Оглавление

[Требования к выполнению лабораторных работ 3](#_Toc129808838)

[Правила оформления отчёта 3](#_Toc129808839)

[ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА №1 5](#_Toc129808840)

[Теоретическое введение 5](#_Toc129808841)

[Варианты заданий 10](#_Toc129808842)

[Порядок выполнения лабораторной работы 10](#_Toc129808843)

[Контрольные вопросы 10](#_Toc129808844)

[ЛАБОРАТОРНАЯ(ПРАКТИЧЕСКАЯ) РАБОТА № 2 11](#_Toc129808845)

[Теоретическое введение 11](#_Toc129808846)

[Варианты заданий 22](#_Toc129808847)

[Порядок выполнения лабораторной работы 22](#_Toc129808848)

[Контрольные вопросы 23](#_Toc129808849)

[ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА №3 24](#_Toc129808850)

[Теоретическое введение 24](#_Toc129808851)

[Варианты заданий 35](#_Toc129808852)

[Порядок выполнения лабораторной работы 35](#_Toc129808853)

[Контрольные вопросы 35](#_Toc129808854)

[ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА № 4 36](#_Toc129808855)

[Теоретическое введение 36](#_Toc129808856)

[Варианты заданий 39](#_Toc129808857)

[Порядок выполнения работы 39](#_Toc129808858)

[Контрольные вопросы 39](#_Toc129808859)

[ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА № 5 40](#_Toc129808860)

[Теоретическое введение 40](#_Toc129808861)

[Варианты заданий 44](#_Toc129808862)

[Порядок выполнения работы 44](#_Toc129808863)

[Контрольные вопросы 44](#_Toc129808864)

[ЛАБОРАТОРНАЯ(ПРАКТИЧЕСКАЯ) РАБОТА № 6 45](#_Toc129808865)

[Теоретическое введение 45](#_Toc129808866)

[Варианты заданий 52](#_Toc129808867)

[Порядок выполнения лабораторной работы 52](#_Toc129808868)

[Контрольные вопросы 52](#_Toc129808869)

[ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА № 7 53](#_Toc129808870)

[Теоретическое введение 53](#_Toc129808871)

[Варианты заданий 58](#_Toc129808872)

[Порядок выполнения лабораторной работы 58](#_Toc129808873)

[Контрольные вопросы 59](#_Toc129808874)

[ЛАБОРАТОРНАЯ(ПРАКТИЧЕСКАЯ) РАБОТА № 8 60](#_Toc129808875)

[Теоретическое введение 60](#_Toc129808876)

[Варианты заданий 72](#_Toc129808877)

[Порядок выполнения лабораторной работы 72](#_Toc129808878)

[Контрольные вопросы 73](#_Toc129808879)

# Требования к выполнению лабораторных работ

В ходе выполнения лабораторной работы студент должен написать программный код на языке программирования Python, выполняющий основное задание лабораторной работы, оформить отчет и отчитаться преподавателю по итогам выполненной лабораторной работы. Далее подробно описаны требования по выполнению каждого этапа лабораторной работы.

**Программный код**

Студент должен написать программный код, выполняющий основное задание лабораторной работы.

К программному коду предъявляются следующие требования:

1) именование переменных должно быть осмысленным;

2) в коде программы должны присутствовать комментарии;

3) код должен производить форматированный вывод результатов;

4) код должен запрашивать пользователя входные данные;

5) код должен иметь базовый уровень проверки входных данных.

# Правила оформления отчёта

Отчёт по лабораторной работе оформляется студентом в отдельной от лекций рабочей тетради, не использующей сменные блоки листов.

Содержание отчёта:

Титульный лист.

Цель лабораторной работы.

Теоретическое введение.

Выполнение работы.

Вывод по лабораторной работе.

Титульный лист выполняется к каждой лабораторной работе. Он содержит: наименования вышестоящей организации, учебного заведения, кафедры, на которой выполняется работа, а также номер и название самой работы. Также указывается, кто выполнил работу (фамилия, инициалы и номер группы студента) и кто проверил работу (учёные степень, звание, фамилия и инициалы). В нижней части титульного листа указывается город и год выполнения работы.

Цель выполнения приведена в описании лабораторной работы. Она размещается на странице, следующей за титульным.

Теоретическое введение содержит основные положения, на которых основана выполняемая работа. Его объем должен составлять не менее половины и не более 1 страницы.

В разделе «Выполнение работы» приводятся исходные данные, этапы решения задания, составленный код на языке программирования Python. Подробное содержание этого раздела приводится в разделе «Порядок выполнения работы» для каждой лабораторной работы.

Вывод содержит изложение результатов работы, полученных в ходе её выполнения, с учётом цели работы, если в разделе «Выполнение работы» не указано иное.

# ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА №1

**Тема:** среда разработки языка **Python** - ANACONDA

**Цель:** познакомиться с возможностями среды разработки языка **Python** – Spyder. Типы данных, операторы «ввода» и «вывода».

## Теоретическое введение

Краткая карточка языка **Python**

| **Характеристика** | **Значение** |
| --- | --- |
| Автор | [Гвидо ван Россум](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D0%BD_%D0%A0%D0%BE%D1%81%D1%81%D1%83%D0%BC,_%D0%93%D0%B2%D0%B8%D0%B4%D0%BE) и [**Python** Software Foundation](https://ru.wikipedia.org/wiki/Python_Software_Foundation) |
| Официальный сайт | [http://www.**Python**.org/](http://www.python.org/) |
| Год создания | 1991 |
| Класс языка | Императивный, функциональный, объектно-ориентированный |
| Тип исполнения | Интерпретируемый, компилируемый в байт-код |
| Расширения файлов | .py, .pyc и др. |
| Кодировка по умолчанию | [UTF-8](https://ru.wikipedia.org/wiki/UTF-8) |
| Основные реализации | [C**Python**](https://ru.wikipedia.org/wiki/CPython) (эталонная реализация), [Jython](https://ru.wikipedia.org/wiki/Jython" \t "_blank), [Iron](https://ru.wikipedia.org/wiki/IronPython" \t "_blank)**[Python](https://ru.wikipedia.org/wiki/IronPython" \t "_blank)** и др. |
| Операционная система | [Кроссплатформенное ПО](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) |
| Лицензия | [**Python** Software Foundation License](https://ru.wikipedia.org/wiki/Python_Software_Foundation_License) |

1. Установка среды разработки ANACONDA;

[WWW.ANACONDA.COM](file:///G:\ВГЛТУ\Прикладное%20программирование\Лабораторные%20работы\WWW.ANACONDA.COM)

1. После установки пакета программ производим загрузку Spyder

(Пуск-Anaconda3-Spyder);

