

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа с переменными окружения в Python3»

ОТЧЕТ
по лабораторной работе №20
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

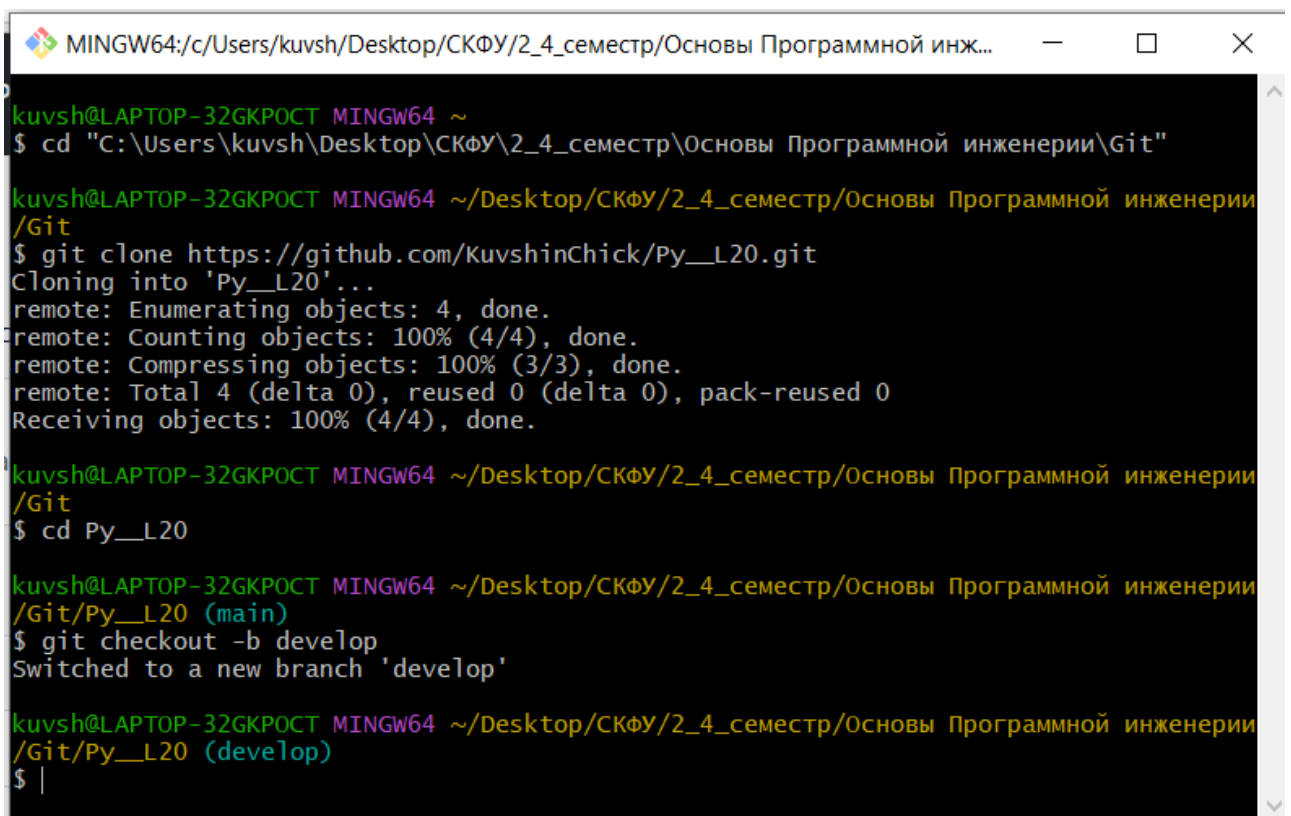
Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

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

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Основы Программной инж...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_4_семестр\Основы Программной инженерии\Git"

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git
$ git clone https://github.com/KuvshinChick/Py__L20.git
Cloning into 'Py__L20'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git
$ cd Py__L20

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py__L20 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии/Git/Py__L20 (develop)
$ |
```

Рисунок 20.1 – Клонирование репозитория и создание ветки develop

```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_4_семестр/Основы Программной инж...
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии
/Git/Py_L20 (develop)
$ git add .

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии
/Git/Py_L20 (develop)
$ git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        modified:   README.md

