
        print(line)
    else:
        print("Список маршрутов пуст.")
def select_routes(routes, name_point):
    """
```

```
    Выбрать маршруты с заданным пунктом отправления или прибытия.
    """
    selected = []
    for route in routes:
        if (
            route["начальный пункт"] == name_point
            or route["конечный пункт"] == name_point
        ):
            selected.append(route)

    return selected
def save_routes(file_name, routes):
    """
    Сохранить все маршруты в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w") as file_out:
        # Записать данные из словаря в формат JSON и сохранить их
        # в открытый файл.
        json.dump(routes, file_out, ensure_ascii=False, indent=4)
def load_routes(file_name):
    """
    Загрузить все маршруты из файла JSON.
    """
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "начальный пункт": {"type": "string"},
                "конечный пункт": {"type": "string"},
                "номер маршрута": {"type": "integer"},
            },
            "required": [
                "начальный пункт",
                "конечный пункт",
                "номер маршрута",
            ],
        },
    }
    if not os.path.exists(file_name):
        return []
    # Открыть файл с заданным именем и прочитать его содержимое.
    with open(file_name, "r") as file_in:
        data = json.load(file_in)  # Прочитать данные из файла
```

```
    try:
        # Валидация
        validate(instance=data, schema=schema)
        print("JSON валиден по схеме.")
        return data
    except ValidationError as e:
        print(f"Ошибка валидации: {e.message}")
        return []
@click.group()
def command():
    pass
@command.command()
@click.argument("filename")
@click.option("-s", "--start", required=True, help="The route start")
@click.option("-e", "--end", required=True, help="The route endpoint")
@click.option(
    "-n", "--number", required=True, type=int, help="The number of route"
)
def add(filename, start, end, number):
    """
    Add a new route.
    """
    routes = load_routes(filename)

    routes, is_dirty = add_route(routes, start.lower(), end.lower(), number)
    if is_dirty:
        save_routes(filename, routes)
@command.command()
@click.argument("filename")
@click.option(
    "-p",
    "--point",
    required=True,
    help="Routes starting or ending at this point",
)
def select(filename, point):
    """
    Select the routes
    """
    point = point.lower()
    routes = load_routes(filename)
    selected_routes = select_routes(routes, point)
    display_routes(selected_routes)
@command.command()
```