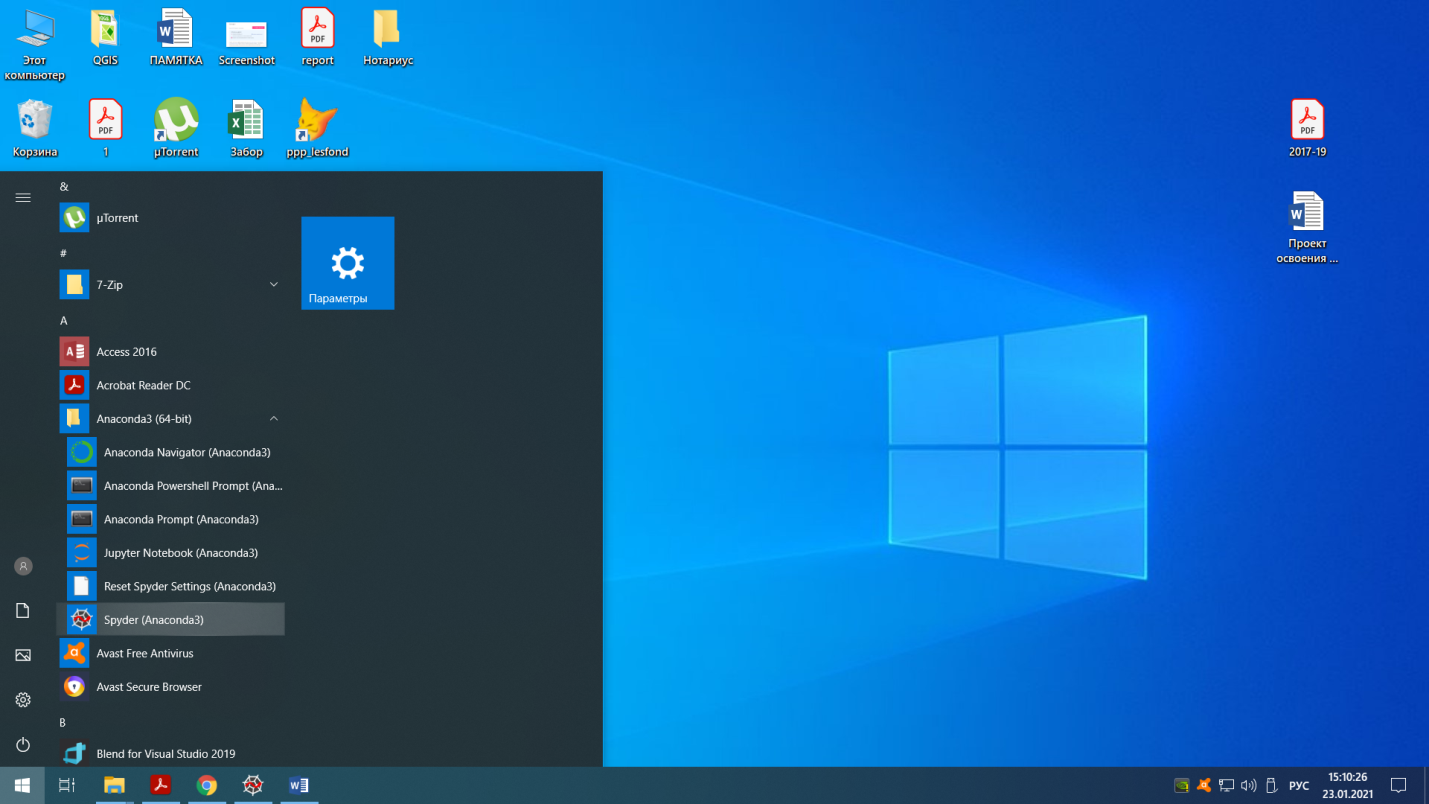


Рис.1 Запуск Spyder

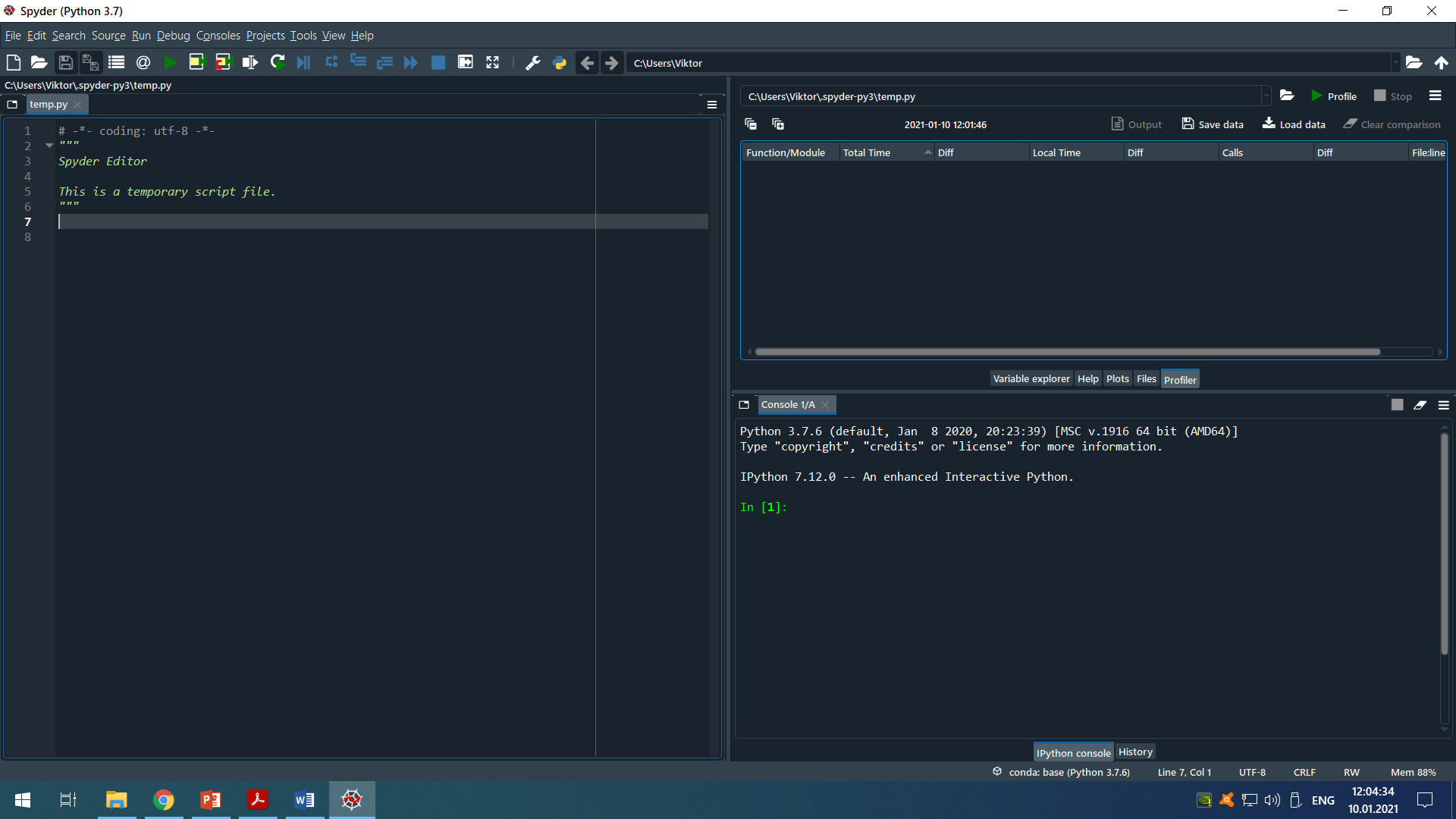


Рис.2 Рабочая область.

Левое окно – окно кода, правое – консоль.

