

Министерство науки и высшего образования Российской Федерации

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4 дисциплины «Основы программной инженерии»

Выполнил:
Звездин Алексей Сергеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

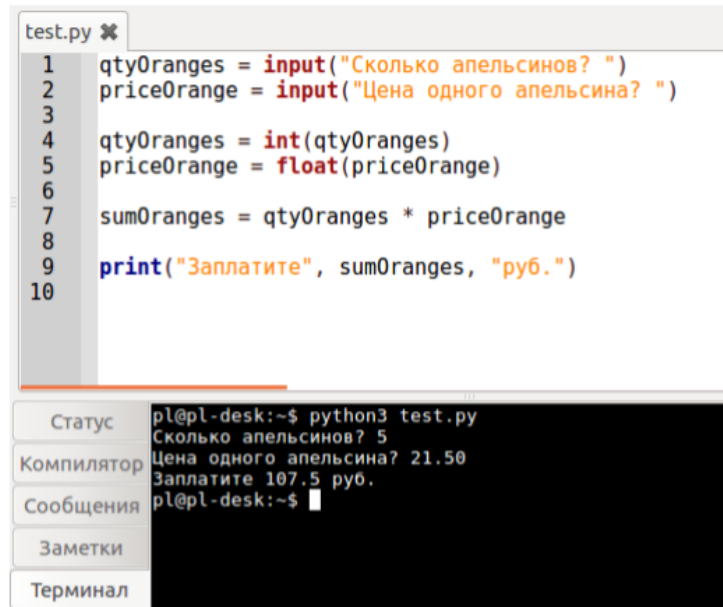
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Я изучил теоретический материал работы



```
test.py ✖
1 qtyOranges = input("Сколько апельсинов? ")
2 priceOrange = input("Цена одного апельсина? ")
3
4 qtyOranges = int(qtyOranges)
5 priceOrange = float(priceOrange)
6
7 sumOranges = qtyOranges * priceOrange
8
9 print("Заплатите", sumOranges, "руб.")
10
```

Статус: pl@pl-desk:~\$ python3 test.py
Компилятор: Сколько апельсинов? 5
Сообщения: Цена одного апельсина? 21.50
Заметки: Заплатите 107.5 руб.
Терминал: pl@pl-desk:~\$

В данном случае с помощью функций `int()` и `float()` строковые значения переменных `qtyOranges` и `priceOrange` преобразуются соответственно в целое число и вещественное число. После этого новые численные значения присваиваются тем же переменным.

Программный код можно сократить, если преобразование типов выполнить в тех же строках кода, где вызывается функция `input()`:

```
qtyOranges = int(input("Сколько апельсинов? "))
priceOrange = float(input("Цена одного апельсина? "))

sumOranges = qtyOranges * priceOrange

print("Заплатите", sumOranges, "руб.")
```

Сначала выполняется функция `input()`. Она возвращает строку, которую функция `int()` или `float()` сразу преобразует в число. Только после этого происходит присваивание переменной, то есть она сразу получает численное значение.

Рисунок 1.1 – Изучение материала для лабораторной работы


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

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *

 InternetHacker1123 ▾

Repository name *

/ Laba4_v1

✔ Laba4_v1 is available.

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

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:



Add a README file

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

Add .gitignore

.gitignore template: Python ▾

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

Choose a license

License: MIT License ▾

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

This will set  main as the default branch. Change the default name in your [settings](#).



