

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 №21
дисциплины «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

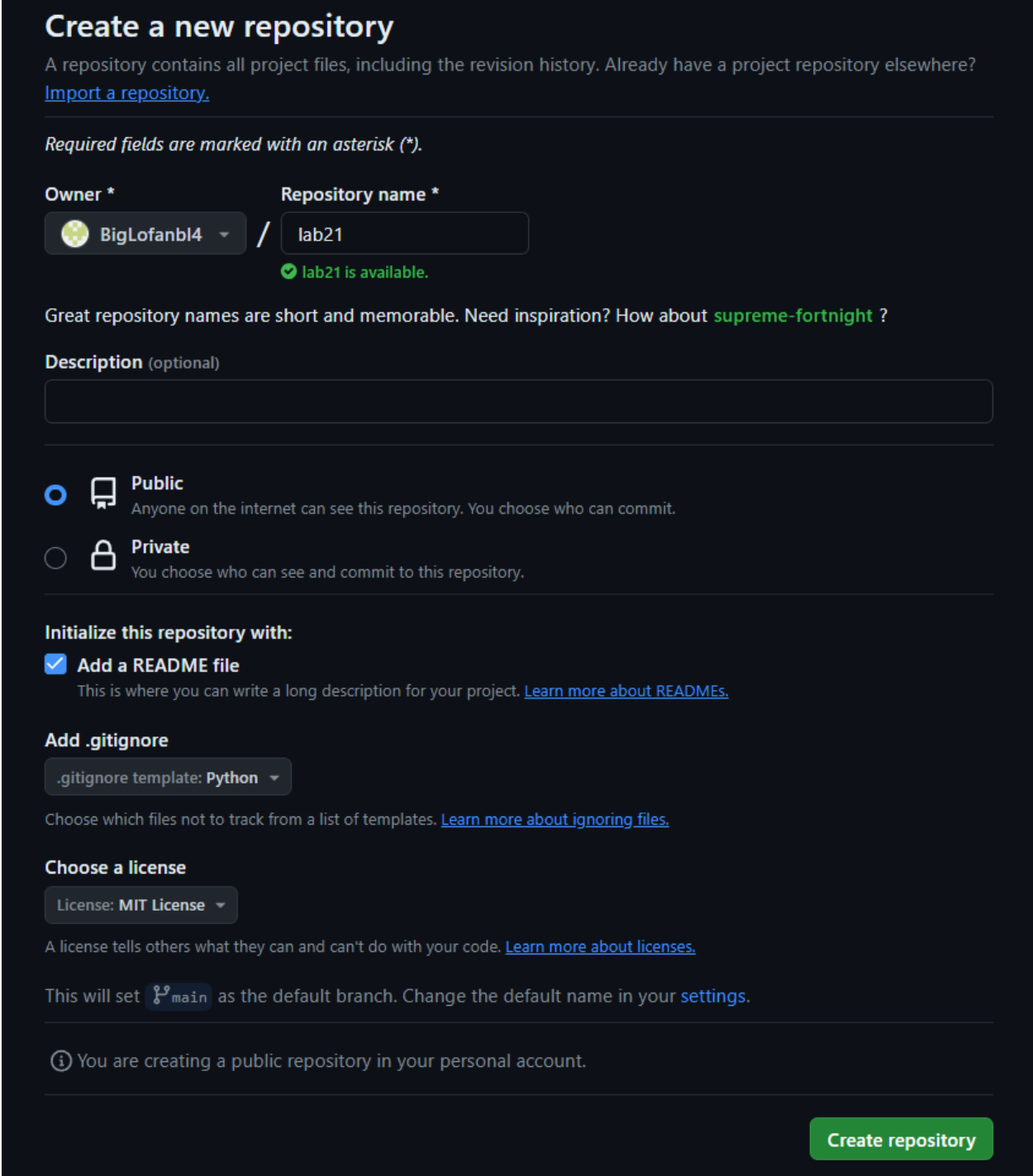
Ставрополь, 2024 г.

Тема: Работа с переменными окружения в Python3

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

Ход работы.

1. Создание репозитория.




Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*


Owner * **Repository name ***


 BigLofanbl4 / lab21

✔ lab21 is available.

Great repository names are short and memorable. Need inspiration? How about **supreme-fortnight** ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: **Python**


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **MIT License**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 – Создание репозитория

2. Выполнение примера из лабораторной работы.

```
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="%(prog)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"
    )
```

Рисунок 2.1 – Код примера

3. Выполнение первого задания: для своего варианта лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

```
# 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("people")
parser.add_argument("--version", action="version", version="%prog s 0.1.0")

subparsers = parser.add_subparsers(dest="command")

# Создать субпарсер для добавления человека.
add = subparsers.add_parser(
    "add", parents=[file_parser], help="Add a new person"
)
add.add_argument(
    "-s",
    "--surname",
    action="store",
    required=True,
    help="The person's surname",
)
add.add_argument(
    "-n", "--name", action="store", required=True, help="The person's name"
)
add.add_argument(
    "-z", "--zodiac", action="store", help="The person's zodiac"
)
add.add_argument(
    "-b",
    "--birthday",
    action="store",
    required=True,
    help="The person's birthday",
)

# Создать субпарсер для отображения всех людей.
_ = subparsers.add_parser(
    "display", parents=[file_parser], help="Display people"
)

# Создать субпарсер для выбора людей по фамилии.
select = subparsers.add_parser(
    "select", parents=[file_parser], help="Select people by surname"
)
select.add_argument(
    "-s",
    "--surname",
    action="store",
    type=str,
    required=True,
    help="The required surname",
)

# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)

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)
```

