**МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**«Работа с данными формата JSON в языке Python»**

**Отчет по лабораторной работе № 2.16 по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Гребенкин Е. А. « » 2022г.

Подпись студента\_\_\_\_\_\_\_\_\_\_\_

Работа защищена « »\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Проверил Воронкин Р.А. \_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись)

**Цель работы:** приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

**Выполнение работы:**

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.
6. Создайте проект PyCharm в папке репозитория.
7. Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории.
8. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.
9. Зафиксируйте сделанные изменения в репозитории.
10. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.
11. Зафиксируйте сделанные изменения в репозитории.
12. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
13. Выполните слияние ветки для разработки с веткой master/main.
14. Отправьте сделанные изменения на сервер GitHub.
15. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

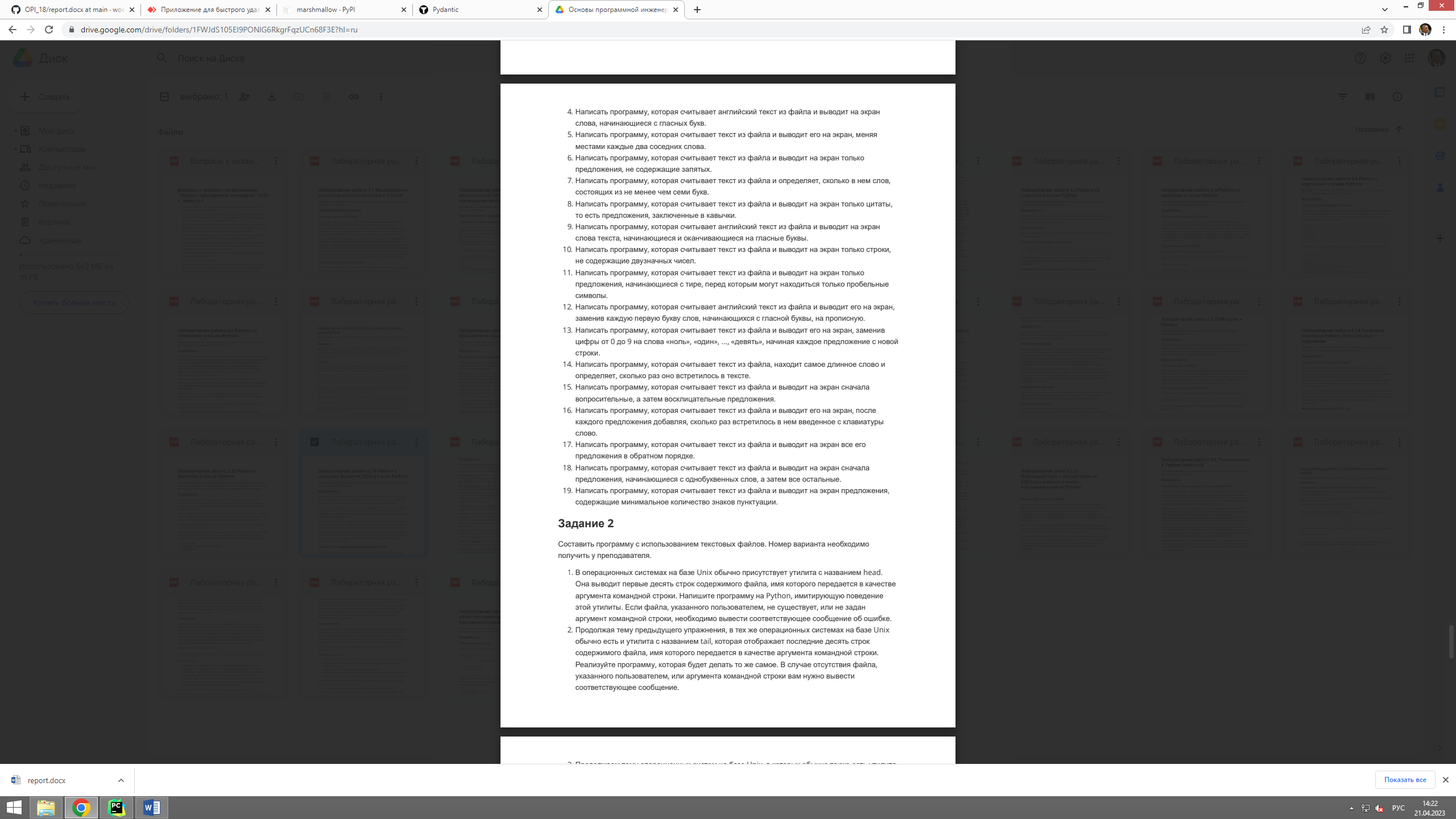
**Проработка примеров:**

#!/user/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
import sys  
import json  
from datetime import date  
  