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии
/Git/Py_L20 (develop)
$ git commit -m "modified .gitignore & readme"
[develop 68bfe5f] modified .gitignore & readme
 2 files changed, 133 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_4_семестр/Основы Программной инженерии
/Git/Py_L20 (develop)
$
```

Рисунок 20.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.

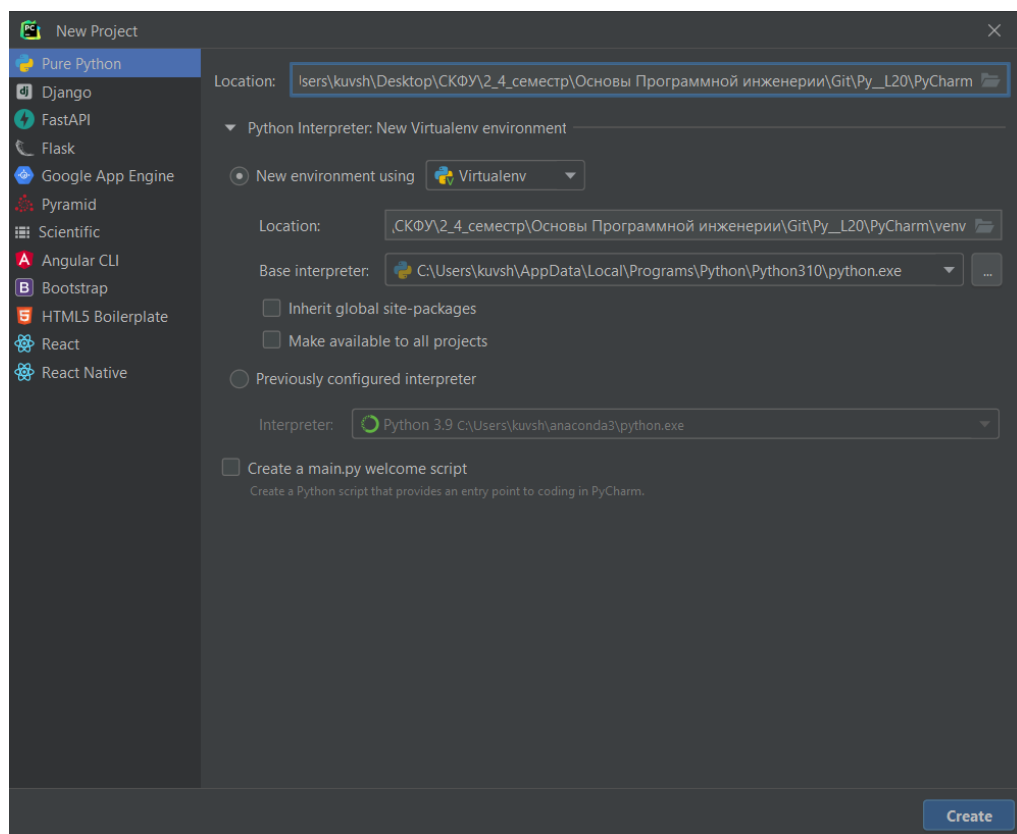


Рисунок 20.3 – Создание проекта и виртуального окружения

7. Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории.

8. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.

ex_1: Считываем одну или все переменные окружения

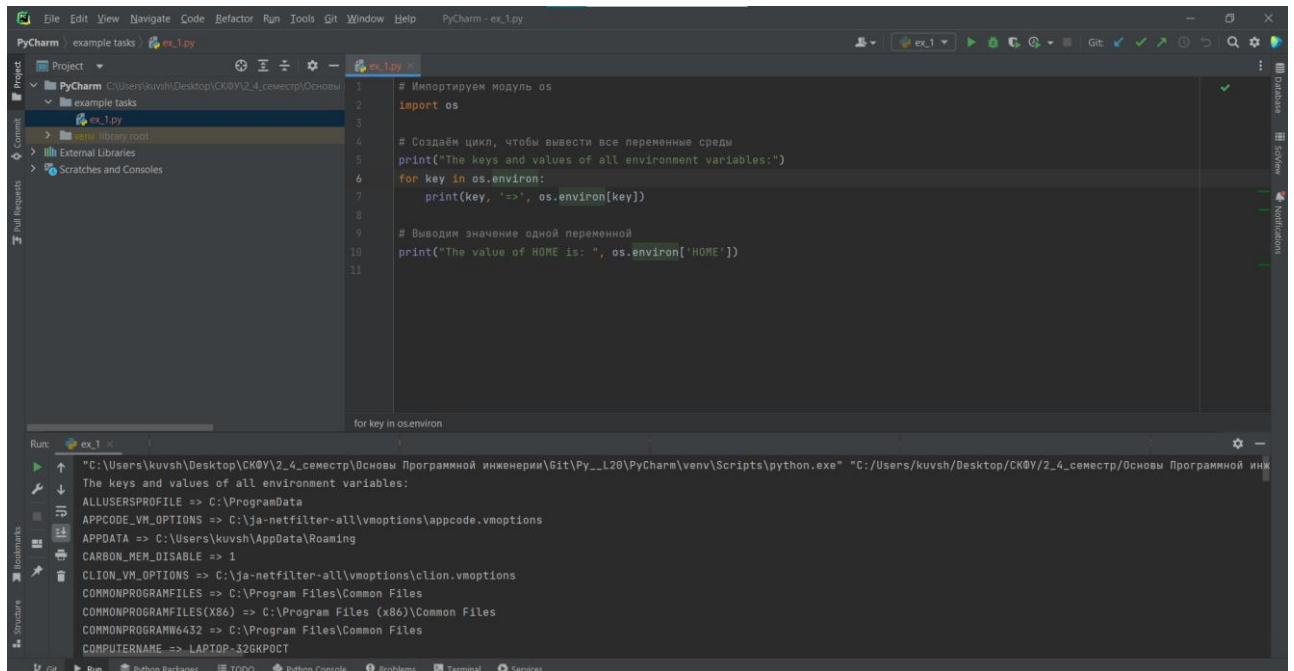


