МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Дисциплина: «Языки программирования»

Отчет по лабораторной работе №17

Установка пакетов в Python. Виртуальные окружения

Выполнила студентка группы ИТС-б-о-21-1

Тимофеева Марина Сергеевна

«__»_____20__г.

Подпись студента_____

Проверил: Доцент, к.т.н, доцент

кафедры инфокоммуникаций

Воронкин Роман Александрович

_____

(подпись)

Ставрополь, 2022

Цель работы: приобретение построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x

Ссылка на репозиторий -

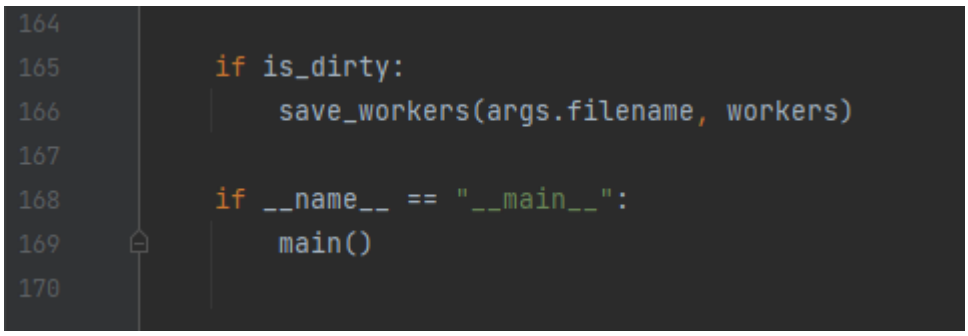
Ход работы:

Пример 1:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import argparse
import json
import os.path
import sys
from datetime import date


def add_worker(staff, name, post, year):
    staff.append(
        {
            "name": name,
            "post": post,
            "year": year
        }
    )

    return staff


def display_workers(staff):
    if staff:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )

        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)

        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
            print(line)

    else:
        print("Список работников пуст.")

```

```
56
57  def select_workers(staff, period):
58      today = date.today()
59
60      result = []
61      for employee in staff:
62          if today.year - employee.get('year', today.year) >= period:
63              result.append(employee)
64
65          return result
66
67
68  def save_workers(file_name, staff):
69      with open(file_name, "w", encoding="utf-8") as fout:
70          json.dump(staff, fout, ensure_ascii=False, indent=4)
71
72
73  def load_workers(file_name):
74      with open(file_name, "r", encoding="utf-8") as fin:
75          return json.load(fin)
76
77
78  def main(command_line=None):
79      file_parser = argparse.ArgumentParser(add_help=False)
80      file_parser.add_argument(
81          "filename",
82          action="store",
83          help="The data file name"
84      )
85
86      parser = argparse.ArgumentParser("workers")
87      parser.add_argument(
88          "--version",
89          action="version",
90          version="%(prog)s 0.1.0"
91      )
92      subparsers = parser.add_subparsers(dest="command")
93
94      add = subparsers.add_parser(
95          "add",
96          parents=[file_parser],
97          help="Add a new worker"
98      )
99      add.add_argument(
100         "-n",
101         "--name",
102         action="store",
103         required=True,
104         help="The worker's name"
105     )
106     add.add_argument(
107         "-p",
108         "--post",
109         action="store",
```

```
        help="The worker's post"
    )
    add.add_argument(
        "-y",
        "--year",
        action="store",
        type=int,
        required=True,
        help="The year of hiring"
    )

    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all workers"
    )

    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the workers"
    )
    select.add_argument(
        "-P",
        "--period",
        action="store",
        type=int,
        required=True,
        help="The required period"
    )

    args = parser.parse_args(command_line)

    is_dirty = False
    if os.path.exists(args.filename):
        workers = load_workers(args.filename)
    else:
        workers = []

    if args.command == "add":
        workers = add_worker(
            workers,
            args.name,
            args.post,
            args.year
        )
        is_dirty = True

    elif args.command == "display":
        display_workers(workers)

    elif args.command == "select":
        selected = select_workers(workers, args.period)
        display_workers(selected)

```