```
@click.argument("filename")
def display(filename):
    """
    Display all routes
    """
    routes = load_routes(filename)
    display_routes(routes)
if __name__ == "__main__":
    command()
```

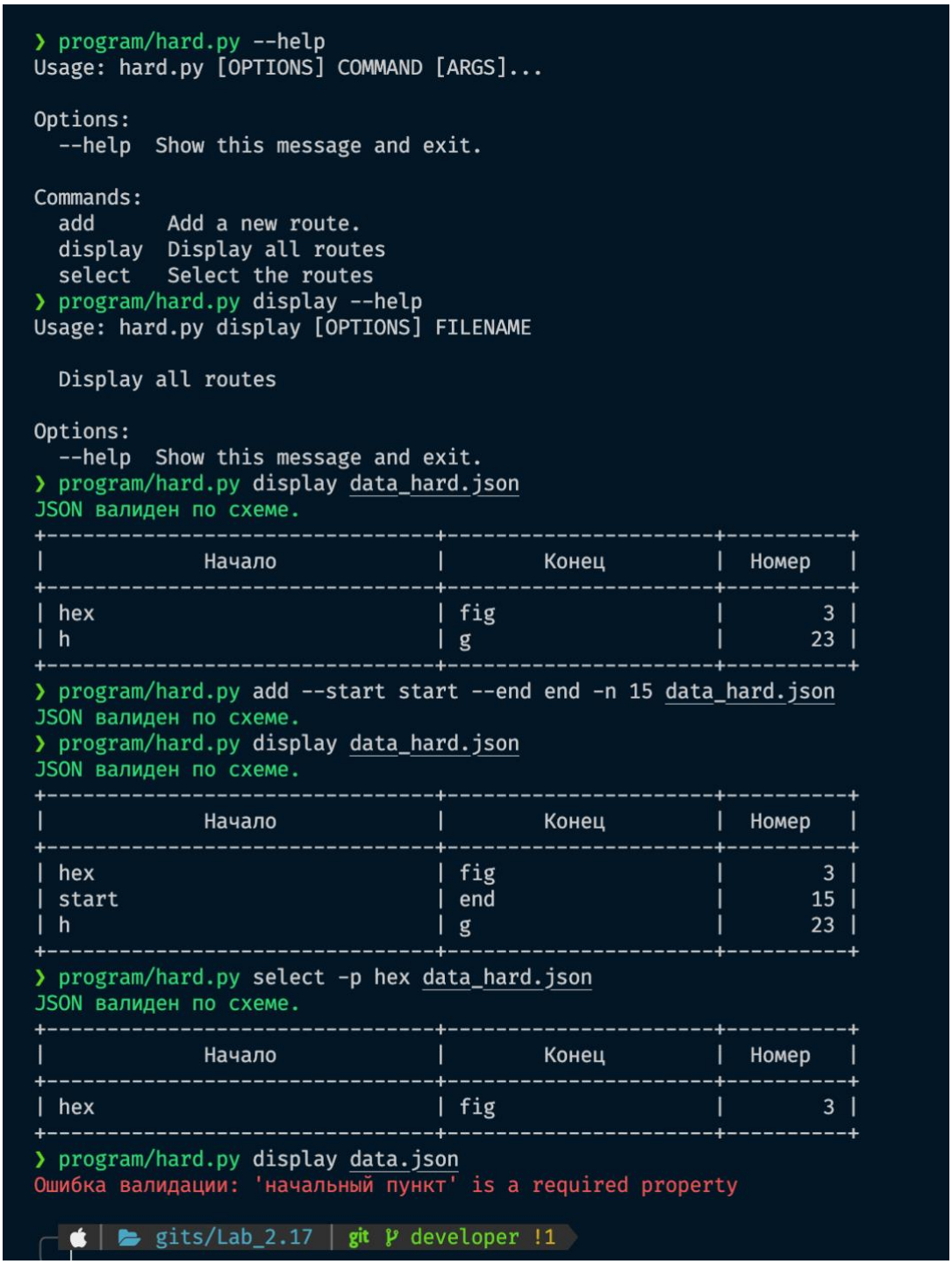


Рисунок 4. Ввод, вывод и выборка маршрутов

Ответы на контрольные вопросы:

1. Каково назначение переменных окружения?

Переменные окружения используются для передачи информации процессам, которые запущены в оболочке.

2. Какая информация может храниться в переменных окружения?

Переменные среды хранят информацию о среде операционной системы.

Эта информация включает такие сведения, как путь к операционной системе, количество процессоров, используемых операционной системой, и расположение временных папок.

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

Нужно открыть окно свойства системы и нажать на кнопку “Переменные среды”.

4. Каково назначение переменных PATH и PATHEXT?

PATH позволяет запускать исполняемые файлы и скрипты, «лежащие» в определенных каталогах, без указания их точного местоположения.

PATHEXT дает возможность не указывать даже расширение файла, если оно прописано в ее значениях.

5. Как создать или изменить переменную окружения в Windows?

В окне “Переменные среды” нужно нажать на кнопку “Создать”, затем ввести имя переменной и путь.

6. Что представляют собой переменные окружения в ОС Linux?

Переменные окружения в Linux представляют собой набор именованных значений, используемых другими приложениями.

7. В чем отличие переменных окружения от переменных оболочки?

Переменные окружения (или «переменные среды») – это переменные, доступные в масштабах всей системы и наследуемые всеми дочерними процессами и оболочками.

Переменные оболочки — это переменные, которые применяются только к текущему экземпляру оболочки. Каждая оболочка, например, bash или zsh, имеет свой собственный набор внутренних переменных.

8. Как вывести значение переменной окружения в Linux?

Наиболее часто используемая команда для вывода переменных окружения – printenv.

9. Какие переменные окружения Linux Вам известны?

USER — текущий пользователь.

PWD – текущая директория.

HOME – домашняя директория текущего пользователя.

SHELL – путь к оболочке текущего пользователя.

EDITOR – заданный по умолчанию редактор. Этот редактор будет вызываться в ответ на команду edit.

LOGNAME – имя пользователя, используемое для входа в систему.

PATH – пути к каталогам, в которых будет производиться поиск вызываемых команд. При выполнении команды система будет проходить по данным каталогам в указанном порядке и выберет первый из них, в котором будет находиться исполняемый файл искомой команды.

LANG – текущие настройки языка и кодировки.

TERM – тип текущего эмулятора терминала.

MAIL – место хранения почты текущего пользователя.

LS_COLORS задает цвета, используемые для выделения объектов.

10. Какие переменные оболочки Linux Вам известны?

BASHOPTS – список задействованных параметров оболочки, разделенных двоеточием.

BASH_VERSION – версия запущенной оболочки bash.

COLUMNS – количество столбцов, которые используются для отображения выходных данных.

HISTFILESIZE – максимальное количество строк для файла истории команд.

HISTSIZE – количество строк из файла истории команд, которые можно хранить в памяти.

HOSTNAME – имя текущего хоста.

IFS – внутренний разделитель поля в командной строке.

PS1 – определяет внешний вид строки приглашения ввода новых команд.

PS2 – вторичная строка приглашения.

UID – идентификатор текущего пользователя.

11. Как установить переменные оболочки в Linux?

$ NEW_VAR='значение'

12. Как установить переменные окружения в Linux?

Команда export используется для задания переменных окружения.

С помощью данной команды мы экспортируем указанную переменную, в результате чего она будет видна во всех вновь запускаемых дочерних командных оболочках.

13. Для чего необходимо делать переменные окружения Linux постоянными?

Чтобы переменная сохранялась после закрытия сеанса оболочки.

14. Для чего используется переменная окружения PYTHONHOME? Переменная среды PYTHONHOME изменяет расположение стандартных библиотек Python.

15. Для чего используется переменная окружения PYTHONPATH?

Переменная среды PYTHONPATH изменяет путь поиска по умолчанию для файлов модуля.

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

PYTHONSTARTUP PYTHONOPTIMIZE PYTHONBREAKPOINT PYTHONDEBUG PYTHONINSPECT PYTHONUNBUFFERED PYTHONVERBOSE PYTHONCASEOK PYTHONDONTWRITEBYTECODE PYTHONPYCACHEPREFIX PYTHONHASHSEED PYTHONIOENCODING PYTHONNOUSERSITE PYTHONUSERBASE PYTHONWARNINGS PYTHONFAULTHANDLER

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

value = os.environ.get('MY_ENV_VARIABLE')

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

if os.environ[key_value]:

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

Для присвоения значения любой переменной среды используется функция os.environ.setdefault(«Переменная», «Значение»)

Вывод: в результате выполнения работы были приобретены навыки по работе с переменными окружения с помощью языка программирования Python версии 3.x.