

РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций
«Наследование и полиморфизм в языке Python»

Отчет по лабораторной работе № 4.4
по дисциплине «Объектно-ориентированное программирование»

Выполнил студент группы ИВТ-б-о-21-1
Толубаев Рамиль Ахметович

Подпись студента _____

Работа защищена « » _____ 20
_____ Г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

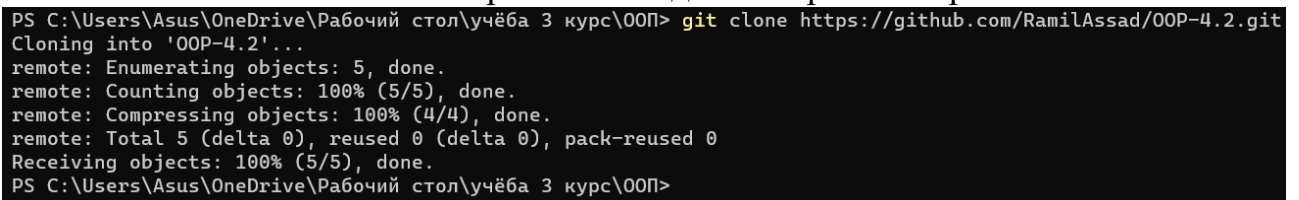
Цель работы: приобретение навыков по созданию иерархии классов при написании программ с помощью языка программирования Python версии 3.x

Порядок выполнения работы:

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.



```
PS C:\Users\Asus\OneDrive\Рабочий стол\учёба 3 курс\ООП> git clone https://github.com/RamilAssad/OOP-4.2.git
Cloning into 'OOP-4.2'...
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.
PS C:\Users\Asus\OneDrive\Рабочий стол\учёба 3 курс\ООП>
```

Рисунок 2 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

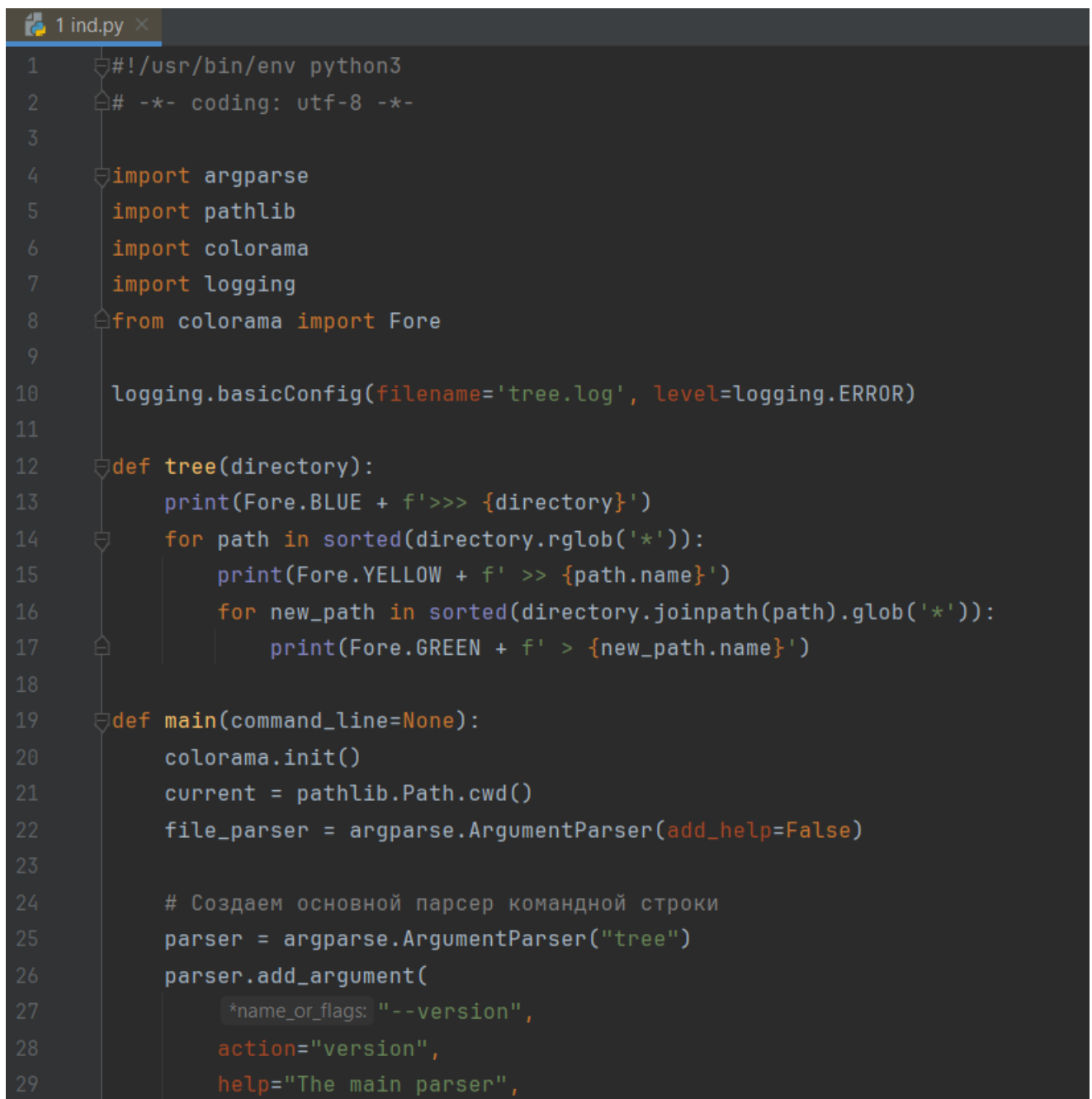
```
C:\Users\Asus\Desktop\Учеба\5 семестр\ООП\ООП_lw_4.1>git checkout -b develop
Switched to a new branch 'develop'

C:\Users\Asus\Desktop\Учеба\5 семестр\ООП\ООП_lw_4.1>
```