You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

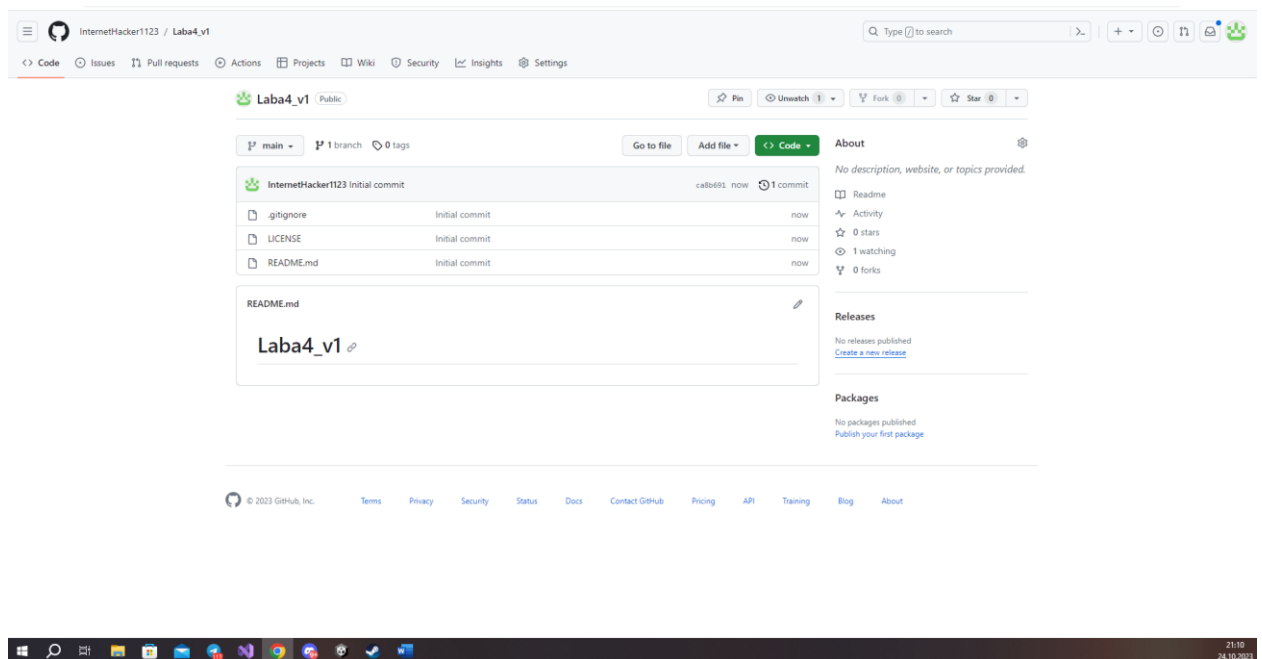


Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория

```

C:\Windows\system32\cmd.exe
20.04.2023 09:50 <DIR> онлайн школа
25.05.2023 18:02 <DIR> от винта webgl
25.05.2023 18:30 <DIR> От винта! webgl Ru
25.05.2023 18:38 11 380 347 От винта! webgl Ru.zip
25.05.2023 09:08 1 594 От винта!.lnk
23.05.2023 21:50 3 224 156 Презентация_История_Звездин.pptx
25.05.2023 09:08 <DIR> Самолеты
28.05.2023 23:05 <DIR> скфу
27.05.2023 16:30 3 884 070 скфу.zip
04.05.2023 10:58 <DIR> швейка
17 файлов 19 633 467 байт
10 папок 19 694 538 752 байт свободно

C:\Users\tyt\Desktop>mkdir SE
C:\Users\tyt\Desktop>cd SE
C:\Users\tyt\Desktop\SE>mkdir Laba4_v1
C:\Users\tyt\Desktop\SE>cd Laba4_v1
C:\Users\tyt\Desktop\SE\Lab4_v1>git clone https://github.com/InternetHacker1123/Laba4_v1.git
Cloning into 'Lab4_v1'...
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.
C:\Users\tyt\Desktop\SE\Lab4_v1>
  
```

Рисунок 3.1 – Клонирование репозитория на локальный диск

4. Дополнил файл .gitignore необходимыми правилами для работы с VScode

```
.vscode/*
!.vscode/settings.json
!.vscode/tasks.json
!.vscode/launch.json
!.vscode/extensions.json
!.vscode/*.code-snippets

# Local History for Visual Studio Code
.history/

# Built Visual Studio Code Extensions
*.vsix
```

Рисунок 4.1 – .gitignore для VScode

5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow

```
C:\TestGit\AgainPython>git branch develop

C:\TestGit\AgainPython>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/MrFinicheck/AgainPython/pull/new/develop
remote:
To https://github.com/MrFinicheck/AgainPython.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\TestGit\AgainPython>git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
```

