Лабораторная работа №1

Основы языка Python

1. Цель работы

Целью работы является ознакомление с основами языка Python.

2. Теоретическая часть

Python — это мощный и гибкий язык программирования, который был создан Гвидо ван Россумом и впервые представлен в 1991 году. Python является многофункциональным языком, который может быть полезен в различных областях.

Python широко используется в различных областях программирования и разработки:

- Web-разработка: Python предлагает несколько полезных фреймворков для разработки веб-сайтов.
- Наука о данных: Python является одним из основных инструментов в науке о данных и машинном обучении.
- Автоматизация: благодаря своей простоте и мощной стандартной библиотеке, Python идеально подходит для автоматизации задач.
- Образование: Python часто используется в качестве первого языка программирования в университетах и школах из-за своей простоты и читаемости.

Python является интерпретируемым языком программирования, это значит, что интерпретатор преобразует исходный код частями, последовательно, строку за строкой, в отличие от компилируемых языков, например, C или Java, для которых исходный код программы преобразуется в машинный код целиком

Другой отличительной чертой языка Python является краткий и понятный синтаксис, который характеризуется ограниченным использованием вспомогательных синтаксических элементов, таких как скобки и точки с запятыми. Вместо них для выделения блоков кода используются отступы, что упрощает зрительное восприятие программ

На основании лицензии подобной General Public License (GNU) интерпретаторы Python распространяются свободно.

2.1 Установка Python

Для работы с Python на Windows будет достаточно скачать установщик с официального сайта https://www.python.org/. Необходимо перейти в раздел «Downloads» и выбрать из выпадающего списка «Windows» (на момент написания методических указаний актуальной версией является Python 3.12.6). Далее потребуется скачать «Windows installer (64-bit)» или «Windows installer (32-bit)» в зависимости от типа вашей операционной системы. После запустить установку, как показано на рисунке 1.

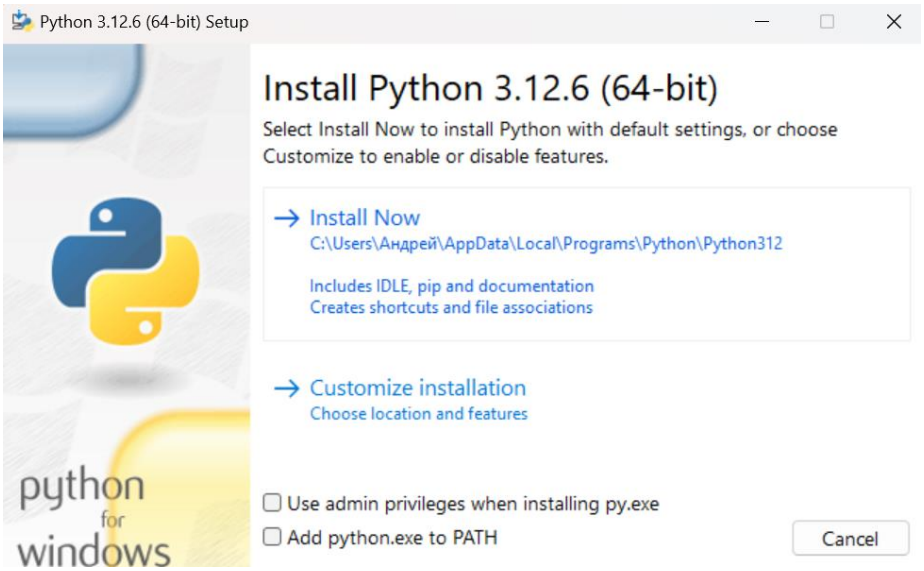

Рис. 1. Windows installer python

Ставим галочку «Add python.exe to PATH» и выбираем «Install Now». После успешного завершения установки, потребуется открыть командный интерпретатор и проверить доступность python, как показано на рисунке 2.

