

**МИНИСТЕРСТВО 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 №2.1

Дисциплина: «Программирование на Python»

Тема: «Основы языка Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-22-1

Сумин Никита Сергеевич

Ставрополь 2023


Выполнение работы:

1. Создал репозиторий GitHub с лицензией MIT, добавил .gitignore с ЯП python, клонировал репозиторий на ПК и организовал репозиторий согласно модели ветвления git-flow:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 BrandooDi ▾

Repository name *

/ mywork4 ✓

Great repository names are short and memorable. Need inspiration? How about [fluffy-octo-winner](#)?

Description (optional)



 Public

Anyone on the internet can see this repository. You choose who can commit.



 Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

Choose a license

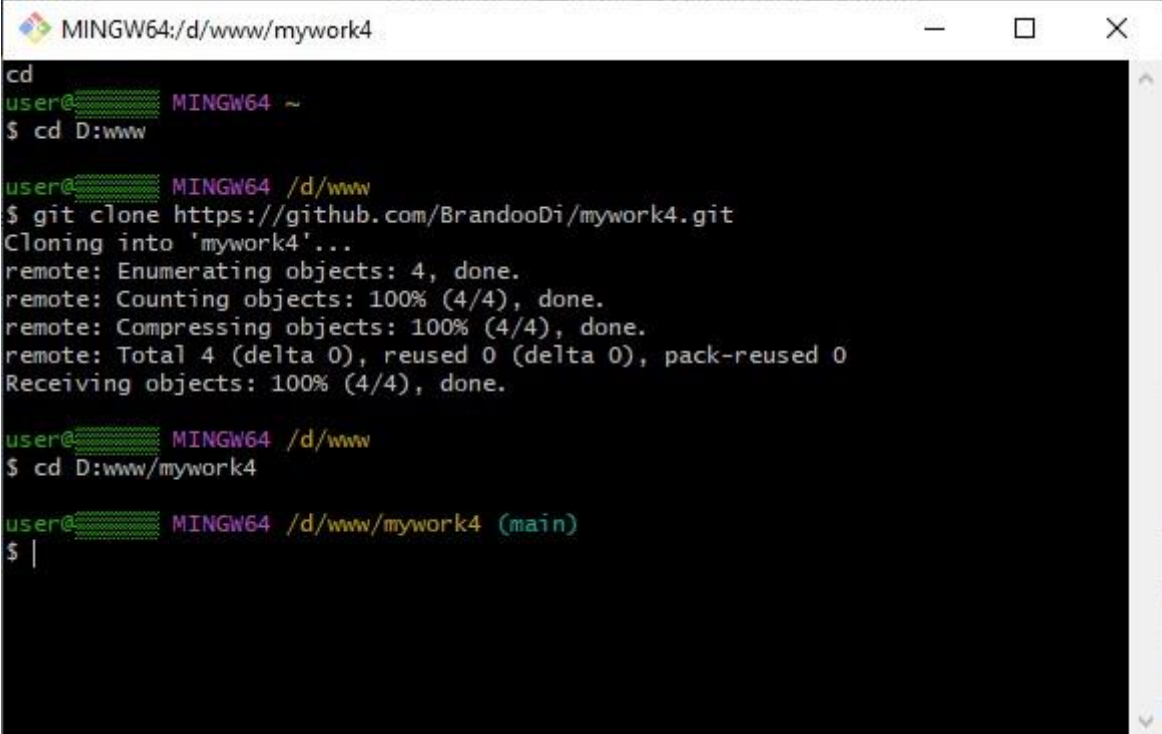
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

