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 «Программирование на Python»**

**Вариант 23**

|  |  |  |
| --- | --- | --- |
|  | Выполнил:  Омонкулов Исомиддин Валижон угли  2 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | Руководитель практики:  Воронкин Р. А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

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

Ставрополь, 2024

**Тема:** Основы языка Python

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

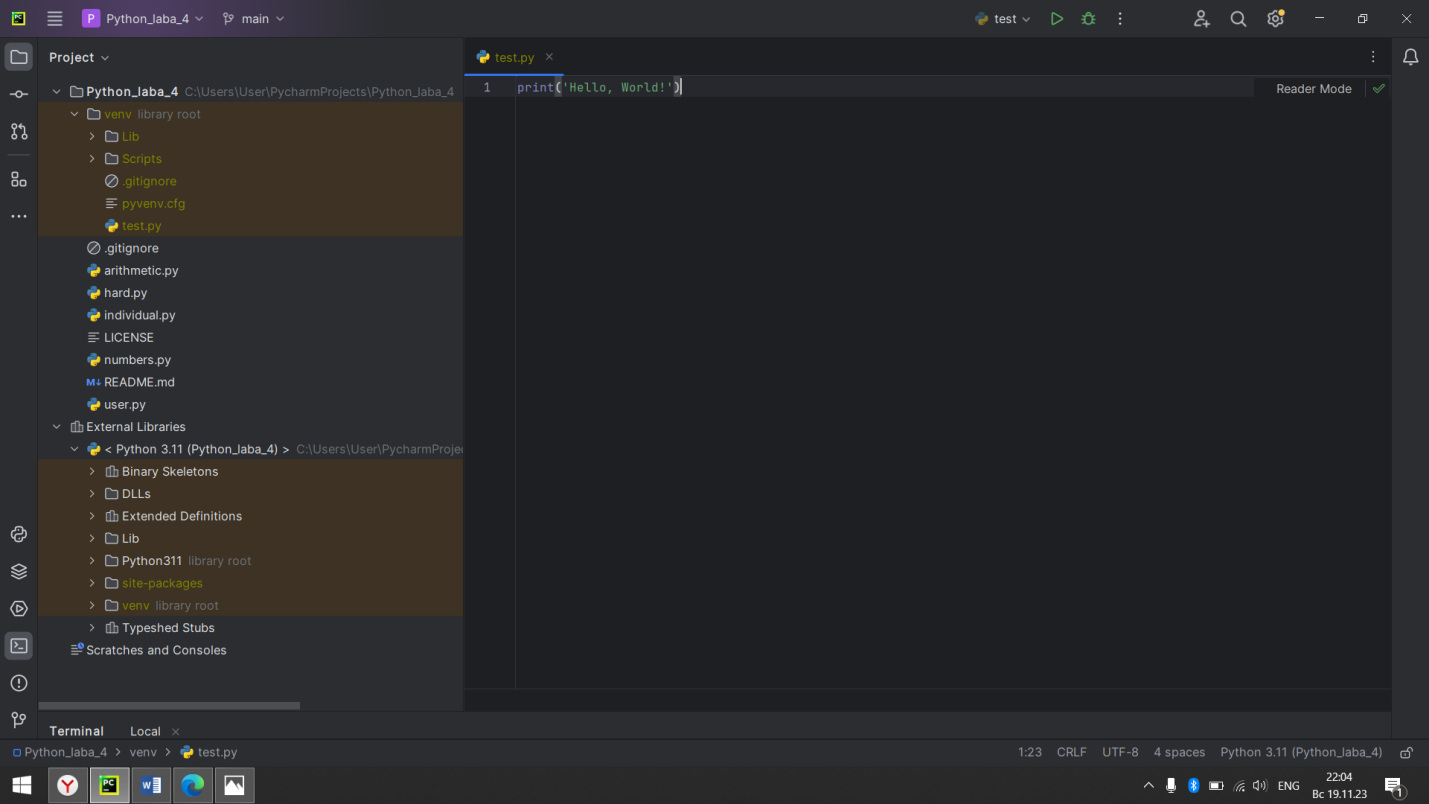


Рисунок 1. Рабочая среда разработки IDE PyCharm

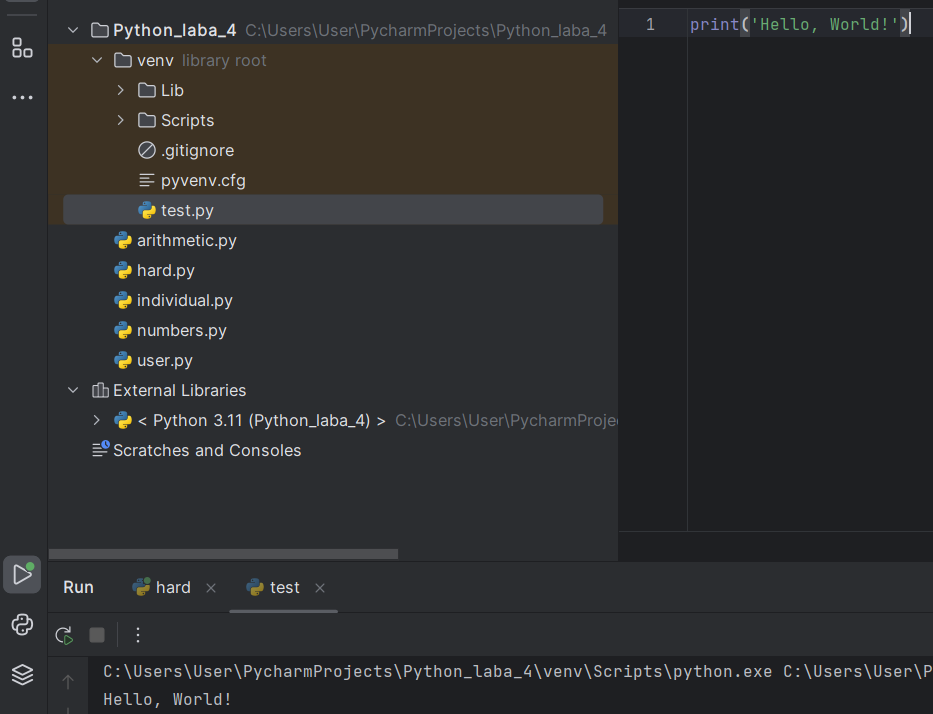


Рисунок 2. Выполнение команды и результат

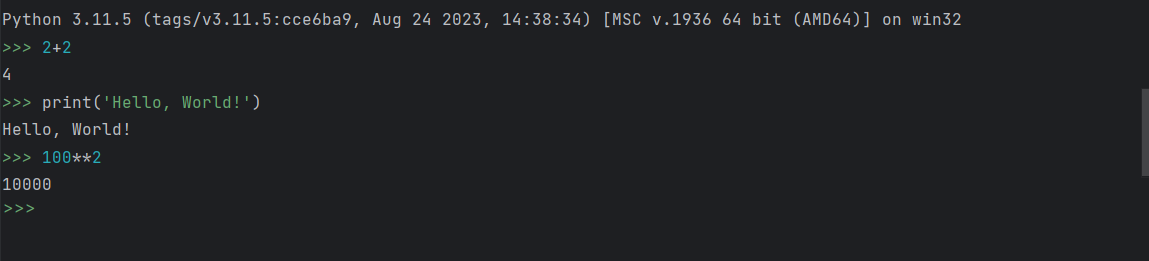


