

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
дисциплины
«Основы кроссплатформенного программирования
Вариант 15

Выполнил:
Мидов Андемир Исланов
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Проверил:

Воронкин Р. А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Основы языка Python

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

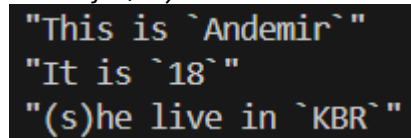
Ссылка GitHub: <https://github.com/midand-hash/laba4.git>

Порядок выполнения работы:

1. Написал программу “user.py” который запрашивает личные данные пользователя.

```
name = input("What is your name? ")
age = input("How old are you? ")
location = input("Where are you live? ")

print(f"\nThis is `{name}`\n")
print(f"\nIt is `{age}`\n")
print(f"\n(s)he live in `{location}`\n")
```



```
"This is `Andemir`"
"It is `18`"
"(s)he live in `KBR`"
```

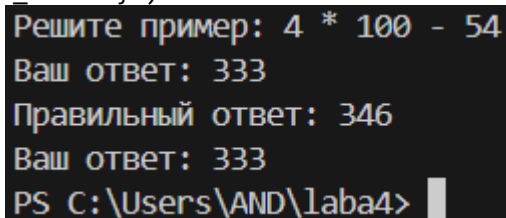
Рисунок 1. Результат user

2. Программа “arithmetic.py” предлагающий решить задачу.

```
print("Решите пример: 4 * 100 - 54")
user_answer = input("Ваш ответ: ")

correct_answer = 4 * 100 - 54

print(f"Правильный ответ: {correct_answer}")
print(f"Ваш ответ: {user_answer}")
```



```
Решите пример: 4 * 100 - 54
Ваш ответ: 333
Правильный ответ: 346
Ваш ответ: 333
PS C:\Users\AND\laba4>
```

Рисунок 2. Результат arithmetic

3. Программа “numbers.py” для умножения и деления введенных чисел

try:

```
num1 = float(input("Введите первое число: "))
num2 = float(input("Введите второе число: "))
num3 = float(input("Введите третье число: "))
num4 = float(input("Введите четвертое число: "))
```

```

except ValueError:
    print("Ошибка: Пожалуйста, введите числовые значения.")
    exit()

sum1 = num1 + num2

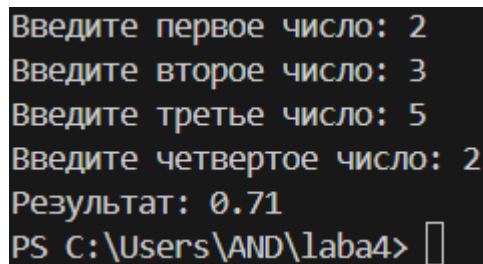
sum2 = num3 + num4

if sum2 == 0:
    print("Ошибка: Вторая сумма равна нулю, деление невозможно.")
    exit()

result = sum1 / sum2

print(f"Результат: {result:.2f}")

```



```

Введите первое число: 2
Введите второе число: 3
Введите третье число: 5
Введите четвертое число: 2
Результат: 0.71
PS C:\Users\AND\laba4>

```

Рисунок 3. Результат numbers

4. Написать программу для индивидуального задания

```

def calculate_distance(v1, v2, s):

    if v1 <= v2:
        raise ValueError("Скорость первого автомобиля должна быть больше скорости второго автомобиля")

    # Время в часах
    time_in_hours = 30 / 60 # 30 минут = 0.5 часа

    # Расстояние, которое первый автомобиль проедет за 30 минут
    distance_1 = v1 * time_in_hours