* Создать новый файл (File – New file)

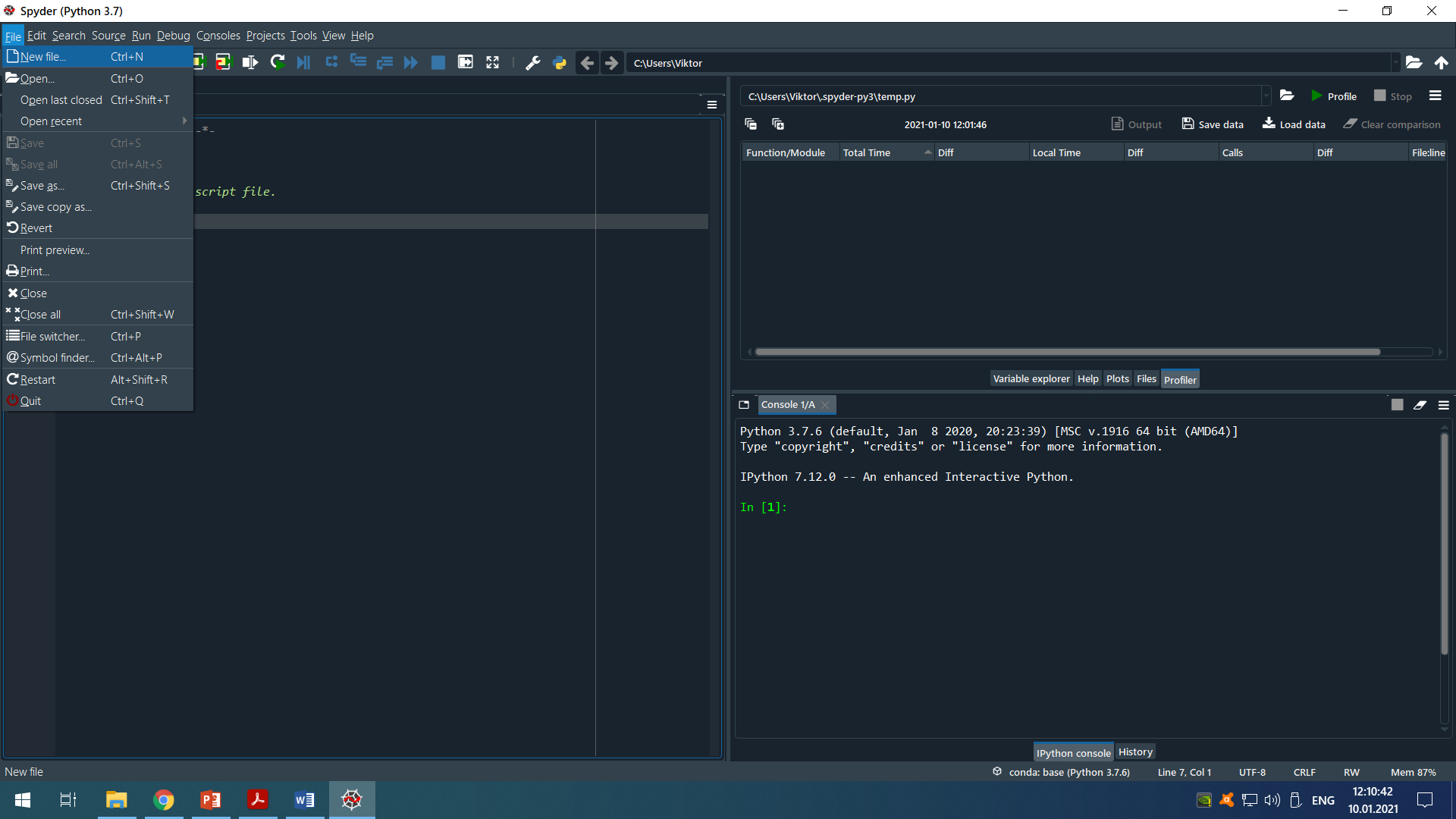


Рис.3 Создание нового файла.

