

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:

Рядская Мария Александровна

2 курс, группа ПИЖ-б-о-22-1,

09.03.04 «Программная инженерия»,

направленность (профиль) «Разработка и

сопровождение программного

обеспечения», очная форма обучения

Проверил Воронкин Роман Александрович

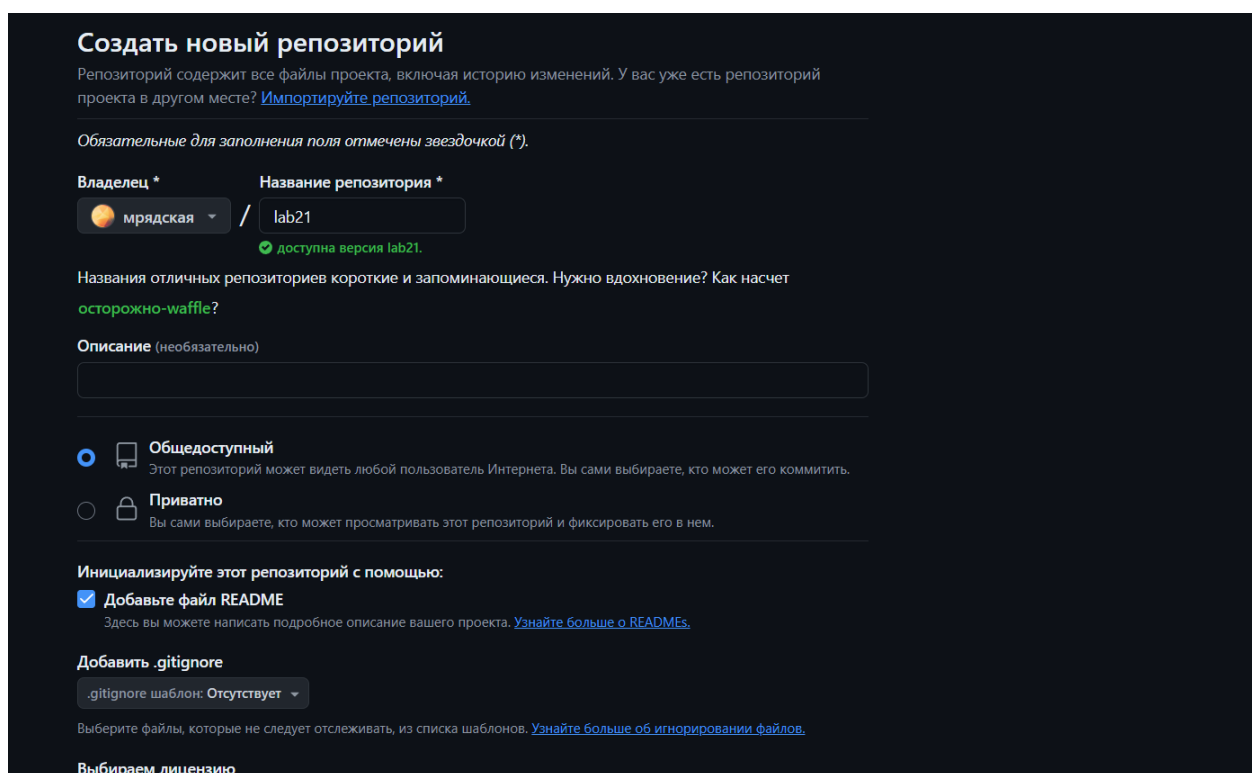
Ставрополь, 2024 г.

Тема: Лабораторная работа 2.18. 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 нового репозитория с лицензией MIT.





Создать новый репозиторий

Репозиторий содержит все файлы проекта, включая историю изменений. У вас уже есть репозиторий проекта в другом месте? [Импортируйте репозиторий.](#)

Обязательные для заполнения поля отмечены звездочкой (*).


Владелец * / Название репозитория *


 mryadskaya / lab21

 доступна версия lab21.

Названия отличных репозитория короткие и запоминающиеся. Нужно вдохновение? Как насчет **осторожно-waffle?**

Описание (необязательно)

☒  **Общедоступный**
Этот репозиторий может видеть любой пользователь Интернета. Вы сами выбираете, кто может его коммитить.