```

```

# Расстояние, которое второй автомобиль проедет за 30 минут
distance_2 = v2 * time_in_hours

# Увеличившееся расстояние между автомобилями
increased_distance = distance_1 - distance_2

# Общее расстояние между автомобилями
total_distance = s + increased_distance

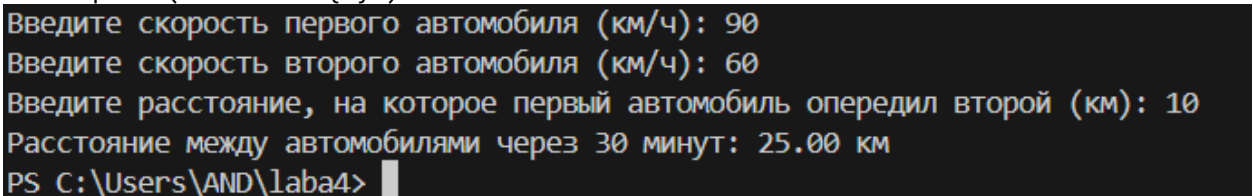
return total_distance

if __name__ == "__main__":
    try:
        v1 = float(input("Введите скорость первого автомобиля (км/ч): "))
        v2 = float(input("Введите скорость второго автомобиля (км/ч): "))
        s = float(input("Введите расстояние, на которое первый автомобиль опередил
второй (км): "))

        distance = calculate_distance(v1, v2, s)
        print(f"Расстояние между автомобилями через 30 минут: {distance:.2f} км")

    except ValueError as e:
        print(f"Ошибка: {e}")

```



```

Введите скорость первого автомобиля (км/ч): 90
Введите скорость второго автомобиля (км/ч): 60
Введите расстояние, на которое первый автомобиль опередил второй (км): 10
Расстояние между автомобилями через 30 минут: 25.00 км
PS C:\Users\AND\laba4>

```

Рисунок 4. Результат кода для индивидуального задания

Ответы на контрольные вопросы:

1. Основные этапы установки Python в Windows и Linux:

- Windows:
1. Скачать установщик с официального сайта python.org.
 2. Запустить установщик и следовать инструкциям (обычно выбирают "Add Python to PATH", чтобы Python был доступен из командной строки).
 3. Проверить установку, запустив команду `python --version` в командной строке.

Linux:

1. Python обычно предустановлен. Проверьте версию командой `python3 --version`.
2. Если Python отсутствует или нужна более новая версия, установите его через менеджер пакетов (например, `sudo apt install python3` для Debian/Ubuntu, `sudo yum install python3` для Fedora/CentOS).
3. Также через менеджер пакетов установите `pip` - `sudo apt install python3-pip`

2. Отличие пакета Anaconda от пакета Python, скачиваемого с официального сайта:

Python с официального сайта: Это минимальная установка Python, включающая только интерпретатор и базовые библиотеки.

Anaconda: Это дистрибутив Python, который включает в себя:

Сам интерпретатор Python.

Менеджер пакетов `conda`.

Большой набор предустановленных библиотек для научных вычислений, анализа данных, машинного обучения (например, NumPy, Pandas, SciPy, Matplotlib).

Множество инструментов и сред разработки.

Anaconda удобнее для тех, кто занимается анализом данных и научными исследованиями, поскольку она предоставляет сразу все необходимые инструменты.

3. Проверка работоспособности пакета Anaconda:

- Запустить Anaconda Prompt (в Windows) или терминал (в Linux/macOS).
- Ввести команду `python --version`, чтобы убедиться, что Python установлен.
- Ввести команду `conda --version`, чтобы убедиться, что conda (менеджер пакетов) установлен.
- Ввести команду `python`, чтобы запустить интерактивный интерпретатор Python.
- Импортировать какую-либо библиотеку (например, `import numpy`) и попробовать использовать её, чтобы проверить, что библиотеки доступны.

4. Задать используемый интерпретатор Python в IDE PyCharm:

- Открыть настройки проекта (File -> Settings -> Project: [имя проекта] -> Python Interpreter).
- Нажать на шестерёнку и выбрать "Add".
- Выбрать "System Interpreter" (если установлен Python в системе), "Virtual Environment" (для создания виртуального окружения) или "Conda Environment" (если используется Anaconda).
- Выбрать путь до исполняемого файла интерпретатора Python.
- Нажать "ОК" и применить изменения.