Рисунок 5.1 – Создание ветки develop от ветки main

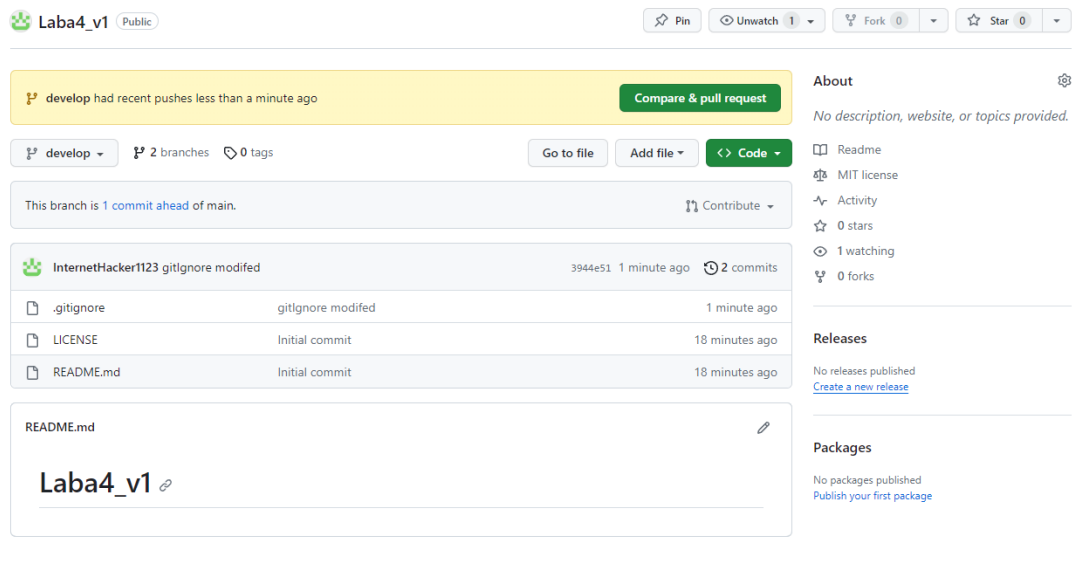


Рисунок 5.2 – Ветка develop на GitHub

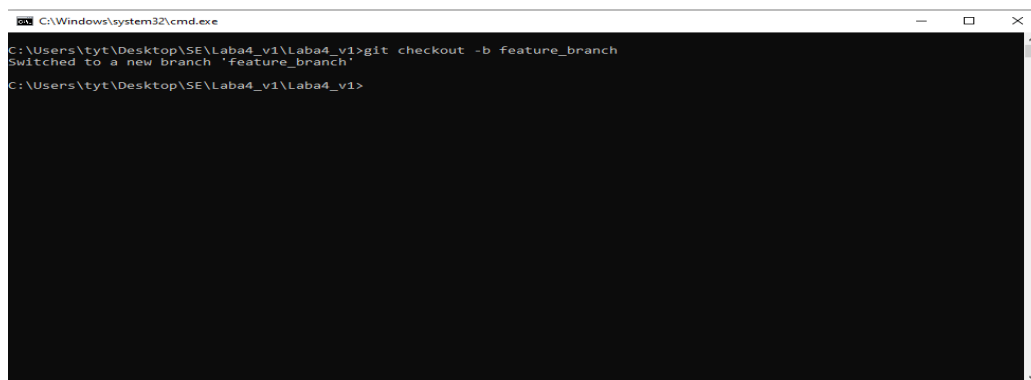


Рисунок 5.3 – Создание ветки feature_branch от ветки develop

6. Создал проект PyCharm в папке репозитория

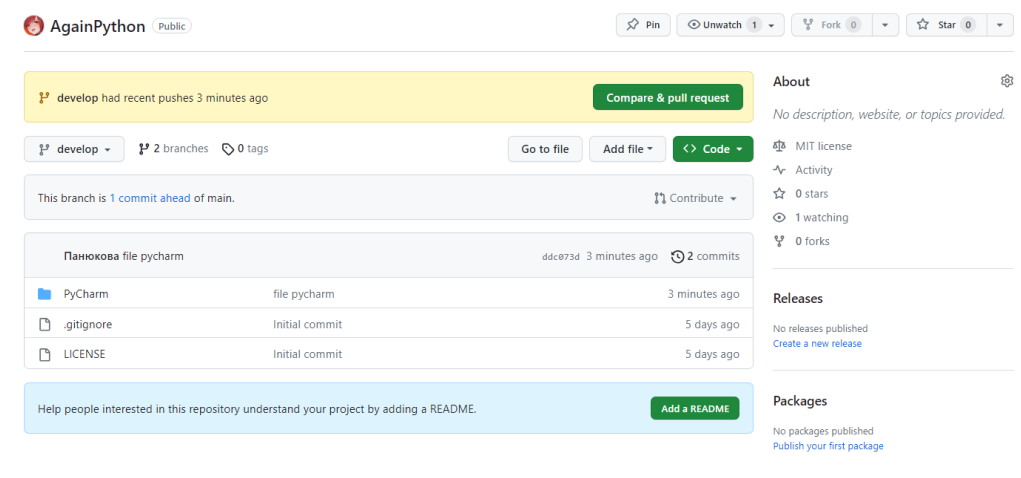


Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Решил следующие задачи с помощью языка программирования Python3 и VScode

8. Напишите программу (файл `user.py`), которая запрашивала бы у пользователя:

- его имя (например, "What is your name?")
- возраст ("How old are you?")
- место жительства ("Where are you live?")

После этого выводила бы три строки:

```
"This is `имя`"  
"It is `возраст`"  
"(S)he live in `место_жительства`"
```

Вместо `имя`, `возраст`, `место_жительства` должны быть данные, введенные пользователем.

Примечание: можно писать фразы на русском языке, но если вы планируете стать профессиональным программистом, привыкайте к английскому.

9. Напишите программу (файл `arithmetic.py`), которая предлагала бы пользователю решить пример $4 * 100 - 54$. Потом выводила бы на экран правильный ответ и ответ пользователя. Подумайте, нужно ли здесь преобразовывать строку в число.

10. Запросите у пользователя четыре числа (файл `numbers.py`). Отдельно сложите первые два и отдельно вторые два. Разделите первую сумму на вторую. Выведите результат на экран так, чтобы ответ содержал две цифры после запятой.

11. Напишите программу (файл `individual.py`) для решения индивидуального задания. Вариант индивидуального задания уточните у преподавателя.

Рисунок 7.1 – Задачи для решения

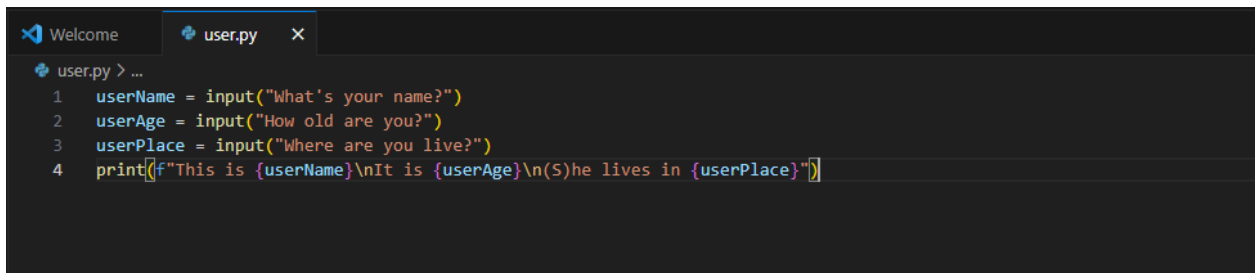
8. Напишите программу (файл `user.py`), которая запрашивала бы у пользователя:

- его имя (например, "What is your name?")
- возраст ("How old are you?")
- место жительства ("Where are you live?")

