

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 №5
дисциплины «Программирование на языке Python»

Вариант_15_

Выполнила:
Маньшина Дарья Алексеевна
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 кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

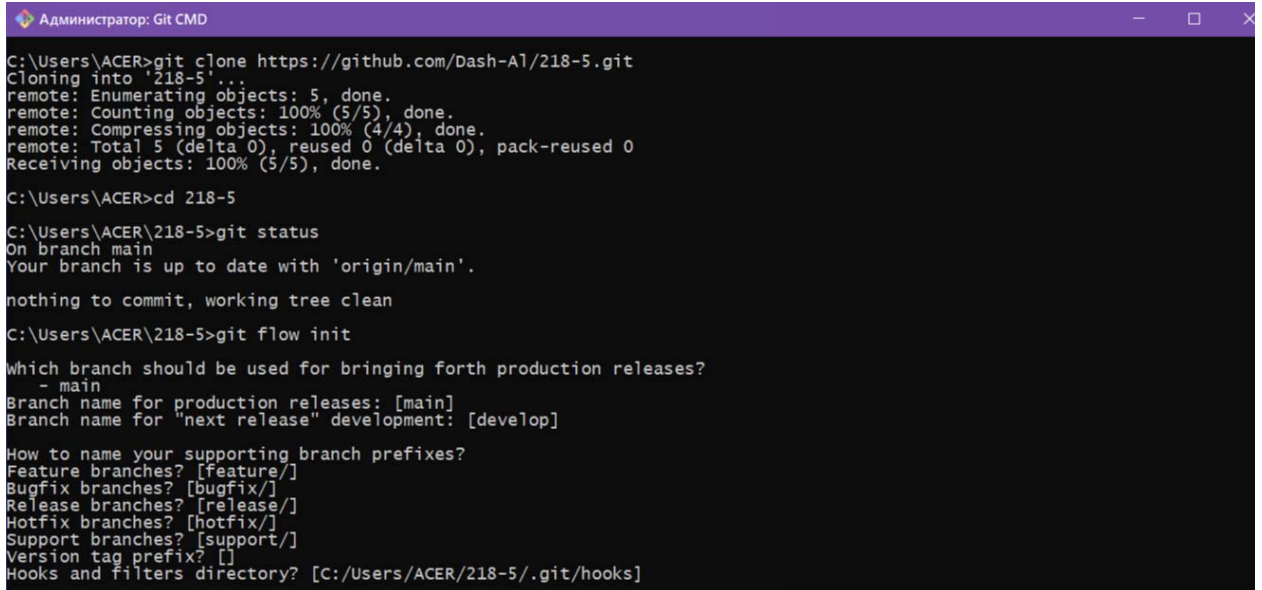
Ставрополь, 2023 г.

Тема: работа с переменными окружения в Python3.

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

Ход работы:

1. Подготовка к выполнению работы. Клонирование репозитория и добавление пакетов black, isort, flake8.

