МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчёт о лабораторной работе №5 по дисциплине основы программной инженерии

Выполнила:
Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ

1. Практическая часть

```
# !/usr/bin/env python3
# -*- cosing: utf-8 -*-

import getopt, sys

full_cmd_argument = sys.argv

argument_list = full_cmd_argument[1:]
print(argument_list)
```

Рисунок 1 – Пример работы с модулем getopt

```
# !/usr/bin/env python3
# -*- cosing: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "square", type=int, help="display a square of a given number"
)
parser.add_argument(
    "-v", "--verbose", action="store_true", help="increase output verbosity"
)
args = parser.parse_args()

answer = args.square**2
if args.verbose:
    print("the square of {} equals {}".format(args.square, answer))
else:
    print(answer)
```

Рисунок 2 – Пример работы с модулем argparse, программа, возводящая в квадрат значение позиционного аргумента и формирующая вывод в зависимости от аргумента опционального

```
# !/usr/bin/env python3
# -*- cosing: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
subparsers = parser.add_subparsers(help='List of commands')

list_parser = subparsers.add_parser('list', help='list contents')
list_parser.add_argument('dirname', action='store', help='Directory to list')

create_parser = subparsers.add_parser('create', help='Create a directory')
create_parser.add_argument('dirname', action='store', help='new directory to create')
create_parser.add_argument(
    '--read-only',
    default=False,
    action='store_true',
    help='Set permissions to prevent writing to the directory'
)
```

Рисунок 3 – Пример работы с использованием субпарсеров

```
# !/usr/bin/env python3
# -*- cosing: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.parse_args()
```

