

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 практической работе №2.1
Дисциплины «Программирование на Python»

Выполнил:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и
вычислительная техника (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических
наук, доцент, доцент кафедры
инфокоммуникаций

(подпись)

Ставрополь, 2023 г.

Тема: Основы языка Python.

Цель: исследовать базовые возможности языка программирования Python версии 3.x.

Ход работы:

Организация репозитория согласно модели ветвления «git flow» (рис. 1).

```
C:\Users\Andrey\Desktop\пайтон\4 лаба\лаба 4\Python_laboratory_work_2.1>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? [C:/Users/Andrey/Desktop/пайтон/4 лаба\лаба 4/Python_laboratory_work_2.1/.git/hooks]
```

Рисунок 1 – модель ветвления «git flow».

Создание проекта PyCharm в папке репозитория (рис. 2).

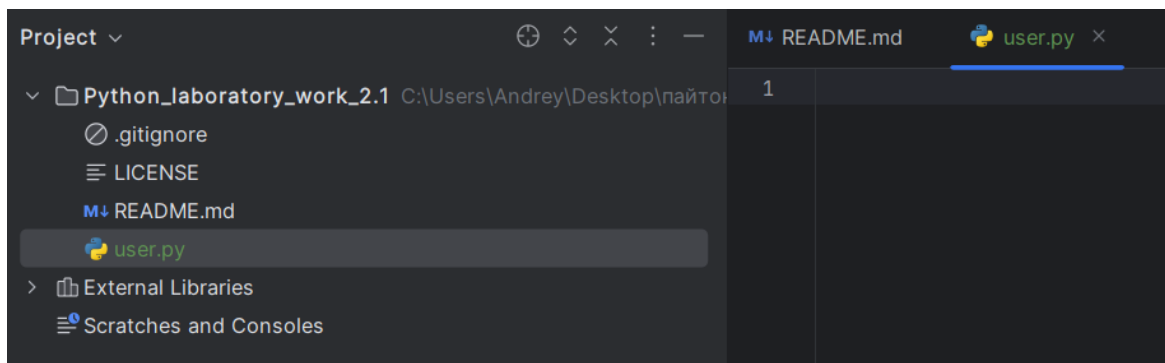
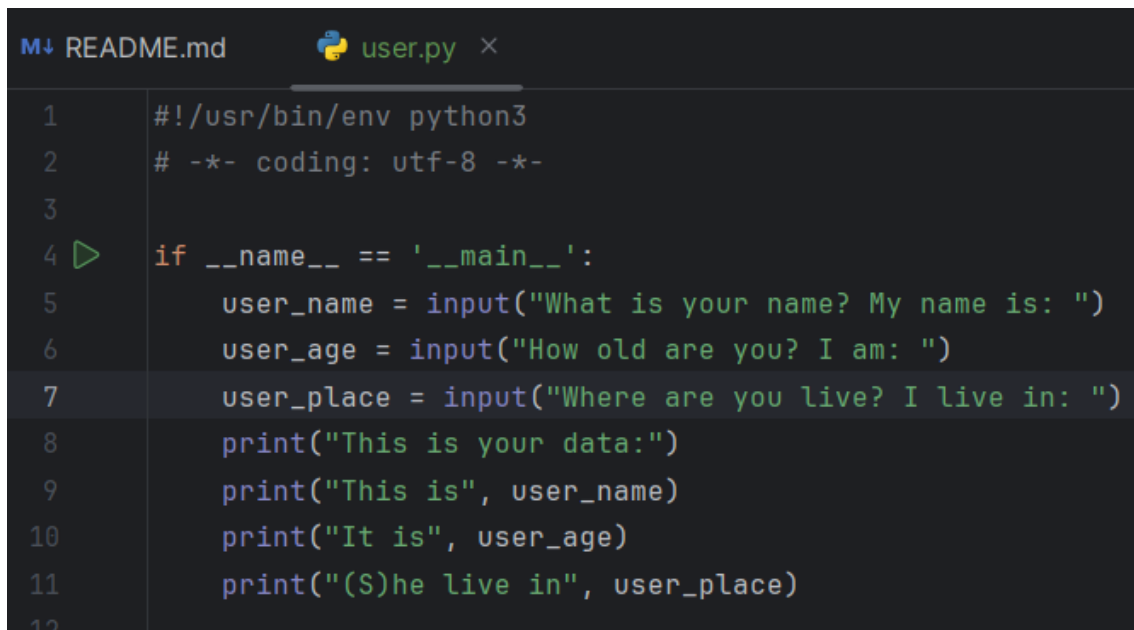