После этого выводила бы три строки:

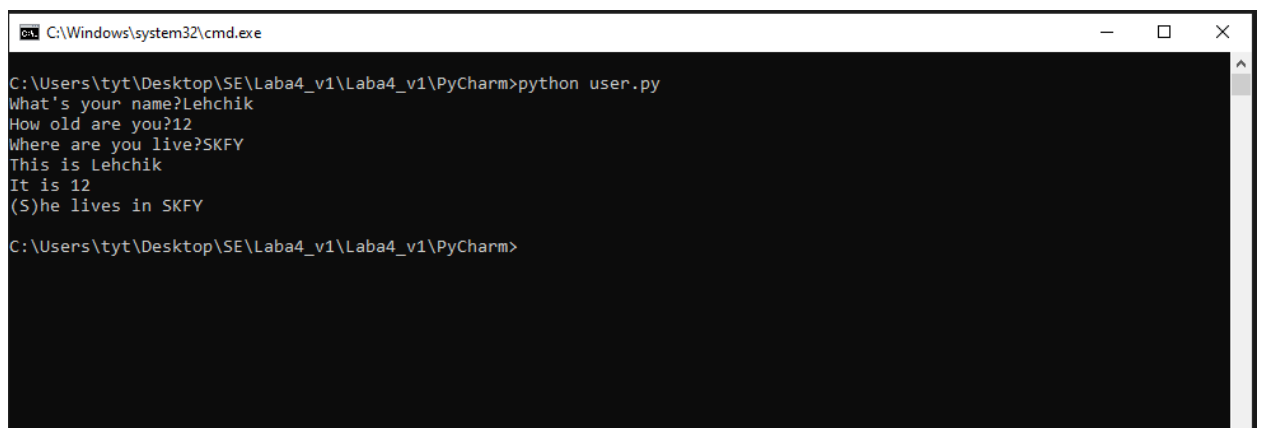
- "This is `имя`"
- "It is возраст"
- "(S)he lives in `место_жительства`"

Вместо `имя`, `возраст`, `место_жительства` должны быть данные, введенные пользователем.



```
Welcome user.py x  
user.py > ...  
1  userName = input("What's your name?")  
2  userAge = input("How old are you?")  
3  userPlace = input("Where are you live?")  
4  print(f"This is {userName}\nIt is {userAge}\n(S)he lives in {userPlace}")
```

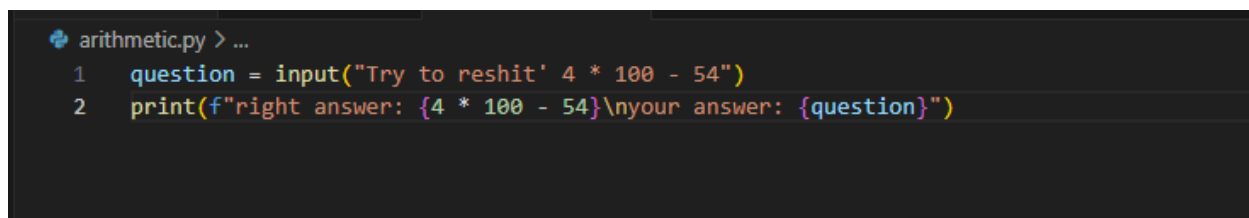
Рисунок 8.1 – Код программы `user.py` в IDE PyCharm



```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm>python user.py
What's your name?Lehchik
How old are you?12
Where are you live?SKFY
This is Lehchik
It is 12
(S)he lives in SKFY
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm>
```

