

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 №1.3**  
**дисциплины «Программирование на Python»**

Выполнил:  
Степанов Леонид Викторович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение  
средств вычислительной  
техники и автоматизирование  
систем», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., канд. техн. наук,  
доцент, доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

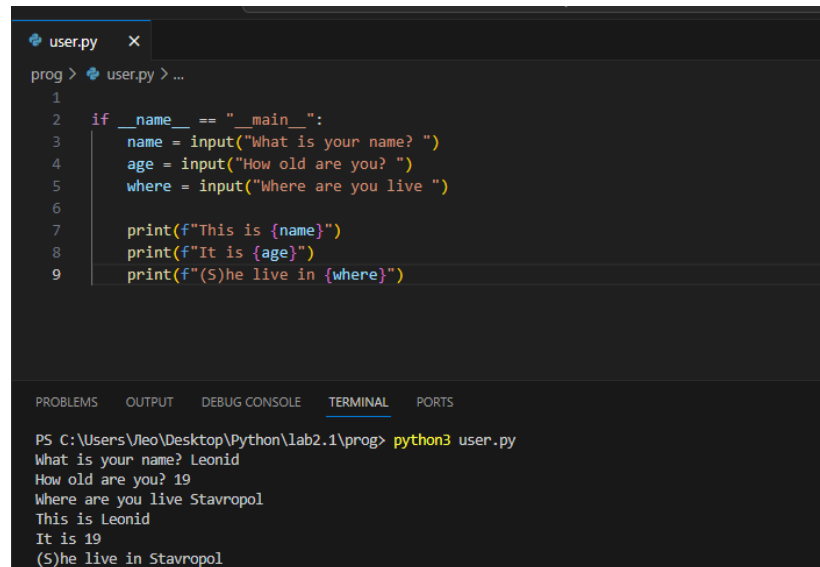
Ставрополь, 2023 г.

Тема: основы языка Python

Цель: исследование процесса установки и базовых возможностей языка Python версии 3.x.

Порядок выполнения работы:

Написал программу (user.py), в которой у пользователя запрашивает имя, возраст, место жительства



```
user.py x
prog > user.py > ...
1
2 if __name__ == "__main__":
3     name = input("What is your name? ")
4     age = input("How old are you? ")
5     where = input("Where are you live ")
6
7     print(f"This is {name}")
8     print(f"It is {age}")
9     print(f"(S)he live in {where}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\leo\Desktop\Python\lab2.1\prog> python3 user.py
What is your name? Leonid
How old are you? 19
Where are you live Stavropol
This is Leonid
It is 19
(S)he live in Stavropol
```

Рисунок 1 – Результат выполнения программы user.py

Написал программу (arithmetic.py) которая предлагает пользователю решить пример, потом выводит правильный ответ и ответ пользователя