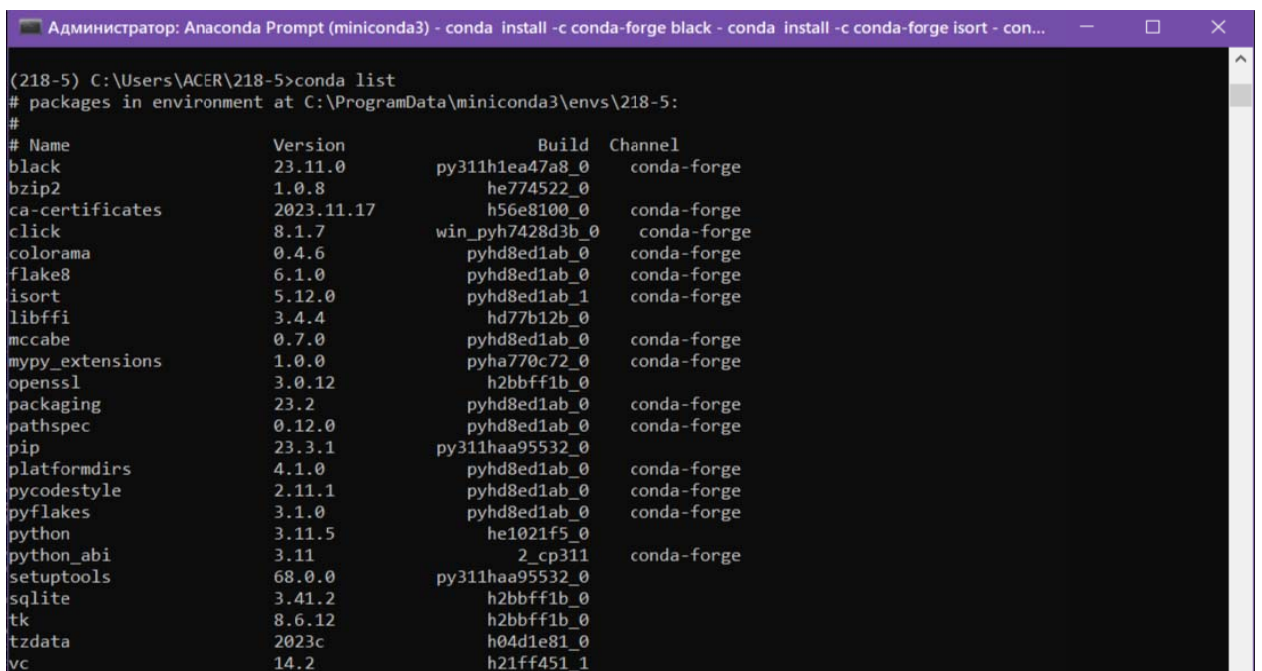


```
Администратор: Git CMD
C:\Users\ACER>git clone https://github.com/Dash-A1/218-5.git
Cloning into '218-5'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\ACER>cd 218-5
C:\Users\ACER\218-5>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
C:\Users\ACER\218-5>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? [C:/Users/ACER/218-5/.git/hooks]
```

Рисунок 1. Клонирование репозитория



```
Администратор: Anaconda Prompt (miniconda3) - conda install -c conda-forge black - conda install -c conda-forge isort - con...
(218-5) C:\Users\ACER\218-5>conda list
# packages in environment at C:\ProgramData\miniconda3\envs\218-5:
#
# Name                        Version      Build                Channel
black                         23.11.0      py311h1ea47a8_0      conda-forge
bzip2                         1.0.8        he774522_0           conda-forge
ca-certificates              2023.11.17   h56e8100_0           conda-forge
click                         8.1.7        win_pyh7428d3b_0     conda-forge
colorama                     0.4.6        pyhd8ed1ab_0         conda-forge
flake8                       6.1.0        pyhd8ed1ab_0         conda-forge
isort                        5.12.0       pyhd8ed1ab_1         conda-forge
libffi                       3.4.4        hd77b12b_0           conda-forge
mccabe                       0.7.0        pyhd8ed1ab_0         conda-forge
mypy_extensions              1.0.0        pyha770c72_0         conda-forge
openssl                      3.0.12       h2bbff1b_0           conda-forge
packaging                    23.2         pyhd8ed1ab_0         conda-forge
pathspec                     0.12.0       pyhd8ed1ab_0         conda-forge
pip                          23.3.1       py311haa95532_0      conda-forge
platformdirs                 4.1.0        pyhd8ed1ab_0         conda-forge
pycodestyle                  2.11.1       pyhd8ed1ab_0         conda-forge
pyflakes                     3.1.0        pyhd8ed1ab_0         conda-forge
python                       3.11.5       he1021f5_0           conda-forge
python_abi                   3.11         2_cp311              conda-forge
setuptools                   68.0.0       py311haa95532_0      conda-forge
sqlite                       3.41.2       h2bbff1b_0           conda-forge
tk                           8.6.12       h2bbff1b_0           conda-forge
tzdata                       2023c        h04d1e81_0           conda-forge
vc                           14.2         h21ff451_1           conda-forge
```

Рисунок 2. Установка пакетов

2. Проработка примеров из методички.