Рисунок 3. Выполнение команд в консоли Python

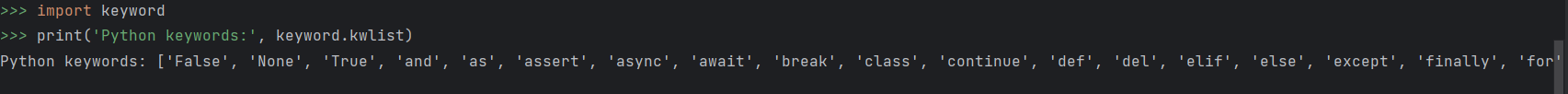


Рисунок 4. Список ключевых слов (фраз) в Python

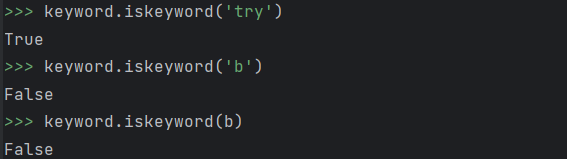


Рисунок 5. Проверка слов: являются ли они ключевыми или нет

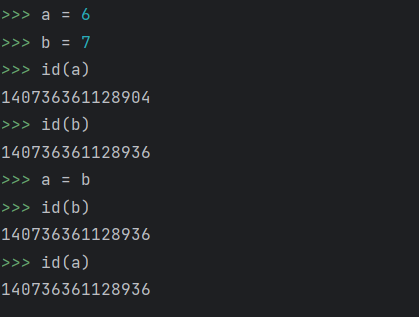


Рисунок 6. Работа с id переменных

Листинг программы **numbers.py**:

x\_1 = float(input('Enter the first number: '))  
x\_2 = float(input('Enter the second number: '))  
x\_3 = float(input('Enter the third number: '))  
x\_4 = float(input('Enter the fourth number: '))  
  
summa\_1 = x\_1 + x\_2  
summa\_2 = x\_3 + x\_4  
  
result = round(summa\_1 / summa\_2, 2)  
  
print(result)