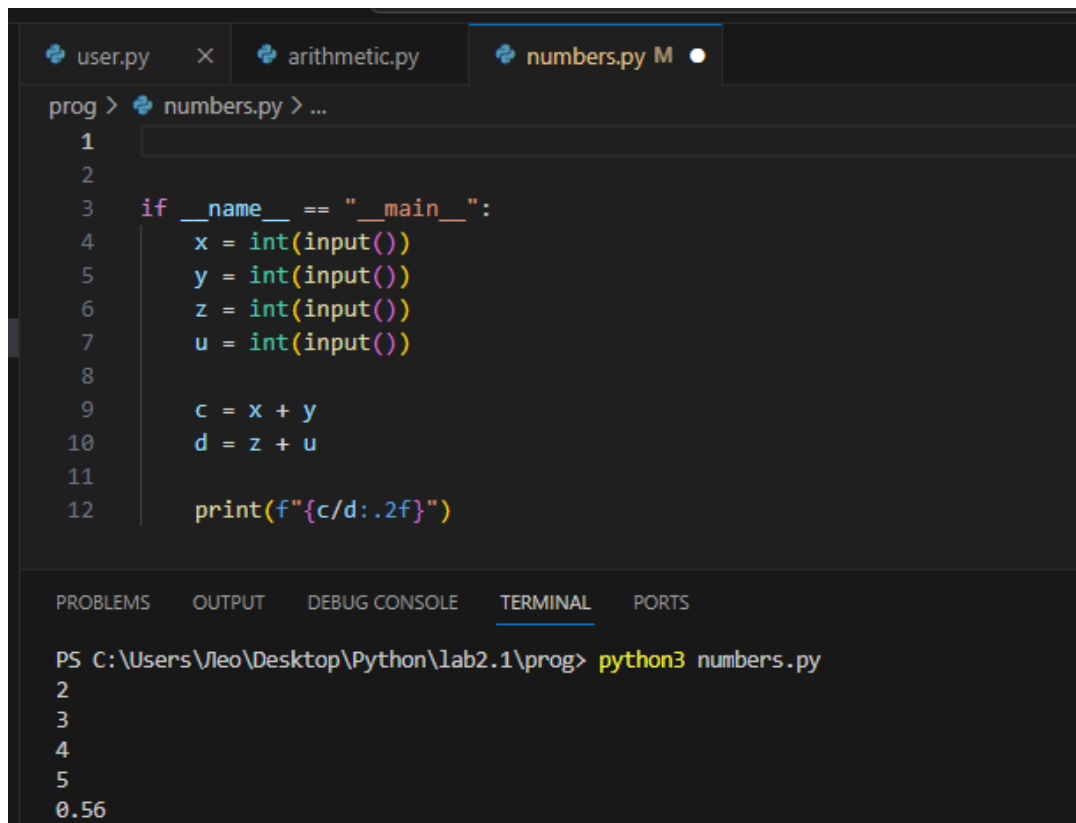


```
user.py arithmetic.py x
prog > arithmetic.py > ...
1 |
2 if __name__ == "__main__":
3     x = input("4 * 100 - 54 = ")
4
5     print(f" Right:{4 * 100 -54} user: {x}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\leo\Desktop\Python\lab2.1\prog> python3 arithmetic.py
4 * 100 - 54 = 100
Right:346 user: 100
PS C:\Users\leo\Desktop\Python\lab2.1\prog>
```

Рисунок 2 – Результат выполнения программы arithmetic.py

Написал программу (numbers.py) которая запрашивает у пользователя 4 числа, первые два из которых необходимо сложить, и остальные. Разделить первую сумму на вторую.



```
user.py x arithmetic.py numbers.py M ●
prog > numbers.py > ...
1
2
3 if __name__ == "__main__":
4     x = int(input())
5     y = int(input())
6     z = int(input())
7     u = int(input())
8
9     c = x + y
10    d = z + u
11
12    print(f"{c/d:.2f}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\leo\Desktop\Python\lab2.1\prog> python3 numbers.py
2
3
4
5
0.56
```

Рисунок 3 – Результат выполнения программы numbers.py

Вариант 17. Задание: Напишите программу (individual.py), в которой вычисляется сумма, разность, произведение, частное и среднее арифметическое двух целых чисел, введенных с клавиатуры.