  
def get\_worker() -> dict:  
 *"""  
 Запросить данные о работнике.* ***:return*** *dict:  
 """* name = input("Фамилия и инициалы >> ")  
 post = input("Должность >> ")  
 year = int(input("Год поступления >> "))  
  
 return {  
 'name': name,  
 'post': post,  
 'year': year  
 }  
  
  
def display\_workers(staff):  
 *"""  
 Отобразить список работников.* ***:param*** *staff:* ***:return****:  
 """* if staff:  
 line = '+-{}-+-{}-+-{}-+-{}-+'.format(  
 '-' \* 4,  
 '-' \* 30,  
 '-' \* 20,  
 '-' \* 8  
 )  
 print(line)  
 print(  
 "| {:^4} | {:^30} | {:^20} | {:^8} |".format(  
 "№",  
 "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 сотрудников пуст.")  
  
  
def select\_workers(staff, period: int) -> list:  
 *"""  
 Выбрать работников с заданным стажем* ***:param*** *staff:* ***:param*** *period:* ***:return*** *result:  
 """* today = date.today()  
  
 result = list()  
  
 for employee in staff:  
 if today.year - employee.get('year', today.year) >= period:  
 result.append(employee)  
  
 return result  
  
  
def save\_workers(file\_name: str, staff):  
 *"""  
 Сохранить всех работников в файл JSON.* ***:param*** *file\_name:* ***:param*** *staff:* ***:return****:  
 """* with open(file\_name, 'w', encoding='utf-8') as fout:  
 json.dump(staff, fout, ensure\_ascii=False, indent=4)  
  
  
def load\_workers(file\_name: str) -> dict:  
 *"""  
 Загрузить всех работников из файла JSON.* ***:param*** *file\_name:* ***:return*** *dict:  
 """* with open(file\_name, 'r', encoding='utf-8') as fin:  
 return json.load(fin)  
  
  
def main():  
 *"""  
 Главная функция программы  
 """* # Список работников.  
 workers = list()  
  
 while True:  
 # Запросить команду из терминала  
 command = input(">>> ").lower()  
  
 # Выполнить действие в соответствии с командой.  
 if command == "exit":  
 break  
  
 elif command == "add":  
 worker = get\_worker()  
  
 workers.append(worker)  
  
 if len(workers) > 1:  
 workers.sort(key=lambda item: item.get('name', ''))  
  
 elif command == "list":  
 display\_workers(workers)  
  
 elif command.startswith("select "):  
 parts = command.split(maxsplit=1)  
  
 period = int(parts[1])  
  
 selected = select\_workers(workers, period)  
  
 display\_workers(selected)  
 elif command.startswith("save "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
  
 save\_workers(file\_name, workers)  
  
 elif command.startswith("load "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
  
 workers = load\_workers(file\_name)  
  
 elif command == "help":  
 print("Список команд:\n")  
 print("add - добавить работника;")  
 print("list - вывести список работников;")  
 print("select <стаж> - запросить работников со стажем;")  
 print("help - отобразить справку;")  
 print("load - загрузить данные из файла;")  
 print("save - сохранить данные в файл;")  
 print("exit - завершиьт работу с программой.")  
  
 else:  
 print(f"Неизвестная команда {command}", file=sys.stderr)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Листинг 1** – Код примера работы с JSON-файламы

**Самостоятельные задания**

**Задание 1.**



#!/user/bin/env python3  
# -\*- coding: utf-8 -\*-  
import json  
  
LETTERS = 'eyuioa'  
  
def load\_words(file\_name: str) -> dict:  
 with open(file\_name, 'r', encoding='utf-8') as f:  
 return json.load(f)  
  
  
def add\_words(file\_name: str, word: dict) -> None:  
 with open(file\_name, 'r', encoding='utf-8') as f:  
 content\_json = json.load(f)  
 content\_json.append(word)  
 with open(file\_name, 'w', encoding='utf-8') as f:  
 json.dump(content\_json, f)  
  
  
def update\_json(file\_name: str, words: list) -> list:  
 for item in words:  
 for letter in item["word"]:  
 if letter in LETTERS:  
 item["result"] = True  
 else:  
 item["result"] = False  
 with open(file\_name, 'w', encoding='utf-8') as f:  
 json.dump(words, f)  
 return words  
  
  
def display(words):  
 if words:  
 line = '+-{}-+-{}-+-{}-+'.format(  
 '-' \* 4,  
 '-' \* 30,  
 '-' \* 20  
 )  
 print(line)  
 print(  
 "| {:^4} | {:^30} | {:^20} |".format(  
 "№",  
 "Слово",  
 "Результат"  
 )  
 )  
 print(line)  
  
 for idx, worker in enumerate(words, 1):  
 print(  
 "| {:^4} | {:^30} | {:^20} |".format(  
 idx,  
 worker.get('word', ''),  
 worker.get('result', ''),  
 )  
 )  
 print(line)  
 else:  
 print("Список пуст.")  
  
