# МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

**ФЕДЕРАЦИИ**

# Федеральное государственное автономное образовательное учреждение высшего образования

**«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

# Кафедра инфокоммуникаций Институт цифрового развития

**ОТЧЁТ**

# по лабораторной работе №2.18

Дисциплина: «Основы программной инженерии»

Тема: «Работа с переменными окружения в Python3»

Выполнил: студент 2 курса группы Пиж-б-о-21-1

Рязанцев Матвей Денисович

Ставрополь 2023

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

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

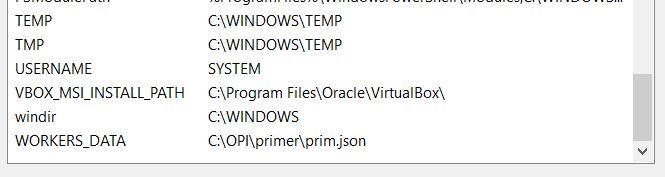


Рисунок 1 – Создание переменной окружения

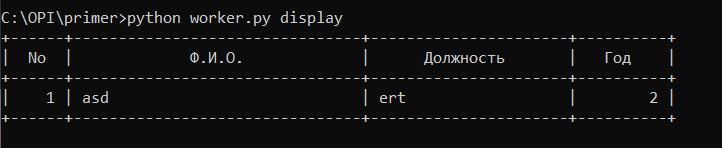


Рисунок 2 – Результат работы программы

1. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.

**Задание 1**

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

Код программы:

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
import json  
import argparse  
import os.path  
import sys  
import os  
  
  
def add\_wrk(pep,name,num,year):  
 pep.append(  
 {  
 'name': name,  
 'num': num,  
 'year': year  
 }  
 )  
 return pep  
  
def li(pep):  
 line = '+-{}-+-{}-+-{}-+-{}-+'.format(  
 '-' \* 4,  
 '-' \* 30,  
 '-' \* 20,  
 '-' \* 8  
 )  
 print(line)  
 print(  
 '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(  
 "№",  
 "F.I.O.",  
 "NUMBER",  
 "BRDAY"  
 )  
 )  
 print(line)  
 for idx, chel in enumerate(pep, 1):  
 print(  
 '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(  
 idx,  
 chel.get('name', ''),  
 chel.get('num', ''),  
 chel.get('year', 0)  
 )  
 )  
 print(line)  
  
  
def sel(pep, numb):  
 ot = []  
 # Проверить сведения работников из списка.  
 for chel in pep:  
 if numb in str(chel.values()):  
 ot.append(chel)  
 return ot  
  
  
def save\_workers(file\_name, staff):  
 *"""  
 Сохранить всех работников в файл JSON.  
 """* # Открыть файл с заданным именем для записи.  
 with open(file\_name, "w", encoding="utf-8", errors="ignore") as fout:  
 # Выполнить сериализацию данных в формат JSON.  
 # Для поддержки кирилицы установим ensure\_ascii=False  
 json.dump(staff, fout, ensure\_ascii=False, indent=4)  
  
  
def load\_workers(file\_name):  
 *"""  
 Загрузить всех работников из файла JSON.  
 """*# Открыть файл с заданным именем для чтения.  
 with open(file\_name, "r", encoding="utf-8", errors="ignore") as fin:  
 return json.load(fin)  
  
  
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("pep")  
 parser.add\_argument(  
 "--version",  
 action="version",  
 help="The main parser",  
 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 human"  
 )  
 add.add\_argument(  
 "-na",  
 "--name",  
 action="store",  
 required=True,  
 help="The human's name"  
 )  
 add.add\_argument(  
 "-n",  
 "--num",  
 action="store",  
 type=int,  
 required=True,  
 help="The human's number"  
 )  
 add.add\_argument(  
 "-y",  
 "--year",  
 action="store",  
 type=int,  
 required=True,  
 help="The date of human's birth"  
 )  
  
 # Создать субпарсер для отображения всех людей.  
 \_ = subparsers.add\_parser(  
 "display",  
 parents=[file\_parser],  
 help="Display all humans"  
 )  
  