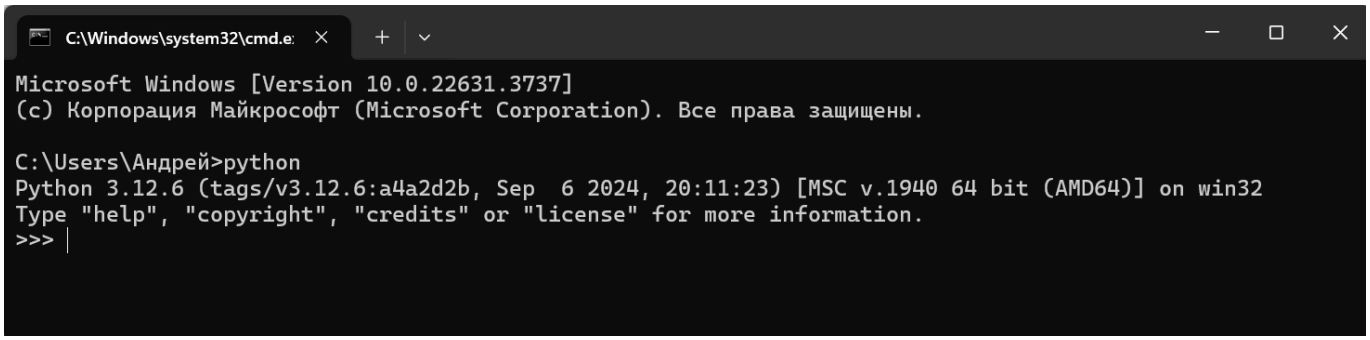

Рис. 2. Python в командной строке

По мимо работы через интерпретатор командной строки, возможно использовать IDLE Shell, который поставляется совместно с Python. В меню «Пуск» в поиске набрать IDEL Python и выбрать соответствующее приложение, как показано на рисунке 3.

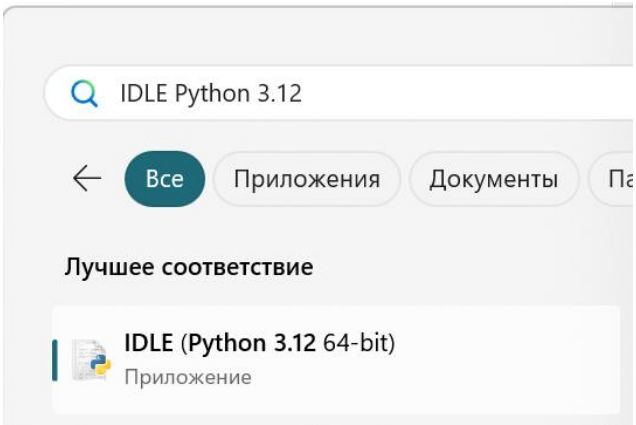

Рис. 3. Поиск меню «Пуск»

В результате откроется окно Python 3.12 Shell, выполняющее все функции интерактивной оболочки, как показано на рисунке 4.

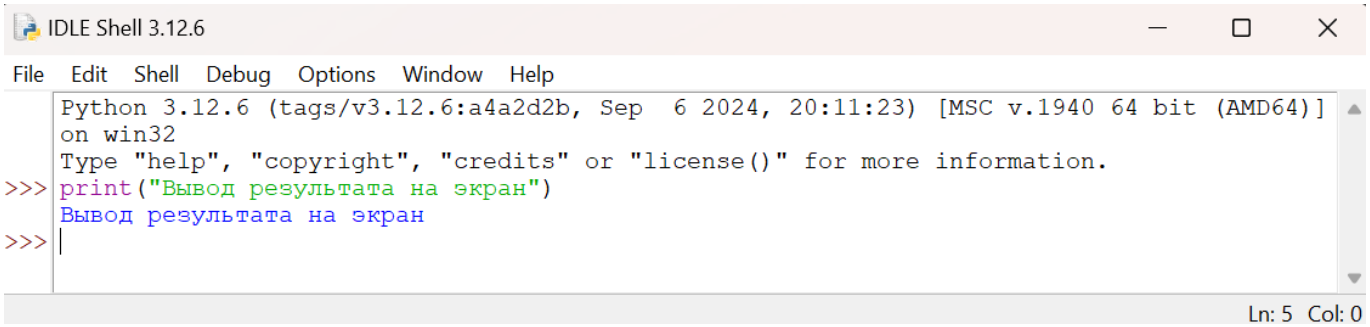

Рис. 4. Окно Python 3.12.6

Также для работы с языком Python возможно использовать интегрированные среды разработки (IDE - Integrated Development Environment) такие как VS Code, Spyder, Atom, PyCharm. Выбор подходящей IDE зависит от требований, рабочего стиля и проекта, над которым ведется работа. Установка и настройка IDE описана на официальном сайте разработчика.

2.2 Запуск программы

После того, как дистрибутив актуальной версии Python установлен, можно перейти к первому запуску программы. Программа будет выполнять сложение двух целых чисел и выводить результат на экран.

