

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 №2.18
дисциплины «Основы кроссплатформенного программирования»**

Выполнил:
Евдаков Евгений Владимирович
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., 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:

Задание 1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер. Организовал свой репозиторий в соответствие с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\Gaming-PC>git clone https://github.com/EvgenyEvdakov/Laba_2.18.git
Cloning into 'Laba_2.18'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
```

Рисунок 1. Клонирование репозитория

Задание 2. Создал виртуальное окружение conda и активировал его, также установил необходимые пакеты isort, black, flake8, pyinputplus.

```
(base) PS C:\Users\Gaming-PC> cd C:\Users\Gaming-PC\Laba_2.18
(base) PS C:\Users\Gaming-PC\Laba_2.18> conda create -n 2.18 python=3.10
WARNING: A conda environment already exists at 'C:\Users\Gaming-PC\.conda\envs\2.18'
Remove existing environment (y/[n])? y

Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
    current version: 23.1.0
    latest version: 23.9.0

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

    conda install conda=23.9.0


## Package Plan ##

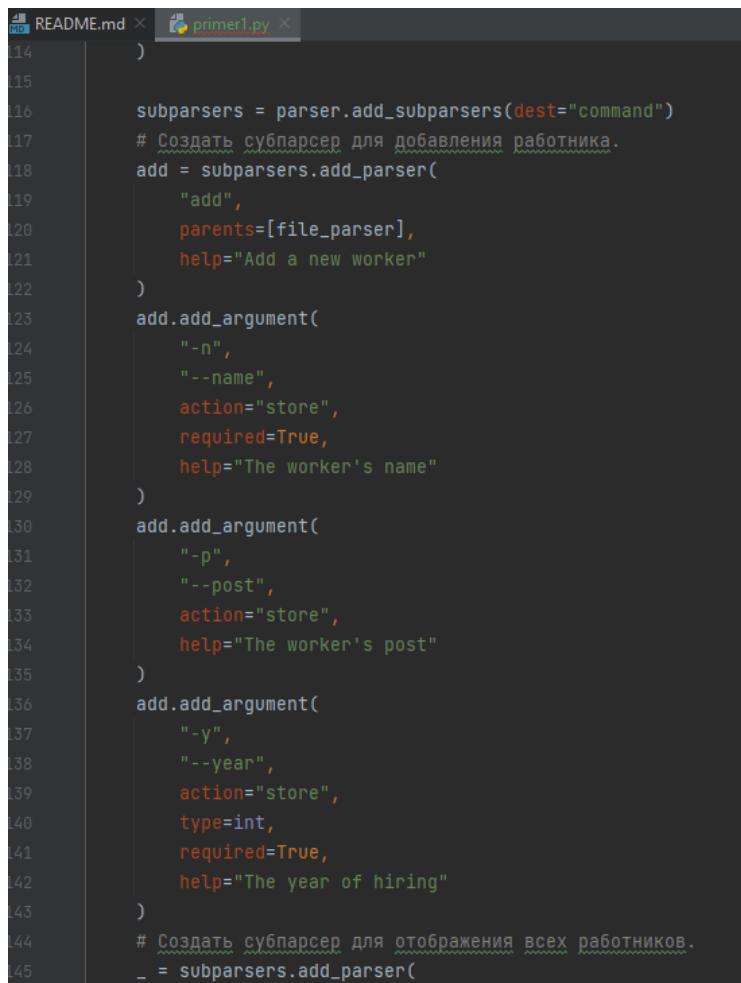
environment location: C:\Users\Gaming-PC\.conda\envs\2.18
```

Рисунок 2. Создание виртуального окружения

Задание 3. Создал проект PyCharm в папке репозитория. Приступил к работе с примером. Добавил новый файл primer1.py.

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

Для хранения имени файла данных будем использовать переменную окружения WORKERS_DATA. При этом сохраним возможность передавать имя файла данных через именной параметр --data. Иными словами, если при запуске программы в командной строке не задан параметр --data, то имя файла данных должно быть взято из переменной окружения WORKERS_DATA.