Сохраняем файл под именем Lab\_1

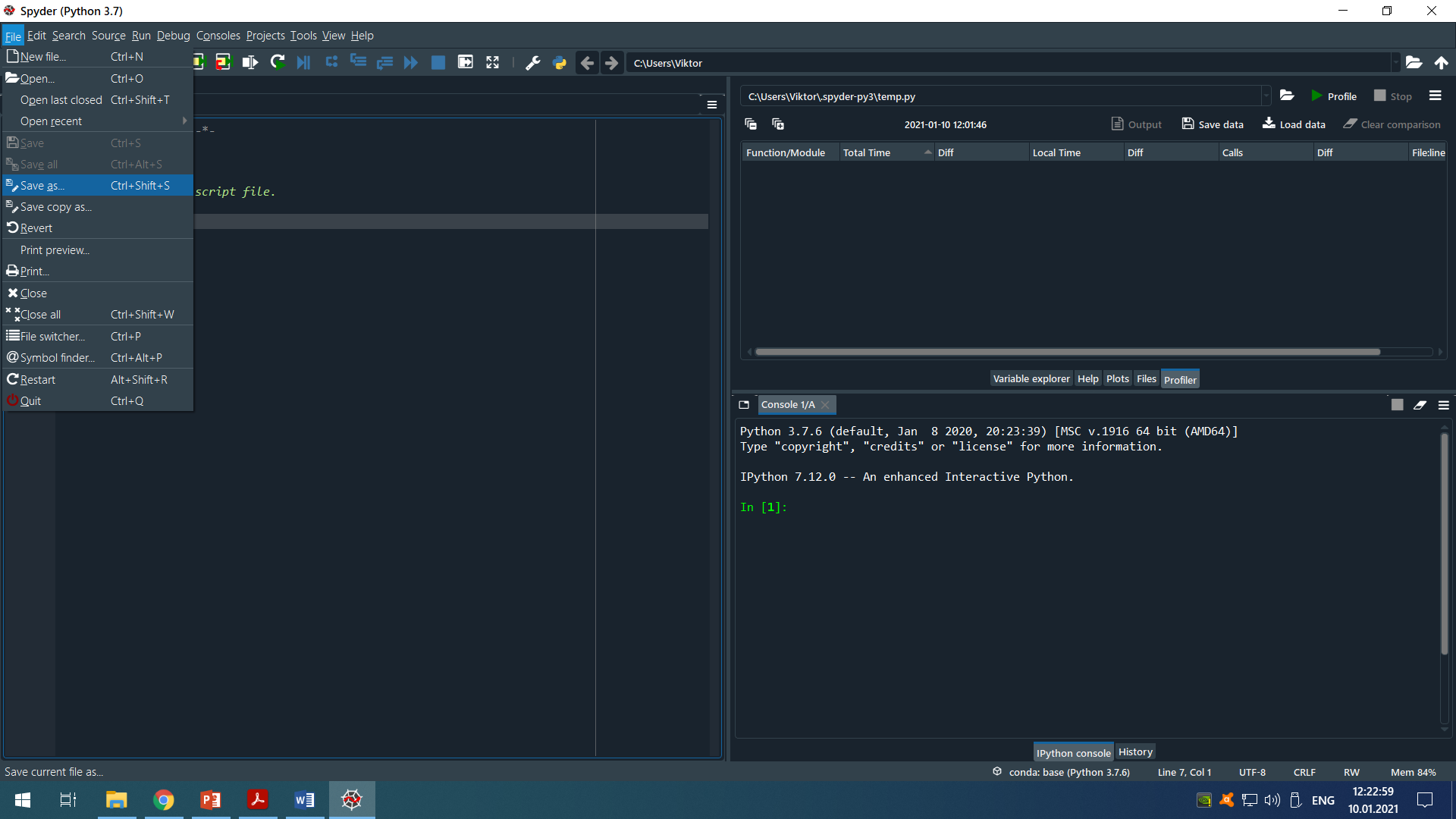
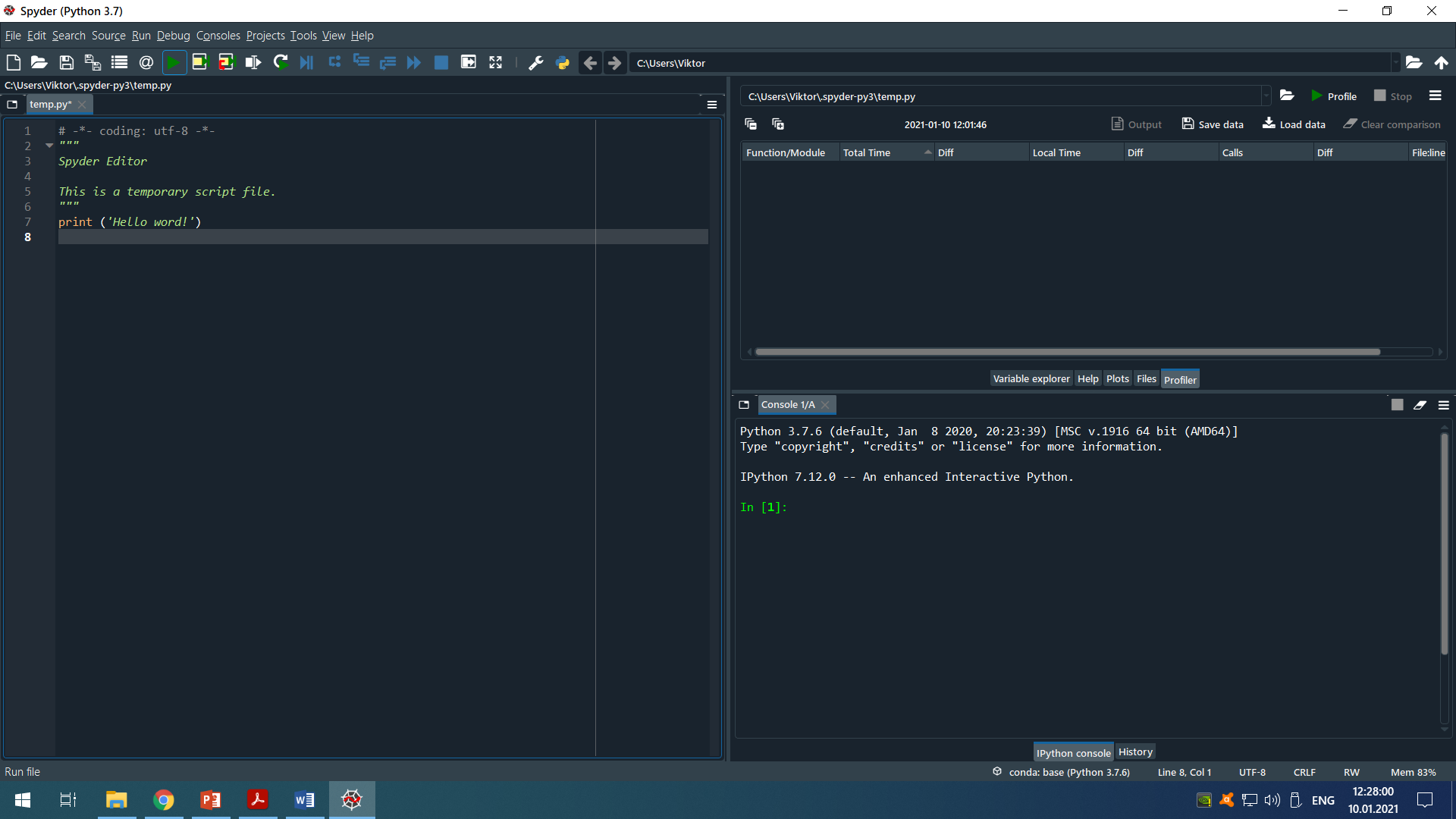


Рис. 4 Сохранения файла.

В окне кода пишем первую программу:

**print** (‘Hello word!’)



**Рис. 5** Первый код на языке **Python**.

И нажимаем F5.

**Типы данных**

В **Python** встроенные типы данных подразделяются на 2 группы:

1. Скалярные (неделимые).
   1. Числа (целое, вещественное).
   2. Логический тип.
   3. NoneType.
2. Структурированные (составные) / коллекции.
   1. Последовательности: строка, список, кортеж.
   2. Множества.
   3. Отображения: словарь.

**Тип данных** (англ. Data type) - характеристика, определяющая множество значений и набор операций на этих значениях:

• множество допустимых значений, которые могут принимать данные, принадлежащие к этому типу (например, объект типа Целое число может принимать только целочисленные значение в определенном диапазоне);

• набор операций, которые можно осуществлять над данными, принадлежащими к этому типу (например, объекты типа Целое число умеют складываться, умножаться и т.д.).

Все типы в **Python** являются объектами (в отличие, например, от C++).

**Переменные**

[Переменная](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F) (*англ.* Variable) - это идентификатор, который указывает на определенную область памяти, где хранятся произвольные данные - созданный объект (*значение* переменной).

Она работает как подписанная коробка или помеченная ячейка, куда можно что-то положить и не потерять.

Когда вы первый раз пишете в коде имя переменной, это называется **объявление переменной**. В **Python** переменную всегда объявляют, присваивая ей какое-нибудь значение. Достаточно просто ввести имя, поставить знак равенства «=» (называется **оператор присваивания**) и написать значение, которое будет храниться в переменной.

**message** = 'Привет, Мир!'

**pages** = 210

**today** = 42

**total** = pages + today

print(total)

# Будет напечатано: 252

**Ввод и вывод данных**

Ввод данных осуществляется при помощи команды **input(**список ввода**)**:

a = **input**()

**print**(a)

В скобках функции можно указать сообщение - комментарий к вводимым данным:

a = **input** ("Введите количество: ")

Команда **input**() по умолчанию воспринимает входные данные как строку символов. Поэтому, чтобы ввести целочисленное значение, следует указать тип данных int():

a = int (**input**())

Для ввода вещественных чисел применяется команда

a=float(**input**())

Вывод данных осуществляется при помощи команды **print** (список вывода):

a = 1

b = 2

**print** (a)

**print** (a + b)

**print** ('сумма = ', a + b)

Существует возможность записи команд в одну строку, разделяя их через**;** Однако не следует часто использовать такой способ, это снижает удобочитаемость:

a = 1; b = 2; **print**(a)

**print** (a + b)

**print** ('сумма = ', a + b)

Для команды **print**может задаваться так называемый сепаратор — разделитель между элементами вывода:

x=2

y=5

**print** ( x, "+", y, "=", x+y, sep = " " )

Результат отобразится с пробелами между элементами: 2 + 5 = 7

Напишите программу, которая запрашивала бы у пользователя:

## Варианты заданий

В данной лабораторной работе варианты заданий выдаются преподавателем.