Рисунок 4 – Пример простого разбора аргументов

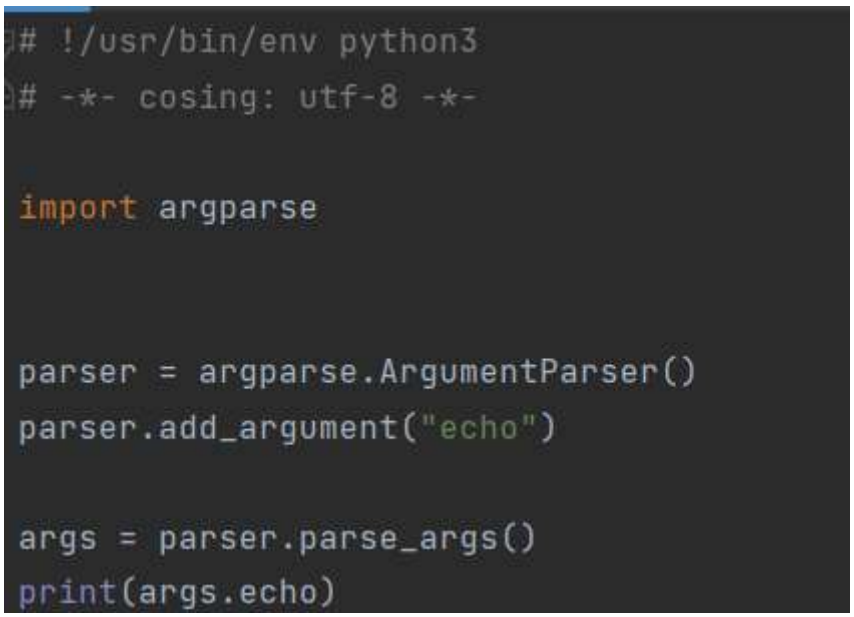


Рисунок 5 – Пример работы с позиционными аргументами

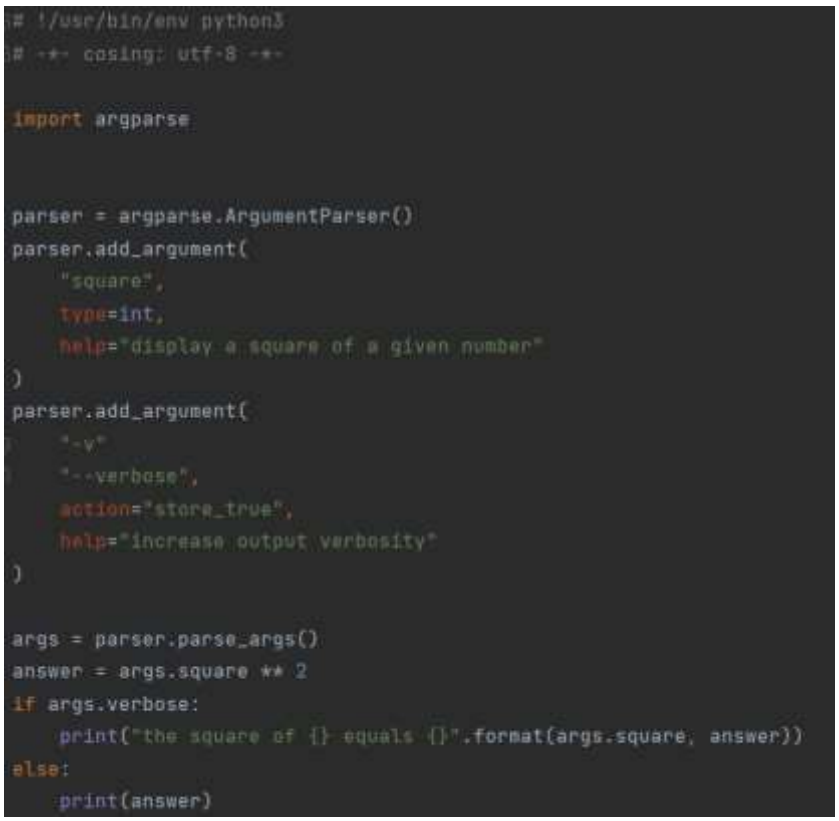
Рисунок 6 – Пример работы с опциональными агрументами

```
# !/usr/bin/env python3
# -*- cosing: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "--verbose",
    help="increase output verbosity",
    action="store_true"
)

args = parser.parse_args()
if args.verbose:
    print("verbosity turned on")
```

Рисунок 7 – Пример работы с булевской переменной

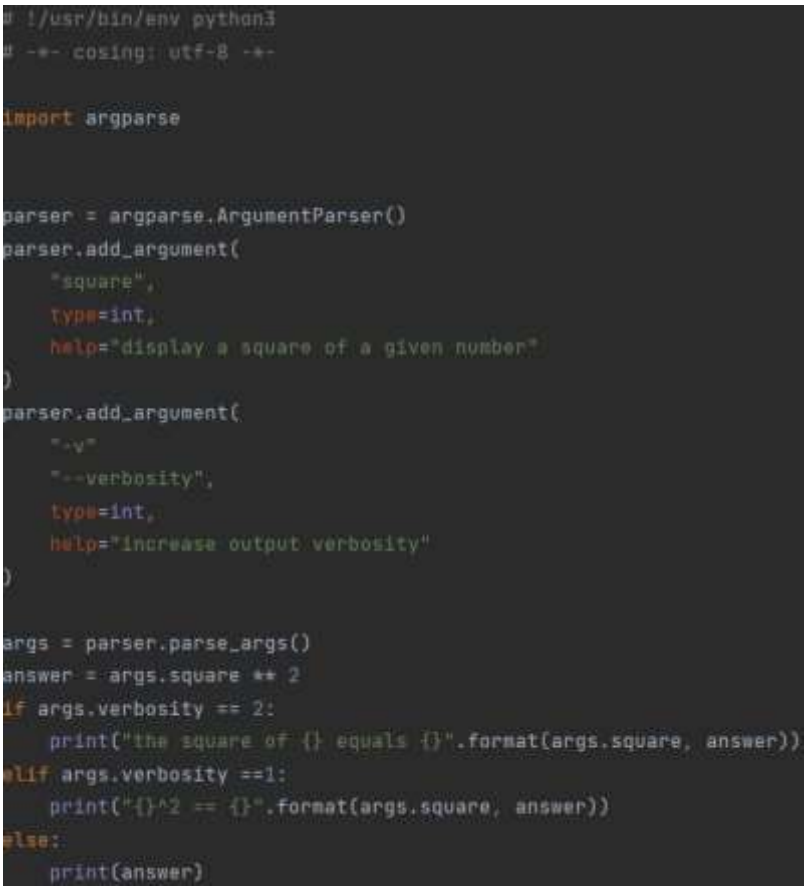
```
# !/usr/bin/env python3
# -*- cosing: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "-v"
    "--verbose",
    help="increase output verbosity",
    action="store_true"
)

args = parser.parse_args()
if args.verbose:
    print("verbosity turned on")
```