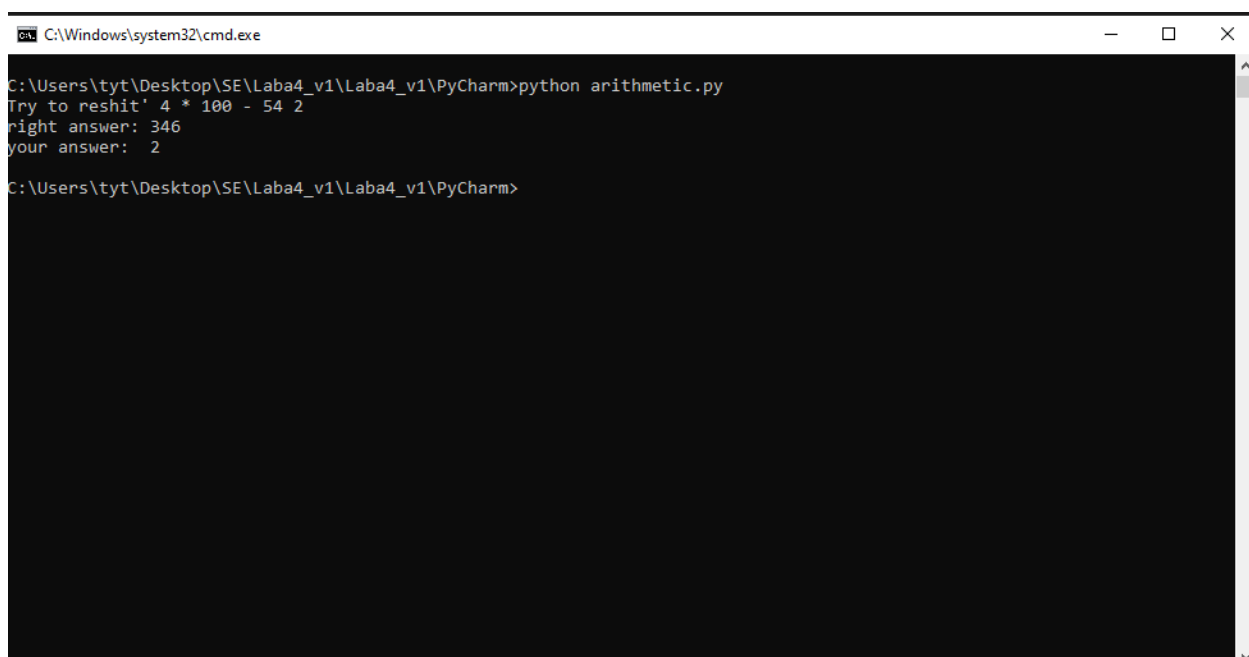
Рисунок 8.2 – Результат выполнения программы user.py

9. Напишите программу (файл arithmetic.py), которая предлагала бы пользователю решить пример $4 * 100 - 54$. Потом выводила бы на экран правильный ответ и ответ пользователя. Подумайте, нужно ли здесь преобразовывать строку в число



```
arithmetic.py > ...
1 question = input("Try to reshit' 4 * 100 - 54")
2 print(f"right answer: {4 * 100 - 54}\nyour answer: {question}")
```

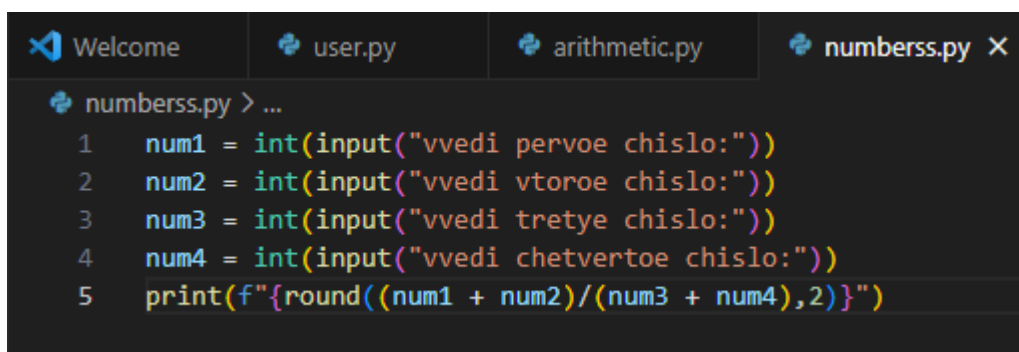
Рисунок 9.1 – Код программы arithmetic.py в IDE PyCharm



```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm>python arithmetic.py
Try to reshit' 4 * 100 - 54 2
right answer: 346
your answer: 2
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm>
```

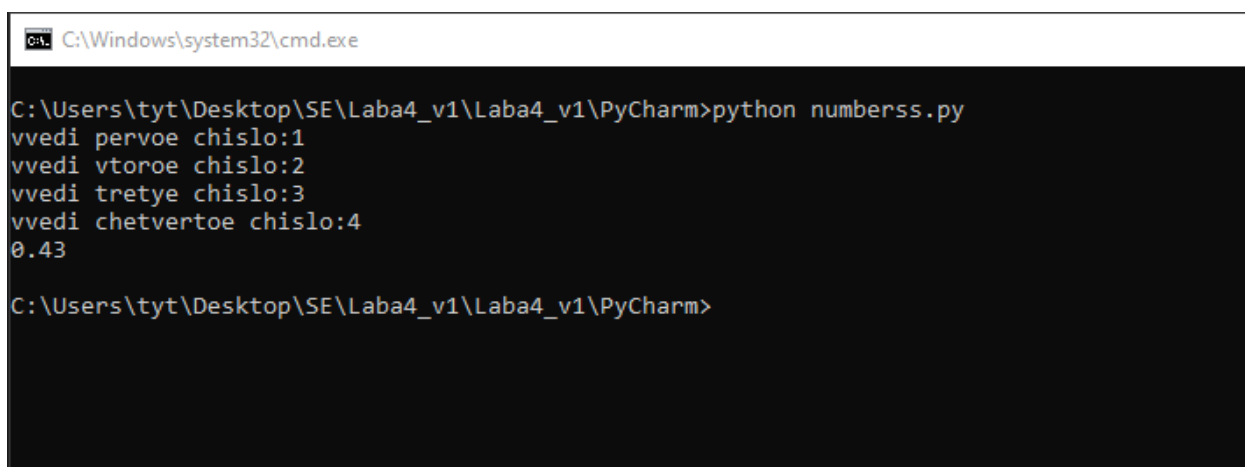

Рисунок 9.2 – Результат выполнения программы arithmetic.py

10. Запросите у пользователя четыре числа (файл numbers.py). Отдельно сложите первые два и отдельно вторые два. Разделите первую сумму на вторую. Выведите результат на экран так, чтобы ответ содержал две цифры после запятой.

