

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:
Наумов Никита Викторович
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 лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер. Организовал свой репозиторий в соответствии 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 ##

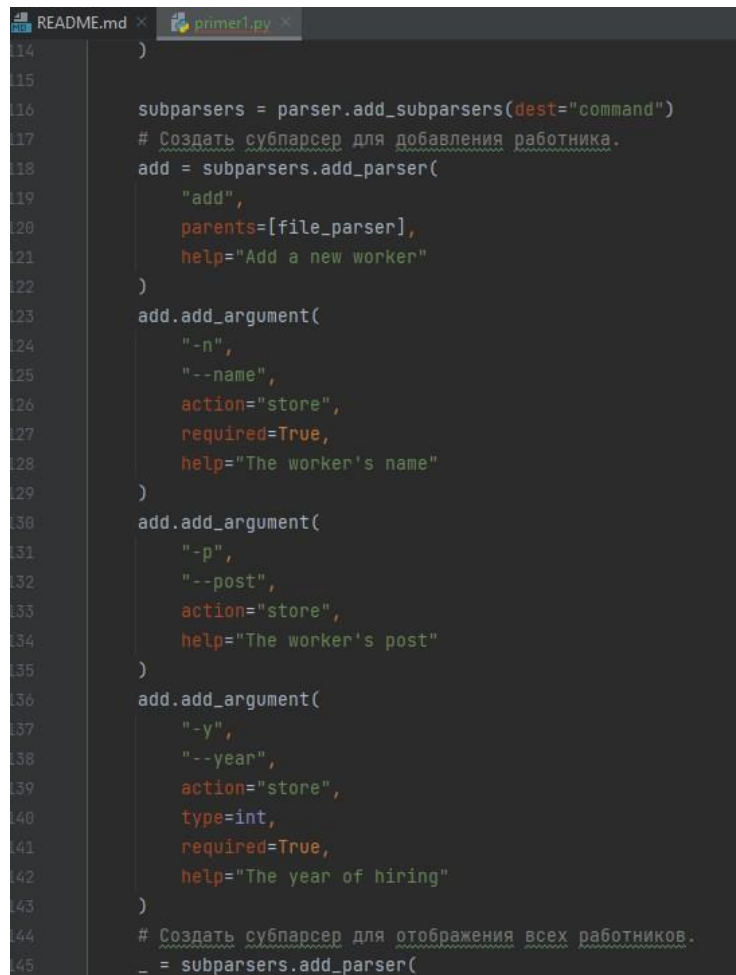
  environment location: C:\Users\Gaming-PC\.conda\envs\2.18
```

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 именованной 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`.



```
114 )
115
116 subparsers = parser.add_subparsers(dest="command")
117 # Создать субпарсер для добавления работника.
118 add = subparsers.add_parser(
119     "add",
120     parents=[file_parser],
121     help="Add a new worker"
122 )
123 add.add_argument(
124     "-n",
125     "--name",
126     action="store",
127     required=True,
128     help="The worker's name"
129 )
130 add.add_argument(
131     "-p",
132     "--post",
133     action="store",
134     help="The worker's post"
135 )
136 add.add_argument(
137     "-y",
138     "--year",
139     action="store",
140     type=int,
141     required=True,
142     help="The year of hiring"
143 )
144 # Создать субпарсер для отображения всех работников.
145 _ = 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:

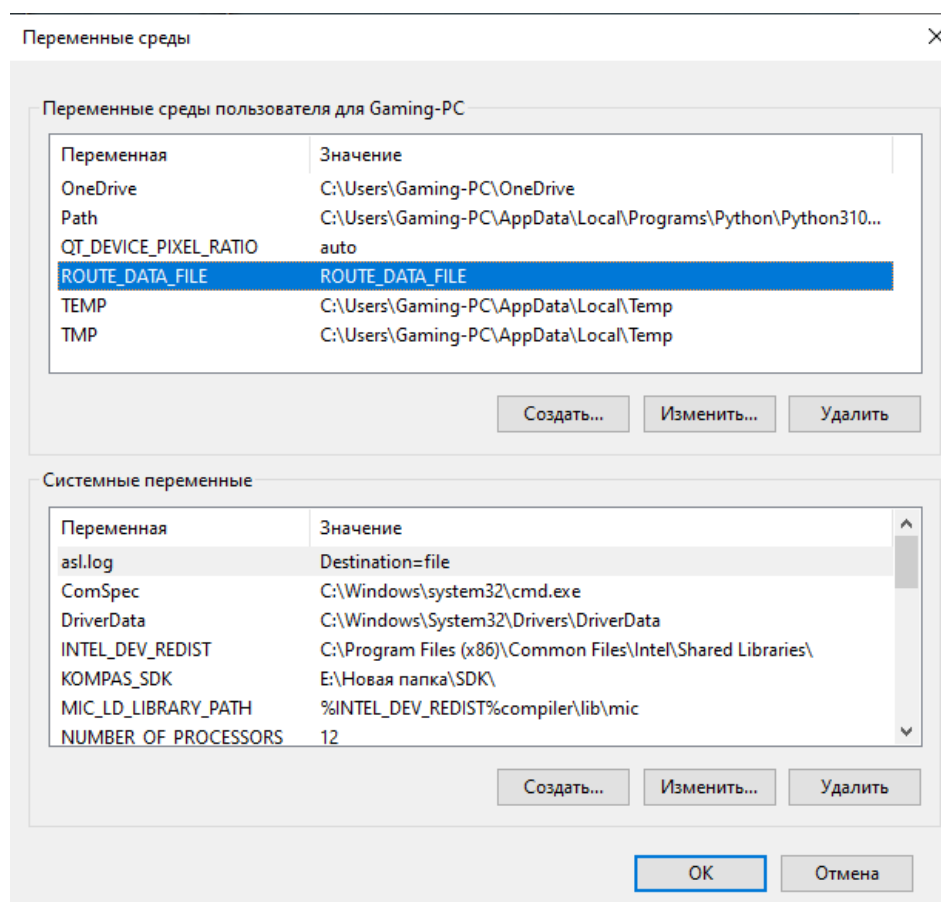


Рисунок 4. Создание переменного окружения

```

1 # idz.py
2
3 import argparse
4 import os
5 import json
6
7 print("Маршрут с таким номером не найден.")
8
9 if __name__ == '__main__':
10     parser = argparse.ArgumentParser(description="Управление маршрутами")
11     parser.add_argument('--add', action='store_true', help='Добавить новый маршрут')
12     parser.add_argument('--number', type=str, help='Номер маршрута для поиска')
13
14     args = parser.parse_args()
15
16     # Получение имени файла данных из переменных окружения
17     file_name = os.getenv('ROUTE_DATA_FILE', 'idz.json')
18
19     try:
20         with open(file_name, "r") as file:
21             routes = json.load(file)
22     except FileNotFoundError:
23         routes = []
24
25     if args.add:
26         start = input("Введите начальный пункт маршрута: ")
27         end = input("Введите конечный пункт маршрута: ")
28         number = input("Введите номер маршрута: ")
29         routes = add_route(routes, start, end, number)
30
31     if args.number:
32         find_route(routes, args.number)
33
34     # Сохранение данных в файл JSON после ввода информации
35     with open(file_name, "w") as file:
36         json.dump(routes, file)
37
38 if __name__ == '__main__':
39     except FileNotFoundError
  
```

Рисунок 5. Код индивидуального задания

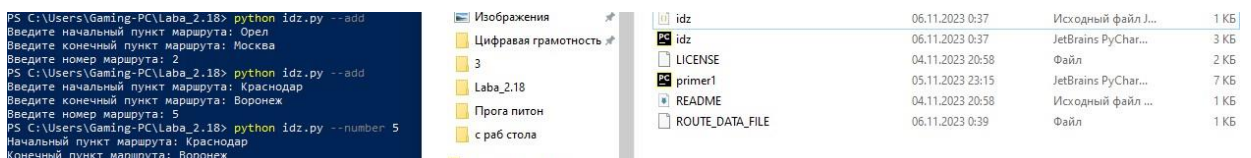


Рисунок 6. Результат индивидуального задания

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`.

- 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

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