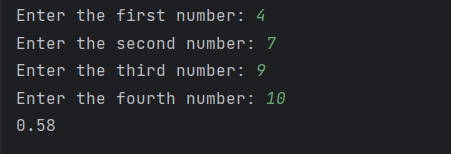


Рисунок 7. Результат к **numbers.py**

Листинг программы **arithmetic.py**:

task = '4 \* 100 - 54'  
print('Calculate and give the answer:', task)  
print()  
answer\_user = int(input('Write an answer here: '))  
print()  
if answer\_user == 400-54:  
 print('You are right!')  
 print('The correct answer:', 400-54)  
else:  
 print('To try again :(')  
 print('The correct answer:', 400 - 54)

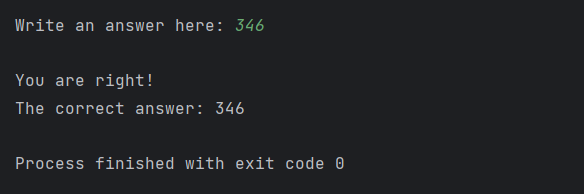


Рисунок 8. Результат к **arithmetic.py**

Листинг программы **individual.py**:

a = float(input('Enter a smaller base: '))  
b = float(input('Enter a larger base: '))  
h = float(input('Enter the height drawn on a larger base: '))  
  
bokovoe\_rebro = (h\*\*2 + ((b - a) / 2)\*\*2)\*\*(1 / 2)  
  
P = a + b + (2 \* bokovoe\_rebro)  
  
print('The perimeter is equal to', P)

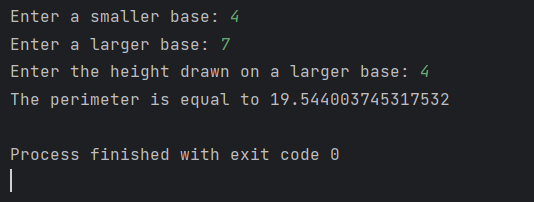


Рисунок 9. Результат к **individual.py**

Листинг программы **hard.py**:

y = int(input('Enter the angle value in DEGREES: '))  
  
if y > 0 and y <= 360:  
 if y % 30 == 0:  
 hours = y // 30  
 print("Full hours: ", hours)  
 minute\_nand\_angle = 0  
 print('Minute nand angle: ', minute\_nand\_angle)  
 minutes = 0  
 print('Full minutes:', minutes)  
 else:  
 x = y % 30 #counting the remaining minutes (1 hour = 30 degrees)  
 minutes = x \* 2  
 minute\_nand\_angle = minutes \* 6  
 print('Full hours:', y // 30)  
 print('Full minutes:', minutes)  
 print('Minute nand angle:', minute\_nand\_angle)  
else:  
 print('The angle should be greater than 0 degrees, but not exceed 360 degrees!')

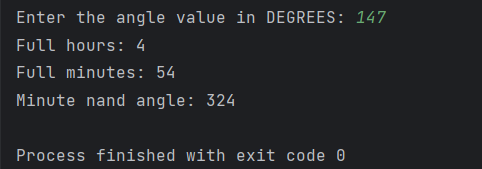


Рисунок 10. Результат к **hard.py**

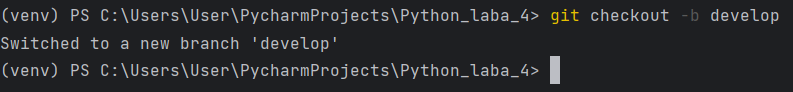


Рисунок 11. Создание ветки разработки

Далее я перешёл на ветку main, произвёл rebase и запушил изменения на удалённый репозиторий.

**Вывод:** в ходе выполнения данной лабораторной работы были исследованы процесс установки и базовые возможности языка Python 3.

**Ответы на контрольные вопросы**

1) Установка Python на Windows:

1. Скачайте установочный файл Python с официального сайта python.org.

2. Запустите установочный файл и выберите опцию "Add Python to PATH" для автоматической установки переменной среды PATH.

3. Выберите папку для установки Python и установите необходимые компоненты.

4. Дождитесь завершения установки и проверьте, что Python успешно установлен, запустив командную строку и введя "python --version".