Рисунок 20.4 – Проработка и результат программы

ex_2: Проверяем, присвоено ли значение переменной окружения

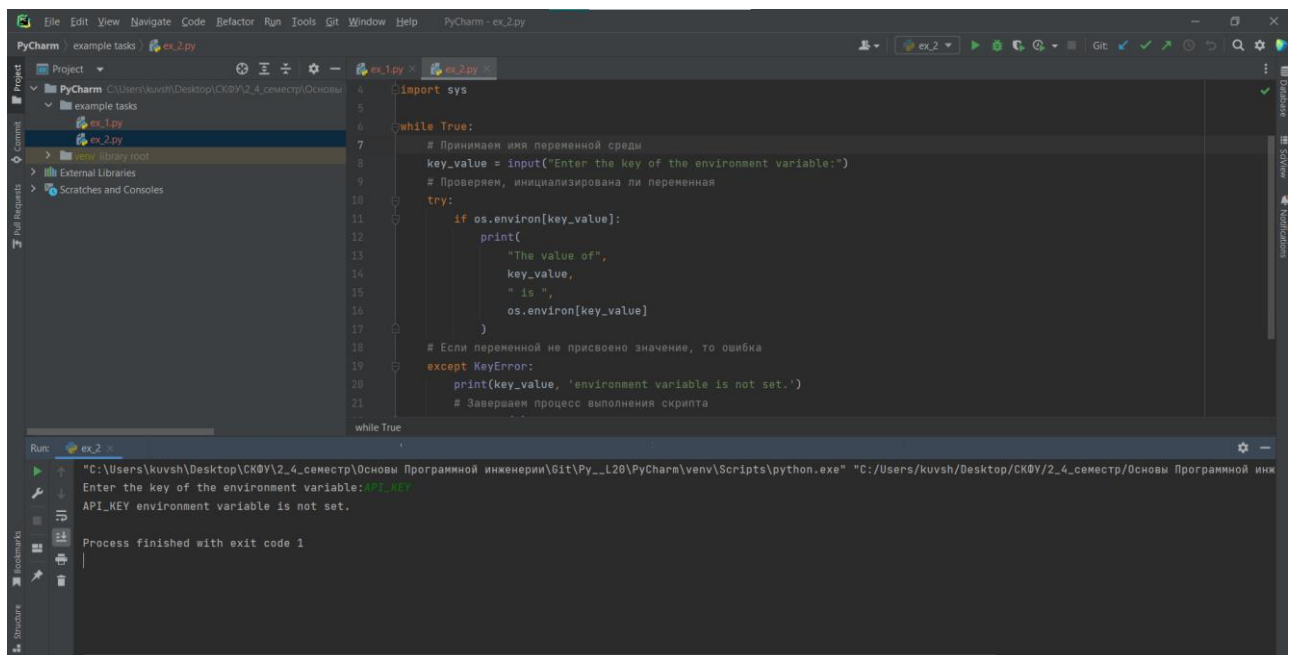


Рисунок 20.5 – Проработка и результат программы

ex_3: Проверяем переменную на истинность

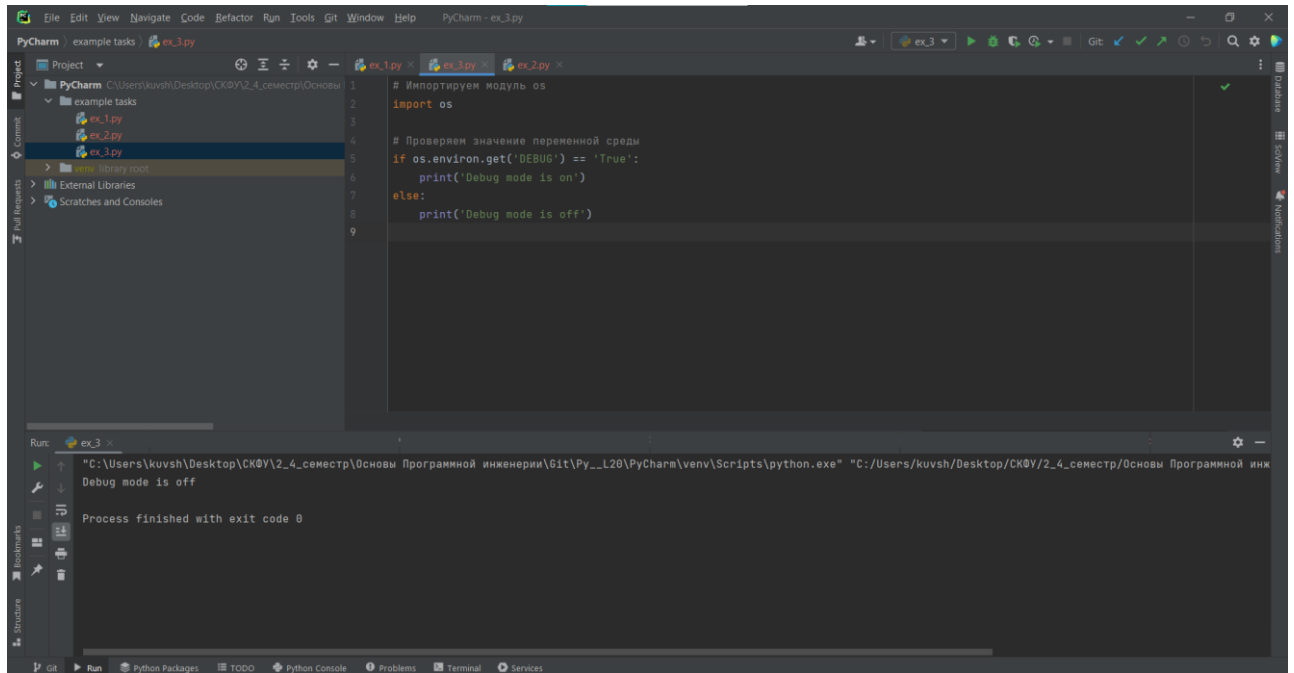


Рисунок 20.6 – Проработка и результат программы