A screenshot of the PyCharm IDE interface. The top bar shows four tabs: 'Welcome', 'user.py', 'arithmetic.py', and 'numbersss.py' (which is active and has a close button). The main editor area displays the code for 'numbersss.py' with line numbers 1 through 5. The code prompts the user for four numbers and calculates the rounded ratio of the sum of the first two to the sum of the last two.

```
1 num1 = int(input("vvedi pervoe chislo:"))
2 num2 = int(input("vvedi vtoroe chislo:"))
3 num3 = int(input("vvedi tretye chislo:"))
4 num4 = int(input("vvedi chetvertoe chislo:"))
5 print(f"{round((num1 + num2)/(num3 + num4),2)}")
```

Рисунок 10.1 – Код программы numbers.py в IDE PyCharm

A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\system32\cmd.exe'. The command prompt shows the execution of 'python numbersss.py' from the directory 'C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm'. The user inputs four numbers: 1, 2, 3, and 4. The program outputs the result '0.43'.

```
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm>python numbersss.py
vvedi pervoe chislo:1
vvedi vtoroe chislo:2
vvedi tretye chislo:3
vvedi chetvertoe chislo:4
0.43
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm>
```

Рисунок 10.2 – Результат выполнения программы numbers.py

11. Напишите программу (файл individual.py) для решения индивидуального задания.

Даны основания равнобедренной трапеции и угол при большем основании. Найти площадь трапеции.

```

individual.py > ...
1  # Пусть основания трапеции равны a и b, а угол при большем основании равен α. Тогда площадь трапеции S вычисляется по формуле:
2
3  #  $S = ((a + b) / 2) * h$ 
4
5  # где h - высота трапеции, которую можно найти по формуле:
6
7  #  $h = (b - a) / 2 * \tan(\alpha/2)$ 
8
9  # Таким образом, полная формула для вычисления площади трапеции будет выглядеть так:
10
11 #  $S = ((a + b) / 2) * ((b - a) / 2 * \tan(\alpha/2))$ 
12
13 import math
14
15 osn1 = int(input("osn 1="))
16 osn2 = int(input("osn 2="))
17
18 ygol = int(input("ygol pri bol'shem osnovanii = "))
19
20 print(f"S = {((osn1 + osn2)/2) * ((osn2 - osn1)/2 * math.tan(ygol/2))}")
21

```

Рисунок 11.1 – Код программы individual.py в IDE PyCharm

```

C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm>python individual.py
osn 1=5
osn 2=10
ygol pri bol'shem osnovanii = 90
S = 30.370784822697402

C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1\PyCharm>

```

Рисунок 11.2 – Результат выполнения программы individual.py

12. Выполнил коммит файлов user.py, arithmetic.py, numbers.py и individual.py в репозиторий git в ветку для разработки.

cls

```
C:\Windows\system32\cmd.exe
On branch feature_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    PyCharm/

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1>git add .
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1>git status
On branch feature_branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   PyCharm/arithmetic.py
    new file:   PyCharm/hard.py
    new file:   PyCharm/individual.py
    new file:   PyCharm/numberss.py
    new file:   PyCharm/user.py

C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1>git commit -m"reshil zadaniya"
[feature_branch 7b3e0ac] reshil zadaniya
 5 files changed, 34 insertions(+)
 create mode 100644 PyCharm/arithmetic.py
 create mode 100644 PyCharm/hard.py
 create mode 100644 PyCharm/individual.py
 create mode 100644 PyCharm/numberss.py
 create mode 100644 PyCharm/user.py
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1>
```