Установка Python на Linux:

1. Откройте терминал и введите команду для установки Python, например, для Ubuntu это может быть "sudo apt-get install python3".

2. Дождитесь завершения установки и проверьте, что Python успешно установлен, запустив терминал и введя "python3 --version".

3. Убедитесь, что переменная среды PATH содержит путь к установленной версии Python.

После установки Python, рекомендуется установить менеджер пакетов pip, который позволит устанавливать дополнительные библиотеки и модули для Python. Для этого на Windows можно использовать команду "python -m ensurepip", а на Linux - установить пакет python3-pip с помощью менеджера пакетов вашего дистрибутива.

2) Пакет Anaconda - это дистрибутив Python, который включает в себя не только сам язык программирования, но и множество дополнительных библиотек и инструментов для анализа данных, машинного обучения, научных вычислений и визуализации. Он также включает в себя менеджер пакетов conda, который упрощает установку и управление библиотеками.

Пакет Python, скачиваемый с официального сайта, представляет собой базовый дистрибутив языка программирования Python без дополнительных инструментов и библиотек.

Таким образом, основное отличие заключается в том, что Anaconda предназначена в первую очередь для работы с данными и научными вычислениями, в то время как базовый пакет Python предоставляет только основные инструменты для разработки программ.

3) Для проверки работоспособности пакета Anaconda можно выполнить следующие шаги:

1. Запустить Anaconda Navigator, который является графическим интерфейсом для управления средами и пакетами в Anaconda. Если он успешно запускается, значит Anaconda установлена и работает корректно.

2. Открыть Jupyter Notebook, который часто используется для анализа данных и научных вычислений в Anaconda. Если Jupyter Notebook открывается без проблем, значит Anaconda установлена правильно.

3. Запустить командную строку и использовать менеджер пакетов conda для установки новых библиотек или обновления существующих. Если conda работает без ошибок, значит Anaconda установлена и функционирует нормально.

4. Запустить простой скрипт на Python, используя любой из установленных редакторов кода (например, Spyder). Если скрипт выполняется без проблем, значит Anaconda установлена и работает правильно.

4) Для задания используемого интерпретатора Python в IDE PyCharm нужно выполнить следующие шаги:

1. Откройте проект в PyCharm.

2. Перейдите в меню "File" -> "Settings" (для пользователей MacOS: "PyCharm" -> "Preferences").

3. В разделе "Project" выберите "Project Interpreter".

4. Нажмите на значок шестеренки рядом с выпадающим списком с текущим интерпретатором Python и выберите "Add..." для добавления нового интерпретатора.

5. Выберите тип интерпретатора (локальный, удаленный, виртуальное окружение и т. д.) и укажите путь к интерпретатору Python.

6. Нажмите "OK" для сохранения изменений.

5) Для запуска программы с помощью IDE PyCharm нужно выполнить следующие шаги:

1. Откройте проект в PyCharm.

2. Откройте файл с программой, которую вы хотите запустить.

3. Нажмите на зеленую кнопку "Run" рядом с функцией или методом, который вы хотите выполнить, или выберите "Run" -> "Run..." из верхнего меню.

4. Выберите конфигурацию запуска (например, файл с программой) и нажмите "OK".

5. После этого программа будет запущена, и вы увидите результат выполнения в консоли PyCharm.

6) Интерактивный режим работы Python позволяет пользователю вводить команды и сразу же получать результат их выполнения. Это удобно для тестирования отдельных команд или небольших фрагментов кода.

Пакетный режим работы Python предполагает написание программы в виде скрипта или модуля, который затем выполняется целиком. Результат выполнения программы выводится после ее завершения. Этот режим используется для создания более сложных программ и скриптов.

7) Python называется языком динамической типизации, потому что в нем тип переменной определяется автоматически во время выполнения программы, а не во время компиляции. Это означает, что переменная может быть присвоена значению любого типа, и тип переменной может быть изменен в любой момент выполнения программы. Это делает Python очень гибким и удобным для разработки, но также требует более внимательного контроля за типами данных в программе.

8) Основные типы данных в языке программирования Python включают:

1. Числовые типы:

- Целые числа (int)

- Вещественные числа (float)

- Комплексные числа (complex)

2. Строковые типы:

- Строки (str)

3. Логический тип:

- Логические значения True и False (bool)

4. Структуры данных:

- Списки (list)

- Кортежи (tuple)

- Множества (set)

- Словари (dict)

5. NoneType:

- Значение None, которое представляет отсутствие значения

9) Объекты в Python создаются при выполнении операций присваивания или создания новых объектов. Когда мы объявляем новую переменную и присваиваем ей значение, Python создает объект этого значения в памяти.