 You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 Создание репозитория



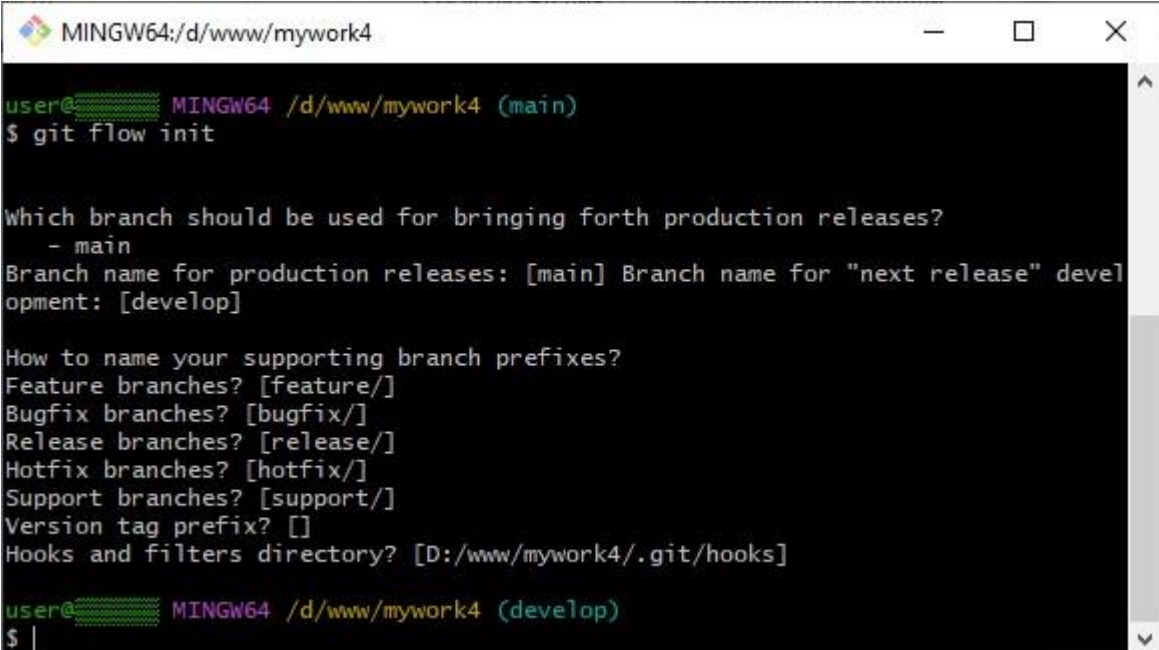
```
MINGW64:/d/www/mywork4
cd
user@MINGW64 ~
$ cd D:\www

user@MINGW64 /d/www
$ git clone https://github.com/BrandooDi/mywork4.git
Cloning into 'mywork4'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

user@MINGW64 /d/www
$ cd D:\www/mywork4

user@MINGW64 /d/www/mywork4 (main)
$ |
```

Рисунок 1.2 Клонирование репозитория



```
MINGW64:/d/www/mywork4
user@MINGW64 /d/www/mywork4 (main)
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main] Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/www/mywork4/.git/hooks]

user@MINGW64 /d/www/mywork4 (develop)
$ |
```

Рисунок 1.3 Организация репозитория согласно модели ветвления git-flow

2. Написал программу user.py, которая запрашивала бы у пользователя имя, возраст и место жительства, после этого выводила бы 3 строки

"This is `имя`"

"It is `возраст`"

"(S)he live in `место_жительства`"

The image shows a screenshot of the Python IDLE 3.10.4 environment. The top window, titled 'user.py - D:/www/mywork4/user.py (3.10.4)', contains the following Python code:

```
name = input("Write your name:\n")
age = input("Write your age:\n")
place = input("Write your place of birth:\n")

print("\nThis is", name)
print("It's ", age)
print("(S)he live in ", place)
```

The bottom window, titled 'IDLE Shell 3.10.4', shows the execution of this script. It displays the Python version and system information, followed by a restart message for the script. The user's input is shown in blue text, and the program's output is in black text.

```
>>> Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: D:/www/mywork4/user.py =====
Write your name:
Nikita
Write your age:
18
Write your place of birth:
Kaluga

This is Nikita
It's 18
(S)he live in Kaluga
>>> |
```

Рисунок 2.1 Программа user в репозитории

4. . Написал программу (файл arithmetic.py), которая предлагала бы пользователю решить пример $4 * 100 - 54$. Потом выводила бы на экран правильный ответ и ответ пользователя.