ex_4: Присваиваем значение переменной окружения

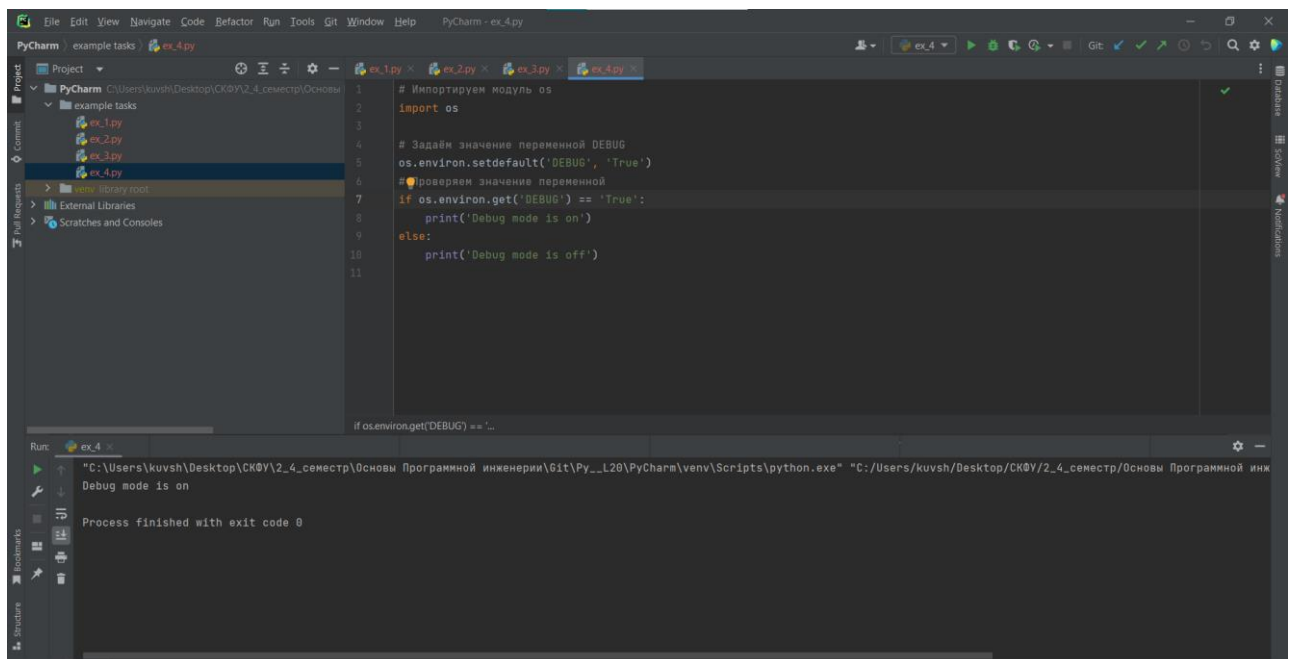


Рисунок 20.7 – Проработка и результат программы

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

ДУ > 2.4_семестр > Основы Программной инженерии > Git > Py_L20 > PyCharm > example tasks

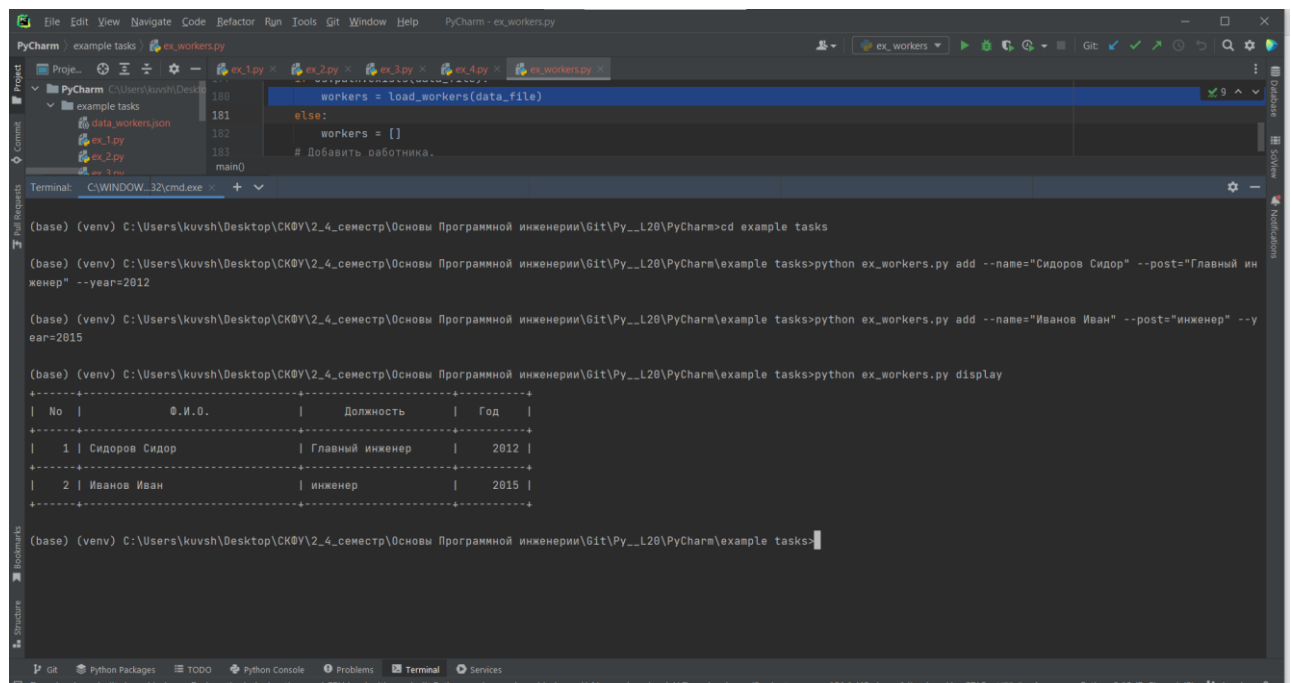
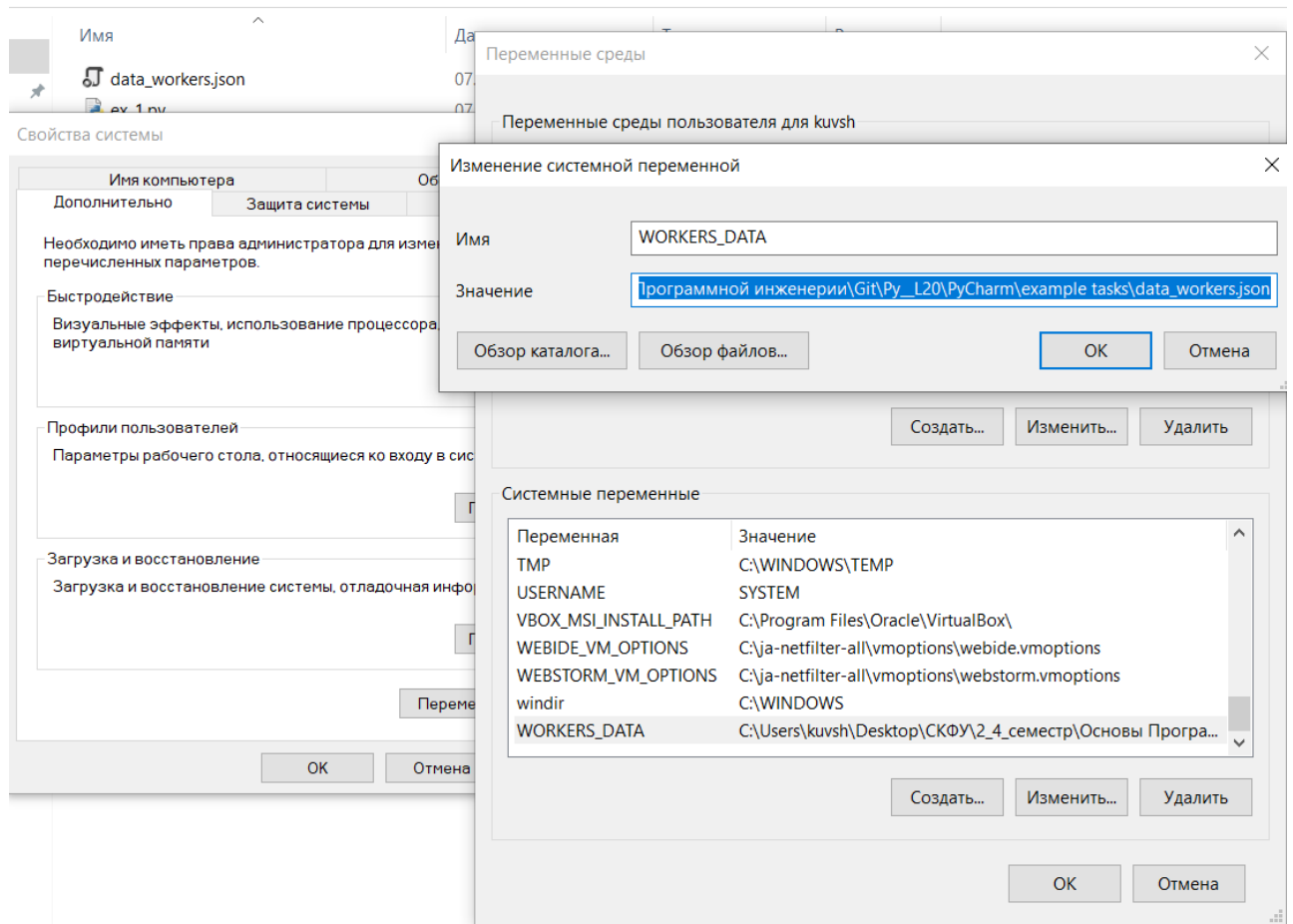