Устройство объектов в памяти Python зависит от их типа. Например, целые числа (int) хранятся как прямые значения, вещественные числа (float) хранятся в формате с плавающей запятой, строки (str) хранятся как последовательность символов и т.д.

10) В Python список ключевых слов можно получить с помощью модуля "keyword". Для этого нужно импортировать модуль и вызвать функцию "kwlist", которая вернет список всех ключевых слов в Python. Пример:

import keyword

print(keyword.kwlist)

11) Функция id() возвращает уникальный идентификатор объекта в Python, который является адресом объекта в памяти. Этот идентификатор может быть использован для сравнения объектов или отслеживания изменений в объекте.

Функция type() возвращает тип объекта в Python. Например, если передать объект в функцию type(), она вернет тип этого объекта, такой как int, str, list и т.д. Функция type() часто используется для проверки типа объекта перед выполнением определенных операций или методов.

12) В Python существуют изменяемые и неизменяемые типы данных. Неизменяемые типы данных означают, что их значения нельзя изменить после создания объекта. Примерами неизменяемых типов данных являются числа, строки и кортежи.

Изменяемые типы данных, наоборот, могут быть изменены после создания объекта. Примерами изменяемых типов данных являются списки, множества и словари.

13) Операция деления возвращает результат с плавающей запятой, то есть десятичную дробь, если она есть. Например, 5 / 2 = 2.5.

Операция целочисленного деления возвращает только целую часть результата деления, отбрасывая дробную часть. Например, 5 // 2 = 2.

14) В Python для работы с комплексными числами используются встроенные типы данных и функции, такие как:

1. Встроенный тип данных complex, который позволяет создавать комплексные числа в виде a + bj, где a и b - это вещественные числа, а j - мнимая единица.

2. Функции для работы с комплексными числами, такие как abs() для вычисления модуля комплексного числа, conjugate() для нахождения сопряженного комплексного числа, и другие.

3. Встроенные операции для работы с комплексными числами, такие как сложение, вычитание, умножение и деление.

Примеры использования комплексных чисел в Python:

# Создание комплексного числа

z1 = 2 + 3j

z2 = complex(4, 5)

# Сложение комплексных чисел

sum = z1 + z2

# Вычитание комплексных чисел

difference = z1 - z2

# Умножение комплексных чисел

product = z1 \* z2

# Деление комплексных чисел

quotient = z1 / z2

# Вывод результатов

print("Сумма:", sum)

print("Разность:", difference)

print("Произведение:", product)

print("Частное:", quotient)

15) Библиотека (модуль) math в Python предоставляет функции для выполнения математических операций. Назначение этой библиотеки - предоставить доступ к математическим функциям, таким как тригонометрические функции, логарифмы, экспоненты, константы и другие.

Модуль cmath, с другой стороны, предоставляет аналогичные функции для работы с комплексными числами. Он содержит функции для вычисления модуля, аргумента, сопряженного числа, а также для выполнения тригонометрических операций с комплексными числами.

Некоторые основные функции модуля cmath:

1. cmath.phase(z) - вычисляет аргумент комплексного числа z.

2. cmath.polar(z) - возвращает модуль и аргумент комплексного числа z в виде кортежа (r, phi).

3. cmath.rect(r, phi) - создает комплексное число по модулю r и аргументу phi.

4. cmath.exp(z) - вычисляет экспоненту комплексного числа z.

16) Именные параметры `sep` и `end` в функции `print()` позволяют управлять разделителями между значениями и окончанием вывода.

Параметр `sep` (сокращение от "separator", разделитель) определяет, какой символ или строка будет использоваться для разделения значений при их выводе на экран. Значение параметра `sep` по умолчанию - пробел. Но вы можете задать любой другой символ или строку.

Параметр `end` определяет, какой символ или строка будет использоваться в конце строки вывода. По умолчанию значение параметра `end` равно символу перевода строки `\n`. Вы также можете задать любой другой символ или строку.

17) Форматирование может выполняться в так называемом старом стиле или с помощью строкового метода format. Старый стиль также называют СИстилем, так как он схож с тем, как происходит вывод на экран в языке C. 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. Сами значения записываются в скобках после знака процента(%) Метод format() В строке в фигурных скобках указаны номера данных, которые будут сюда 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.

18) Каким образом осуществить ввод с консоли значения целочисленной и вещественной переменной в языке Python? Указать перед input() тип данных: int(input()), а для вещественной переменной float(input()).