В окне Python 3.12.6 нажать «File» и из выпадающего списка выбрать «New File». Код программы показан на рисунке 5.

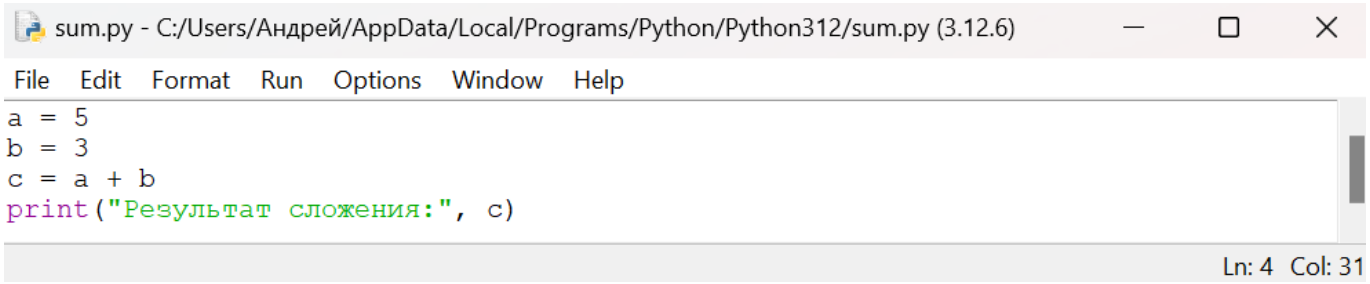

Рис. 5. Файл с программой

Для запуска программы и получения результата работы потребуется нажать на «Run» и выбрать «Run Module» или нажать F5. Результат выполнения показан на рисунке 6.

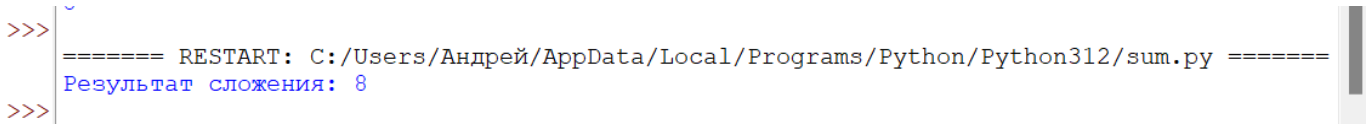

Рис. 6 Результат выполнения

Работа в редакторе IDLE Python представляет собой интерактивный режим, но при сохранение исходного кода с расширением **.py* получаем программу.

2.3 Синтаксис языка Python

Синтаксис языка Python предусматривает соблюдение следующих основных правил:

1. конец строки является концом инструкции;

2. вложенные инструкции объединяются в блоки по величине отступов (индентация), каждый из отступов может быть любым, но должен иметь одинаковое значение в пределах одного вложенного блока;
3. вложенные инструкции записываются в соответствии с одним и тем же шаблоном: основная инструкция завершается двоеточием, а вложенный блок кода располагается с отступом под строкой основной инструкции;
4. возможна запись нескольких инструкций в одной строке при условии разделения их точкой с запятой, например,

   a=5;b=4;c=a+b;print(c)
5. возможна запись одной инструкции на нескольких строках при условии ее заключения в круглые, квадратные или фигурные скобки, например,

   if (a==1 and
   b==2)
6. тело составной инструкции может располагаться в той же строке, что и тело основной, если оно не содержит составных инструкций, например,

   if a>b: print(c)

Одной из особенностей Python является его читаемый и чистый синтаксис. Это одна из причин, почему Python стал таким популярным. Код Python выполняется построчно, с верхней строки вниз. Каждая строка кода обычно содержит одну команду. В Python блоки кода (например, тело функции или цикла) определяются отступами. Каждый уровень отступа обычно состоит из 4 пробелов или одного табулятора. Это делает код Python легко читаемым и понятным.

В Python есть комментарии – строки, которые не выполняются интерпретатором Python. В Python комментарии начинаются с символа #.

```
# Это однострочный комментарий
print("Привет, мир!")
""" Это
    многострочный
    комментарий"""
```

2.4 Типы данных и переменные Python

Python относится к языкам с неявной сильной динамической типизацией. Неявная типизация означает, что при объявлении переменной нет необходимости указывать ее тип, при этом динамическая означает, что тип переменной определяется непосредственно при выполнении программы, а сильная свидетельствует, что нельзя производить операции в выражениях с данными различных типов.