## Порядок выполнения лабораторной работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги.

3. Разработать алгоритм решения задачи. Уточнить последовательность выполнения его пунктов.

4. Изучить примеры, приведенные в литературе, в том числе и в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.

5. Подготовить свой вариант решения и отладить его с помощью компьютера.

## Контрольные вопросы

1. Назовите две группы данных языка Python.
2. Какие типы данных вы знаете?
3. Что такое переменная?

# ЛАБОРАТОРНАЯ(ПРАКТИЧЕСКАЯ) РАБОТА № 2

**Тема:** Основы синтаксиса. Математические операции в **Python**.

**Цель:** освоить основные правила синтаксиса языка **Python,** математические операции.

## Теоретическое введение

**PEP 8 - руководство по написанию кода на Python (для самостоятельного изучения)**

Функция проверки в среде разработки Spyder на соответствие PEP 8- Pylint. Клавиша F8.

**Знаки пунктуации**

В алфавит **Python** входит достаточное количество знаков пунктуации, которые используются для различных целей. Например, знаки "+" или «\*» могут использоваться для сложения и умножения, а знак запятой ","  - для разделения параметров функций.

**Идентификаторы**

Идентификаторы в **Python** это имена, используемые для обозначения переменной, функции, класса, модуля или другого объекта.

**Ключевые слова**

Некоторые слова имеют в **Python** специальное назначение и представляют собой управляющие конструкции языка.

Ключевые слова в **Python**:

[**'False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'**]

**Отступы**

Используйте 4 пробела на каждый уровень отступа.

**Табуляция или пробелы?**

Пробелы - самый предпочтительный метод отступов.

**Кодировка исходного файла**

Кодировка **Python** должна быть UTF-8.

В Python нет символа, который бы отвечал за отделение выражений друг от друга в исходном коде, как, например, точка с запятой (;) в C++ или Java. Также отсутствует такая конструкция, как фигурные скобки {}, позволяющая объединить группу инструкций в единый блок. Физические строки разделяются самим символом конца строки, но если выражение слишком длинное для одной строки, то две физических строки можно объединить в одну логическую. Для этого необходимо в конце первой строки ввести символ обратного слэша (\), и тогда следующую строку интерпретатор будет трактовать как продолжение первой.

Для выделения блоков кода используются исключительно отступы. Логические строки с одинаковым размером отступа формируют блок, и заканчивается блок в том случае, когда появляется логическая строка с отступом меньшего размера:

Основная инструкция:

Вложенный блок инструкций

Именно поэтому первая строка в сценарии Python не должна иметь отступа.

Используются стандартные правила для заданий идентификаторов переменных, методов и классов – имя не должно начинаться с подчеркивания или латинского символа любого регистра и не может содержать символов @, $, %. Также не может использоваться в качестве идентификатора только один символ подчеркивания.

Для обозначения комментариев используется символ #. Комментарии и пустые строки интерпретатор игнорирует.

Все имена в Python чувствительны к регистру — имена переменных, функций, классов, модулей, исключений. Всё, что можно прочитать, записать, вызвать, создать или импортировать, чувствительно к регистру.

**18 правил написания кода на Python:**

1. На каждый уровень отступа используйте по **4 пробела**.
2. Длину строки рекомендуется ограничить **79 символами**.
3. Функции верхнего уровня и [определения классов](https://smartiqa.ru/courses/python/lesson-6#class-python) отделяются **двумя** пустыми строками. Определения методов внутри класса разделяются **одной** пустой строкой. Дополнительные пустые строки используются для логического разделения.
4. Для каждого [**импорта**](https://smartiqa.ru/courses/python/lesson-5#import) - отдельная строка. Порядок расположения - [стандартные библиотеки](https://smartiqa.ru/courses/python/lesson-5#standard-library), [сторонние библиотеки](https://smartiqa.ru/courses/python/lesson-5#site-packages), локальные модули приложения. Импорты всегда помещаются в начале файла, сразу после комментариев к модулю и строк документации, и перед объявлением констант.
5. Избегайте лишних пробелов внутри круглых, квадратных или фигурных скобок.
6. Используйте одиночный пробел с каждой стороны у операторы присваивания **=, +=, -=,** операторов сравнения**==, <, >, !=, <>, <=, >=, in, not in, is, is not** и логических операторов **and, or, not**. Не используйте пробелы вокруг знака **=**, если он используется для обозначения именованного аргумента или значения параметров по умолчанию.
7. **Комментарии**, которые противоречат коду, хуже, чем их отсутствие. Всегда исправляйте комментарии при обновлении кода.
8. Никогда не используйте символы **l** (маленькая латинская буква «**L**»), **O** (заглавная латинская буква «**o**») или **I**(заглавная латинская буква «**i**») в качестве имен.
9. [**Модули**](https://smartiqa.ru/courses/python/lesson-5#module) должны иметь короткие имена, состоящие из маленьких букв.
10. [**Имена классов**](https://smartiqa.ru/courses/python/lesson-6#class-python) должны обычно следовать соглашению CapitalizedWords (слова с заглавными буквами).
11. **Имена**[**функций**](https://smartiqa.ru/courses/python/lesson-3#function) должны состоять из маленьких букв, а слова разделяться символами подчеркивания (lower\_case\_with\_underscores).
12. Для **имен**[**методов**](https://smartiqa.ru/courses/python/lesson-6#method) и [**переменных**](https://smartiqa.ru/courses/python/lesson-6#field) экземпляров классов используйте тот же стиль, что и для имен функций.
13. Всегда используйте **[self](https://smartiqa.ru/courses/python/lesson-6" \l "self" \t "_blank)** в качестве первого аргумента метода экземпляра объекта. Всегда используйте **[cls](https://smartiqa.ru/courses/python/lesson-6" \l "cls" \t "_blank)** в качестве первого аргумента метода класса.
14. Сравнения с **None** должны обязательно выполняться с использованием операторов **is** или **is not**, а не с помощью операторов сравнения.
15. Наследуйте свой класс исключения от **Exception**. Перехватывайте конкретные ошибки вместо простого выражения **except**.
16. Всегда используйте выражение **def**, а не присваивание лямбда-выражения к имени.
17. Постарайтесь заключать в каждую конструкцию **try...except** минимум кода, чтобы легче отлавливать ошибки.
18. Для [**последовательностей**](https://smartiqa.ru/courses/python/lesson-3#sequence)([строк](https://smartiqa.ru/courses/python/lesson-4), [списков](https://smartiqa.ru/courses/python/lesson-2#list), [кортежей](https://smartiqa.ru/courses/python/lesson-2#tuple)) используйте тот факт, что пустая последовательность есть **false**.

**Создать переменные типа и присвоить значения:**

english = 554.2

russian = 863.9

german = 25

chinese = ‘456’

top3\_total = english+russian+german

**print** (top3\_total)

**Преобразовать:**

int -> float;

float – int;

str -> int;

int -> str.