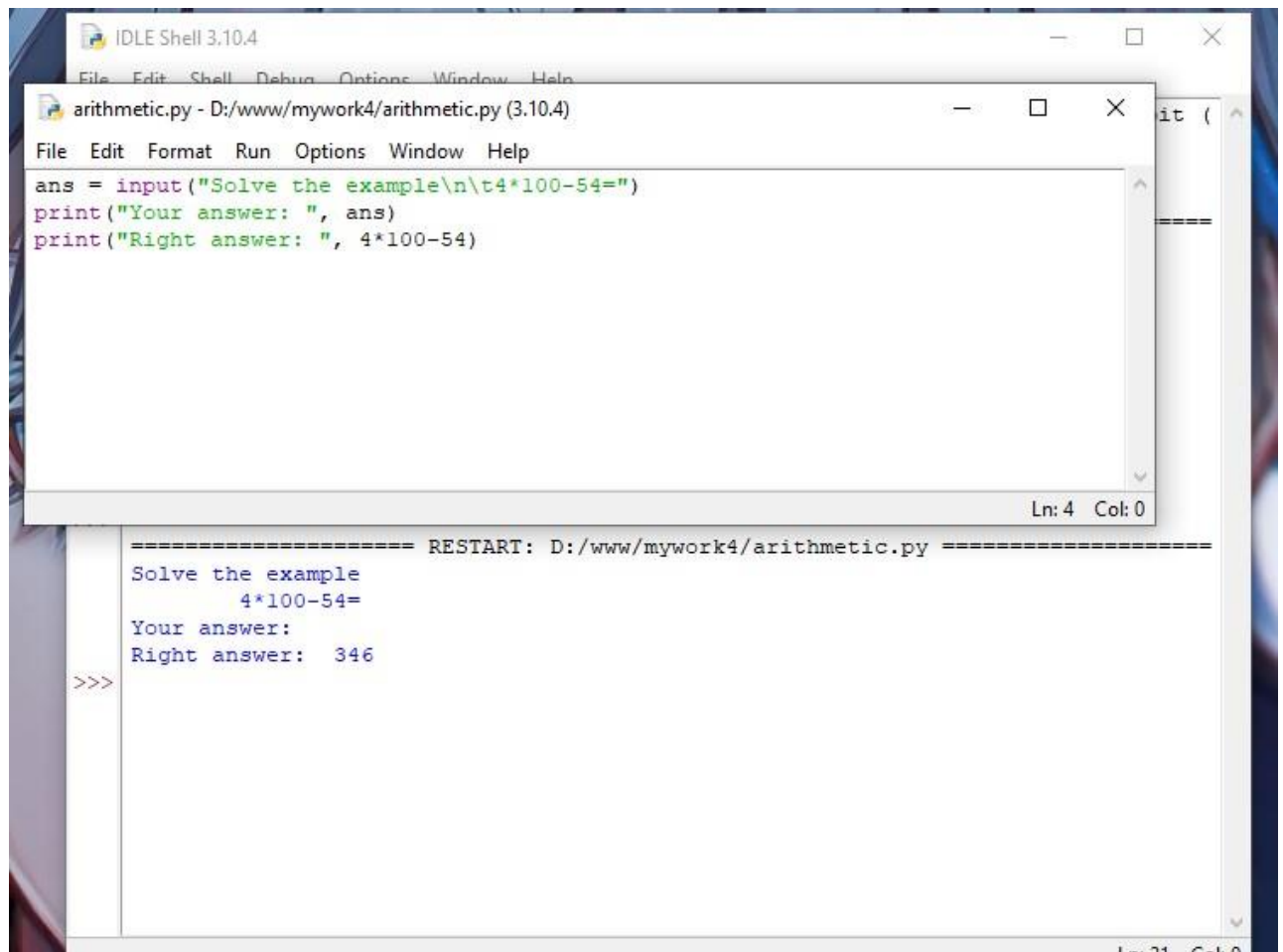


Рисунок 2.2 Программа `arithmetic.py`

5. Написал программу `numbers.py`, которая запрашивает у пользователя 4 числа, отдельно складывает первые два и вторые два, затем делит первую сумму на вторую, после выводит рез-т на экран с точностью до сотен.