  
def main():  
 words = list()  
  
 while True:  
 command = input(">>> ")  
 if command.startswith("load "):  
 words = load\_words(command.split(maxsplit=1)[1])  
 elif command == "list":  
 display(words)  
 elif command == "exit":  
 break  
 elif command.startswith("add "):  
 word = input("Введите слово >> ")  
 words.append({"word": word, "result": False})  
 add\_words(command.split(maxsplit=1)[1], {"word": word, "result": False})  
 elif command.startswith("update "):  
 words = update\_json(command.split(maxsplit=1)[1], words)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

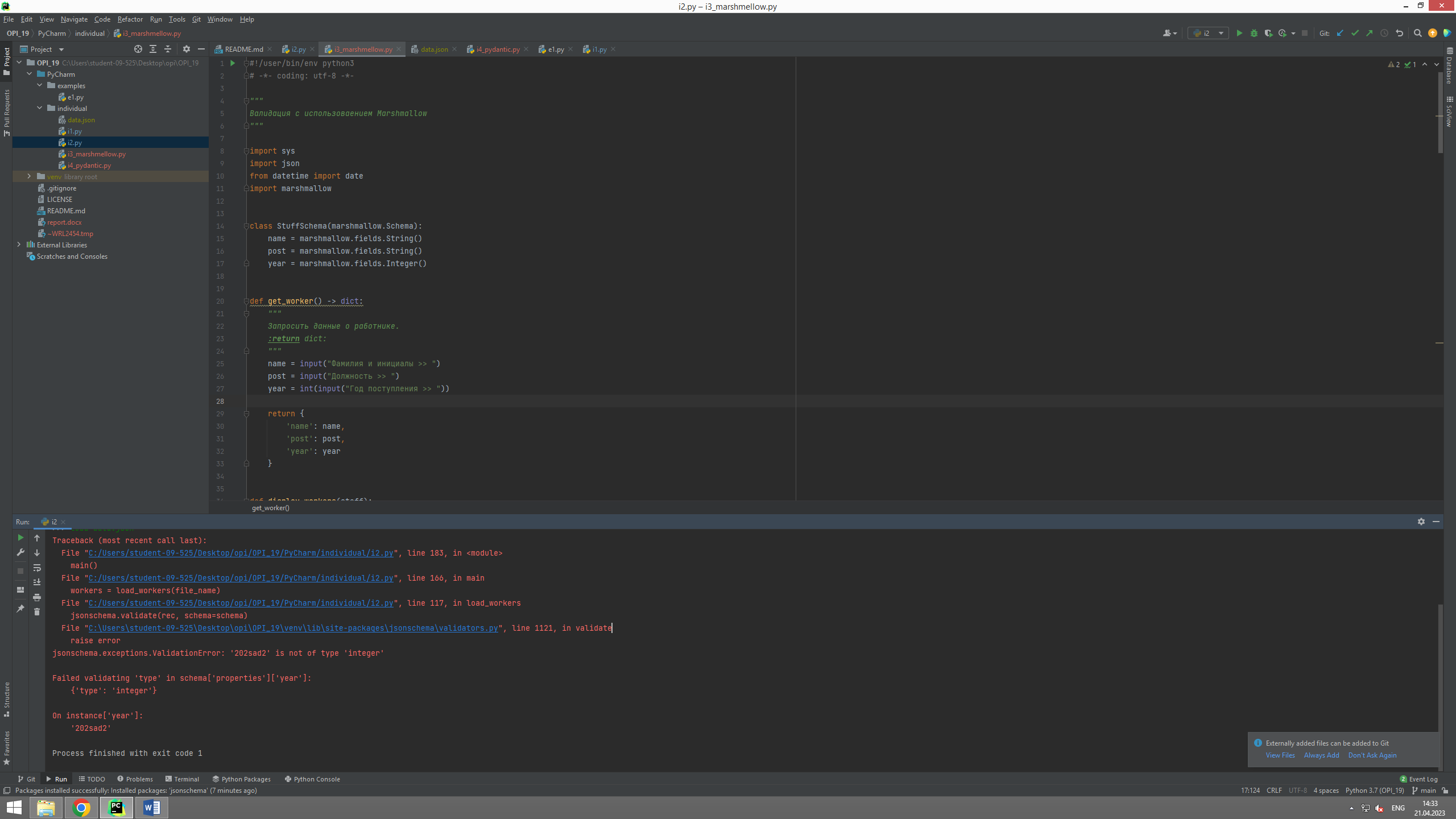
**Листинг 2** – Код задания для ЛР №2.15 с использованием JSON

**Задание 2.**

Прописать валидацию данных с использованием библиотек marshmallow, jsonschema, pydantic.