```
164
165     if is_dirty:
166         save_workers(args.filename, workers)
167
168     if __name__ == "__main__":
169         main()
170
```

Рисунок 1. Код примера

Задание

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

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import argparse
import json
import os.path
import sys


def add_shop(list_shop, name_shop, name_product, prise):
    """
    Добавить данные магазина.
    """
    list_shop.append(
        {
            "name_shop": name_shop,
            "name_product": name_product,
            "prise": prise
        }
    )
    return list_shop


def display_shop(list_shop):
    """
    Отобразить список магазинов.
    """
    if list_shop:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 6,
            '-' * 20,
            '-' * 30,
            '-' * 20
        )
        print(line)

        print(
            '| {:^6} | {:^20} | {:^30} | {:^20} |'.format(
                "No",
                "Название магазина",
                "Название продукта",
                "Стоимость товара"
            )
        )

        print(line)

        for idx, listshop in enumerate(list_shop, 1):
            print(
                '| {:>6} | {:<20} | {:<30} | {:>20} |'.format(
                    idx,
                    listshop.get('name_shop', ''),
                    listshop.get('name_product', ''),
                    listshop.get('prise', 0)
                )
```

```
            )
        print(line)
    else:
        print("Список магазинов пуст.")


def select_product(list_shop, shop_sear):
    """
    Выбрать продукт.
    """
    search_shop = []
    for shop_sear_itme in list_shop:
        if shop_sear == shop_sear_itme['name_product']:
            search_shop.append(shop_sear_itme)
    return search_shop


def save_shop(file_name, list_shop):
    """
    Сохранить всех работников в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(list_shop, fout, ensure_ascii=False, indent=4)


def load_list_shop(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def main(command_line=None):
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )

    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
        "-vr",
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )

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

    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new shop"
```

```
    )
    add.add_argument(
        "-ns",
        "--name_shop",
        action="store",
        required=True,
        help="The shop's name"
    )
    add.add_argument(
        "-np",
        "--name_product",
        action="store",
        help="The product's name"
    )
    add.add_argument(
        "--prise",
        action="store",
        type=int,
        required=True,
        help="Cost of goods"
    )

    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all shops"
    )

    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the product"
    )
    select.add_argument(
        "-ss",
        "--shop_sear",
        action="store",
        type=str,
        required=True,
        help="The name product"
    )

    args = parser.parse_args(command_line)

    is_dirty = False
    if os.path.exists(args.filename):
        shop = load_list_shop(args.filename)
    else:
        shop = []

    if args.command == "add":
        shop = add_shop(
            shop,
            args.name_shop,
            args.name_product,
```

Рисунок 2. Код первого задания

Контрольные вопросы:

1. В чем отличие терминала и консоли?

Терминал (от лат. terminus — граница) — устройство или ПО, выступающее посредником между человеком и вычислительной системой. Обычно данный термин используется, когда точка доступа к системе вынесена в отдельное физическое устройство и предоставляет свой пользовательский интерфейс на основе внутреннего интерфейса (например, сетевых протоколов).

Консоль console — исторически реализация терминала с клавиатурой и текстовым дисплеем. В настоящее время это слово часто используется как синоним сеанса работы или окна оболочки командной строки. В том же смысле иногда применяется и слово “терминал”.

2. Что такое консольное приложение?

Консольное приложение console application — вид ПО, разработанный с расчётом на работу внутри оболочки командной строки, т.е. опирающийся на текстовый ввод-вывод.

3. Какие существуют средства языка программирования Python для построения приложений командной строки?

Sys, getopt, argparse, click

4. Какие особенности построение CLI с использованием модуля sys

Модуль sys в Python предоставляет простые функции, которые позволяют нам напрямую взаимодействовать с интерпретатором. Функции, предоставляемые модулем sys, позволяют нам работать с базовым интерпретатором, независимо от того, является ли он платформой Windows, Macintosh или Linux

5. Какие особенности построение CLI с использованием модуля getopt?

Модуль getopt в Python – это анализатор, используемый для параметров командной строки, которые основаны на соглашении, организованном функцией UNIX getopt(). Он в основном используется для анализа последовательности аргументов, например sys.argv. Мы также можем истолковать этот модуль как помощника сценариям анализировать аргументы командной строки в sys.argv

6. Какие особенности построение CLI с использованием модуля argparse ?

Модуль argparse является рекомендуемым к использованию модулем стандартной библиотеки Python, предназначенным для работы с аргументами командной строки

Вывод: приобрела знания в построении приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x