**Узнать тип переменной**

russian\_web\_part = 0.061

**print** (**type**(russian\_web\_part))

**Получить результат и прокомментировать код**

**print** (0.017 \* 10000000)

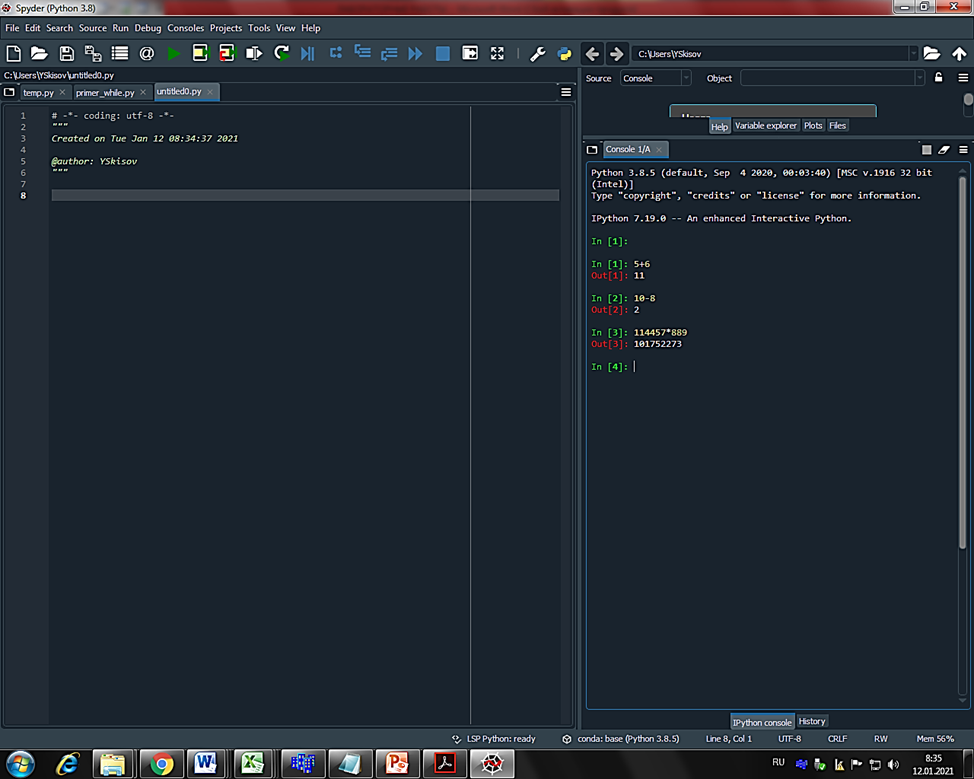


Рис. 6 Первые математические вычисления в **Python**

|  |  |
| --- | --- |
| **x + y** | Сложение |
| **x - y** | Вычитание |
| **x \* y** | Умножение |
| **x / y** | Деление |
| **x // y** | Получение целой части от деления |
| **x % y** | Остаток от деления |
| **-x** | Смена знака числа |
| **abs(x)** | Модуль числа |
| **divmod(x, y)** | Пара (x // y, x % y) |
| **x \*\* y** | Возведение в степень |

1. Арифметические операции

# Тип результата операции определяется типом аргументов

# -х

# Меняет знак х на обратный

**>>>** x = 5

**>>>** -x

-5

# х + у, х - у, х \* у

# Сложение, разность, произведение чисел х и у

**>>>** 5 + 3

8

**>>>** 5 - 7

-2

**>>>** 5 \* 3

15

х / у

# Делит х на у (результат типа float)

**>>>** 5 / 3

1.6666666666666667

x // y

# Делит х на у нацело - усекает дробную часть (результат типа int)

**>>>** 5 // 3

1

х % у

# Возвращает модуль (остаток) от деления х на у

**>>>** 5 % 3

2

# x\*\*y

# Возводит х в степень у

**>>>** 5\*\*3

125

# abs(x)

# Возвращает абсолютное значение х

**>>>** abs(-5)

5

Для форматированного вывода используется **format**:

Строковый метод format() возвращает отформатированную версию строки, заменяя идентификаторы в фигурных скобках {}. Идентификаторы могут быть позиционными, числовыми индексами, ключами словарей, именами переменных.

Синтаксис команды **format**:

поле замены    -   =  "{" [имя поля] ["!" преобразование] [":" спецификация] "}"

имя поля          -     :=  arg\_name ("." имя атрибута | "[" индекс "]")\*

преобразование - :=  "r" (внутреннее представление) | "s" (человеческое представление)

спецификация -   :=  см. ниже

Аргументов в format() может быть больше, чем идентификаторов в строке. В таком случае оставшиеся игнорируются.

Спецификация формата:

|  |  |
| --- | --- |
| спецификация | :=  [[fill]align][sign][#][0][width][,][.precision][type] |
| заполнитель | :=  символ кроме '{' или '}' |
| выравнивание | :=  "<" | ">" | "=" | "^" |
| знак | :=  "+" | "-" | " " |
| ширина | :=  integer |
| точность | :=  integer |
| тип | :=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%" |

|  |  |
| --- | --- |
| **Тип** | **Значение** |
| 'd', 'i', 'u' | Десятичное число. |
| 'o' | Число в восьмеричной системе счисления. |
| 'x' | Число в шестнадцатеричной системе счисления (буквы в нижнем регистре). |
| 'X' | Число в шестнадцатеричной системе счисления (буквы в верхнем регистре). |
| 'e' | Число с плавающей точкой с экспонентой (экспонента в нижнем регистре). |
| 'E' | Число с плавающей точкой с экспонентой (экспонента в верхнем регистре). |
| 'f', 'F' | Число с плавающей точкой (обычный формат). |
| 'g' | Число с плавающей точкой. с экспонентой (экспонента в нижнем регистре), если она меньше, чем -4 или точности, иначе обычный формат. |
| 'G' | Число с плавающей точкой. с экспонентой (экспонента в верхнем регистре), если она меньше, чем -4 или точности, иначе обычный формат. |
| 'c' | Символ (строка из одного символа или число - код символа). |
| 's' | Строка. |
| '%' | Число умножается на 100, отображается число с плавающей точкой, а за ним знак %. |

Вычислить в консоли следующие выражения:

Вариант 1. a + (b\*(c\*\*d)) / f

Вариант 2. a + b\*(c\*\*d) / f

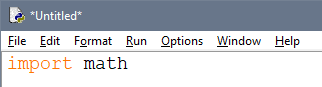
Вариант 3. (a + b\*(c\*\*d)) / f

Вариант 4. (a + b\*(c\*d)) / f

**Библиотека (модуль) math**

В стандартную поставку **Python** входит библиотека math, в которой содержится большое количество часто используемых математических функций.