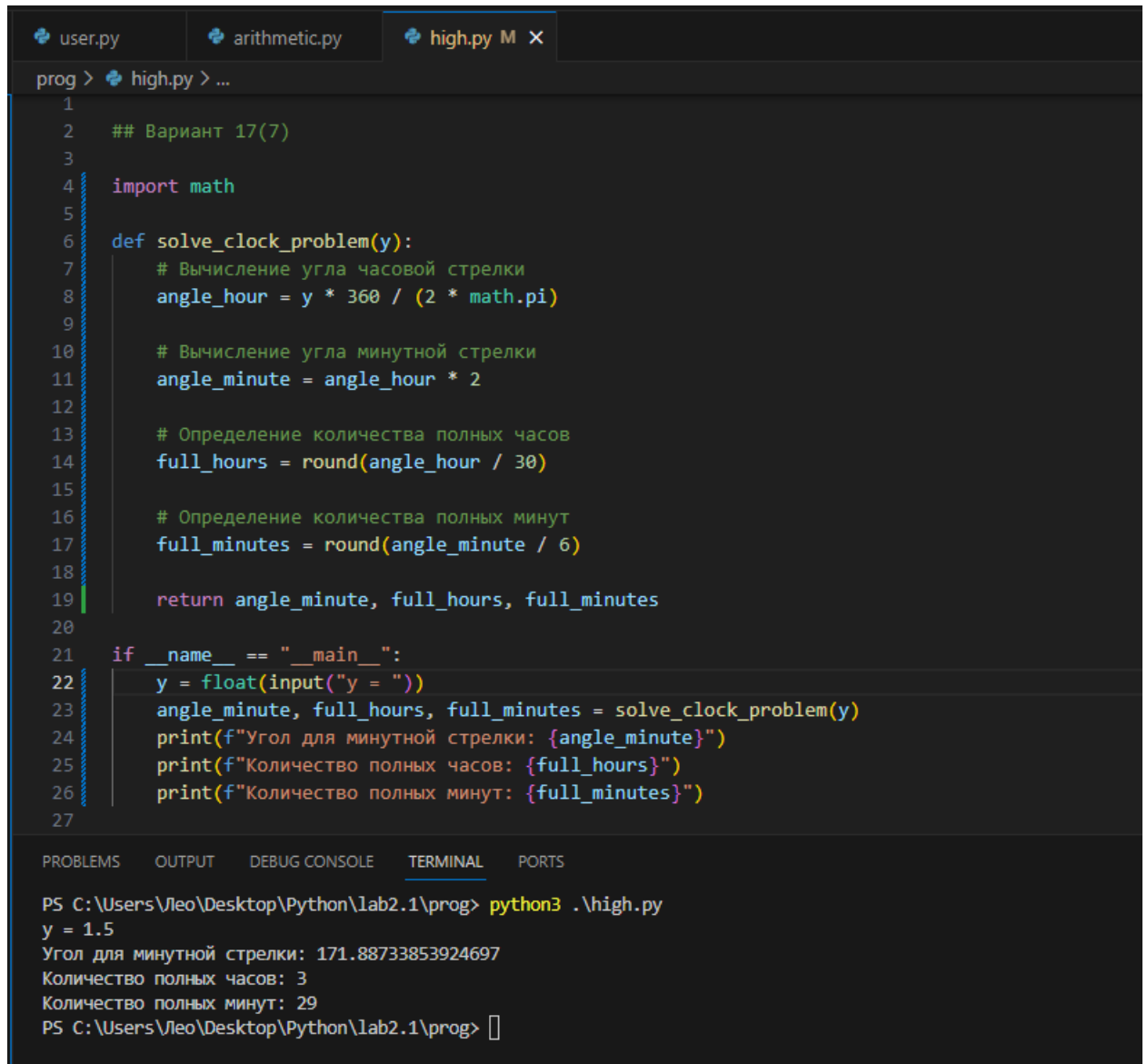
```
user.py arithmetic.py individual.py x
prog > individual.py > ...
1
2 ## 17 Вариант
3
4 if __name__ == "__main__":
5     x = int(input("x = "))
6     y = int(input("y = "))
7
8     print(f"{x}+{y}={x+y} {x}-{y}={x-y} {x}*{y}={x*y} {x}/{y}={x/y} ({x}+{y})/2={({x+y})/2}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\leo\Desktop\Python\lab2.1\prog> python3 .\individual.py
x = 2
y = 7
2+7=9 2-7=-5 2*7=14 2/7=0.2857142857142857 (2+7)/2=4.5
```

Рисунок 4 – Результат выполнения программы individual.py

Вариант 17(7). Задание: 7 Часовая стрелка образует угол  $y$  с лучом, проходящим через центр и через точку, соответствующую 12 часам на циферблате,  $0 < y < 2\pi$ . Определить значение угла для минутной стрелки, а также количество полных часов и полных минут.