Рисунок 20.8 – Проработка и результат программы

Индивидуальное задание.

Задание 1

Для своего варианта лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответствующую переменную окружения

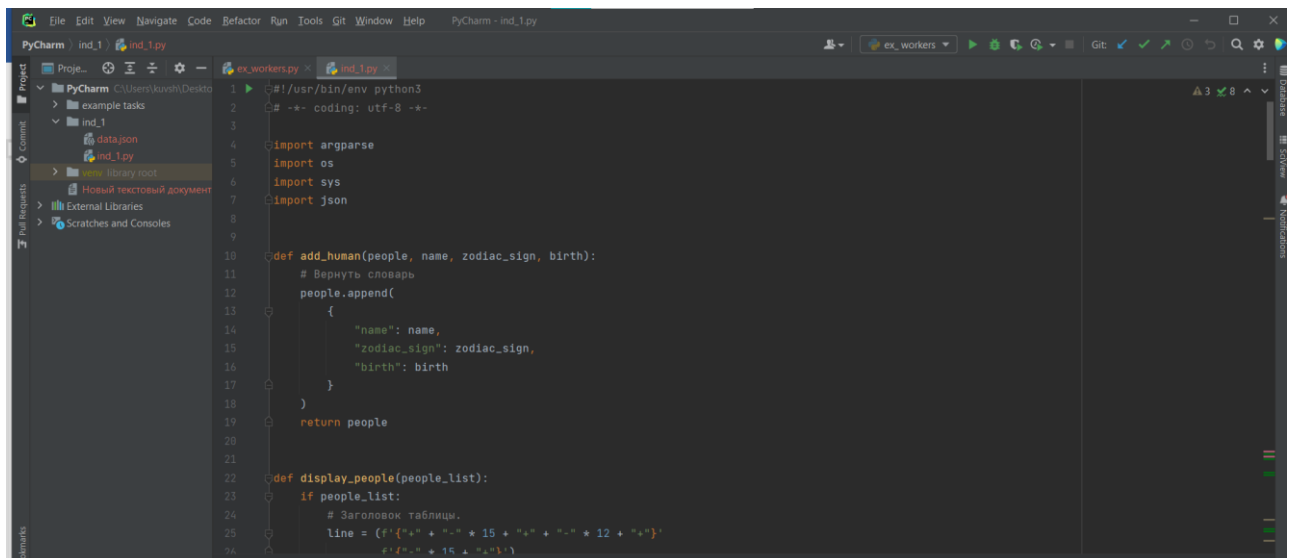
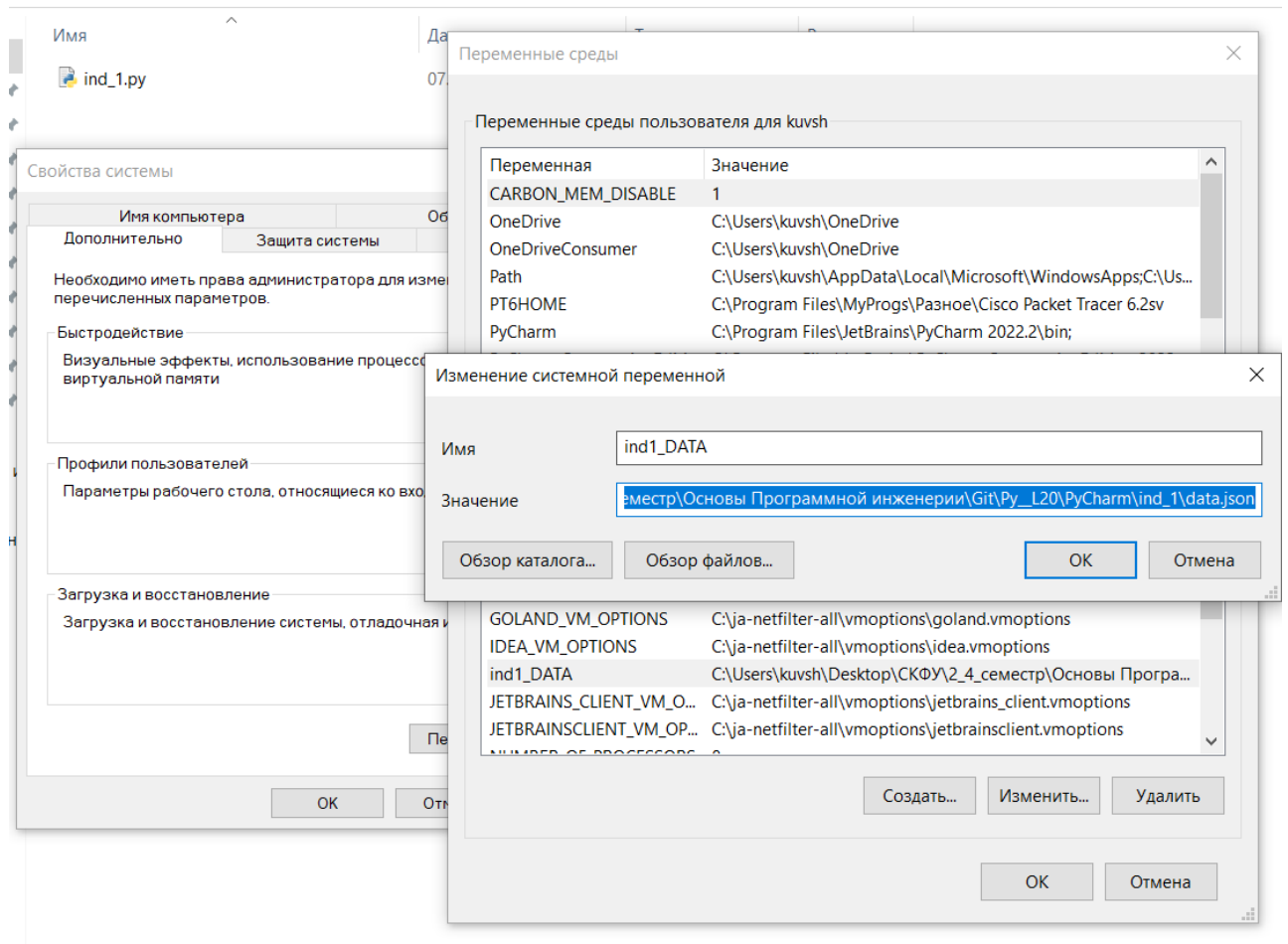


Рисунок 20.9 – Проработка индивидуального задания