☐  **Приватно**
Вы сами выбираете, кто может просматривать этот репозиторий и фиксировать его в нем.

Инициализируйте этот репозиторий с помощью:

☒ **Добавьте файл README**
Здесь вы можете написать подробное описание вашего проекта. [Узнайте больше о READMEs.](#)

Добавить .gitignore

.gitignore шаблон: **Отсутствует**

Выберите файлы, которые не следует отслеживать, из списка шаблонов. [Узнайте больше об игнорировании файлов.](#)

Выбираем лицензию

Рисунок 1 – создание репозитория

2. Клонировал репозиторий на рабочий ПК.

```

C:\git1>git clone https://github.com/mryadskaya/lab21.git
Cloning into 'lab21'...
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:\git1>cd lab

C:\git1\lab>

```

Рисунок 2 – клонирование репозитория

```

C:\git1>cd lab

C:\git1\lab>git checkout -b develop
Switched to a new branch 'develop'

C:\git1\lab>

```

Рисунок 3– организация ветки

4. Создал виртуальное окружение и установил нужные пакеты.

```

Collecting flake8
  Obtaining dependency information for flake8 from https://files.pythonhosted.org/packages/e3/01/cc8cdec7b61db0315c2ab62d80677a138e
6832ec17f04d87e6ef858f7f/flake8-7.0.0-py2.py3-none-any.whl.metadata
  Downloading flake8-7.0.0-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting black
  Obtaining dependency information for black from https://files.pythonhosted.org/packages/0f/11/fa05ac9429d971d0fc10da85f24dafc3fa5
8733fbd0d1c186b7577cefd/black-24.4.0-cp311-cp311-win_amd64.whl.metadata
  Downloading black-24.4.0-cp311-cp311-win_amd64.whl.metadata (76 kB)
----- 76.4/76.4 kB 1.4 MB/s eta 0:00:00
Collecting pre-commit
  Obtaining dependency information for pre-commit from https://files.pythonhosted.org/packages/53/1a/1870d52d0d86d534d4d5f7a58e65d4d

```

Рисунок 5 - создание виртуального окружения

5.Проработала примеры из лабораторной работы.

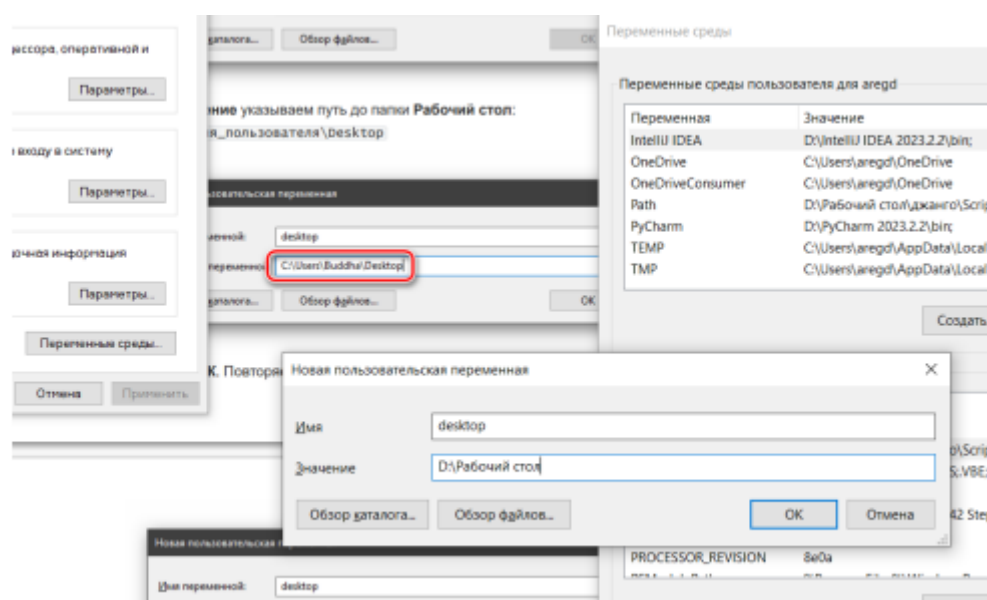


Рисунок 6 – добавление нового сотрудника

6. Для своего варианта лабораторной работы 2.16 необходимо дополнительно реализовать интерфейс командной строки (CLI).