Считываем одну или все переменные окружения

Программа:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Импортируем модуль os
import os
# Создаём цикл, чтобы вывести все переменные среды
print("The keys and values of all environment variables:")
for key in os.environ:
    print(key, '=>', os.environ[key])
# Выводим значение одной переменной
print("The value of HOME is: ", os.environ['HOME'])
```

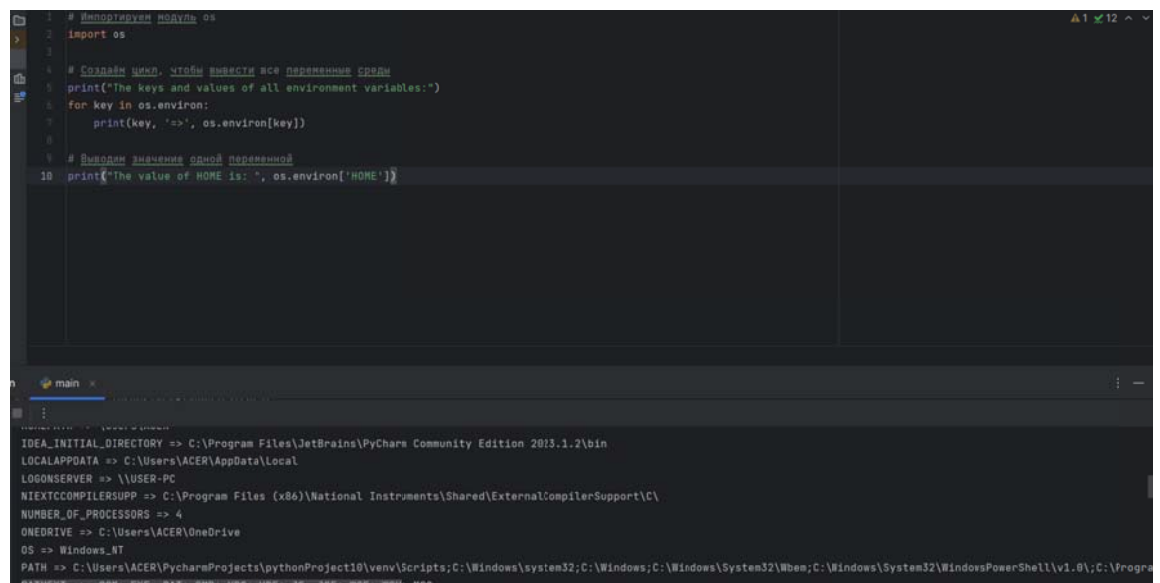


Рисунок 3. Результат программы

Проверяем, присвоено ли значение переменной окружения

Программа:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Импортируем модуль os
import os
# Импортируем модуль sys
import sys
while True:
    # Принимаем имя переменной среды
    key_value = input("Enter the key of the environment variable:")
    # Проверяем, инициализирована ли переменная
    try:
        if os.environ[key_value]:
```

```

print(
    "The value of",
    key_value,
    " is ",
    os.environ[key_value]
)

# Если переменной не присвоено значение, то ошибка
except KeyError:
    print(key_value, 'environment variable is not set.')

# Завершаем процесс выполнения скрипта
sys.exit(1)