```
Terminal: C:\WINDOWS\system32\cmd.exe + -
(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm>cd ind_1

(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_1>python ind_1.py display
Список пуст.

(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_1>python ind_1.py add --name="Иванов Иван" --z="овен" --b="2001.03.21"

(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_1>python ind_1.py display
+-----+-----+-----+
| Name      | Birth    | Zodiac_sign |
+-----+-----+-----+
| Иванов Иван | 2001.03.21 | овен        |
+-----+-----+-----+

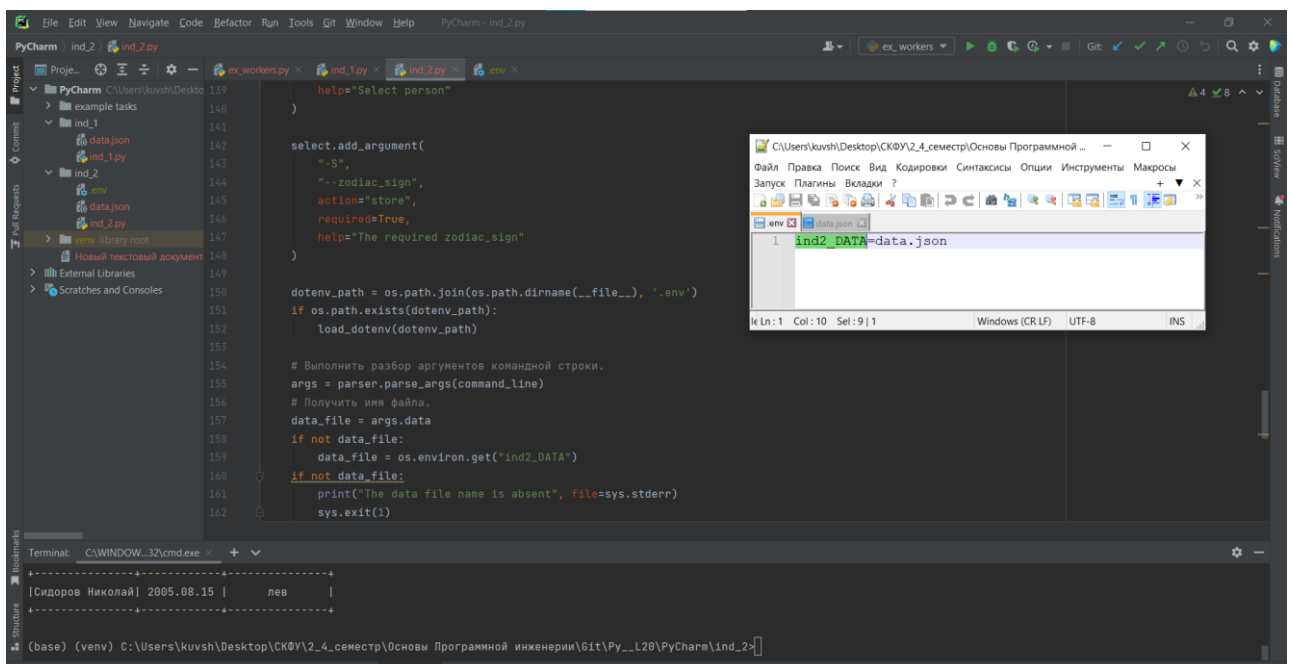
(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_1>
```

Рисунок 20.10 – Результат проработки программы

Задание 2

Самостоятельно изучите работу с пакетом `python-dotenv`.

Модифицируйте программу задания 1 таким образом, чтобы значения необходимых переменных окружения считывались из файла `.env`.



```
File Edit View Navigate Code Refactor Run Tools Git Window Help PyCharm - ind_2.py
PyCharm ind_2.py ind_1.py ind_2.py
Project: C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm
ind_1 ind_2 data.json .env
venv library root
External Libraries
Scratches and Consoles

139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162

help="Select person"
)

select.add_argument(
    "-s",
    "--zodiac_sign",
    action="store",
    required=True,
    help="The required zodiac sign"
)

dotenv_path = os.path.join(os.path.dirname(__file__), '.env')
if os.path.exists(dotenv_path):
    load_dotenv(dotenv_path)

# 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.getenv("ind2_DATA")
if not data_file:
    print("The data file name is absent", file=sys.stderr)
    sys.exit(1)

Terminal: C:\WINDOWS\system32\cmd.exe + -
+-----+-----+-----+
| Сидоров Николай | 2005.08.15 | лев |
+-----+-----+-----+

(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_2>
```

Рисунок 20.11 – Проработка индивидуального задания

```
Terminal: C:\WINDOWS\system32\cmd.exe + -
(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_2>
(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_2>python ind_2.py add --name="Иванов Иван" --z="овен" --b="2001.03.21"
(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_2>python ind_2.py add --name="Сидоров Николай" --z="лев" --b="2005.08.15"
(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_2>python ind_2.py select --z="лев"
+-----+-----+-----+
| Name      | Birth    | Zodiac_sign |
+-----+-----+-----+
| Сидоров Николай | 2005.08.15 | лев        |
+-----+-----+-----+