Рисунок 8 – Пример работы с короткими именами –v и -verbosity



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "square",
    type=int,
    help="display a square of a given number"
)
parser.add_argument(
    "-v"
    "--verbose",
    action="store_true",
    help="increase output verbosity"
)

args = parser.parse_args()
answer = args.square ** 2
if args.verbose:
    print("the square of {} equals {}".format(args.square, answer))
else:
    print(answer)
```

Рисунок 9 – Пример совместной работы с позиционными и опциональными аргументами



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "square",
    type=int,
    help="display a square of a given number"
)
parser.add_argument(
    "-v"
    "--verbosity",
    type=int,
    help="increase output verbosity"
)

args = parser.parse_args()
answer = args.square ** 2
if args.verbosity == 2:
    print("the square of {} equals {}".format(args.square, answer))
elif args.verbosity ==1:
    print("{}^2 == {}".format(args.square, answer))
else:
    print(answer)
```

Рисунок 10 – Пример работы с опциональными аргументами с параметром + позиционным аргументов

```
# !/usr/bin/env python3
# -*- cosing: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "square",
    type=int,
    help="display a square of a given number"
)
parser.add_argument(
    "-v"
    "--verbosity",
    type=int,
    choices=[0, 1, 2],
    help="increase output verbosity"
)

args = parser.parse_args()
answer = args.square ** 2
if args.verbosity == 2:
    print("the square of {} equals {}".format(args.square, answer))
elif args.verbosity == 1:
    print("{}^2 == {}".format(args.square, answer))
else:
    print(answer)
```

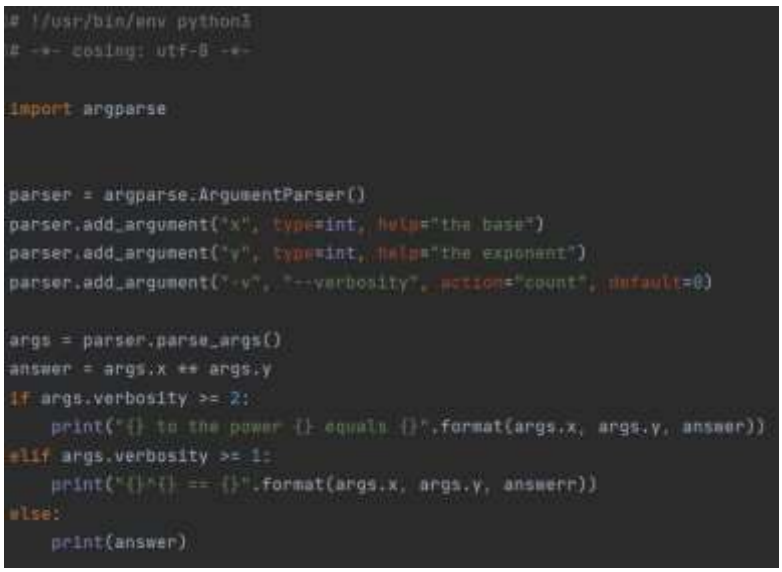
Рисунок 11 – Пример выбора значения из заранее определённого списка

```
# !/usr/bin/env python3
# -*- cosing: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "square",
    type=int,
    help="display a square of a given number"
)
parser.add_argument(
    "-v"
    "--verbosity",
    action="count",
    help="increase output verbosity"
)

args = parser.parse_args()
answer = args.square ** 2
if args.verbosity == 2:
    print("the square of {} equals {}".format(args.square, answer))
elif args.verbosity == 1:
    print("{}^2 == {}".format(args.square, answer))
else:
    print(answer)
```

Рисунок 12 – Пример подсчёта количества заданных аргументов –v action=’count’