```

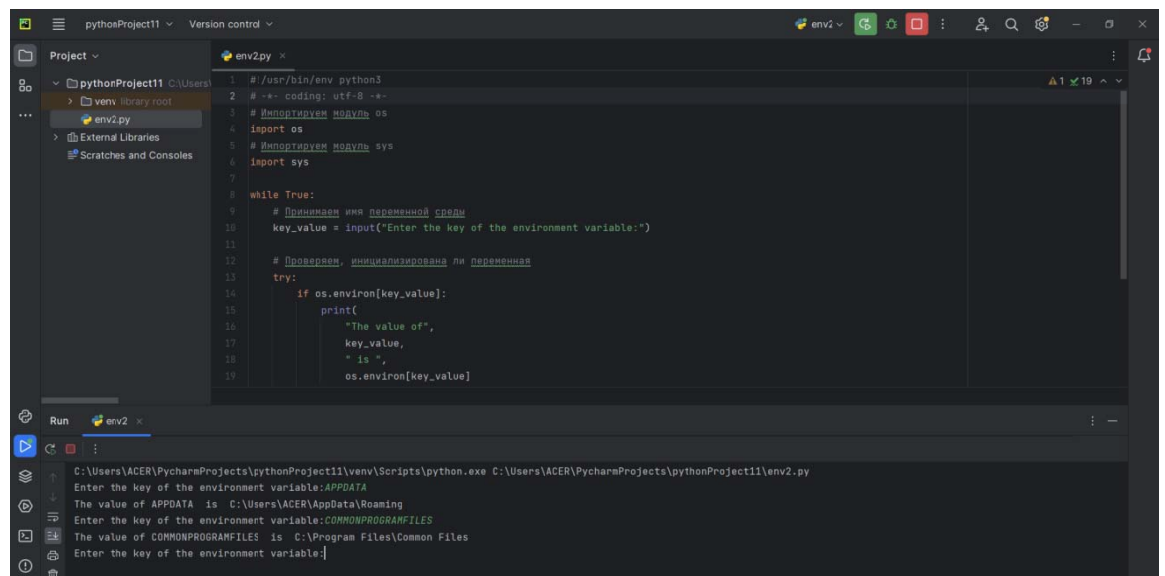


Рисунок 4. Результат программы

Проверяем переменную на истинность

Программа:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Импортируем модуль os
import os
# Проверяем значение переменной среды
if os.environ.get('DEBUG') == 'True':
    print('Debug mode is on')
else:
    print('Debug mode is off')

```

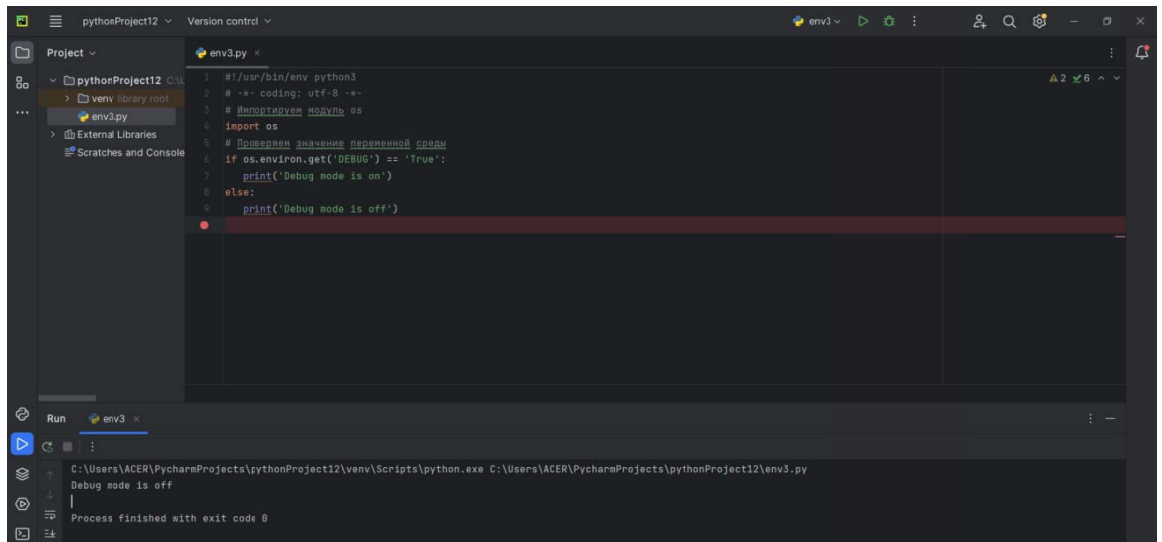


Рисунок 5. Результат программы

Присваиваем значение переменной окружения

Программа:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Импортируем модуль os
import os
# Задаём значение переменной DEBUG
os.environ.setdefault('DEBUG', 'True')
# Проверяем значение переменной
if os.environ.get('DEBUG') == 'True':
    print('Debug mode is on')
else:
    print('Debug mode is off')