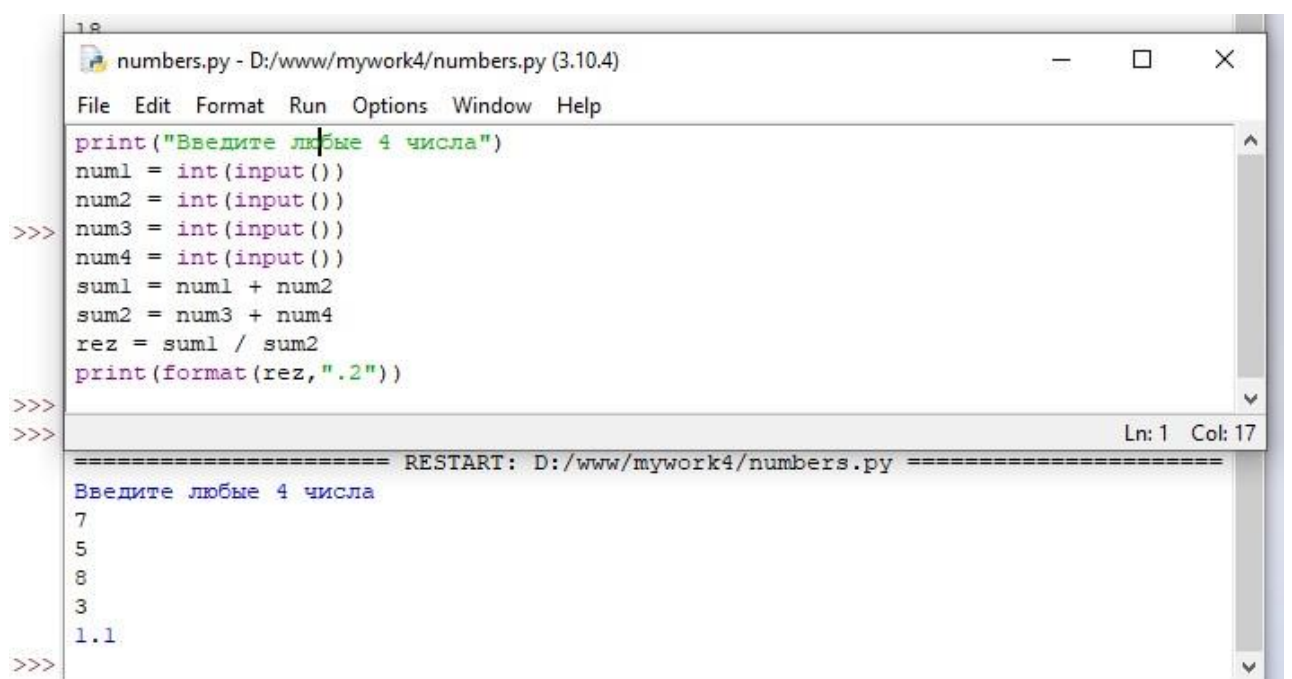
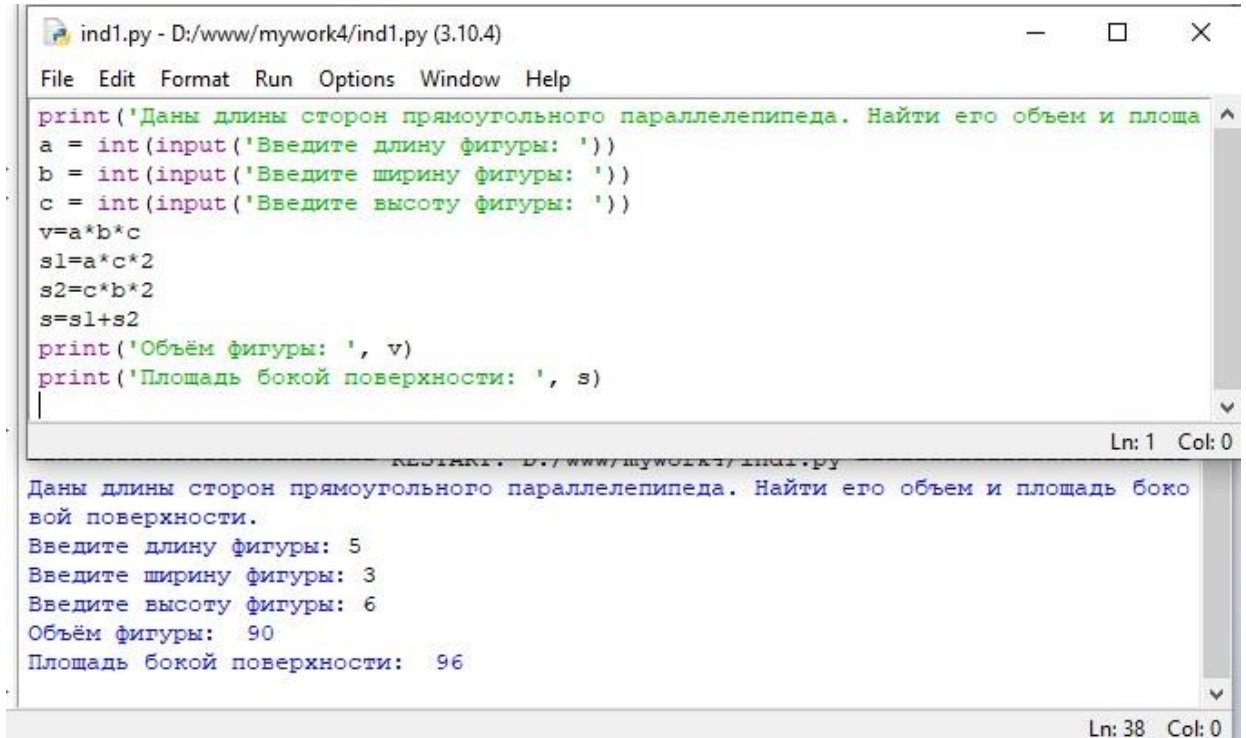


Рисунок 2.3 Программа numbers.py

5. Написал программу для индивидуального задания:

Даны длины сторон прямоугольного параллелепипеда. Найти его объем и площадь боковой поверхности.



```
ind1.py - D:/www/mywork4/ind1.py (3.10.4)
File Edit Format Run Options Window Help
print('Даны длины сторон прямоугольного параллелепипеда. Найти его объем и площадь боковой поверхности.')
a = int(input('Введите длину фигуры: '))
b = int(input('Введите ширину фигуры: '))
c = int(input('Введите высоту фигуры: '))
v=a*b*c
s1=a*c*2
s2=c*b*2
s=s1+s2
print('Объем фигуры: ', v)
print('Площадь боковой поверхности: ', s)
```

Ln: 1 Col: 0

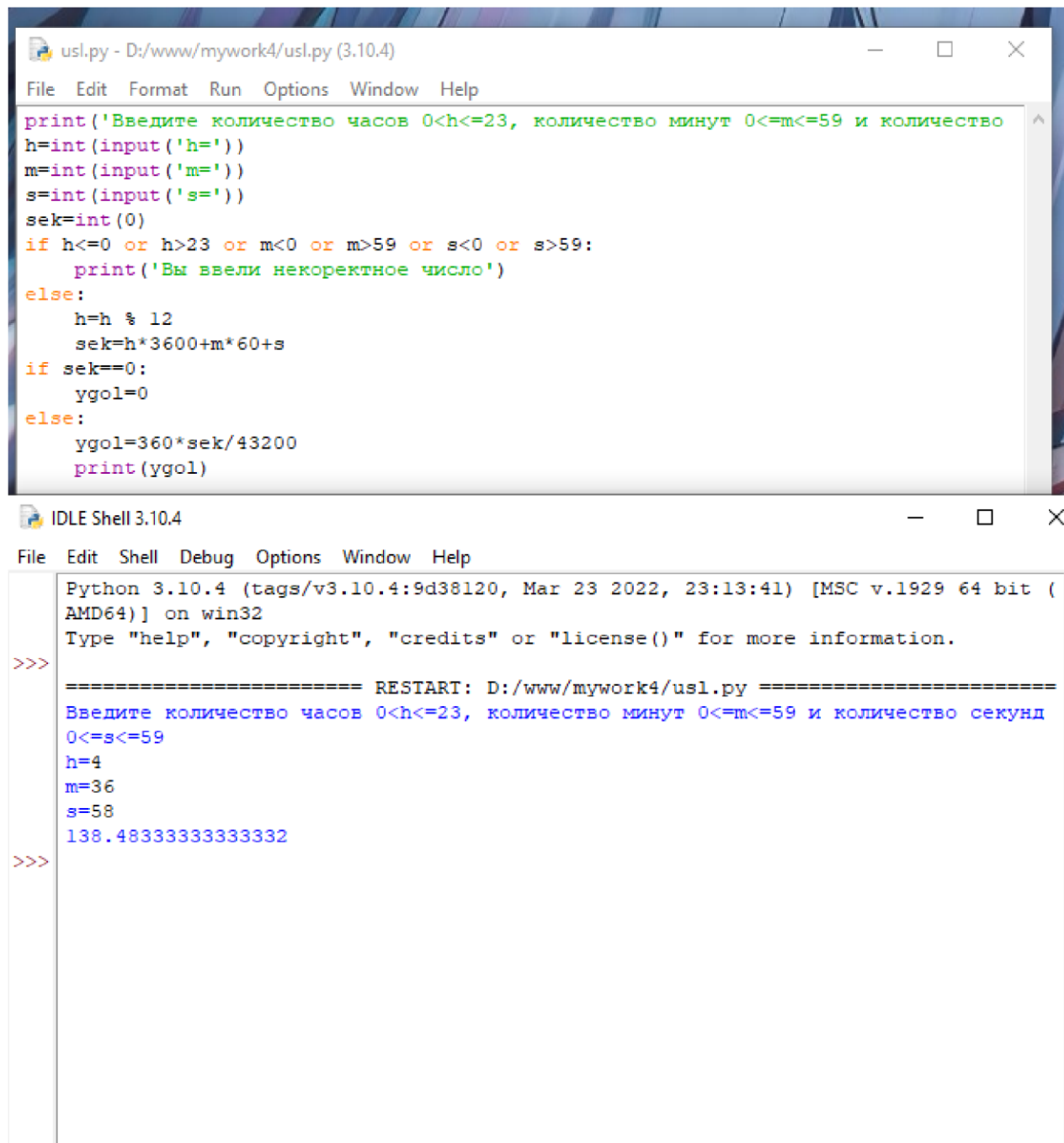
```
Даны длины сторон прямоугольного параллелепипеда. Найти его объем и площадь боковой поверхности.
Введите длину фигуры: 5
Введите ширину фигуры: 3
Введите высоту фигуры: 6
Объем фигуры: 90
Площадь боковой поверхности: 96
```