Рисунок 2 – Открытие репозитория в PyCharm.

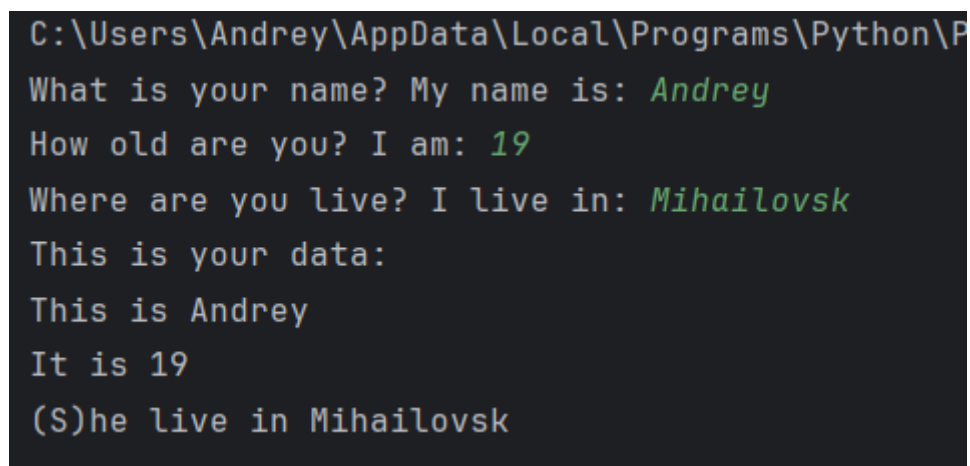
Создание файла «user.py», в котором запрашиваются данные о пользователе (имя, возраст и место жительства) и выводятся на экран (рис. 3).

A screenshot of a code editor with two tabs: 'README.md' and 'user.py'. The 'user.py' tab is active, showing a Python script. The script starts with a shebang line and a UTF-8 encoding declaration. It then has an if-statement block that runs when the script is executed directly. Inside this block, it prompts the user for their name, age, and place of residence, and then prints the collected data in a specific format.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     user_name = input("What is your name? My name is: ")
6     user_age = input("How old are you? I am: ")
7     user_place = input("Where are you live? I live in: ")
8     print("This is your data:")
9     print("This is", user_name)
10    print("It is", user_age)
11    print("(S)he live in", user_place)
12
```

Рисунок 3 – Код программы запроса и вывода данных.

Результат выполнения данной программы (рис. 4).

A screenshot of a terminal window showing the execution of the Python script. The prompt shows the file path 'C:\Users\Andrey\AppData\Local\Programs\Python\Python39\Scripts\python.exe user.py'. The user enters 'Andrey' for the name, '19' for the age, and 'Mihailovsk' for the place of residence. The program then outputs the collected data in the format specified in the code.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\Scripts\python.exe user.py
What is your name? My name is: Andrey
How old are you? I am: 19
Where are you live? I live in: Mihailovsk
This is your data:
This is Andrey
It is 19
(S)he live in Mihailovsk
```

Рисунок 4 – Результат работы программы.

Создание программы «arithmetic.py», которая предлагает пользователю решить пример и сверяла его с правильным ответом (рис. 5).