 # Создать субпарсер для выбора людей.  
 select = subparsers.add\_parser(  
 "select",  
 parents=[file\_parser],  
 help="Select the humans"  
 )  
 select.add\_argument(  
 "-s",  
 "--select",  
 action="store",  
 required=True,  
 help="The required select"  
 )  
  
 # Выполнить разбор аргументов командной строки.  
 args = parser.parse\_args(command\_line)  
  
 data\_file = args.data  
 if not data\_file:  
 data\_file = os.environ.get("IDZ1")  
 if not data\_file:  
 print("The data file name is absent", file=sys.stderr)  
 sys.exit(1)  
  
 # Загрузить всех людей из файла, если файл существует.  
 is\_dirty = False  
 if os.path.exists(data\_file):  
 pep = load\_workers(data\_file)  
 else:  
 pep = []  
  
 # Добавить человека.  
 if args.command == "add":  
 pep = add\_wrk(  
 pep,  
 args.name,  
 args.num,  
 args.year  
 )  
 is\_dirty = True  
  
  
 # Отобразить всех людей.  
 elif args.command == "display":  
 li(pep)  
  
 # Выбрать требуемых людей.  
 elif args.command == "select":  
 selected = sel(pep, args.select)  
 li(selected)  
  
 # Сохранить данные в файл, если список людей был изменен.  
 if is\_dirty:  
 save\_workers(data\_file, pep)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

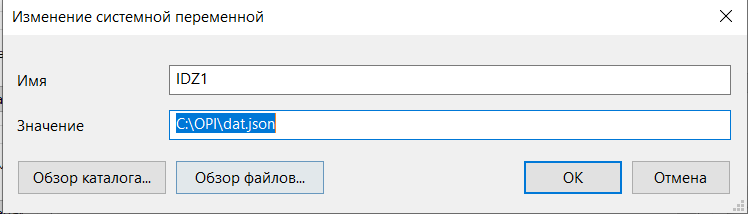


Рисунок 3 – Создание переменной окружения

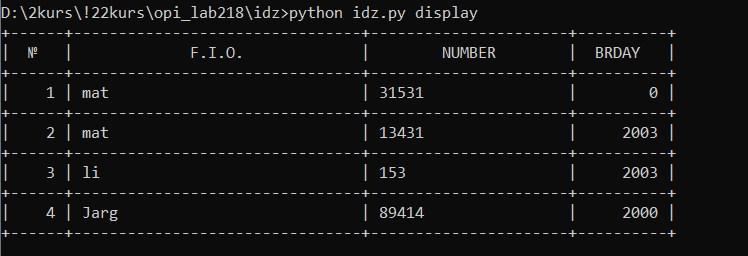
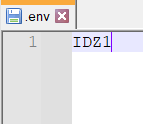


Рисунок 4 – Результат работы программы

Задание 2



Файл .env

data\_file = args.data  
dotenv\_path = os.path.join(os.path.dirname(\_\_file\_\_), ".env")  
if os.path.exists(dotenv\_path):  
 load\_dotenv(dotenv\_path)  
if not data\_file:  
 data\_file = os.environ.get("IDZ1")  
if not data\_file:  
 print("The data file name is absent", file=sys.stderr)  
 sys.exit(1)

правка в коде

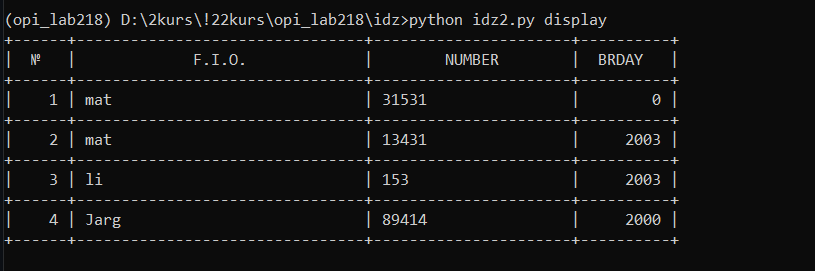


Рисунок 5 – Результат работы программы