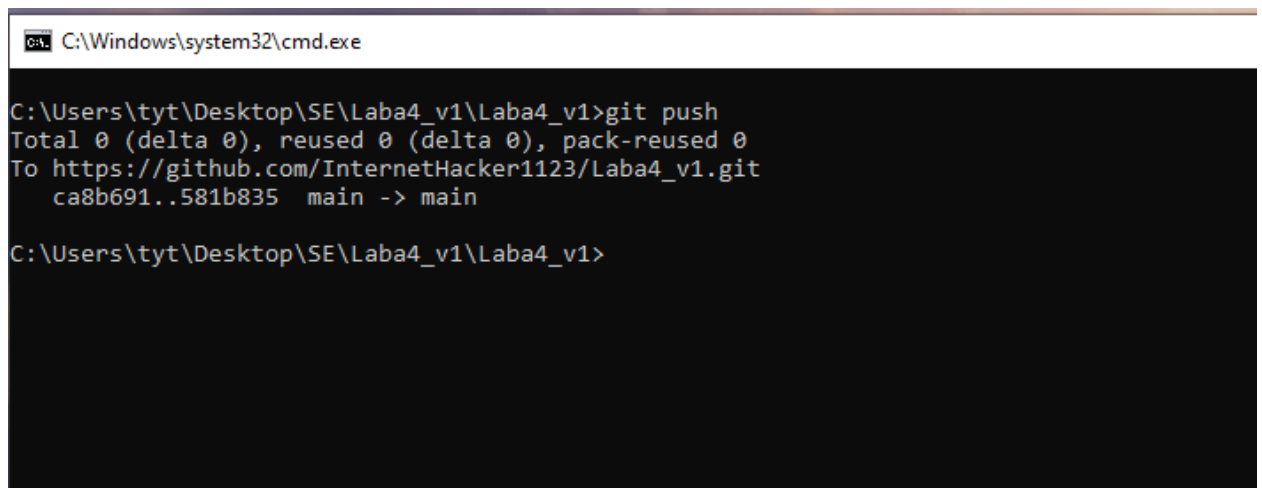
Рисунок 12.1 – Коммит файлов в репозиторий git

13. Выполнил слияние ветки для разработки с веткой main

```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1>git merge develop
Updating 3944e51..581b835
Fast-forward
 PyCharm/arithmetic.py | 2 ++
 PyCharm/hard.py       | 3 +++
 PyCharm/individual.py | 20 ++++++
 PyCharm/numberss.py   | 5 +++++
 PyCharm/user.py       | 4 +++++
 5 files changed, 34 insertions(+)
 create mode 100644 PyCharm/arithmetic.py
 create mode 100644 PyCharm/hard.py
 create mode 100644 PyCharm/individual.py
 create mode 100644 PyCharm/numberss.py
 create mode 100644 PyCharm/user.py
C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1>
```

Рисунок 13.1 – Слияние ветки main с веткой develop

14. Отправил сделанные изменения на сервер GitHub



```
C:\Windows\system32\cmd.exe

C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/InternetHacker1123/Laba4_v1.git
   ca8b691..581b835  main -> main

C:\Users\tyt\Desktop\SE\Laba4_v1\Laba4_v1>
```

Рисунок 14.1 – Отправка изменений на сервер GitHub

Контрольные вопросы

1. Опишите основные этапы установки Python в Windows и Linux

Установка Python в Windows

Для операционной системы Windows дистрибутив распространяется либо в виде исполняемого файла (с расширением exe), либо в виде архивного файла (с расширением zip). Если вы используете Windows 7, не забудьте установить Service Pack 1.

Порядок установки.

1. Запустите скачанный установочный файл.
2. Выберите способ установки
3. Отметьте необходимые опции установки (доступно при выборе Customize installation)
4. Выберите место установки (доступно при выборе Customize installation)
5. После успешной установки вас ждет сообщение об успешной установке

Установка Python в Linux

Чаще всего интерпретатор Python уже входит в состав дистрибутива. Это можно проверить, набрав в терминале python или python 3. В первом случае вы запустите Python 2 во втором – Python 3. Если у вас, при попытке запустить

Python, выдается сообщение о том, что он не установлен, или установлен, но не тот, что вы хотите, то у вас есть два пути: а) собрать Python из исходников; б) взять из репозитория. Для установки из репозитория в Ubuntu воспользуйтесь командой `sudo apt-get install python3`.

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

Пакет Anaconda включает в себя интерпретатор языка Python (есть версии 2 и 3), набор наиболее часто используемых библиотек и удобную среду разработки и исполнения, запускаемую в браузере.

3. Как осуществить проверку работоспособности пакета Anaconda?

Для выполнения проверки работоспособности Anaconda необходимо вначале запустить командный процессор с поддержкой виртуальных окружений Anaconda. В Windows это можно сделать, выбрав следующий пункт главного меню системы Пуск → Anaconda3 (64-bit) → Anaconda Prompt. 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.

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

Запустите PyCharm и выберите Create New Project в появившемся окне. Укажите путь до проекта Python и интерпретатор, который будет использоваться для запуска и отладки..

5. Как осуществить запуск программы с помощью IDE PyCharm?

После того, как исходный код написан, чтобы первый раз запустить программу, проще всего нажать `Ctrl+Shift+F10`.

6. В чем суть интерактивного и пакетного режимов работы Python?

В интерактивный режим можно войти, набрав в командной строке `python` или `python3`. В результате Python запустится в интерактивном режиме и будет ожидать ввод команд пользователя.

Чтобы выполнить запуск в пакетном режиме, надо ввести в командной строке имя интерпретатора, плюс имя файла.

7. Почему язык программирования Python называется языком динамической типизации?

Если достаточно формально подходить к вопросу о типизации языка Python, то можно сказать, что он относится к языкам с неявной сильной динамической типизацией. Неявная типизация означает, что при объявлении переменной вам не нужно указывать её тип, при явной – это делать необходимо. Также языки бывают с динамической и статической типизацией. В первом случае тип переменной определяется непосредственно при выполнении программы, во втором – на этапе компиляции. Как уже было сказано Python – это динамически типизированный язык.