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

if __name__ == '__main__':
    print("Hello, please solve the example: 4 * 100 - 54")
    user_answer = int(input("Your answer: "))
    real_answer = 4 * 100 - 54
    if real_answer == user_answer:
        print("This answer is correct!", "Your answer:",
              user_answer, "Right answer:", real_answer)
    else:
        print("This answer is not correct!", "Your answer:",
              user_answer, "Right answer:", real_answer)
```

Рисунок 5 – Код программы «arithmetic».

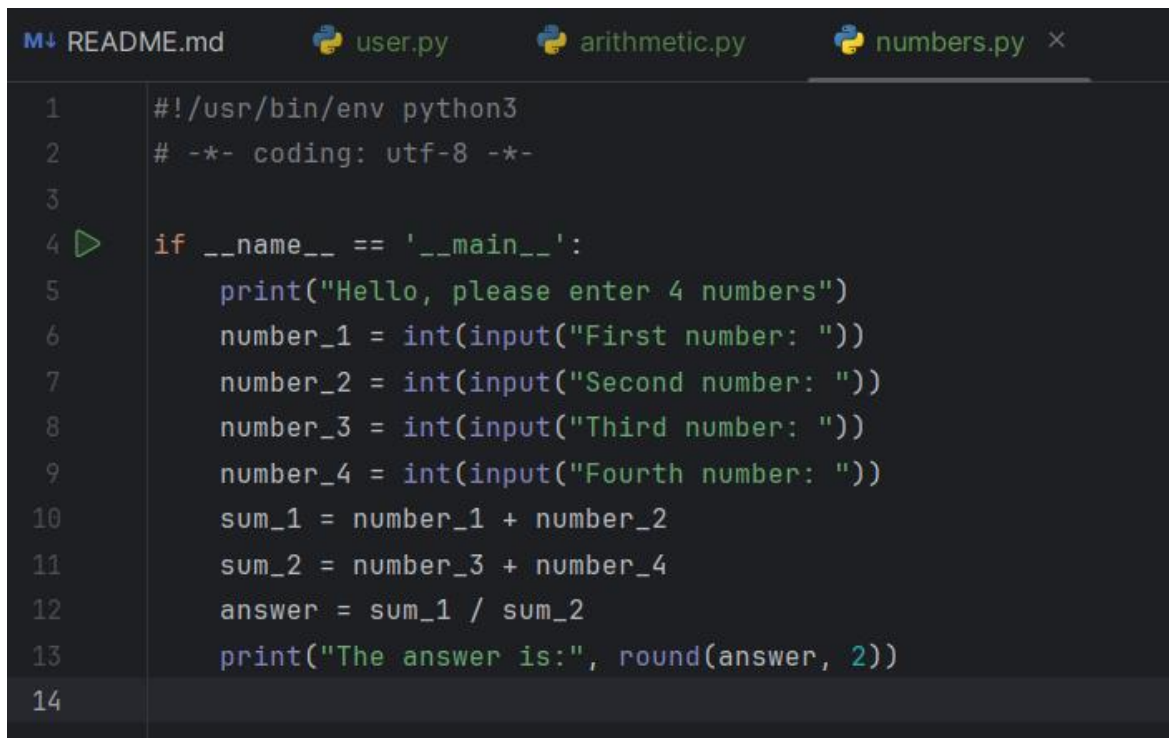
Результаты работы программы решения примера (рис. 6).