Рисунок 3 - Ветвление по модели git-flow

Задание 1. Вариант 15

Выполнить индивидуальное задание 1 лабораторной работы 2.19, добавив возможность работы с исключениями и логгирование.

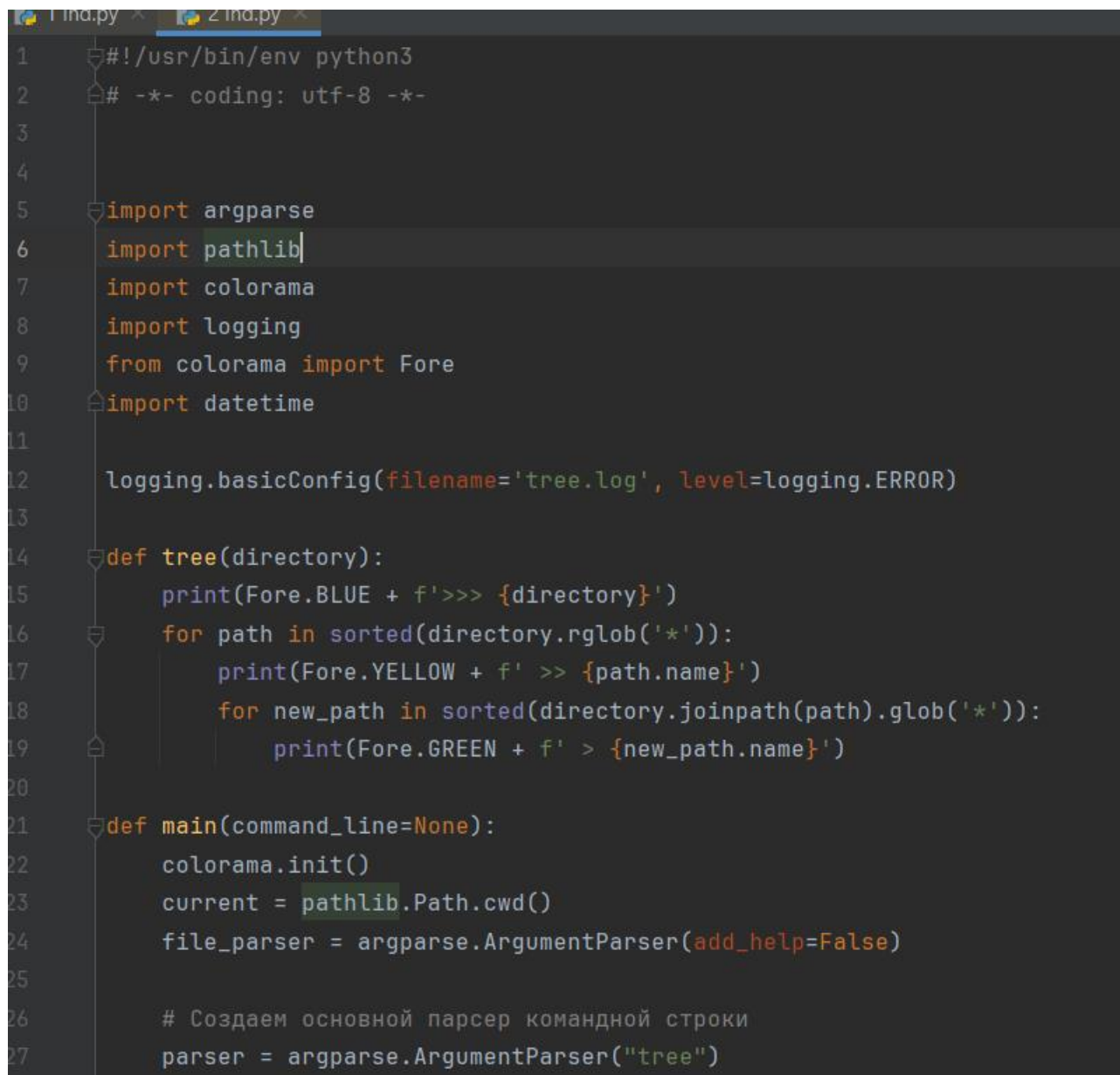


```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import argparse
5  import pathlib
6  import colorama
7  import logging
8  from colorama import Fore
9
10 logging.basicConfig(filename='tree.log', level=logging.ERROR)
11
12 def tree(directory):
13     print(Fore.BLUE + f'>>> {directory}')
14     for path in sorted(directory.rglob('*')):
15         print(Fore.YELLOW + f' >> {path.name}')
16         for new_path in sorted(directory.joinpath(path).glob('*')):
17             print(Fore.GREEN + f' > {new_path.name}')
18
19 def main(command_line=None):
20     colorama.init()
21     current = pathlib.Path.cwd()
22     file_parser = argparse.ArgumentParser(add_help=False)
23
24     # Создаем основной парсер командной строки
25     parser = argparse.ArgumentParser("tree")
26     parser.add_argument(
27         *name_or_flags: "--version",
28         action="version",
29         help="The main parser",
```