Python упаковывает каждое значение, например, целые числа, числа с плавающей точкой, строки и даже крупные структуры данных, функции и программы, в памяти как объекты. Объектом является фрагмент данных, в котором содержится, как минимум его уникальный номер (id – представляет собой адрес первого байта объекта в памяти), тип объекта (целый, вещественный и др.), счётчик ссылок на данные объект.

Для того чтобы объявить и сразу инициализировать переменную, следует написать ее имя, поставить знак равенства, а затем – значение.

Объекты, описываемые на языке Python, могут быть представлены следующими типами данных:

1. *None Type* (неопределенное значение переменной) – объект со значением None, обозначающих отсутствие значения.
2. *Boolean Type* – логический тип данных, обозначаемый через bool и принимающий значения True и False.
3. *Numeric Type* – числовой тип данных, к которому относятся:
    a. *int – целые числа*
    b. *float – вещественные числа (числа с плавающей точкой)*
    c. *complex – комплексные числа*
4. *Sequence Type* – тип данных список, который может быть представлен одним из следующих видов:
    a. list – список
    b. tuple – кортеж
    c. range – диапазон
5. *Text sequence Type* – строковый тип данных, обозначаемый через *str*.
6. *Binary Sequence Type* – бинарные списки, включающие в себя:
    a. *bytes – байты*
    b. *bytearray – массивы байт*
    c. *memoryview – специальные объекты, предназначенные для доступа к внутренним данным*
7. *Set Type* – тип данных множества, состоящий из:
    a. set – множество
    b. frozen set – неизменяемое множество
8. *Mapping Types* – тип данных словари, обозначаемый через *dict*.

2.5 Ввод и вывод данных

Ввод и вывод данных являются важными аспектами программирования. Без возможности ввода данных программы выполняли бы одну и ту же последовательность действий, за исключением случаев, когда значения генерируются случайным образом внутри программы. Вывод данных позволяет визуализировать, использовать и передавать результаты работы программы.

В Python за вывод данных отвечает функция *print()*, которая выводит содержимое, указанное внутри ее скобок, на экран. В скобках могут быть любые типы данных.

```
print("Hello, World!")
```

За ввод данных в программу с клавиатуры в Python отвечает функция *input()*. При вызове данной функции, программа останавливает свое выполнение и ожидает, когда пользователь введет текст и нажмет Enter. После того как текст был введен, функция передает его программе

```
name = input("Введите ваше имя: ")
print("Привет,", name)
```

Обратите внимание, что функция input() всегда возвращает данные в виде строки и всегда завершается на переводе строки (то есть при нажатии Enter). Если

вы хотите работать с числами, вам нужно преобразовать эту строку в число с помощью функций int() или float().

```
age_str = input("Введите ваш возраст: ")
age = int(age_str)
print("Через год вам будет", age + 1)
```

2.6 Арифметические операторы

Арифметические операторы используются для выполнения математических операций:

Оператор	Операция	Пример	Результат
+	Сложение	4 + 3	7
-	Вычитание	3 - 1	2
*	Умножение	5 * 2	10
/	Деление	7 / 2	3.5
//	Целочисленное деление	5 // 2	2
%	Модуль (остаток от деления)	7 % 2	1
**	Возведение в степень	6 ** 2	36

2.7 Операторы сравнения

Операторы сравнения используются для сравнения двух значений:

Оператор	Операция	Пример	Результат
==	Равно	3 == 2	False
!=	Не равно	4 != 9	True
>	Больше	5 > 2	True
<	Меньше	5 < 2	False
>=	Больше или равно	3 >= 2	True
<=	Меньше или равно	5 <= 2	False

2.8 Логические операторы

Логические операторы используются для комбинирования условных выражений:

Оператор	Операция
and	Логическое И
or	Логические ИЛИ
not	Отрицание НЕ

2.9 Контроль потока

2.9.1 Оператор if

Оператор if используется для создания условной инструкции, которая выполняет определенный блок кода, если условие истинно.

```
x = 10
if x > 0:
    print("x - положительное число")
```

В этом примере код внутри блока if будет выполнен, только если x больше нуля.

2.9.2 Оператор else

Оператор else используется вместе с if для определения блока кода, который будет выполнен, если условие if ложно.

```
x = -5
if x > 0:
    print("x - положительное число")
else:
    print("x - не положительное число")
```

В этом примере, если x больше нуля, будет напечатано «x — положительное число». Если же x не больше нуля (то есть меньше или равно), будет напечатано «x — не положительное число».