```
Hello, please solve the example: 4 * 100 - 54
Your answer: 22
This answer is not correct! Your answer: 22 Right answer: 346
```

Рисунок 6 – Результат работы программы «arithmetic».

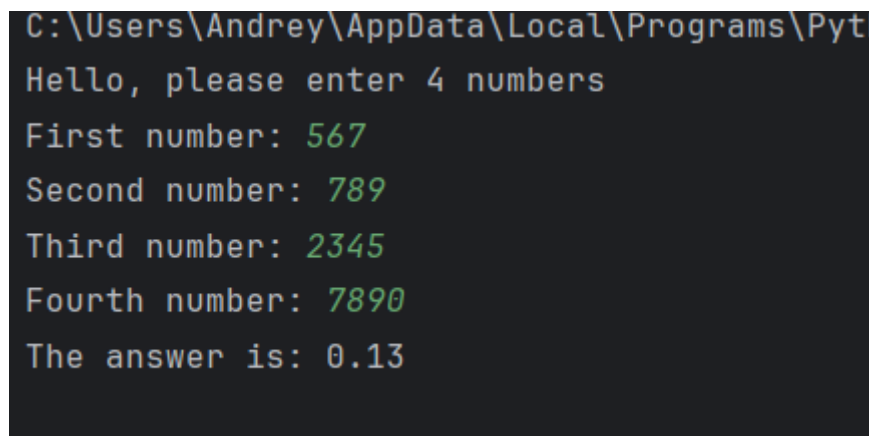
Необходимо написать программу, которая запрашивала у пользователя четыре числа. Сложить по отдельности два первых и два последних, а затем разделить первую сумму на вторую, причем результат должен содержать две цифры после запятой.

Код программы нахождения частного двух сумм «numbers.py» и результаты работы программы (рис. 7, 8).

The image shows a code editor with four tabs: 'README.md', 'user.py', 'arithmetic.py', and 'numbers.py'. The 'numbers.py' tab is active, displaying the following Python code:

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      print("Hello, please enter 4 numbers")
6      number_1 = int(input("First number: "))
7      number_2 = int(input("Second number: "))
8      number_3 = int(input("Third number: "))
9      number_4 = int(input("Fourth number: "))
10     sum_1 = number_1 + number_2
11     sum_2 = number_3 + number_4
12     answer = sum_1 / sum_2
13     print("The answer is:", round(answer, 2))
14
```

Рисунок 7 – Код программы нахождения частного сумм.

The image shows a terminal window with the following output:

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe numbers.py
Hello, please enter 4 numbers
First number: 567
Second number: 789
Third number: 2345
Fourth number: 7890
The answer is: 0.13
```

Рисунок 8 – Результаты работы программы.

Создание программы по индивидуальному заданию (Вариант 26 (№2)).
Даны стороны прямоугольника, необходимо найти периметр и длину диагонали.

Код программы нахождения периметра прямоугольника и длины диагонали «individual.py» и результаты работы программы (рис. 9, 10).

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

import math

if __name__ == '__main__':
    length_rectangle = int(input("Enter the length of rectangle:"))
    width_rectangle = int(input("Enter the width of rectangle:"))
    perimetr = (length_rectangle + width_rectangle) * 2
    diagonal_rectangle = math.sqrt(length_rectangle**2 + width_rectangle**2)
    print("Perimetr of rectangle =", perimetr)
    print("Diagonal of rectangle =", diagonal_rectangle)
```

Рисунок 9 – Код программы нахождения периметра и длины диагонали.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe
Enter the length of rectangle:5
Enter the width of rectangle:5
Perimetr of rectangle = 20
Diagonal of rectangle = 7.0710678118654755

Process finished with exit code 0
```

Рисунок 10 – Результаты работы программы.

Выполнение индивидуального задания повышенной сложности:

Задание №6

Необходимо определить число полных часов и число полных минут прошедших с начала суток, если часовая стрелка повернулась на определенное количество градусов.

Код программы нахождения количества полных часов и количества полных минут с начала суток «difficult.py» и результаты работы программы (рис. 11, 12).

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

import math

if __name__ == '__main__':
    degree = int(input("Enter the number of full degrees:"))
    hours_number = degree * 12 // 360
    minutes_number = (12 * degree / 360 - hours_number) * 60
    minutes_number = int(minutes_number)
    print("Now", hours_number, ",", minutes_number, "minutes")
```

Рисунок 11 – Код программы задания повышенной сложности.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\py
Enter the number of full degrees:100
Now 3 , 20 minutes

Process finished with exit code 0
```

Рисунок 12 – Результаты работы программы повышенной сложности.

Выполнение коммита файлов с созданными программами (рис. 13).