```

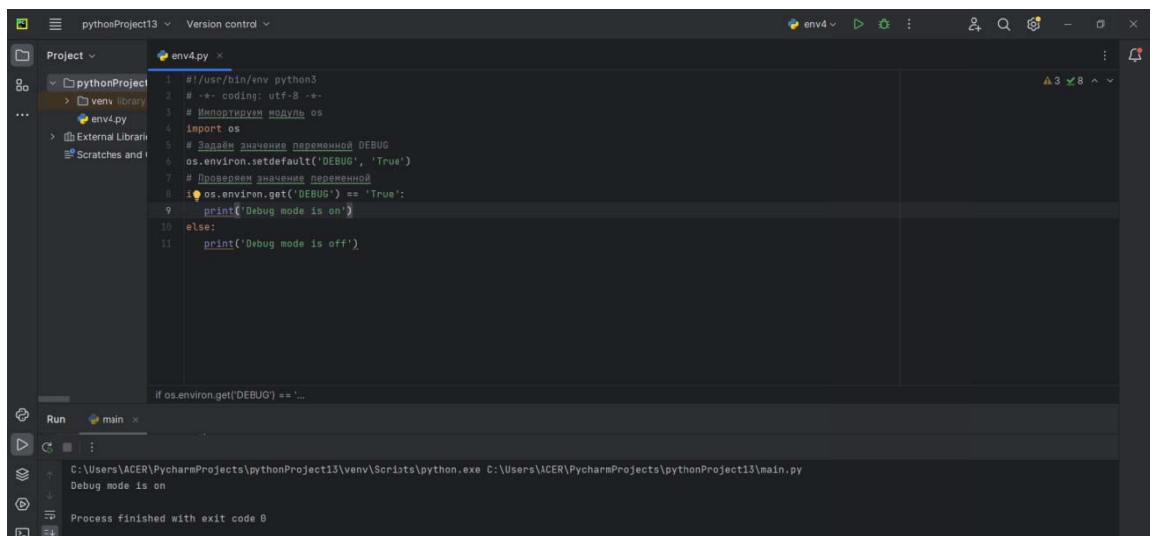


Рисунок 6. Результат программы

Пример 1. Для примера 1 лабораторной работы 2.17 добавим возможность получения имени файла данных, используя соответствующую переменную окружения.

Программа:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import argparse
import json
import os
import sys
from datetime import date

def add_worker(staff, name, post, year):
    staff.append(
        {
            "name": name,
            "post": post,
            "year": year
        }
    )

    return staff

def display_workers(staff):
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
```

```

)
print(line)
else:
print("Список работников пуст.")

def select_workers(staff, period):

# Получить текущую дату.
today = date.today()

# Сформировать список работников.
result = []
for employee in staff:
if today.year - employee.get('year', today.year) >= period:
result.append(employee)
# Возвратить список выбранных работников.
return result

def save_workers(file_name, staff):
# Открыть файл с заданным именем для записи.
with open(file_name, "w", encoding="utf-8") as fout:
# Выполнить сериализацию данных в формат JSON.
# Для поддержки кириллицы установим ensure_ascii=False
json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_workers(file_name):
# Открыть файл с заданным именем для чтения.
with open(file_name, "r", encoding="utf-8") as fin:
return json.load(fin)

def main(command_line=None):
# Создать родительский парсер для определения имени файла.
file_parser = argparse.ArgumentParser(add_help=False)
file_parser.add_argument(
"-d",
"--data",
action="store",
required=False,
help="The data file name"
)
# Создать основной парсер командной строки.
parser = argparse.ArgumentParser("workers")
parser.add_argument(
"--version",
action="version",
version="%s 0.1.0"
)

subparsers = parser.add_subparsers(dest="command")
# Создать субпарсер для добавления работника.
add = subparsers.add_parser(
"add",
parents=[file_parser],
help="Add a new worker"

```

```

)
add.add_argument(
    "-n",
    "--name",
    action="store",
    required=True,
    help="The worker's name"
)
add.add_argument(
    "-p",
    "--post",
    action="store",
    help="The worker's post"
)
add.add_argument(
    "-y",
    "--year",
    action="store",
    type=int,
    required=True,
    help="The year of hiring"
)
# Создать субпарсер для отображения всех работников.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all workers"
)
# Создать субпарсер для выбора работников.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the workers"
)
select.add_argument(
    "-p",
    "--period",
    action="store",
    type=int,
    required=True,
    help="The required period"
)

# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)
# Получить имя файла.
data_file = args.data
if not data_file:
    data_file = os.environ.get("WORKERS_DATA")
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):
    workers = load_workers(data_file)

```



```

else:
workers = []
# Добавить работника.
if args.command == "add":
workers = add_worker(
workers,
args.name,
args.post,
args.year
)
is_dirty = True

# Отобразить всех работников.
elif args.command == "display":
display_workers(workers)

# Выбрать требуемых работников.
elif args.command == "select":
selected = select_workers(workers, args.period)
display_workers(selected)

# Сохранить данные в файл, если список работников был изменен.
if is_dirty:
save_workers(data_file, workers)

if __name__ == "__main__":
main()

```

Выполним индивидуальное задание.

Условие: использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата рождения (список из трёх чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по датам рождения; вывод на экран информацию о людях, родившихся под знаком, название которого введено с клавиатуры; если таких нет, выдать на дисплей соответствующее сообщение. Оформив каждую команду в виде отдельной функции. Добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

Программа:

```

#Использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата
рождения (список из трёх чисел).
# Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в
список, состоящий из словарей заданной структуры;
# записи должны быть упорядочены по датам рождения; вывод на экран информацию о
людях, родившихся под знаком, название которого введено с клавиатуры;
# если таких нет, выдать на дисплей соответствующее сообщение. Оформив каждую
команду в виде отдельной функции.

```

Добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

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

import os
def input_data():
    data = []
    while True:
        surname = input("Введите фамилию: ")
        name = input("Введите имя: ")
        zodiac = input("Введите знак Зодиака: ")
        birthday = input("Введите дату рождения (через пробел: ").split()
        if len(birthday) != 3:
            print("Неверный формат даты. Повторите ввод.")
            continue
        try:
            birthday = [int(x) for x in birthday]
        except ValueError:
            print("Неверный формат даты. Повторите ввод.")
            continue
        data.append({
            "фамилия": surname,
            "имя": name,
            "знак Зодиака": zodiac,
            "дата рождения": birthday
        })
        if input("Желаете добавить еще запись? (y/n): ") != 'y':
            break
    data.sort(key=lambda x: x["дата рождения"])
    return data

def find_people_by_zodiac(data, zodiac):
    people = []
    for person in data:
        if person["знак Зодиака"] == zodiac:
            people.append(person)
    return people

def print_people(people):
    if len(people) == 0:
        print("Нет людей с таким знаком Зодиака.")
    else:
        for person in people:
            print("Фамилия: {}".format(person["фамилия"]))
            print("Имя: {}".format(person["имя"]))
            print("Знак Зодиака: {}".format(person["знак Зодиака"]))
            print("Дата рождения: {}/{}/{}".format(person["дата рождения"][0], person["дата
рождения"][1],
                                                    person["дата рождения"][2]))
            print()

def main():
```

```

filename = os.environ.get("DATA_FILE")
if filename:
    with open(filename, "r") as file:
        data = eval(file.read())
else:
    data = input_data()

zodiac = input("Введите знак Зодиака для поиска: ")
people = find_people_by_zodiac(data, zodiac)
print_people(people)

if __name__ == "__main__":
    main()

```

Результат:

```

pythonProject15  Version control
IDZ.py
1  #Использовать словарь, содержащий следующие ключи: фамилия, имя, знак Зодиака; дата рождения (список из трёх чисел).
2  # Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из элементов заданной структуры;
3  # запись должна быть упорядочена по дате рождения; вывод на экран информации о людях, выделенных под знаком, название которого введено с клавиатуры;
4  # если таких нет, выдать на дисплей соответствующее сообщение. Оформить каждую команду в виде отдельной функции.
5  # Добавить возможность получения имени файла данных, используя соответствующую переменную окружения.
6
7  #!/usr/bin/env python3
8  # -*- coding: utf-8 -*-
9
10 import os
11
12 def input_data():
13     data = []
14     while True:
15         surname = input("Введите фамилию: ")
16         name = input("Введите имя: ")
17         zodiac = input("Введите знак Зодиака: ")
18         birthday = input("Введите дату рождения (через пробел): ").split()
19         if len(birthday) != 3:
20             continue
21         data.append((surname, name, zodiac, birthday))
22         y_n = input("Желаете добавить еще запись? (y/n): ")
23         if y_n.lower() != 'y':
24             break
25     return data
26
27 if __name__ == '__main__':
28     data = input_data()
29     print(data)

```

main

Введите знак Зодиака: Дмитриев
Введите дату рождения (через пробел): 30 02 5000
Желаете добавить еще запись? (y/n): y
Введите знак Зодиака для поиска: Лев
Фамилия: Петр
Имя: Петр
Знак Зодиака: Лев
Дата рождения: 17/7/2900

Рисунок 7. Результат программы индивидуального задания

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

Контрольные вопросы

1. Каково назначение переменных окружения?

Переменные окружения используются для хранения информации, которая может использоваться различными программами или компонентами системы. Они используются для определения путей к исполняемым файлам, библиотекам, каталогам и другим ресурсам, которые требуются программам для выполнения своих задач.

2. Какая информация может храниться в переменных окружения?