Рисунок 3.1 – Код примера

```
(Lab21) C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\21\lab21\python ind1.py add --surname="Plugatyrev" --name="Vladislav" --zodiac="Corycorn" --birthday="12.01.2005"
The data file name is absent
```

Рисунок 3.2 – Вывод программы

4. Выполнение второго задания: самостоятельно изучите работу с пакетом `python-dotenv`. Модифицируйте программу задания 1 таким образом, чтобы значения необходимых переменных окружения считывались из файла `.env`.

```
load_dotenv()
# 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("people")
parser.add_argument("--version", action="version", version=f"%(prog)s 0.1.0")

subparsers = parser.add_subparsers(dest="command")

# Создать субпарсер для добавления человека.
add = subparsers.add_parser(
    "add", parents=[file_parser], help="Add a new person"
)
add.add_argument(
    "-s",
    "--surname",
    action="store",
    required=True,
    help="The person's surname",
)
add.add_argument(
    "-n", "--name", action="store", required=True, help="The person's name"
)
add.add_argument(
    "-z", "--zodiac", action="store", help="The person's zodiac"
)
add.add_argument(
    "-b",
    "--birthday",
    action="store",
    required=True,
    help="The person's birthday",
)

# Создать субпарсер для (variable) file_parser: ArgumentParser
display = subparsers.add_parser(
    "display", parents=[file_parser], help="Display people"
)

# Создать субпарсер для выбора людей по фамилии.
select = subparsers.add_parser(
    "select", parents=[file_parser], help="Select people by surname"
)
select.add_argument(
    "-s",
    "--surname",
    action="store",
    type=str,
    required=True,
    help="The required surname",
)

# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)

data_file = args.data
if not data_file:
    data_file = os.getenv("WORKERS_DATA")
if not data_file:
    print("The data file name is absent", file=sys.stderr)
    sys.exit(1)
```

Рисунок 4.1 – Код программы

```
python ind2.py add --surname="Plugatyrev" --name="Vladislav" --zodiac="Corycorn" --birthday="12.01.2005"
```

Рисунок 4.2 – Запуск программы



```
{  
  "surname": "Plugatyrev",  
  "name": "Vladislav",  
  "zodiac": "Corycorn",  
  "birthday": [  
    "12",  
    "01",  
    "2005"  
  ]  
}
```

Рисунок 4.3 – Вывод программы

Ответы на контрольные вопросы

1—3: Назначение переменных окружения и хранение информации:

Переменные окружения используются для хранения информации о конфигурации системы для текущего пользователя или текущего процесса (например, пути к файлам, параметры сети, идентификаторы сессии). В Windows доступ к переменным окружения осуществляется через "Система" в "Панели Управления" или через командную строку с помощью команды `echo %VARIABLE_NAME%`.

4: PATH и PATHEXT

`PATH` содержит перечень каталогов, в которых операционная система ищет исполняемые файлы (приложения). `PATHEXT` задает расширения файлов, которые распознаются как исполняемые.

5: Создание и изменение переменных окружения в Windows

Переменные окружения можно создавать и изменять через "Свойства системы" -> "Дополнительные параметры системы" -> "Переменные среды", либо с помощью команд `set` и `setx` в командной строке.

6—8: Переменные окружения в Linux и их вывод

В Linux переменные окружения содержат информацию о предпочтениях и поведении окружения, используемого вашим пользователем или shell-процессами, и могут отличаться от переменных оболочки. ``echo $VARIABLE_NAME`` позволяет вывести их значение в терминале.

9—12: Известные переменные и их установка в Linux

Среди известных переменных окружения Linux – ``HOME``, ``PATH``, ``EDITOR``, а среди переменных оболочки – ``PS1``, ``BASH_VERSION``. Для их установки используется команда ``export VARIABLE_NAME=value`` или указание их в файле профайла, например ``~/.bashrc``.

13: Постоянство переменных окружения

Чтобы переменные окружения сохранялись между сессиями и перезапусками, их определяют в файлах ``~/.profile``, ``~/.bashrc``, ``/etc/environment`` или в других конфигурационных файлах.

14—15: PYTHONHOME и PYTHONPATH

``PYTHONHOME`` определяет местоположение стандартной библиотеки Python. ``PYTHONPATH`` определяет дополнительные директории, где Python будет искать модули и пакеты при их импорте.

16: Другие переменные для интерпретатора Python

Примеры включают ``PYTHONIOENCODING`` для кодировки ввода/вывода и ``PYTHONUNBUFFERED`` для небуферизованного вывода.

17—19: Работа с переменными окружения в Python

В программе на Python переменные окружения читаются с помощью модуля ``os`` и его функции ``os.getenv('VARIABLE_NAME')``. Чтобы установить или изменить значение, можно использовать ``os.environ['VARIABLE_NAME'] = 'value'``. Для проверки наличия переменной применяется оператор ``in``: ``'VARIABLE_NAME' in os.environ``.