#!/user/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
*"""  
Валидация с использоваением Marshmallow  
"""*import sys  
import json  
from datetime import date  
import marshmallow  
  
  
class StuffSchema(marshmallow.Schema):  
 name = marshmallow.fields.String()  
 post = marshmallow.fields.String()  
 year = marshmallow.fields.Integer()  
  
  
def get\_worker() -> dict:  
 *"""  
 Запросить данные о работнике.* ***:return*** *dict:  
 """* name = input("Фамилия и инициалы >> ")  
 post = input("Должность >> ")  
 year = int(input("Год поступления >> "))  
  
 return {  
 'name': name,  
 'post': post,  
 'year': year  
 }  
  
  
def display\_workers(staff):  
 *"""  
 Отобразить список работников.* ***:param*** *staff:* ***:return****:  
 """* if staff:  
 line = '+-{}-+-{}-+-{}-+-{}-+'.format(  
 '-' \* 4,  
 '-' \* 30,  
 '-' \* 20,  
 '-' \* 8  
 )  
 print(line)  
 print(  
 "| {:^4} | {:^30} | {:^20} | {:^8} |".format(  
 "№",  
 "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 сотрудников пуст.")  
  
  
def select\_workers(staff, period: int) -> list:  
 *"""  
 Выбрать работников с заданным стажем* ***:param*** *staff:* ***:param*** *period:* ***:return*** *result:  
 """* today = date.today()  
  
 result = list()  
  
 for employee in staff:  
 if today.year - employee.get('year', today.year) >= period:  
 result.append(employee)  
  
 return result  
  
  
def save\_workers(file\_name: str, staff):  
 *"""  
 Сохранить всех работников в файл JSON.* ***:param*** *file\_name:* ***:param*** *staff:* ***:return****:  
 """* with open(file\_name, 'w', encoding='utf-8') as fout:  
 json.dump(staff, fout, ensure\_ascii=False, indent=4)  
  
  
def load\_workers(file\_name: str) -> list:  
 *"""  
 Загрузить всех работников из файла JSON.* ***:param*** *file\_name:* ***:return*** *dict:  
 """* with open(file\_name, 'r', encoding='utf-8') as fin:  
 schema = StuffSchema()  
 res = list()  
 for rec in json.load(fin):  
 res.append(schema.load(rec))  
 return res  
  
  
def main():  
 *"""  
 Главная функция программы  
 """* # Список работников.  
 workers = list()  
  
 while True:  
 # Запросить команду из терминала  
 command = input(">>> ").lower()  
  
 # Выполнить действие в соответствии с командой.  
 if command == "exit":  
 break  
  
 elif command == "add":  
 worker = get\_worker()  
  
 workers.append(worker)  
  
 if len(workers) > 1:  
 workers.sort(key=lambda item: item.get('name', ''))  
  
 elif command == "list":  
 display\_workers(workers)  
  
 elif command.startswith("select "):  
 parts = command.split(maxsplit=1)  
  
 period = int(parts[1])  
  
 selected = select\_workers(workers, period)  
  
 display\_workers(selected)  
 elif command.startswith("save "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
  
 save\_workers(file\_name, workers)  
  
 elif command.startswith("load "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
  
 workers = load\_workers(file\_name)  
  
 elif command == "help":  
 print("Список команд:\n")  
 print("add - добавить работника;")  
 print("list - вывести список работников;")  
 print("select <стаж> - запросить работников со стажем;")  
 print("help - отобразить справку;")  
 print("load - загрузить данные из файла;")  
 print("save - сохранить данные в файл;")  
 print("exit - завершиьт работу с программой.")  
  
 else:  
 print(f"Неизвестная команда {command}", file=sys.stderr)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