(base) (venv) C:\Users\kuvsh\Desktop\CK0Y\2_4_семестр\Основы Программной инженерии\Git\Py_L20\PyCharm\ind_2>
```

Рисунок 20.12 – Результат проработки программы

Контрольные вопросы

1. Каково назначение переменных окружения?

Переменная среды (переменная окружения) – это короткая ссылка на какой-либо объект в системе. С помощью таких сокращений, например, можно создавать универсальные пути для приложений, которые будут работать на любых ПК, независимо от имен пользователей и других параметров.

2. Какая информация может храниться в переменных окружения?

Значением такой переменной может быть, например, место размещения исполняемых файлов в системе, имя предпочитаемого текстового редактора или настройки системной локали.

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

Для этого следует в Проводнике щелкнуть правой кнопкой мыши по иконке компьютера («Этот компьютер» в Windows 10, «Мой компьютер» в Windows 7) и выбрать «Свойства». Далее следует открыть «Дополнительные параметры системы», а в появившемся окне «Свойства системы» — «Переменные среды».

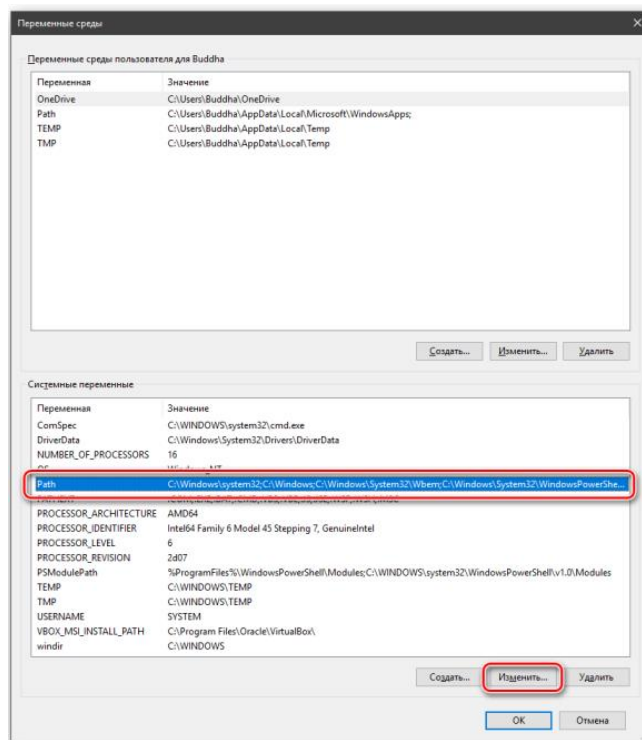
4. Каково назначение переменных PATH и PATHEXT?

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

Например, если ввести в «Командную строку»

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

5. Как создать или изменить переменную окружения в Windows?



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 можно разделить на две основные категории: 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?

команда `printenv` — выводит список всех переменных окружения (или какую-то отдельно заданную переменную);

9. Какие переменные окружения Linux Вам известны?

Ниже приведены некоторые из наиболее распространенных **переменных окружения**:

- `USER` — текущий пользователь.
- `PWD` — текущая директория.
- `OLDPWD` — предыдущая рабочая директория. Используется оболочкой для того, чтобы вернуться в предыдущий каталог при выполнении команды `cd -`.
- `HOME` — домашняя директория текущего пользователя.
- `SHELL` — путь к оболочке текущего пользователя (например, `bash` или `zsh`).
- `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 (например, различные типы файлов в выводе команды `ls` будут выделены разными цветами).

10. Какие переменные оболочки Linux Вам известны?

Наиболее распространенные **переменные оболочки**:

- `BASHOPTS` — список задействованных параметров оболочки, разделенных двоеточием.
- `BASH_VERSION` — версия запущенной оболочки `bash`.
- `COLUMNS` — количество столбцов, которые используются для отображения выходных данных.
- `DIRSTACK` — стек директорий, к которому можно применять команды `pushd` и `popd`.
- `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 (по умолчанию используется пробел).
 - `PS1` — определяет внешний вид строки приглашения ввода новых команд.
 - `PS2` — вторичная строка приглашения.
 - `SHELLOPTS` — параметры оболочки, которые можно устанавливать с помощью команды `set`.
 - `UID` — идентификатор текущего пользователя.

11. Как установить переменные оболочки в Linux?

Чтобы создать новую переменную оболочки с именем, например, `NEW_VAR` и значением `Ravesli.com`, просто введите:

```
$ NEW_VAR='Ravesli.com'
```

12. Как установить переменные окружения в Linux?

Для создания переменной окружения экспортируем нашу недавно созданную переменную оболочки:

```
$ export NEW_VAR
```

13. Для чего необходимо делать переменные окружения Linux постоянными?

Если вы хотите, чтобы переменная сохранялась после закрытия сеанса оболочки, то необходимо прописать её в специальном файле. Прописать переменную можно как для текущего пользователя, так и для всех пользователей.

14. Для чего используется переменная окружения PYTHONHOME ?

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**...

17. Как осуществляется чтение переменных окружения в программах на

языке программирования Python?

Следующий код позволяет прочитать и вывести все переменные окружения, а также определенную переменную. Для вывода имен и значений всех переменных используется цикл `for`. Затем выводится значение переменной `HOME`.

```
# 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'])
```

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

Проверяем, присвоено ли значение переменной окружения

Давайте создадим Python-файл со следующим скриптом для проверки переменных. Для чтения значений переменных мы используем модуль `os`, а модуль `sys` — для прекращения работы приложения.

Бесконечный цикл `while` непрерывно принимает от пользователя имена переменных и проверяет их значения до тех пор, пока пользователь не введёт имя переменной, которой не присвоено значение.

Если пользователь вводит имя переменной окружения, которой присвоено значение, это значение выводится, если же нет — выводится соответствующее сообщение и процесс останавливается.

```
# 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)
```

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

Присваиваем значение переменной окружения

Для присвоения значения любой переменной среды используется функция `setdefault()`.

Давайте напишем код, чтобы с помощью функции `setdefault()` изменить значение переменной `DEBUG` на `True` (по умолчанию установлено `False`). После установки значения мы проверим его функцией `get()`.

Если мы сделали всё правильно, выведется сообщение «Режим отладки включен», в противном случае – «Режим отладки выключен».

```
# 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')
```