2.9.3 Оператор elif

Оператор elif (сокращенно от «else if») используется для добавления дополнительных условий к конструкции if/else.

```
x = 0
if x > 0:
    print("x - положительное число")
elif x < 0:
    print("x - отрицательное число")
else:
    print("x равно нулю")
```

В этом примере, в зависимости от значения x, будет напечатано либо «x — положительное число», либо «x — отрицательное число», либо «x равно нулю».

Операторы if, else и elif являются основой управления потоком выполнения в Python и позволяют вашему коду реагировать на различные ситуации.

2.9.4 Цикл for

Цикл for в Python используется для итерации по последовательности (это может быть список, кортеж, строка или диапазон чисел).

```
for i in range(5):  # range(5) создает последовательность чисел от 0 до 4
    print(i)  # Этот код будет выполнен 5 раз, с каждым числом от 0 до 4
```

2.9.5 Цикл while

Цикл while в Python повторяет блок кода, пока условие истинно.

```
i = 0
while i < 5:  # Этот код будет выполняться, пока i меньше 5
    print(i)
    i += 1  # Увеличиваем i на 1 после каждого прохода цикла
```

2.9.6 Управление циклами: break, continue

break используется для преждевременного выхода из цикла.

continue используется для пропуска оставшейся части текущей итерации цикла и немедленного перехода к следующей итерации.

```
for i in range(10):
    if i == 3:
        continue  # Если i равно 3, пропустим остаток этой итерации и перейдем к следующей
    if i == 7:
        break  # Если i равно 7, прервем цикл
    print(i)  # Этот код будет выполнен для каждого числа от 0 до 6
```

В этом примере числа 0-2 и 4-6 будут напечатаны. Когда i равно 3, оператор continue пропускает остаток итерации, поэтому 3 не выводится. Когда i равно 7, оператор break прерывает цикл, поэтому числа 7-9 не выводятся.

Циклы и операторы управления циклами являются важной частью Python и позволяют автоматизировать и повторять операции в вашем коде.

2.9.7 Функция range()

Функция range() в Python используется для генерации последовательности чисел. Она часто используется в циклах for для контроля количества итераций.

Основное использование функции range() выглядит следующим образом:

```
for i in range(5):
    print(i)  # Выведет числа от 0 до 4
```

Функция range() может принимать от одного до трех аргументов:
range(stop): Генерирует числа от 0 до stop - 1.
range(start, stop): Генерирует числа от start до stop - 1.
range(start, stop, step): Генерирует числа от start до stop - 1 с шагом step.

2.9.8 Функция enumerate()

Функция enumerate() применяется к последовательности (например, списку) и создает объект enumerate, который генерирует пары, состоящие из индекса и соответствующего ему элемента из последовательности.

```
fruits = ["apple", "banana", "cherry"]
for i, fruit in enumerate(fruits):
    print(i, fruit)  # Выведет "0 apple", "1 banana", "2 cherry"
```

В этом примере enumerate(fruits) генерирует пары (0, «apple»), (1, «banana») и (2, «cherry»). Цикл for затем итерирует по этим парам, и переменные i и fruit принимают значения каждой пары.

Функции range() и enumerate() являются полезными инструментами при работе с циклами и последовательностями в Python.

3. Задание на лабораторную работу

Выполнить установку актуальной версии Python. Настроить IDE для работы с языком Python. Реализовать и выполнить простые программы на Python.

4. Методика выполнения задания

1. Изучить теоретическую часть.
2. Выполнить установку Python.
3. Выполнить установку и настройку одной из IDE.
4. С использованием IDLE Python реализовать программу, которая запрашивает ваше имя и фамилию, выполняет приветствие с указанием вашего имени и фамилии.
5. Выполнить в интерактивном режиме по одном примеру для арифметических, логических и операцией сравнений. Каждый пример должен включать однострочные комментарии для пояснения операции. Для арифметических примеров первое число должно быть номером студенческого билета.
6. Для пункта 2.9 раздела 2 реализовать примеры программ в виде одного выполняемого файла с расширением *.py*.

5. Требования к содержанию и оформлению отчета

Отчет по лабораторной работе должен содержать:

а) титульный лист;

б) описание хода выполнения работа;

в) заключение по выполненной работе.