Рисунок 4 – Код индивидуального задания 1

Задание 2. Вариант 15

Добавить для предыдущего задания вывод в файлы лога даты и времени выполнения пользовательской команды с точностью до миллисекунды.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  import argparse
6  import pathlib
7  import colorama
8  import logging
9  from colorama import Fore
10 import datetime
11
12 logging.basicConfig(filename='tree.log', level=logging.ERROR)
13
14 def tree(directory):
15     print(Fore.BLUE + f'>>> {directory}')
16     for path in sorted(directory.rglob('*')):
17         print(Fore.YELLOW + f' >> {path.name}')
18         for new_path in sorted(directory.joinpath(path).glob('*')):
19             print(Fore.GREEN + f' > {new_path.name}')
20
21 def main(command_line=None):
22     colorama.init()
23     current = pathlib.Path.cwd()
24     file_parser = argparse.ArgumentParser(add_help=False)
25
26     # Создаем основной парсер командной строки
27     parser = argparse.ArgumentParser("tree")
```

Рисунок 5 – Код индивидуального задания 2

Контрольные вопросы:

1. Какие существуют виды ошибок в языке программирования Python?

В Python существуют различные виды ошибок, такие как синтаксические ошибки, исключения (ошибки выполнения программы), ошибки логики программы и другие.

2. Как осуществляется обработка исключений в языке программирования Python?

Обработка исключений в Python осуществляется с помощью конструкций try-except. Код, который может вызвать исключение, помещается в блок try, а обработчики

3. Для чего нужны блоки `finally` и `else` при обработке исключений?

Блок `finally` используется для выполнения кода независимо от того, возникло исключение или нет. Блок `else` используется для выполнения кода, если исключение не было вызвано.

4. Как осуществляется генерация исключений в языке Python?

Исключения могут быть сгенерированы с помощью ключевого слова `raise`, за которым следует имя исключения или экземпляр класса исключения.

5. Как создаются классы пользовательских исключений в языке Python?

Классы пользовательских исключений в Python создаются путем наследования от встроенного класса `Exception` или его подклассов.

6. Каково назначение модуля `logging`?

Модуль `logging` предназначен для регистрации информации (логгирования) о работе программы, включая сообщения об ошибках, предупреждения, информационные сообщения и другие.

7. Какие уровни логгирования поддерживаются модулем `logging`? Приведите примеры, в

которых могут быть использованы сообщения с этим уровнем журналирования.

Модуль `logging` поддерживает несколько уровней логгирования, такие как `DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`. Примеры использования: `DEBUG` — для отладочной информации, `INFO` — для информационных сообщений, `WARNING` — для предупреждений, `ERROR` — для сообщений об ошибках, `CRITICAL` — для критически важных ошибок.