Решение: написал программу (high.py), которая принимает на вход значение  $y$  в радианах, и выводит угол для минутной стрелки, количество полных часов, количество полных минут



```
user.py arithmetic.py high.py M X
prog > high.py > ...
1
2  ## Вариант 17(7)
3
4  import math
5
6  def solve_clock_problem(y):
7      # Вычисление угла часовой стрелки
8      angle_hour = y * 360 / (2 * math.pi)
9
10     # Вычисление угла минутной стрелки
11     angle_minute = angle_hour * 2
12
13     # Определение количества полных часов
14     full_hours = round(angle_hour / 30)
15
16     # Определение количества полных минут
17     full_minutes = round(angle_minute / 6)
18
19     return angle_minute, full_hours, full_minutes
20
21 if __name__ == "__main__":
22     y = float(input("y = "))
23     angle_minute, full_hours, full_minutes = solve_clock_problem(y)
24     print(f"Угол для минутной стрелки: {angle_minute}")
25     print(f"Количество полных часов: {full_hours}")
26     print(f"Количество полных минут: {full_minutes}")
27
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Leo\Desktop\Python\lab2.1\prog> python3 .\high.py
y = 1.5
Угол для минутной стрелки: 171.88733853924697
Количество полных часов: 3
Количество полных минут: 29
PS C:\Users\Leo\Desktop\Python\lab2.1\prog> 
```

Рисунок 5 – Результат выполнения программы high.py

Вывод: было проведено исследование процесса установки и базовых возможностей языка Python версии 3.x.

## Контрольные вопросы

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

В Windows: скачать дистрибутив, начать установку, выбрать все дополнения, выбрать директорию.

В Linux: команда `sudo apt-get install python3`

2. В чем отличие пакета Anaconda от пакета Python, скачиваемого с официального сайта: Основное отличие между пакетом Anaconda и пакетом Python, скачиваемым с официального сайта Python, заключается в том, что Anaconda уже предустановлен и настроен для работы с множеством популярных библиотек и инструментов, что делает его идеальным выбором для научных и аналитических задач. Он также обеспечивает простоту установки и управления пакетами, что упрощает работу с различными зависимостями и версиями библиотек.

3. Как осуществить проверку работоспособности пакета Anaconda: в anaconda Prompt ввести `jupyter notebook`, после чего в веб-сервере ввести команды `print ("hello, world!")` должен появиться вывод.

4. Как задать используемый интерпретатор языка Python в IDE PyCharm: при создании нового проекта.

5. Как осуществить запуск программы с помощью IDE PyCharm: нажать на Run "main"

6. В чём суть интерактивного и пакетного режимов работы Python: интерактивный – последовательный ввод и вывод, а пакетный берет и интерпретирует файл с кодом.

7. Почему язык программирования Python называется языком динамической типизации: тип переменной в Python определяется при выполнении программы.

8. Какие существуют основные типы в языке программирования Python: None, логические, числа, списки, строки, бинарные списки, множества, словари.

9. Как создаются объекты в памяти? Какого их устройство? В чем заключается процесс объявления новых переменных, и работа операции присваивания: Объекты в памяти создаются в Python при выполнении операции присваивания. При объявлении новой переменной, Python выделяет память для хранения значения этой переменной и связывает имя переменной с этой памятью. Устройство объектов в памяти зависит от их типа. В Python все является объектами, включая числа, строки, списки и даже функции.

Процесс объявления новых переменных в Python прост и не требует явного указания типа переменной. Просто присвойте значение переменной с помощью оператора "=" и Python автоматически определит тип переменной на основе присвоенного значения.

Работа операции присваивания в Python заключается в связывании имени переменной с объектом в памяти. Когда вы присваиваете значение переменной, Python создает объект в памяти, содержащий это значение, и связывает имя переменной с этим объектом. Если переменная уже существует, операция присваивания просто изменяет связанное с ней значение.

10. Как получить список ключевых слов в Python: В Python можно получить список ключевых слов с помощью модуля keyword. Этот модуль предоставляет список всех ключевых слов, которые используются в языке Python.

Вот пример кода, который демонстрирует, как получить список ключевых слов в Python:

```
import keyword  
keywords = keyword.kwlist  
print(keywords)
```

11. Каково назначение функций id() и type(): Функция id() в Python возвращает уникальный идентификатор объекта. Этот идентификатор является целым числом, которое является адресом объекта в памяти. Каждый объект в Python имеет свой уникальный идентификатор, даже если значения объектов одинаковы. Функция type() в Python возвращает тип объекта.

12. Что такое изменяемые и неизменяемые типы в Python: В Python существуют изменяемые (mutable) и неизменяемые (immutable) типы данных.