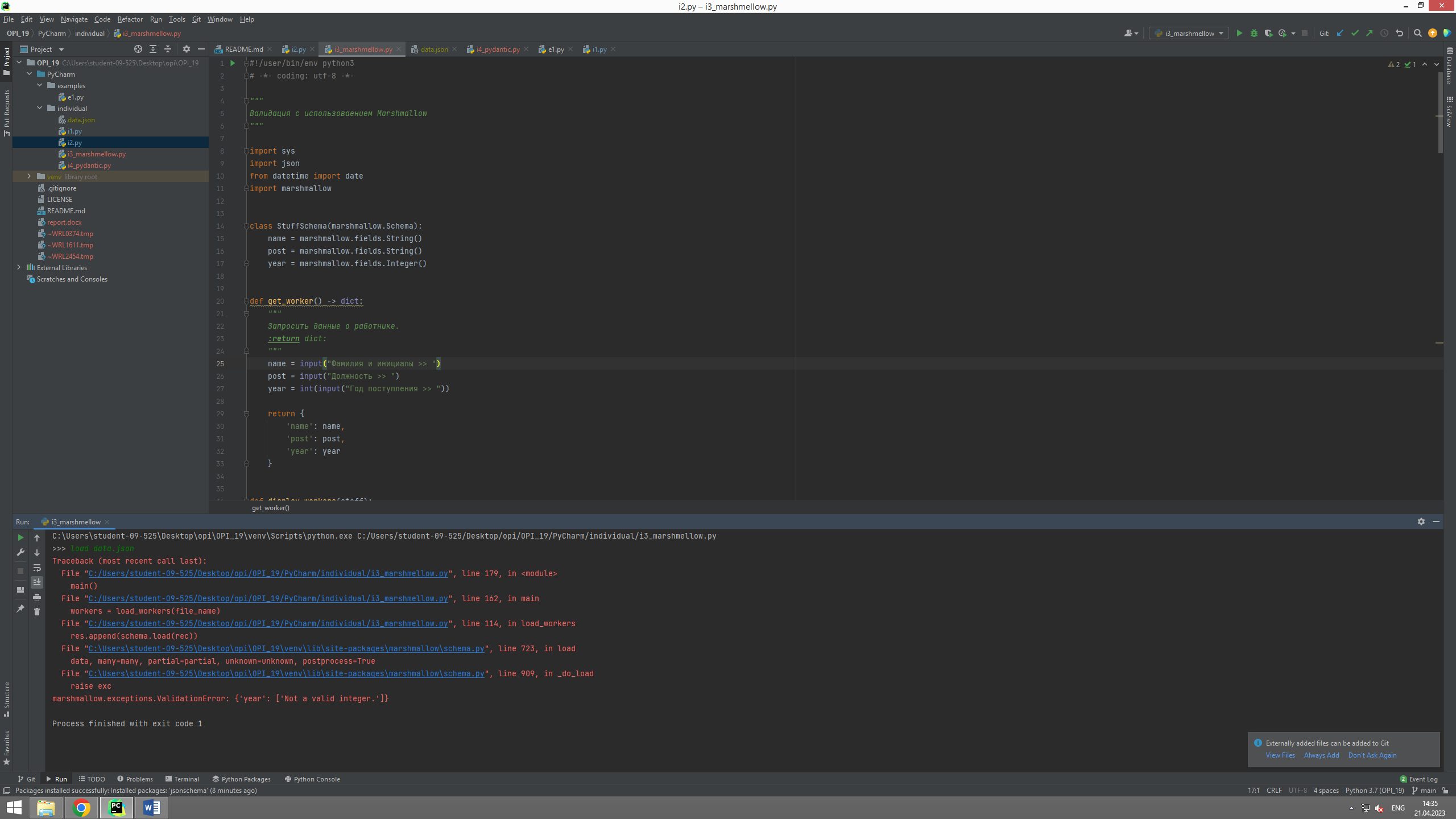
**Листинг 3** – Валидация данных с помощью jsonschema



**Рисунок 1** – Исключение при ошибке валидации

#!/user/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
*"""  
Валидация с использоваением Marshmallow  
"""*import sys  
import json  
from datetime import date  
import marshmallow  
  
  
class StuffSchema(marshmallow.Schema):  
 name = marshmallow.fields.String()  
 post = marshmallow.fields.String()  
 year = marshmallow.fields.Integer()  
  
  
def get\_worker() -> dict:  
 *"""  
 Запросить данные о работнике.* ***:return*** *dict:  
 """* name = input("Фамилия и инициалы >> ")  
 post = input("Должность >> ")  
 year = int(input("Год поступления >> "))  
  
 return {  
 'name': name,  
 'post': post,  
 'year': year  
 }  
  
  
def display\_workers(staff):  
 *"""  
 Отобразить список работников.* ***:param*** *staff:* ***:return****:  
 """* if staff:  
 line = '+-{}-+-{}-+-{}-+-{}-+'.format(  
 '-' \* 4,  
 '-' \* 30,  
 '-' \* 20,  
 '-' \* 8  
 )  
 print(line)  
 print(  
 "| {:^4} | {:^30} | {:^20} | {:^8} |".format(  
 "№",  
 "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 сотрудников пуст.")  
  