```
14     )
15
16     subparsers = parser.add_subparsers(dest="command")
17     # Создать субпарсер для добавления работника.
18     add = subparsers.add_parser(
19         "add",
20         parents=[file_parser],
21         help="Add a new worker"
22     )
23     add.add_argument(
24         "-n",
25         "--name",
26         action="store",
27         required=True,
28         help="The worker's name"
29     )
30     add.add_argument(
31         "-p",
32         "--post",
33         action="store",
34         help="The worker's post"
35     )
36     add.add_argument(
37         "-y",
38         "--year",
39         action="store",
40         type=int,
41         required=True,
42         help="The year of hiring"
43     )
44     # Создать субпарсер для отображения всех работников.
45     _ = subparsers.add_parser(
```

Рисунок 3. Пример 1

Задание 4.

Индивидуальное задание

Вариант 10

Создал новый файл под названием idz.py.

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

Для начала необходимо создать переменное окружение:

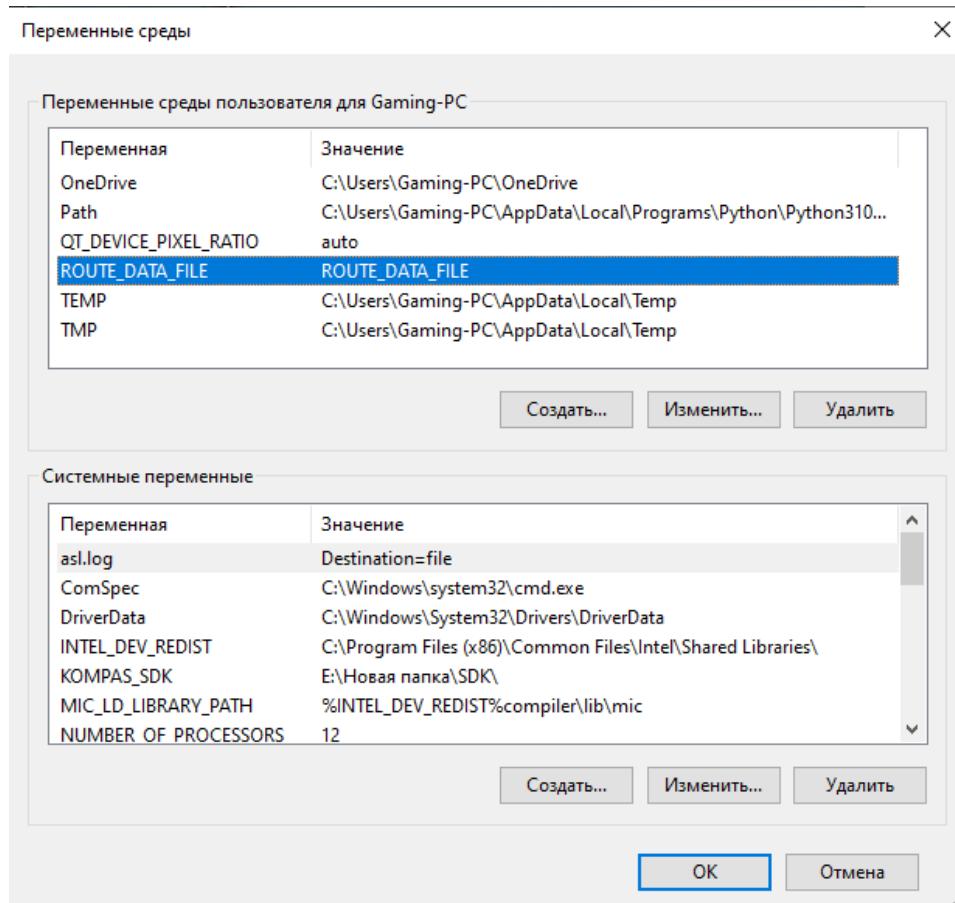


Рисунок 4. Создание переменного окружения

Рисунок 5. Код индивидуального задания

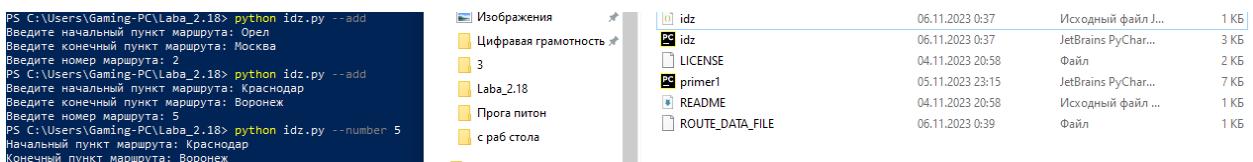


Рисунок 6. Результат индивидуального задания

В результате выполнения кода если переменная окружения ROUTE_DATA_FILE установлена, она будет использоваться в качестве имени файла данных. Если переменная окружения не установлена, будет использоваться значение по умолчанию "data.json" (в данном случае установлена).

Создадим файл idz2.py

Условия задания: Самостоятельно изучите работу с пакетом python-dotenv. Модифицируйте программу задания 1 таким образом, чтобы значения необходимых переменных окружения считывались из файла .env.

```

lbd2.py
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import json
5 import argparse
6 import os
7 from dotenv import load_dotenv
8
9 # Загрузка переменных окружения из файла .env
10 load_dotenv()
11
12 # Функция для ввода данных о маршруте
13 def add_route(routes, start, end, number):
14     route = {
15         "start": start,
16         "end": end,
17         "number": number
18     }
19     routes.append(route)
20
21     return routes
22
23 # Функция для вывода информации о маршруте по номеру
24 def find_route(routes, number):
25     found = False
26
27     for route in routes:
28         if route["number"] == number:
29             print("Начальный пункт маршрута:", route["start"])
30             print("Конечный пункт маршрута:", route["end"])
31             found = True
32
33     if not found:
34         print("Маршрут с таким номером не найден")
35
36 if __name__ == '__main__':

```

Рисунок 7. Выполнение второго индивидуального задания

Задание 5.

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер. Создал файл envirement.yml и деактивировал виртуальное окружение.