5. Осуществить запуск программы с помощью IDE PyCharm:

Создать новый файл Python (File -> New -> Python File).

Написать код программы.

Нажать правой кнопкой мыши на код и выбрать "Run 'имя_файла.py'" или нажать зеленую кнопку "Run" сверху.

6. Суть интерактивного и пакетного режимов работы Python:

Интерактивный режим:

Запускается, когда вы вводите команду `python` в терминале/командной строке.

Позволяет вводить команды Python по одной и сразу видеть результат.

Удобно для тестирования кода, изучения Python и быстрых вычислений.

Пакетный режим:

Запускается, когда вы выполняете файл Python (например, `python my_script.py`).

Программа считывает и выполняет все команды из файла по порядку.

Используется для создания программ, которые выполняют сложные задачи.

7. Почему Python называют языком с динамической типизацией:

В Python тип переменной определяется динамически во время выполнения, а не явно при объявлении.

Одна и та же переменная может хранить значения разных типов в разное время.

Это делает Python более гибким, но может привести к ошибкам во время выполнения, если не следить за типами данных.

8. Основные типы данных в Python:

None, Логические переменные, Числа, Списки, Строки, Бинарные списки, Множества, Словари.

9. Как создаются объекты в памяти, их устройство и процесс объявления новых переменных:

Когда в Python объявляется переменная (например, `x = 5`), интерпретатор Python:

1. Создает в памяти объект, представляющий значение 5 (целое число).

2. Привязывает имя переменной `x` к этому объекту.

Объекты в Python - это области памяти, которые хранят:

- * Значение данных.
 - * Тип данных.
 - * Ссылку на другие объекты (при необходимости).
 - * Методы и атрибуты, позволяющие объекту выполнять определенные действия.
- * Присваивание: Операция присваивания (`=`) связывает имя переменной с конкретным объектом. Если переменная уже существует, то она просто перенаправляется на новый объект.

10. Как получить список ключевых слов в Python:

Импортировать модуль `keyword`: `import keyword`.

Вызвать функцию `keyword.kwlist`: `print(keyword.kwlist)`.

11. Назначение функций `id()` и `type()`:

`id(object)`: Возвращает уникальный идентификатор (адрес в памяти) объекта.

`type(object)`: Возвращает тип объекта.

12. Изменяемые и неизменяемые типы данных в Python:

Изменяемые:

`list`

`dict`

`set`

Неизменяемые:

`int, float, complex`

`bool`

`str`

`tuple`

13. Отличие операций деления (/) и целочисленного деления (//):

/: Выполняет обычное деление и всегда возвращает результат типа float

//: Выполняет целочисленное деление (деление с отбрасыванием остатка) и возвращает результат типа int если оба операнда целые, и float если один из операндов вещественный.

14. Средства Python для работы с комплексными числами:

Комплексные числа в Python представлены типом complex.

Можно создавать с помощью complex(real, imaginary) или a + bj.

Для доступа к действительной и мнимой частям использовать атрибуты .real и .imag.

15. Назначение и основные функции модуля math и модуля cmath:

math: Предоставляет математические функции для обычных чисел (вещественных, целых) - math.sqrt(), math.sin(), math.cos(), math.log(), math.pow() и др.

cmath: Предоставляет математические функции для комплексных чисел.

16. Назначение именованных параметров sep и end в функции print():

sep: Определяет разделитель между аргументами, по умолчанию пробел.

end: Определяет, что будет добавлено в конце вывода, по умолчанию перенос строки.

17. Назначение метода format() и другие средства форматирования строк:

Метод format(): Используется для форматирования строк, подставляя значения в {} на основе позиции или имени.

Другие средства:

f-строки (удобнее и быстрее, чем `format()`).

Форматирование строк с помощью `%` (устаревший способ).

18. Осуществление ввода целочисленной и вещественной переменных с консоли:

`int(input("Введите целое число: "))` для ввода целого числа.

`float(input("Введите вещественное число: "))` для ввода вещественного числа.

Вывод: исследован процесс установки и базовых возможностей языка Python версии 3.x.