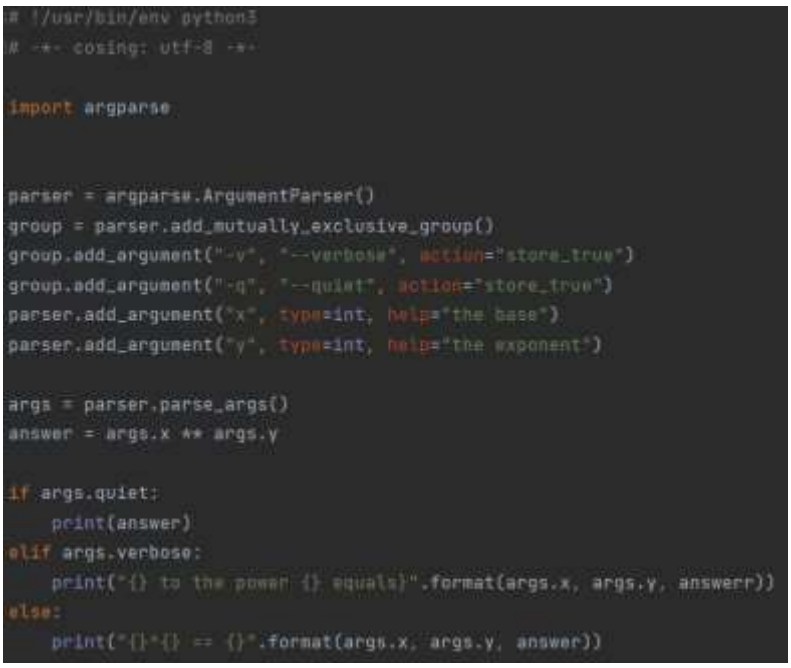
```
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument(
    "square",
    type=int,
    help="display a square of a given number"
)
parser.add_argument(
    "-v"
    "--verbosity",
    action="count",
    default=0,
    help="increase output verbosity"
)

args = parser.parse_args()
answer = args.square ** 2
if args.verbosity == 2:
    print("the square of {} equals {}".format(args.square, answer))
elif args.verbosity == 1:
    print("{}^2 == {}".format(args.square, answer))
else:
    print(answer)
```

Рисунок 13 – Пример добавления опции default=0, чтобы когда аргумент на задан, значение переменной было не None, а 0



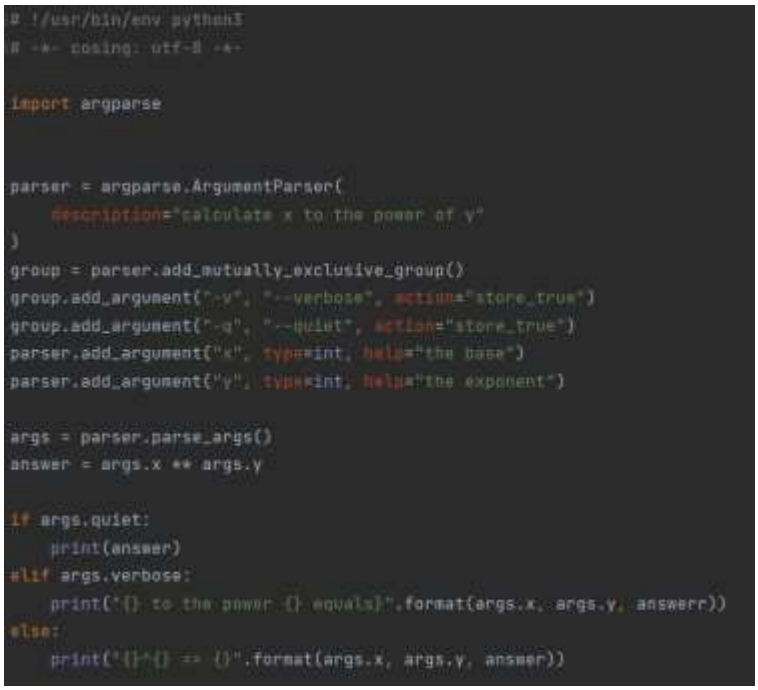
```
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
parser.add_argument("x", type=int, help="the base")
parser.add_argument("y", type=int, help="the exponent")
parser.add_argument("-v", "--verbosity", action="count", default=0)

args = parser.parse_args()
answer = args.x ** args.y
if args.verbosity >= 2:
    print("{} to the power {} equals {}".format(args.x, args.y, answer))
elif args.verbosity >= 1:
    print("{}^{} == {}".format(args.x, args.y, answerr))
else:
    print(answer)
```