  
def select\_workers(staff, period: int) -> list:  
 *"""  
 Выбрать работников с заданным стажем* ***:param*** *staff:* ***:param*** *period:* ***:return*** *result:  
 """* today = date.today()  
  
 result = list()  
  
 for employee in staff:  
 if today.year - employee.get('year', today.year) >= period:  
 result.append(employee)  
  
 return result  
  
  
def save\_workers(file\_name: str, staff):  
 *"""  
 Сохранить всех работников в файл JSON.* ***:param*** *file\_name:* ***:param*** *staff:* ***:return****:  
 """* with open(file\_name, 'w', encoding='utf-8') as fout:  
 json.dump(staff, fout, ensure\_ascii=False, indent=4)  
  
  
def load\_workers(file\_name: str) -> list:  
 *"""  
 Загрузить всех работников из файла JSON.* ***:param*** *file\_name:* ***:return*** *dict:  
 """* with open(file\_name, 'r', encoding='utf-8') as fin:  
 schema = StuffSchema()  
 res = list()  
 for rec in json.load(fin):  
 res.append(schema.load(rec))  
 return res  
  
  
def main():  
 *"""  
 Главная функция программы  
 """* # Список работников.  
 workers = list()  
  
 while True:  
 # Запросить команду из терминала  
 command = input(">>> ").lower()  
  
 # Выполнить действие в соответствии с командой.  
 if command == "exit":  
 break  
  
 elif command == "add":  
 worker = get\_worker()  
  
 workers.append(worker)  
  
 if len(workers) > 1:  
 workers.sort(key=lambda item: item.get('name', ''))  
  
 elif command == "list":  
 display\_workers(workers)  
  
 elif command.startswith("select "):  
 parts = command.split(maxsplit=1)  
  
 period = int(parts[1])  
  
 selected = select\_workers(workers, period)  
  
 display\_workers(selected)  
 elif command.startswith("save "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
  
 save\_workers(file\_name, workers)  
  
 elif command.startswith("load "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
  
 workers = load\_workers(file\_name)  
  
 elif command == "help":  
 print("Список команд:\n")  
 print("add - добавить работника;")  
 print("list - вывести список работников;")  
 print("select <стаж> - запросить работников со стажем;")  
 print("help - отобразить справку;")  
 print("load - загрузить данные из файла;")  
 print("save - сохранить данные в файл;")  
 print("exit - завершиьт работу с программой.")  
  
 else:  
 print(f"Неизвестная команда {command}", file=sys.stderr)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