```
(2.18) PS C:\Users\Gaming-PC> conda env export > envirement.yml
(2.18) PS C:\Users\Gaming-PC> conda deactivate
```

Рисунок 7. Деактивация ВО

Ссылка: https://github.com/EvgenyEvdakov/Laba_2.18

Ответы на контрольные вопросы:

1. Каково назначение переменных окружения?

Переменные окружения используются для хранения информации, доступной для всех процессов, запущенных в операционной системе. Они предоставляют программам и системе информацию о конфигурации, путях поиска файлов, доступе к ресурсам, языковых настройках и многом другом.

2. Какая информация может храниться в переменных окружения?

- Пути к исполняемым файлам (например, переменная PATH).
- Конфигурационные настройки программ.
- Языковые параметры (например, LANG, LC_ALL).
- Данные о временных директориях, пользователях и системе.

- Параметры, управляющие поведением операционной системы и программ.

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

- Для получения доступа к переменным окружения в Windows можно использовать команду echo %VARIABLE_NAME% в командной строке, где VARIABLE_NAME - имя переменной.

- В окне "Свойства системы" можно просмотреть и изменить переменные окружения через панель управления.

4. Каково назначение переменных PATH и PATHEXT?

- PATH: Переменная, хранящая пути к исполняемым файлам. Она определяет, где операционная система будет искать исполняемые файлы, когда команда вводится в командной строке.
- PATHEXT: Список расширений файлов, который интерпретируется как исполняемые файлы в Windows.

5. Как создать или изменить переменную окружения в Windows?

Для создания или изменения переменной окружения в Windows можно использовать "Свойства системы" -> "Дополнительные параметры системы" -> "Переменные окружения". Можно добавить новую переменную или изменить значение существующей.

6. Что представляют собой переменные окружения в ОС Linux?

В Linux переменные окружения представляют собой параметры, хранящиеся в системе, доступные для всех процессов. Они определяют окружение, в котором запускаются процессы, включая пути поиска, языковые настройки и другие параметры.

7. В чем отличие переменных окружения от переменных оболочки?

- Переменные оболочки (shell variables) - это переменные, специфичные для конкретной оболочки и доступные только для этой оболочки.

- Переменные окружения (environment variables) - это переменные, доступные для всех процессов, запущенных в операционной системе, их значения наследуются от родительских процессов.

8. Как вывести значение переменной окружения в Linux?

В командной строке Linux можно использовать команду echo \$VARIABLE_NAME, где VARIABLE_NAME - имя переменной.

9. Какие переменные окружения Linux Вам известны?

PATH, HOME, USER, LANG, SHELL, PWD и другие.

10. Какие переменные оболочки Linux Вам известны?

PS1, PS2, HISTSIZE, HISTFILE и другие, специфичные для определенных оболочек (например, BASH, Zsh).

11. Как установить переменные оболочки в Linux?

Для установки переменных оболочки в Linux используются команды экспорта переменной с ключевым словом export (например, export VARIABLE_NAME=value).

12. Как установить переменные окружения в Linux?

Переменные окружения устанавливаются в Linux также, как и переменные оболочки, но они будут доступны для всех процессов. Эти переменные часто устанавливаются в файлах конфигурации системы, таких как .bashrc, .bash_profile, /etc/environment, и т. д.

13. Для чего необходимо делать переменные окружения Linux постоянными?

Переменные окружения могут быть установлены постоянно, добавив их в файлы инициализации оболочки, такие как .bashrc или .bash_profile в домашнем каталоге пользователя.

14. Для чего используется переменная окружения PYTHONHOME?

PYTHONHOME - это переменная окружения Python, которая определяет базовый каталог установки Python.

15. Для чего используется переменная окружения PYTHONPATH?

PYTHONPATH - это переменная окружения Python, определяющая пути поиска Python для модулей.

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

PYTHONHOME, PYTHONPATH, PYTHONSTARTUP, PYTHONCASEOK, PYTHONIOENCODING и другие.

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

В Python переменные окружения можно читать с помощью модуля os с функцией os.getenv().

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

Для проверки установленного значения переменной окружения в Python используйте функцию os.getenv('VARIABLE_NAME').

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

Для установки значения переменной окружения в Python используйте `os.putenv('VARIABLE_NAME', 'value')`.

Вывод: приобрел навыки по работе с переменными окружения с помощью языка программирования Python версии 3.x.