Ln: 38 Col: 0

Рисунок 2.4 Программа индивидуального задания

6. Написал программу для задачи повышенной сложности

Рисунок 2.5 Задача повышенной сложности



The image shows two windows from a Python IDE. The top window, titled 'usl.py - D:/www/mywork4/usl.py (3.10.4)', contains the following Python code:

```
print('Введите количество часов 0<h<=23, количество минут 0<=m<=59 и количество секунд 0<=s<=59')
h=int(input('h='))
m=int(input('m='))
s=int(input('s='))
sek=int(0)
if h<=0 or h>23 or m<0 or m>59 or s<0 or s>59:
    print('Вы ввели некорректное число')
else:
    h=h % 12
    sek=h*3600+m*60+s
if sek==0:
    ygol=0
else:
    ygol=360*sek/43200
    print(ygol)
```

The bottom window, titled 'IDLE Shell 3.10.4', shows the execution output:

```
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/www/mywork4/usl.py =====
Введите количество часов 0<h<=23, количество минут 0<=m<=59 и количество секунд 0<=s<=59
h=4
m=36
s=58
138.48333333333332
>>>
```

6. Сделал коммит изменений в ветку разработки, выполнил ее слияние с веткой main и отправил сделанные изменения на уд. репозиторий.


```
MINGW64:/d/www/mywork4

user@MINGW64 /d/www/mywork4 (develop)
$ git add .

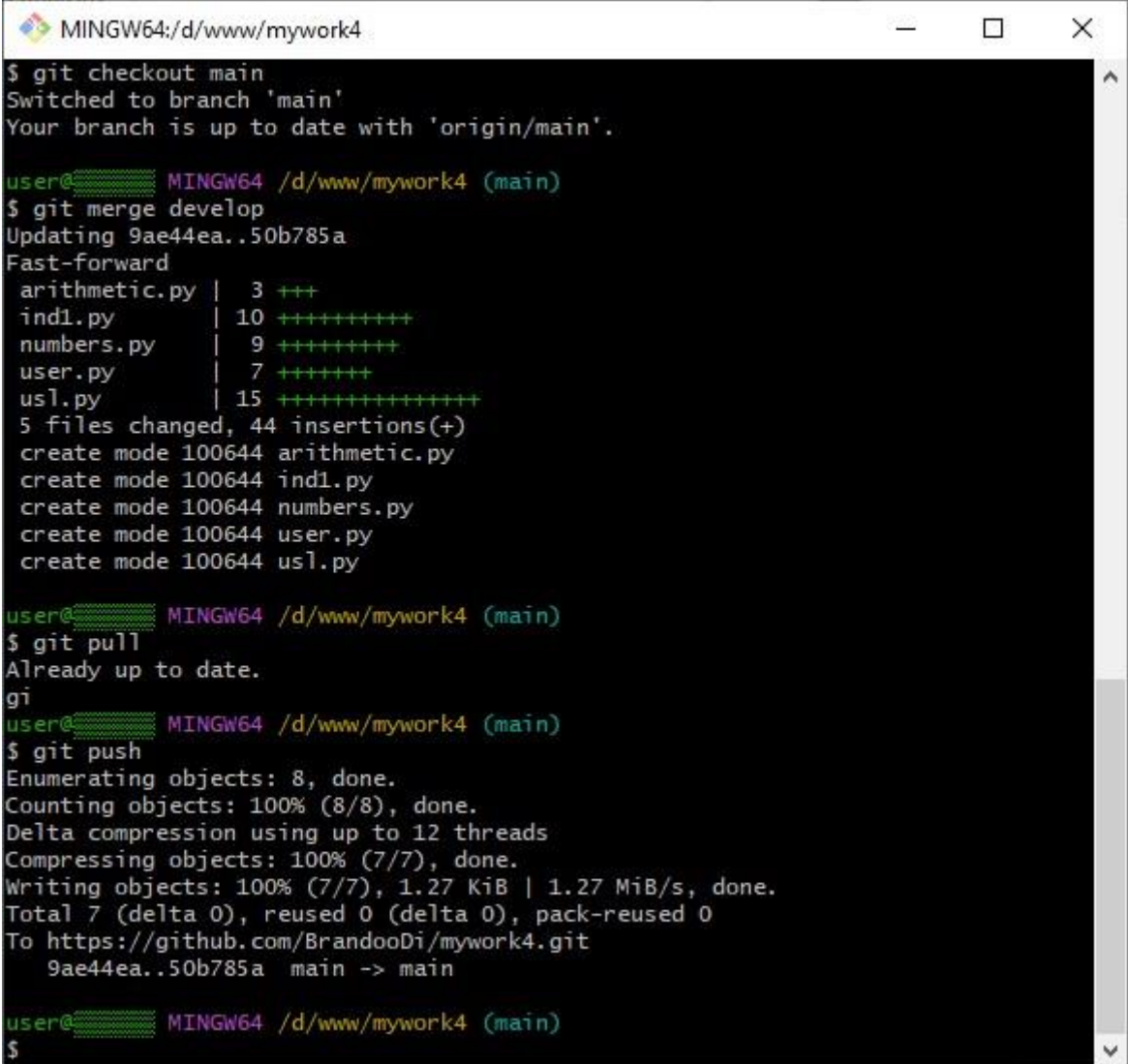
user@MINGW64 /d/www/mywork4 (develop)
$ git commit -m "Programs"
[develop 50b785a] Programs
 5 files changed, 44 insertions(+)
 create mode 100644 arithmetic.py
 create mode 100644 ind1.py
 create mode 100644 numbers.py
 create mode 100644 user.py
 create mode 100644 usl.py

user@MINGW64 /d/www/mywork4 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

user@MINGW64 /d/www/mywork4 (main)
$ git merge develop
Updating 9ae44ea..50b785a
Fast-forward
 arithmetic.py | 3 +++
 ind1.py       | 10 ++++++++
 numbers.py    | 9 ++++++++
 user.py       | 7 ++++++
 usl.py        | 15 ++++++++
 5 files changed, 44 insertions(+)
 create mode 100644 arithmetic.py
 create mode 100644 ind1.py
 create mode 100644 numbers.py
 create mode 100644 user.py
 create mode 100644 usl.py

user@MINGW64 /d/www/mywork4 (main)
$
```