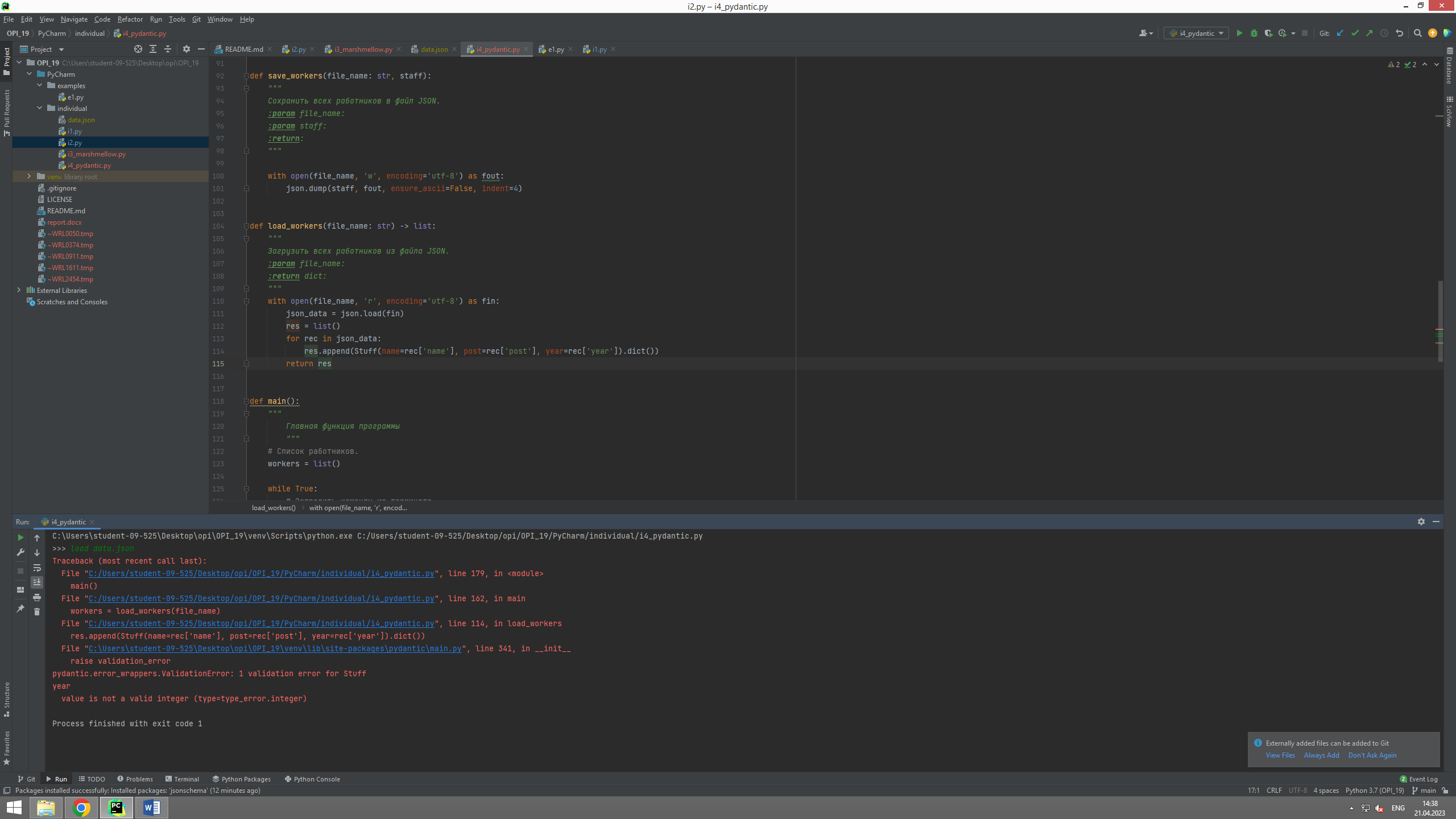
**Листинг 4** – Валидация при помощи marshmallow



**Картинка 2** – Исключение при ошибке валидации

#!/user/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
*"""  
Валидация с использоваением Pydantic  
"""*import sys  
import json  
from datetime import date  
import pydantic  
  
  
class Stuff(pydantic.BaseModel):  
 name: str  
 post: str  
 year: int  
  
  
def get\_worker() -> dict:  
 *"""  
 Запросить данные о работнике.* ***:return*** *dict:  
 """* name = input("Фамилия и инициалы >> ")  
 post = input("Должность >> ")  
 year = int(input("Год поступления >> "))  
  
 return {  
 'name': name,  
 'post': post,  
 'year': year  
 }  
  
  
def display\_workers(staff):  
 *"""  
 Отобразить список работников.* ***:param*** *staff:* ***:return****:  
 """* if staff:  
 line = '+-{}-+-{}-+-{}-+-{}-+'.format(  
 '-' \* 4,  
 '-' \* 30,  
 '-' \* 20,  
 '-' \* 8  
 )  
 print(line)  
 print(  
 "| {:^4} | {:^30} | {:^20} | {:^8} |".format(  
 "№",  
 "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 сотрудников пуст.")  
  
  
def select\_workers(staff, period: int) -> list:  
 *"""  
 Выбрать работников с заданным стажем* ***:param*** *staff:* ***:param*** *period:* ***:return*** *result:  
 """* today = date.today()  
  
 result = list()  
  
 for employee in staff:  
 if today.year - employee.get('year', today.year) >= period:  
 result.append(employee)  
  
 return result  
  
  
def save\_workers(file\_name: str, staff):  
 *"""  
 Сохранить всех работников в файл JSON.* ***:param*** *file\_name:* ***:param*** *staff:* ***:return****:  
 """* with open(file\_name, 'w', encoding='utf-8') as fout:  
 json.dump(staff, fout, ensure\_ascii=False, indent=4)  
  
  
def load\_workers(file\_name: str) -> list:  
 *"""  
 Загрузить всех работников из файла JSON.* ***:param*** *file\_name:* ***:return*** *dict:  
 """* with open(file\_name, 'r', encoding='utf-8') as fin:  
 json\_data = json.load(fin)  
 try:  
 res = list()  
 for rec in json\_data:  
 res.append(Stuff(name=rec['name'], post=rec['post'], year=rec['year']).dict())  
 return res  
 except pydantic.ValidationError as e:  
 print(e.json())  
 sys.exit()  
  
  
def main():  
 *"""  
 Главная функция программы  
 """* # Список работников.  
 workers = list()  
  
 while True:  
 # Запросить команду из терминала  
 command = input(">>> ").lower()  
  