Для работы с данным модулем его предварительно нужно импортировать.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/4.png?attredirects=0)

**Рассмотрим наиболее часто используемые функции модуля math**

|  |  |
| --- | --- |
| **math.ceil(x)** | Возвращает ближайшее целое число большее, чем x |
| **math.fabs(x)** | Возвращает абсолютное значение числа x |
| **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 |

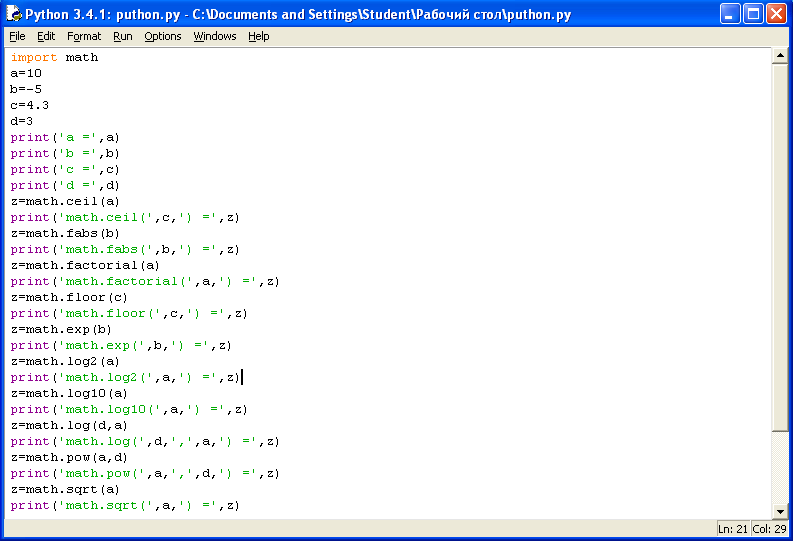
**Пример применения вышеописанных функций над числами**

В программе определены 4 переменные - a, b, c, d, каждая из которых является либо целым числом, либо вещественным, либо отрицательным.

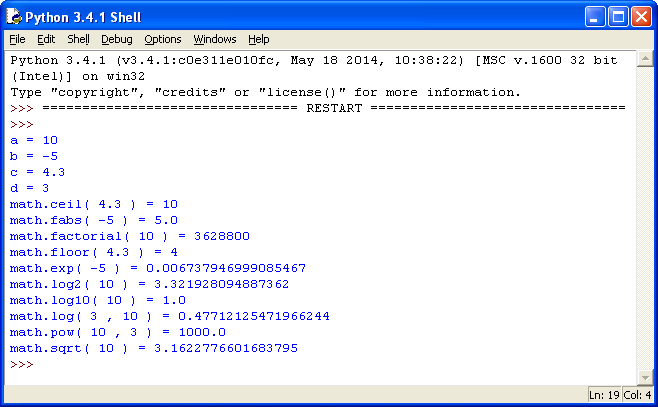
Командой **print**() выводится значение каждой переменной на экран при выполнении программы.

В переменную z помещается результат выполнения функции модуля math.

Затем командой **print**() выводится сообщение в виде используемой функции и её аргумента и результат её выполнения.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/math.png?attredirects=0)

Пример программы на **Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/math_result.png?attredirects=0)

Результат выполнения программы с применением функций модуля **math**

**Тригонометрические функции модуля math**

|  |  |
| --- | --- |
| math.cos(x) | Возвращает cos числа X |
| math.sin(x) | Возвращает sin числа X |
| math.tan(x) | Возвращает tan числа X |
| math.acos(x) | Возвращает acos числа X |
| math.asin(x) | Возвращает asin числа X |
| math.atan(x) | Возвращает atan числа X |

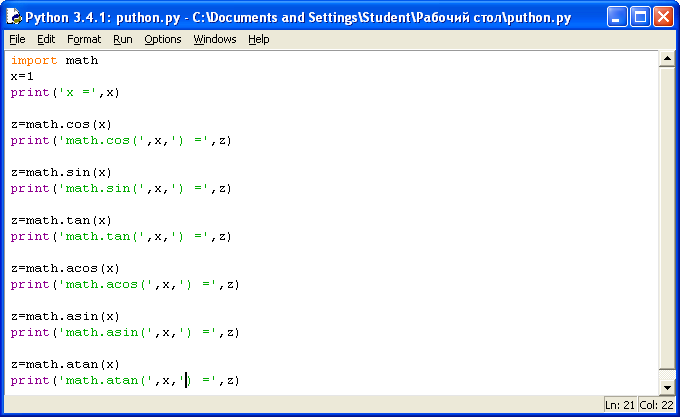
**Пример применения вышеописанных функций над числами**

В программе определена переменная x, содержащая целое число.

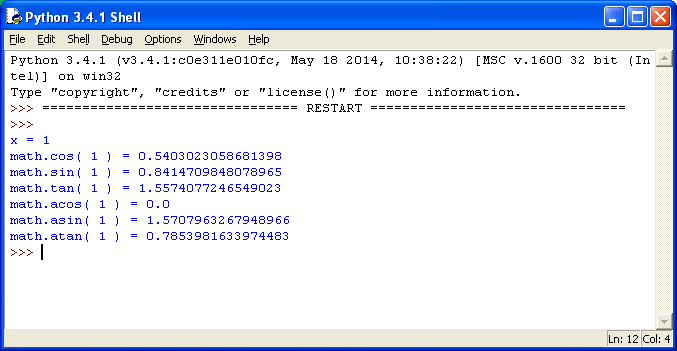
Значение переменной выводится командой **print**() на экран.

В переменную z помещается результат выполнения тригонометрической функции модуля math.

Затем командой **print** () выводится сообщение в виде используемой функции и её аргумента и результат её выполнения.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/sincos.png?attredirects=0)

Пример программы с использованием тригонометрических функций модуля math

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/sincos_result.png?attredirects=0)

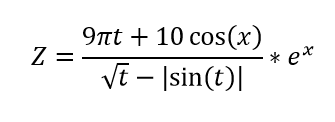
Результат выполнения программы с применением  тригонометрических функций модуля **math**

**Константы:**

* **math.pi**-число Pi.
* **math.e**-число е (экспонента).

**Пример**

Напишите программу, которая бы вычисляла заданное арифметическое выражение при заданных переменных. Ввод переменных осуществляется с клавиатуры. Вывести результат с 2-мя знаками после запятой.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/lab2_example.png?attredirects=0)

x=10; t=1

**Решение**

Сначала импортируем модуль **math.** Для этого воспользуемся командой

**import** **math**

Затем следует ввести значения двух переменных целого типа x и t.   
Для ввода данных используется команда **input**, но так как в условии даны целые числа, то нужно сначала определить тип переменных: **x=int(), t=int().**  
Определив тип переменных, следует их ввести, для этого в скобках команды **int()** нужно написать команду **input().**

Для переменной x это выглядит так: **x=int(input("сообщение при вводе значения")).**  
Для переменной t аналогично: **t=int(input("сообщение при вводе значения")).**  
Следующий шаг - это составление арифметического выражения, результат которого поместим в переменную z.  
Сначала составим числитель.

Выглядеть он будет так:

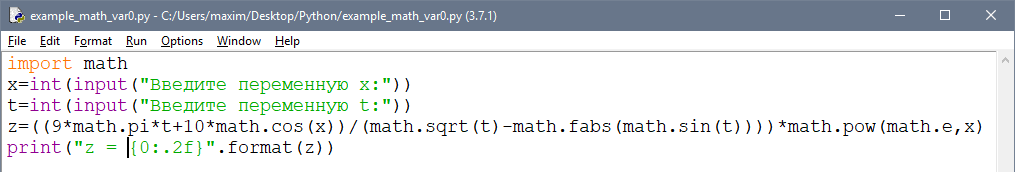
**9\*math.pi\*t+10\*math.cos(x)**

Затем нужно составить знаменатель, при этом обратим внимание на то, что числитель делится на знаменатель, поэтому и числитель, и знаменатель нужно поместить в скобки (), а между ними написать знак деления /.