8. Какие существуют основные типы в языке программирования Python?

К основным встроенным типам относятся:

- None (неопределенное значение переменной);
- Логические переменные (Boolean Type);
- Числа (Numeric Type);
- Списки (Sequence Type);
- Строки (Text Sequence Type);
- Бинарные списки (Binary Sequence Types);
- Множества (Set Types);
- Словари (Mapping Types).

9. Как создаются объекты в памяти? Каково их устройство? В чем заключается процесс объявления новых переменных и работа операции присваивания?

Объект – это абстракция для представления данных, данные – это числа, списки, строки и т. п. При этом, под данными следует понимать как непосредственно сами объекты, так и отношения между ними. Каждый объект имеет три атрибута – это идентификатор, значение и тип. Идентификатор – это

уникальный признак объекта, позволяющий отличать объекты друг от друга, а значение – непосредственно информация, хранящаяся в памяти, которой управляет интерпретатор

При инициализации переменной (например, 5), на уровне интерпретатора, происходит следующее:

- Создается целочисленный объект 5 (можно представить, что в этот момент создается ячейка и 5 кладется в эту ячейку);
- Данный объект имеет некоторый идентификатор, значение: 5, и тип: целое число;
- Посредством оператора «=» создается ссылка между переменной b и целочисленным объектом 5 (переменная b ссылается на объект 5).

10. Как получить список ключевых слов в Python?

Проверить является ли идентификатор ключевым словом можно с помощью команды `keyword.iskeyword("название_переменной")`.

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

Для того, чтобы посмотреть на объект с каким идентификатором ссылается данная переменная, можно использовать функцию `id()`. Тип переменной можно определить с помощью функции `type()`.

12. Что такое изменяемые и неизменяемые типы в Python

В Python существуют изменяемые и неизменяемые типы.

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).

Неизменяемость типа данных означает, что созданный объект больше не изменяется. Если тип данных изменяемый, то можно менять значение объекта.

13. Чем отличаются операции деления и целочисленного деления?

Целочисленное деление (`//`) отличается от обычной операции деления тем, что возвращает целую часть частного, а дробная часть 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 комплексно сопряженного 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`

В стандартную поставку Python входит библиотека `math`, в которой содержится большое количество часто используемых математических функций. Основные функции `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 π
- `math.e` число e

Модуль `math` предоставляет доступ к математическим функциям для комплексных чисел. Функции в этом модуле принимают в качестве аргументов целые числа, числа с плавающей запятой или комплексные числа. Они также будут принимать любой объект Python, у которого есть метод `__complex__()` или `__float__()`: данные методы используются для преобразования объекта в комплексное число или число с плавающей запятой соответственно, а затем функция применяется к результату преобразования.

Множество функций подобные, но не идентичные тому, что в модуле `math`. Причина наличия двух модулей в том, что некоторые пользователи не интересуются комплексными числами и, возможно, даже не знают, что они собой представляют. Функции, определённые в `cmath`, всегда возвращают комплексное число, даже если ответ может быть выражен в виде действительного числа (в этом случае комплексное число имеет нулевую мнимую часть). Основные функции `cmath`:

- `cmath.polar(x)` возвращает представление x в полярных координатах.
- `cmath.rect(r, phi)` возвращает комплексное число x с полярными координатами r и ϕ
- `cmath.phase(x)` возвращает фазу x (также известную как `argument` x) в виде числа с плавающей запятой
- `cmath.isfinite(x)` возвращает `True`, если и действительная, и мнимая части x конечны, и `False` в противном случае
- `cmath.isinf(x)` возвращает `True`, если действительная или мнимая часть x равна бесконечности, и `False` в противном случае

- `cmath.isnan(x)` возвращает True, если действительная или мнимая часть `x` является NaN, и False в противном случае
- `cmath.infj` комплексное число с нулевой действительной частью и положительной бесконечностью мнимой части
- `cmath.nanj` комплексное число с нулевой действительной частью и NaN мнимой частью.`math.sqrt(x)` корень квадратный от `x`
- `math.nan` Значение с плавающей запятой «не число» (NaN)

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.

В строке в фигурных скобках указаны номера данных, которые будут подставлены. Далее к строке применяется метод `format()`. В его скобках указываются сами данные (можно использовать переменные). На нулевое место подставится первый аргумент метода `format()`, на место с номером 1 – второй и т. д. Существует также Старый стиль его называют Си-стилем, так как он схож с тем, как происходит вывод на экран в языке C. Вместо трех комбинаций символов `%s` , `%d` , `%f` подставляются значения переменных. Буквы `s`, `d`, `f` обозначают типы данных – строку, целое число, вещественное число. Если бы требовалось подставить три строки, то во всех случаях использовалось бы сочетание `%s`.

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

За ввод в программу данных с клавиатуры в Python отвечает функция `input()`. Когда вызывается эта функция, программа останавливает свое выполнение и ждет, когда пользователь введет текст. Функция `input()` передает

введенные данные в программу. Их можно присвоить переменной. Чтобы получить целочисленное и вещественное значение нужно воспользоваться функцией преобразования типов. В данном случае с помощью функций `int()` и `float()`.