Неизменяемые типы данных в Python означают, что их значения не могут быть изменены после создания объекта. Когда значение неизменяемого объекта изменяется, на самом деле создается новый объект с новым значением. Примерами неизменяемых типов данных в Python являются числа (int, float, complex), строки (str) и кортежи (tuple).

Изменяемые типы данных в Python, наоборот, позволяют изменять их значения после создания объекта. При изменении значения изменяемого объекта не создается новый объект, а изменяется сам объект в памяти. Примерами изменяемых типов данных в Python являются списки (list), множества (set) и словари (dict).

13. Чем отличаются операции деления и целочисленного деления: Операции деления и целочисленного деления отличаются в Python следующим образом:

Операция деления (/): Оператор деления (/) выполняет обычное деление двух чисел и возвращает результат в виде числа с плавающей запятой (float). Независимо от типа операндов, результатом операции деления всегда будет число с плавающей запятой. Например, результатом деления 5 на 2 будет 2.5.

Операция целочисленного деления (//): Оператор целочисленного деления (//) выполняет деление двух чисел и возвращает результат в виде целого числа (int), округленного вниз до ближайшего целого значения. Результатом операции целочисленного деления всегда будет целое число, даже если операнды являются числами с плавающей запятой. Например, результатом целочисленного деления 5 на 2 будет 2.

14. Какие имеются средства в языке Python для работы с комплексными числами: В языке Python для работы с комплексными числами имеются следующие средства:

15. Встроенный тип данных complex: В Python есть встроенный тип данных complex, который позволяет создавать и работать с комплексными

числами. Комплексные числа представляются в виде  $a + bj$ , где  $a$  и  $b$  - это действительная и мнимая части соответственно.

16. Каково назначение и основные функции библиотеки (модуля) `math`? По аналогии с модулем `math` изучите самостоятельно назначение и основные функции модуля `cmath`: Модуль `math` в Python предоставляет функции для выполнения математических операций. Он содержит различные математические функции, константы и методы для работы с числами. Вот некоторые из основных функций и констант модуля `math`:

1) Функции тригонометрии: Модуль `math` содержит функции для вычисления тригонометрических значений, таких как синус (`sin()`), косинус (`cos()`), тангенс (`tan()`), арксинус (`asin()`), арккосинус (`acos()`), арктангенс (`atan()`), и другие.

2) Функции экспоненты и логарифмы: Модуль `math` предоставляет функции для вычисления экспоненты (`exp()`), натурального логарифма (`log()`), логарифма по основанию 10 (`log10()`), а также возведения в степень (`pow()`).

3) Математические константы: Модуль `math` также содержит некоторые математические константы, такие как число  $\pi$  (`pi`), экспонента (`e`), бесконечность (`inf`) и не число (`nan`).

Модуль `cmath` в Python предоставляет аналогичные функции для работы с комплексными числами. Он содержит функции для вычисления модуля (`abs()`), аргумента (`phase()`), сопряженного числа (`conjugate()`), а также другие операции с комплексными числами.

Каково назначение именных параметров `sep` и `end` в функции `print()`: Параметр `sep`: Этот параметр определяет разделитель между значениями, которые нужно вывести. По умолчанию `sep` равен пробелу. Вы можете изменить разделитель, установив значение параметра `sep` на нужное вам значение.

Параметр `end`: Этот параметр определяет, какой символ должен быть добавлен в конце вывода. По умолчанию `end` равен символу новой строки (`\n`).



Вы можете изменить это значение, установив значение параметра `end` на нужное вам значение.

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: Метод `format()` в Python используется для форматирования строк. Он позволяет вставлять значения переменных в определенные места в строке, называемые заполнителями. Заполнители указываются в фигурных скобках `{ }` внутри строки, и значения переменных подставляются вместо этих заполнителей. f-строки (f-strings): Введены в Python 3.6, f-строки позволяют встраивать значения переменных непосредственно в строку, используя префикс `f`. Внутри f-строки можно использовать выражения и переменные, заключенные в фигурные скобки.

18. Каким образом осуществить ввод с консоли значения целочисленной и вещественной переменной в языке Python: Для ввода значения целочисленной и вещественной переменной с консоли в языке Python можно использовать функцию `input()`. Если целочисленной перед `input` – `int()`, а вещественной – `float()`.