Рисунок 3.1 Слияние ветки develop с веткой main



```
MINGW64:/d/www/mywork4
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

user@MINGW64 /d/www/mywork4 (main)
$ git merge develop
Updating 9ae44ea..50b785a
Fast-forward
 arithmetic.py | 3 +++
 ind1.py       | 10 ++++++++
 numbers.py    | 9 ++++++++
 user.py       | 7 ++++++
 us1.py        | 15 ++++++++
 5 files changed, 44 insertions(+)
 create mode 100644 arithmetic.py
 create mode 100644 ind1.py
 create mode 100644 numbers.py
 create mode 100644 user.py
 create mode 100644 us1.py

user@MINGW64 /d/www/mywork4 (main)
$ git pull
Already up to date.

user@MINGW64 /d/www/mywork4 (main)
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.27 KiB | 1.27 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/BrandooDi/mywork4.git
 9ae44ea..50b785a main -> main

user@MINGW64 /d/www/mywork4 (main)
$
```

Рисунок 3.2 Пуш файлов

1. Опишите основные этапы установки Python в Windows и Linux.

Linux: Чаще всего интерпретатор Python уже входит в состав дистрибутива.

Windows: Оsn. этапы установки Python на Windows:

- 1) Скачать дистрибутив с официального сайта;
- 2) Запустить скачанный установочный файл;
- 3) Выбрать способ установки;
- 4) Отметить необходимые опции установки; 5) Выбрать место установки; 6) Готово.

2. В чем отличие пакета Anaconda от пакета Python, скачиваемого с официального сайта?

Пакет Anaconda содержит версии языка Python 2 и 3, набор наиболее часто используемых библиотек и удобную среду разработки и исполнения, запускаемую в браузере, а также на Anaconda удобнее запускать примеры.

3. Как осуществить проверку работоспособности пакета Anaconda?

Для выполнения проверки работоспособности Anaconda необходимо вначале запустить командный процессор с поддержкой виртуальных окружений Anaconda. В появившейся командной строке необходимо ввести `> jupyter notebook`, в результате чего отобразится процесс загрузки веб-среды Jupyter Notebook, после чего запустится веб-сервер и среда разработки в браузере. Создать ноутбук для разработки, для этого нажать на кнопку New и в появившемся списке выбрать Python. В результате будет создана новая страница в браузере с ноутбуком. Ввести в первой ячейке команду `print("Hello, World!")` и нажать `Alt+Enter` на компьютере. Ниже ячейки должна появиться соответствующая надпись.

4. Как задать используемый интерпретатор языка Python в IDE PyCharm?

Указать путь до интерпретатора в настройках IDE, для этого:

- 1) Нажмите на шестеренку в верхнем правом углу, выберите "Add..".
- 2) Далее выберите "System Interpreter";
- 3) Нажмите на 3 точки "..." справа от поля в выборе интерпретатора;
- 4) Укажите путь до интерпретатора.

5. Как осуществить запуск программы с помощью IDE PyCharm? Сочетанием клавиш Shift+F10.