```
C:\Users\Andrey\Desktop\пайтон\4 лаба\лаба 4\Python_laboratory_work_2.1>git add .
C:\Users\Andrey\Desktop\пайтон\4 лаба\лаба 4\Python_laboratory_work_2.1>git commit -m "add tasks and edit .gitignore"
[develop db17a19] add tasks and edit .gitignore
7 files changed, 63 insertions(+), 3 deletions(-)
create mode 100644 programs/arithmetic.py
create mode 100644 programs/difficult.py
create mode 100644 programs/individual.py
create mode 100644 programs/numbers.py
create mode 100644 programs/user.py
```

Рисунок 13 – Коммит файлов программ.

Выполнение слияния ветки «develop» с веткой «main» (рис. 14).

```

C:\Users\Andrey\Desktop\пайтон\4 лаба\лаба 4\Python_laboratory_work_2.1>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\Andrey\Desktop\пайтон\4 лаба\лаба 4\Python_laboratory_work_2.1>git merge release/1.0
Updating 0521827..db17a19
Fast-forward
 .gitignore      | 2 +-
 README.md       | 4 ++--
 programs/arithmetic.py | 13 ++++++++
 programs/difficult.py | 11 ++++++++
 programs/individual.py | 12 ++++++++
 programs/numbers.py   | 13 ++++++++
 programs/user.py      | 11 ++++++++

```

Рисунок 14 – Слияние веток «develop» и «main» согласно модели ветвления «git-flow».

Ответы на контрольные вопросы:

1. Для операционной системы Windows дистрибутив распространяется либо в виде исполняемого файла (с расширением exe), либо в виде архивного файла (с расширением zip). Порядок установки, следующий: необходимо запустить скачанный установочный файл; выбрать способ установки. Install Now - Python установится в папку по указанному пути. Помимо самого интерпретатора будет установлен IDLE (интегрированная среда разработки), pip (пакетный менеджер) и документация, также будут созданы соответствующие ярлыки и установлены связи файлов, имеющие расширение «.py» с интерпретатором Python. Customize installation – вариант настраиваемой установки. На этом шаге предлагается отметить дополнения, устанавливаемые вместе с интерпретатором Python. Documentation – установка документаций. pip – установка пакетного менеджера pip. tcl/tk and IDLE – установка интегрированной среды разработки (IDLE) и библиотеки для построения графического интерфейса. Выбрать место установки (доступно при выборе Customize installation. Помимо указания пути, данное окно позволяет внести дополнительные изменения в процесс установки с помощью опций: Install for all users – Установить для всех пользователей. Если не выбрать данную опцию, то будет предложен вариант инсталляции в папку пользователя, устанавливающего интерпретатор. Associate files with Python – Связать файлы, имеющие расширение .py, с Python. При выборе данной опции будут внесены изменения в Windows, позволяющие запускать Python скрипты

по двойному щелчку мыши. Create shortcuts for installed applications – Создать ярлыки для запуска приложений. Add Python to environment variables – Добавить пути до интерпретатора Python в переменную PATH. Precompile standard library – Провести прекомпиляцию стандартной библиотеки). Установка Python в Linux: чаще всего интерпретатор Python уже входит в состав дистрибутива. Это можно проверить, набрав в терминале «python python 3» (Запустится Python3). Если при попытке запустить Python, выдается сообщение о том, что он не установлен, или установлен, но не тот, то можно собрать Python из исходников или взять из репозитория (Для установки из репозитория в Ubuntu есть команда `sudo apt-get install python3`).

2. Пакет «Anaconda» в отличие от «Python» 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. Для выполнения проверки работоспособности «Anaconda» необходимо запустить командный процессор с поддержкой виртуальных окружений Anaconda. В Windows это можно сделать, выбрав пункт Anaconda Prompt в Пуске. В появившейся командной строке необходимо ввести `jupyter notebook`, в результате чего отобразится процесс загрузки веб-среды Jupyter Notebook и запустится браузер со средой разработки.

4. Для того, чтобы задать используемый интерпретатор языка Python необходимо открыть настройки нужного проекта в Pycharm, выбрав пункт меню «File» Settings». Найти «Python Interpreter», после чего на открывшейся странице нажать на «+» и выбрать «Add...». Выбрать нужное расположение интерпретатора Python и нажать «ОК».