Информация, хранящаяся в переменных окружения, может включать пути к исполняемым файлам и библиотекам, идентификаторы пользователей и групп, параметры конфигурации и т.д.

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

В ОС Windows доступ к переменным окружения можно получить с помощью командной строки. Для этого нужно открыть командную строку (например, введя “cmd” в поиске меню “Пуск”) и ввести команду “set”. Это выведет список всех переменных окружения и их значений.

4. Каково назначение переменных PATH и PATHEXT?

Переменная PATH используется для указания путей к каталогам, в которых система будет искать исполняемые файлы при запуске программ. Переменная PATHEXT используется для указания расширений файлов, которые система будет рассматривать как исполняемые.

5. Как создать или изменить переменную окружения в Windows?

В Windows переменные окружения можно создать или изменить, используя панель управления. Откройте панель управления, выберите “Система”, затем “Дополнительные параметры системы”. В открывшемся окне выберите вкладку “Дополнительно”, затем нажмите на кнопку “Переменные среды”. Здесь вы можете создать или изменить переменные окружения.

6. Что представляют собой переменные окружения в ОС Linux?

В Linux переменные окружения хранятся в файле “/etc/environment”. Этот файл содержит строки вида “имя_переменной=значение”, где “имя_переменной” - это имя переменной окружения, а “значение” - соответствующее значение.

7. В чем отличие переменных окружения от переменных оболочки?

Переменные оболочки используются конкретной оболочкой (например, `bash`), в то время как переменные окружения доступны для всех программ и процессов в системе.

8. Как вывести значение переменной окружения в Linux?

Чтобы вывести значение переменной окружения, используйте команду `printenv`. Например, чтобы вывести значение переменной `PATH`, введите `printenv | grep PATH`.

9. Какие переменные окружения Linux Вам известны?

Некоторые переменные окружения Linux включают `HOME`, `PATH`, `LANG` и `USER`. Полный список переменных окружения можно найти в файле `/etc/environment`.

10. Какие переменные оболочки Linux Вам известны?

Некоторые переменные оболочки Linux включают `BASH_ENV`, `ENV`, `PS1` и `PS2`. Полный список переменных оболочки можно найти в документации по вашей оболочке (например, `man bash` для `bash`).

11. Как установить переменные оболочки в Linux?

Чтобы установить переменные оболочки, используйте команду `export`. Например, чтобы установить переменную `MYVAR` со значением `hello`, введите `export MYVAR=hello`.

12. Как установить переменные окружения в Linux?

Чтобы установить переменные окружения, отредактируйте файл `/etc/environment` и добавьте новые строки в формате

имя_переменной=значение. После сохранения изменений перезапустите систему, чтобы изменения вступили в силу.

13. Для чего необходимо делать переменные окружения Linux постоянными?

Переменные окружения Linux обычно не нужно делать постоянными, так как они предназначены для хранения временной информации, такой как пути к исполняемым файлам или настройки системы. Если вам нужно, чтобы переменная окружения была постоянной, вы можете добавить ее в файл `/etc/environment` или создать файл `.pam_environment` в домашнем каталоге пользователя.

14. Для чего используется переменная окружения PYTHONHOME?

Переменная окружения PYTHONHOME используется для указания пути к каталогу, в котором установлен интерпретатор Python. Если эта переменная не установлена, Python будет использовать каталог, указанный в переменной окружения PATH.

15. Для чего используется переменная окружения PYTHONPATH?

Переменная окружения PYTHONPATH используется для указания дополнительных путей, по которым Python будет искать модули при импорте. Если PYTHONPATH не установлена, Python использует путь, указанный в настройках проекта (если таковые имеются).

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

Кроме PYTHONHOME и PYTHONPATH, есть еще несколько переменных окружения, используемых для управления работой Python. Например, переменная LD_LIBRARY_PATH может использоваться для указания путей к библиотекам динамической компоновки. Также есть

переменные окружения, управляющие поведением интерпретатора, такие как PYTHONOPTIMIZE, PYTHONHASHSEED и PYTHONBREAKPOINT.

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

В Python чтение переменных окружения осуществляется с помощью встроенной функции `os.environ`.

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

Чтобы проверить, установлено ли значение переменной окружения, вы можете использовать оператор `in`

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

Чтобы присвоить значение переменной окружения из Python-скрипта, вы можете использовать функцию `os.putenv()`