6. В чем суть интерактивного и пакетного режимов работы

Python?

Интерактивный.

Python можно использовать как калькулятор для различных вычислений, а если дополнительно подключить необходимые математические библиотеки, то по своим возможностям он становится практически равным таким пакетам как Matlab, Octave и т.п.

Проектный.

В этом режиме сначала записывается вся программа, а потом эта программа выполняется полностью.

7. Почему язык программирования Python называется языком динамической типизации?

Т. к. в ЯП Python проверка типа происходит во время выполнения, а не компиляции.

8. Какие существуют основные типы в языке программирования Python?

Типы в ЯП Python:

1. None
2. Логические переменные
3. Числа
4. Списки
5. Строки
6. Бинарные списки
7. Множества
8. Словари
9. Как создаются объекты в памяти? Каково их устройство?

В чем заключается процесс объявления новых переменных и работа операции присваивания?

Для того, чтобы объявить и сразу инициализировать переменную необходимо написать её имя, потом поставить знак равенства и значение, с которым эта переменная будет создана.

При инициализации переменной, на уровне интерпретатора, создается целочисленный объект, который имеет некоторый идентификатор, значение и тип. Посредством оператора “=” создается ссылка между переменной и объектом.

10. Как получить список ключевых слов в Python?

Список ключевых слов можно получить непосредственно в программе, для этого нужно подключить модуль keyword и воспользоваться командой keyword.kwlist.

11. Каково назначение функций id() и type()?

Функция id() предназначена для получения значения идентичности объекта.

С помощью функции type() можно получить тип конкретного объекта.

12. Что такое изменяемые и неизменяемые типы в Python.

К неизменяемым (immutable) типам относятся: целые числа (int), числа с плавающей точкой (float), комплексные числа (complex), логические переменные (bool), кортежи (tuple), строки (str) и неизменяемые множества (frozen set).

К изменяемым (mutable) типам относятся: списки (list), множества (set), словари (dict).

13. Чем отличаются операции деления и целочисленного деления?

При целочисленном делении отбрасывается дробная часть от деления чисел, при операции деления дробная часть не отбрасывается.

14. Какие имеются средства в языке Python для работы с комплексными числами?

Для создания комплексного числа можно использовать функцию `complex(a, b)`, в которую, в качестве первого аргумента, передается действительная часть, в качестве второго – мнимая.

Либо записать число в виде $a + bj$. Комплексные числа можно складывать, вычитать, умножать, делить и возводить в степень. У комплексного числа можно извлечь действительную (`x.real`) и мнимую части (`x.imag`).

Для получения комплексносопряженного числа необходимо использовать метод `conjugate()`.

15. Каково назначение и основные функции библиотеки (модуля) `math`? По аналогии с модулем `math` изучите самостоятельно назначение и основные функции модуля `cmath`.

Для выполнения математических операций необходим модуль `math`.

Осн. операции библиотеки `math`:

`math.ceil(x)` - возвращает ближайшее целое число большее, чем x .
`math.fabs(x)` - возвращает абсолютное значение числа. `math.factorial(x)` - вычисляет факториал x . `math.floor(x)` - возвращает ближайшее целое число меньшее, чем x . `math.exp(x)` - вычисляет e^x . `math.log2(x)` - логарифм по основанию 2. `math.log10(x)` - логарифм по основанию 10. `math.log(x[, base])` - по умолчанию вычисляет логарифм по основанию e , дополнительно можно указать основание логарифма. `math.pow(x, y)` - вычисляет значение x в степени y . `math.sqrt(x)` - корень квадратный от x . `math.cos(x)` - косинус от x . `math.sin(x)` - синус от x . `math.tan(x)` - тангенс от x .

`math.acos(x)` - арккосинус от x . `math.asin(x)` - арксинус от x . `math.atan(x)` - арктангенс от x .

`math.pi` - число пи. `math.e` - число e .

16. Каково назначение именных параметров `sep` и `end` в функции `print()`?

Через параметр `sep` можно указать отличный от пробела разделитель строк.

Параметр `end` позволяет указывать, что делать, после вывода строки.

17. Каково назначение метода `format()`? Какие еще существуют средства для форматирования строк в Python? Примечание: в дополнение к рассмотренным средствам изучите самостоятельно работу с f-строками в Python.

Форматирование может выполняться в так называемом старом стиле или с помощью строкового метода `format`.

Символы `%s`, `%d`, `%f` подставляются значения переменных. Буквы `s`, `d`, `f` обозначают типы данных – строку, целое число, вещественное число.

18. Каким образом осуществить ввод с консоли значения целочисленной и вещественной переменной в языке Python?

Указать перед `input` тип данных: `int(input())`.