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import argparse
5  import json
6  import os.path
7  import sys
8
9
10 def add_flight(flights, destination, departure_date, aircraft_type):
11
12     flights.append(
13         {
14             "name": name,
15             "знак зодиака": post,
16             "year": year,
17         }
18     )
19
20     return flights
21
22
23 def display_flights(flights):
24
25     if flights:
26         line = "+-{}-+-{}-+-{}-+-{}-+".format(
27             *args: "-" * 4, "-" * 30, "-" * 20, "-" * 8
28         )
29         print(line)
30         print(
31             "| {:^4} | {:^30} | {:^20} | {:^8} |".format(
32                 *args: "No", "Name", "знак зодиака ", "year"
33             )

```

Рисунок 7 – код для реализации задания

9. Самостоятельно изучите работу с пакетом click для построения интерфейса командной строки (CLI). Для своего варианта лабораторной работы 2.16 необходимо реализовать интерфейс командной строки с использованием пакета click .

```

1  ✓ #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  ✓ import argparse
5  import json
6  import os.path
7  import sys
8
9  from dotenv import load_dotenv
10
11
12  ✓ def add_flight(flights, destination, departure_date, aircraft_type):
13
14      flights.append(
15          {
16              "name": name,
17              "знак зодиака": post,
18              "year": year,
19          }
20      )
21
22      return flights
23
24
25  ✓ def display_flights(flights):
26
27      if flights:
28          line = "+-{}-+-{}-+-{}-+-{}-+ ".format(
29              *args: "-" * 4, "-" * 30, "-" * 20, "-" * 8
30          )
31          print(line)
32          print(
33              "| {:^4} | {:^30} | {:^20} | {:^8} | ".format(
main() > if args.command == "add"

```

Рисунок 8 – код для выполнения задания повышенной сложности

Контрольные вопросы:

1. Отличие терминала и консоли:

Терминал (или терминальное окно) - это программа, которая

предоставляет пользователю интерфейс для взаимодействия с операционной

системой через текстовый интерфейс. Терминал обычно предоставляет доступ

к командной строке, где пользователь может вводить команды для выполнения различных задач.

Консоль - это текстовый интерфейс, предоставляемый операционной системой для взаимодействия с пользователем или другими программами. Он

может быть встроенным в окружение рабочего стола или виртуальной машины. Консоль обычно работает поверх терминала и обеспечивает вывод

текстовой информации и ввод команд пользователя.

2.Консольное приложение:

Консольное приложение - это приложение, которое работает в текстовом режиме и взаимодействует с пользователем через командную

строку или терминал. Оно обычно используется для выполнения задач в

операционной системе или автоматизации определенных процессов.

Примеры

консольных приложений включают текстовые редакторы, утилиты командной

строки и скрипты.

3.Средства Python для построения приложений командной строки:

Модуль `sys`: для доступа к аргументам командной строки и другим системным параметрам.

Модуль `getopt`: для парсинга аргументов командной строки в стиле POSIX.

Модуль `argparse`: для создания гибких и мощных интерфейсов командной строки с поддержкой подпрограмм, позиционных и именованных

аргументов, справочной информации и др.

4. Особенности построения CLI с использованием модуля `sys`:

Модуль `sys` предоставляет базовые инструменты для работы с аргументами командной строки, доступом к системным переменным и другими системными функциями.

С его помощью можно получить доступ к аргументам командной строки через список `sys.argv` и осуществить простой парсинг аргументов.

5. Особенности построения CLI с использованием модуля `getopt`:

Модуль `getopt` позволяет парсить аргументы командной строки в стиле

POSIX с короткими и длинными опциями.

Он предоставляет более гибкие возможности по обработке аргументов,

чем модуль `sys`, но требует более сложного кода для использования.

6. Особенности построения CLI с использованием модуля `argparse`:

Модуль `argparse` предоставляет мощные средства для создания гибких интерфейсов командной строки.

Он поддерживает позиционные и именованные аргументы, подпрограммы, группы аргументов, справочную информацию и многое

другое.

`argparse` обеспечивает автоматическую проверку типов аргументов, генерацию справки и сообщений об ошибках.