Рисунок 14 – Пример того, что опциональный аргумент может содержать парметр



```
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse


parser = argparse.ArgumentParser()
group = parser.add_mutually_exclusive_group()
group.add_argument("-v", "--verbose", action="store_true")
group.add_argument("-q", "--quiet", action="store_true")
parser.add_argument("x", type=int, help="the base")
parser.add_argument("y", type=int, help="the exponent")

args = parser.parse_args()
answer = args.x ** args.y

if args.quiet:
    print(answer)
elif args.verbose:
    print("{} to the power {} equals}".format(args.x, args.y, answerr))
else:
    print("{}^{} == {}".format(args.x, args.y, answer))
```

Рисунок 15 – Пример программы, которая возводит не в квадрат, а в указанную степень



```
# !/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse


parser = argparse.ArgumentParser(
    description="calculate x to the power of y"
)
group = parser.add_mutually_exclusive_group()
group.add_argument("-v", "--verbose", action="store_true")
group.add_argument("-q", "--quiet", action="store_true")
parser.add_argument("x", type=int, help="the base")
parser.add_argument("y", type=int, help="the exponent")

args = parser.parse_args()
answer = args.x ** args.y

if args.quiet:
    print(answer)
elif args.verbose:
    print("{} to the power {} equals}".format(args.x, args.y, answerr))
else:
    print("{}^{} == {}".format(args.x, args.y, answer))
```

Рисунок 16 – Пример взаимоисключающих аргументов

```
PS C:\Users\тома нестеренко\PycharmProjects\1> python main.py add file.json -s Sviridov -n Maksim -f Rak -d 17.01.2002
```

Рисунок 17 – Пример ввода команды для индивидуального задания

Рисунок 18 – Пример результата работы программы индивидуального задания

2. Вопросы для защиты

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”.

Терминал, это уже надстройка над консолью и под собой больше подразумевает удалённый доступ с мало мощной машины.

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?

Python 3 поддерживает несколько различных способов обработки аргументов командной строки. Встроенный способ – использовать модуль sys . С точки зрения имен и использования, он имеет прямое отношение к библиотеке C (libc). Второй

способ – это модуль getopt, который обрабатывает как короткие, так и длинные параметры, включая оценку значений параметров.

Кроме того, существуют два других общих метода. Это модуль argparse , производный от модуля optparse, доступного до Python 2.7. Другой метод – использование модуля docopt, доступного на GitHub.

4. Какие особенности построение CLI с использованием модуля sys?

Это базовый модуль, который с самого начала поставлялся с Python. Он использует подход, очень похожий на библиотеку C, с использованием argc и argv для доступа к аргументам.

Модуль sys реализует аргументы командной строки в простой структуре списка с именем sys.argv. Каждый элемент списка представляет собой единственный аргумент. Первый элемент в списке sys.argv [0] – это имя скрипта Python. Остальные элементы списка, от sys.argv [1] до sys.argv [n] , являются аргументами командной строки с 2 по n. В качестве разделителя между аргументами используется пробел. Значения аргументов, содержащие пробел, должны быть заключены в кавычки, чтобы их правильно проанализировал sys. Эквивалент argc – это просто количество элементов в списке.

5. Какие особенности построение CLI с использованием модуля getopt?

Модуль getopt в Python идет немного дальше и расширяет разделение входной строки проверкой параметров. Основанный на функции C getopt, он позволяет использовать как короткие, так и длинные варианты, включая присвоение значений.

6. Какие особенности построение CLI с использованием модуля argparse?

Для начала рассмотрим, что интересного предлагает argparse:

- анализ аргументов sys.argv ;
- конвертирование строковых аргументов в объекты Вашей программы и работа с ними;
- форматирование и вывод информативных подсказок.

Ссылка на репозиторий: https://github.com/tamaranesterenko/Python_LR_5-2