5. Для запуска программы с помощью IDE PyCharm есть команда «python название файла и его расширение». Или же можно просто вводить необходимые команды, например «`print()`» (Это работает только в интерактивном режиме, в командной строке можно только запускать файлы).

6. Интерактивный пакетный режим Python используется для различных вычислений. Пакетный - сначала записывается вся программа, потом она выполняется полностью.

7. Python является языком динамической типизации так как в нем нет необходимости определять тип переменной, т.к. это происходит автоматически в процессе выполнения программы.

8. Типы языка Python: логические переменные (boolean), числа (int – целое число, float – число с плавающей точкой, complex – комплексное число), Списки (list – список, tuple – кортеж, range – диапазон), Строки (str), Бинарные списки (bytes – байты, bytearray – массивы байт, memoryview – специальные объекты для доступа к внутренним данным объекта через protocol buffer), Множества (set – множество, frozenset – неизменяемое множество), Словари (dict – словарь).

9. Для объявления и инициализации переменной необходимо написать её имя, потом поставить знак равенства и значение, с которым эта переменная будет создана. Передаваемое значение в Python является объектом (Объект – абстракция для представления данных, данные – числа, списки, строки и т.п. – объекты и отношения между ними. Каждый объект имеет три атрибута: идентификатор (уникальный признак объекта, позволяющий отличать объекты друг от друга), значение (информация, хранящаяся в памяти, которой управляет интерпретатор), тип).

10. Для вывода списка ключевых слов необходимо подключить «keyword» (import keyword) и вывести список ключевых слов (print(keyword.kwlist)).

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

12. Изменяемые типы данных – списки, множества, словари (Объект может измениться, например, добавятся или заменятся значения). Неизменяемые типы данных – числа (целые, с плавающей точкой,

комплексные), логические переменные, строки, неизменяемые множества (Объект неизменен, например – создание целочисленной переменной и присваивание ей значения).

13. Целочисленное деление возвращает только целую часть частного, а обычное деление возвращает частное полностью, вместе с дробной частью.

14. Создание комплексного числа возможно при помощи функции «complex(действительная часть, мнимая часть)». Над комплексными числами возможны операции, деления, умножения, суммы, вычитания.

15. «math» – библиотека, содержащая математические и тригонометрические функции (sin, cos, floor, ceil, fabs и т.д.).

Содержание модуля «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.

Константы: math.pi - число Пи, math.e - число e.

16. Параметр «sep» позволяет указать разделитель строк (По умолчанию в качестве разделителя используется пробел). Параметр «end» позволяет указать, что нужно добавить после последней строки (По умолчанию добавляется управляющий символ «\n» (перевод строки на новую)).

17. Метод `format()` содержит в скобках данные, которые необходимо указать в фигурных скобках в `print`. На нулевое место подставится первый аргумент метода `format()`, на место с номером 1 – второй и т. д.

18. За ввод в программу данных с клавиатуры в Python отвечает функция `input()`. Когда вызывается эта функция, программа останавливает свое выполнение и ждет ввода данных. После нажатия «Enter», функция `input()` заберет введенный текст и передаст его программе, которая уже будет обрабатывать его согласно своим алгоритмам. Для преобразования полученных строковых значений в целочисленные и вещественные, необходимо обернуть вызов `input` в `int()` или `float()` соответственно.

Вывод: в ходе выполнения лабораторной работы были исследованы базовые средства программирования в Python. Были изучены основные функции ввода вывода, некоторые библиотеки (модули), а также способы форматирования вывода. В ходе лабораторной работы были изучены основные типы данных в языке программирования Python. Были изучены основные способы установки Python на различные операционные системы.