 # Выполнить действие в соответствии с командой.  
 if command == "exit":  
 break  
  
 elif command == "add":  
 worker = get\_worker()  
  
 workers.append(worker)  
  
 if len(workers) > 1:  
 workers.sort(key=lambda item: item.get('name', ''))  
  
 elif command == "list":  
 display\_workers(workers)  
  
 elif command.startswith("select "):  
 parts = command.split(maxsplit=1)  
  
 period = int(parts[1])  
  
 selected = select\_workers(workers, period)  
  
 display\_workers(selected)  
 elif command.startswith("save "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
  
 save\_workers(file\_name, workers)  
  
 elif command.startswith("load "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
  
 workers = load\_workers(file\_name)  
  
 elif command == "help":  
 print("Список команд:\n")  
 print("add - добавить работника;")  
 print("list - вывести список работников;")  
 print("select <стаж> - запросить работников со стажем;")  
 print("help - отобразить справку;")  
 print("load - загрузить данные из файла;")  
 print("save - сохранить данные в файл;")  
 print("exit - завершиьт работу с программой.")  
  
 else:  
 print(f"Неизвестная команда {command}", file=sys.stderr)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Листинг 5 –** Валидация при помощи pydantic



**Картинка 3** – Исключение при ошибке валидации

**Вопросы для защиты работы**

1. Для чего используется JSON?

*За счёт своей лаконичности по сравнению с XML формат JSON может быть более подходящим для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения).*

*Легкочитаемый и компактный, JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента.*

1. Какие типы значений используются в JSON?

*В качестве значений в JSON могут быть использованы:*

* ***запись*** *— это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.*
* ***массив*** *(одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, т.е. не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.*
* ***число*** *(целое или вещественное).*
* ***литералы*** *true (логическое значение «истина»), false (логическое значение «ложь») и null.*
* ***строка*** *— это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape-последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты ', ", \, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.*

1. Как организована работа со сложными данными в JSON?

*JSON может содержать другие вложенные объекты в JSON, в дополнение к вложенным массивам.*

*Такие объекты и массивы будут передаваться, как значения назначенные ключам и будут*

*представлять собой связку ключ-значение.*

1. Самостоятельно ознакомьтесь с форматом данных JSON5? В чем отличие этого формата от формата данных JSON?

***Объекты***

* *Объекты могут иметь одну запятую.*

***Массивы***

* *Массивы могут иметь одну запятую.*

***Строки***

* *Строки могут заключаться в одинарные кавычки.*
* *Строки могут охватывать несколько строк, экранируя символы новой строки.*
* *Строки могут включать в себя экранирование символов.*

***Числа***

* *Числа могут быть шестнадцатеричными.*
* *Числа могут иметь ведущую или последующую десятичную точку.*
* *Числа могут быть Infinity, -Infinity2 и NaN.*
* *Числа могут начинаться с явно определенного знака +.*

***Комментарии***

* *Допускаются однострочные и многострочные комментарии.*

***Пробельные символы***

* *Разрешены дополнительные пробельные символы.*

1. Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5?

*Библиотека JSON, позволяющая работать с данными dict, str в python для преобразования их в json-формат, а также в обратную сторону, для преобразования json-файлов в читаемые python форматы*.

1. Какие средства предоставляет язык Python для сериализации данных в формате JSON?

*json.dump() и json.dumps()*

1. В чем отличие функций json.dump() и json.dumps()?

*json.dump() – конвертировать python объект в json и записать в файл*

*json.dumps() – тоже самое, но в строку*

1. Какие средства предоставляет язык Python для десериализации данных из формата JSON?

*json.load() – прочитать json из файла и конвертировать в python объект*

*json.loads() – тоже самое, но из строки с json (s на конце от string/строка)*

1. Какие средства необходимо использовать для работы с данными формата JSON, содержащими кирилицу?

*Нужно использоавть аргумент encoding со значением utf-8 в функции open при открытии JSON-файла.*

1. Самостоятельно ознакомьтесь со спецификацией JSON Schema? Что такое схема данных? Приведите схему данных для примера 1.

*Схема данных – это перечисление полей и их типов, которые должны быть в JSON. При несоблюдении названия, количества или типа полей, выдаётся ошибка валидации данных.*

*Схема данных для примера 1:*

schema = {  
 "type": "object",  
 "properties": {  
 "name": {"type": "string"},  
 "post": {"type": "string"},  
 "year": {"type": "integer"}  
 }  
}