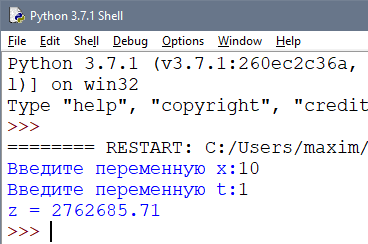
Выглядеть это будет так:

**(9\*math.pi\*t+10\*math.cos(x))/(math.sqrt(t)-math.fabs(math.sin(t)))**  
Последним шагом является умножение дроби на экспоненту в степени x.  
Так как умножается вся дробь, то следует составленное выражение поместить в скобки (), а уже потом написать функцию **math.pow(math.e,x).**  
В результате выражение будет иметь вид:  
**z=((9\*math.pi\*t+10\*math.cos(x))/(math.sqrt(t)-math.fabs(math.sin(t))))\*math.pow(math.e,x).**

При составлении данного выражения следует обратить внимание на количество открывающихся и закрывающихся скобок.  
Командой **print()** выведем значение переменной, отформатировав его командой **format.**  
Сам формат записывается в апострофах в фигурных скобках **{}**.  
В задаче требуется вывести число с двумя знаками после запятой, значит вид формата будет выглядеть следующим образом: {0:.2f}, где 2 - это количество знаков после запятой, а f указывает на то, что форматируется вещественное число. При этом перед 2 нужно поставить точку, указав тем самым на то, что форматируем именно дробную часть числа.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/lab2_example_prog.png?attredirects=0)

**Результат**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-2/lab2_example_res.png?attredirects=0)

## Варианты заданий

В данной лабораторной работе варианты заданий выдаются преподавателем.

## Порядок выполнения лабораторной работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги.

3. Разработать алгоритм решения задачи. Уточнить последовательность выполнения его пунктов.

4. Изучить примеры, приведенные в литературе, в том числе и в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.

5. Подготовить свой вариант решения и отладить его с помощью компьютера.

## Контрольные вопросы

1. Какая операция с целыми числами никогда не дает в виде результата целое число?
2. Отличаются ли чем-то варианты объявления целого числа: 1230 или int(1230)?
3. Какой результат дает применение операции int(a) к любому числу «а» с плавающей точкой?
4. Каким методом можно проверить, является ли заданное вещественное число целым?
5. Какой тип данных получится в результате вычисления выражения: 3 + 3.0 + 4?

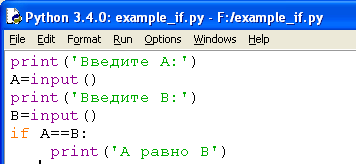
# ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА №3

**Тема:** условная инструкция if-elif-else.

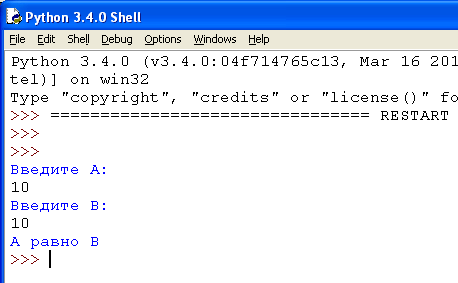
**Цель работы:**познакомиться со структурой ветвление **(if, if-else, if-elif-else),** научиться работать с числами и строками используя данную структуру.

## Теоретическое введение

Программа запрашивает у пользователя два числа, затем сравнивает их и если числа равны, то есть логическое выражение A==B истинно, то выводится соответствующее сообщение.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if.png?attredirects=0)

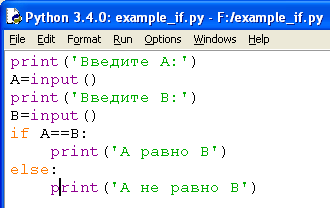
Пример программы на **Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if_res.png?attredirects=0)

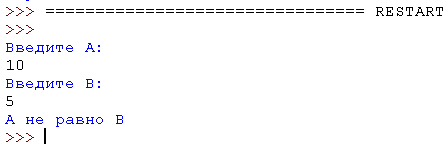
Результат выполнения программы с использованием условного оператора if

**2. Конструкция if – else**  
Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т.е. при истинном условии нужно выполнить один набор инструкций, при ложном – другой. Для этого используется конструкция if – else.  
  
Синтаксис оператора if – else выглядит так:  
**if** **логическое выражение:**  
    команда\_1  
    команда\_2  
    ...  
    команда\_n  
**else:**  
    команда\_1      
    команда\_2  
    ...  
    команда\_n

Программа запрашивает у пользователя два числа, затем сравнивает их и если числа равны, то есть логическое выражение A==B истинно, то выводится соответствующее сообщение. В противном случае выводится сообщение, что числа не равны.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if-else.png?attredirects=0)

Пример программы на **Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if-else_res.png?attredirects=0)

Результат выполнения программы с использованием условного оператора if-else

**3. Конструкция if – elif – else**  
    Для реализации выбора из нескольких альтернатив можно использовать конструкцию if – elif – else.  
Синтаксис оператора if – elif – else выглядит так:

**if  логическое выражение\_1:**    команда\_1

    команда\_2

    ...

    команда\_n  
**elif логическое выражение\_2:**    команда\_1

    команда\_2

    ...

    команда\_n  
**elif** **логическое выражение\_3:**

    команда\_1

    команда\_2

    ...

    команда\_n

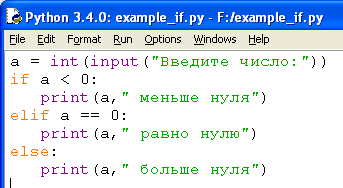
**else:**  
    команда\_1

    команда\_2

    ...

    команда\_n

Программа запрашивает число у пользователя и сравнивает его с нулём a<0. Если оно меньше нуля, то выводится сообщение об этом. Если первое логическое выражение не истинно, то программа переходит ко второму - a==0. Если оно истинно, то программа выведет сообщение, что число равно нулю, в противном случае, если оба вышеуказанных логических выражения оказались ложными, то программа выведет сообщение, что введённое число больше нуля.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if-elif-else.png?attredirects=0)

Пример программы на **Python**

**Условный оператор ветвления if, if-else, if-elif-else**

Оператор ветвления **if** позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Возможны следующие варианты использования.

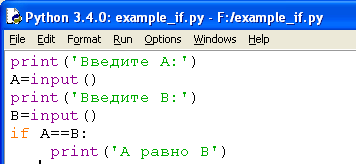
**1. Конструкция if**  
Синтаксис оператора if выглядит так:  
**if логическое выражение:**  
    команда\_1  
    команда\_2  
    ...  
    команда\_n  
  
После оператора if  записывается логическое выражение.

**Логическое выражение** — конструкция [языка программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), результатом вычисления которой является «истина» или «ложь».

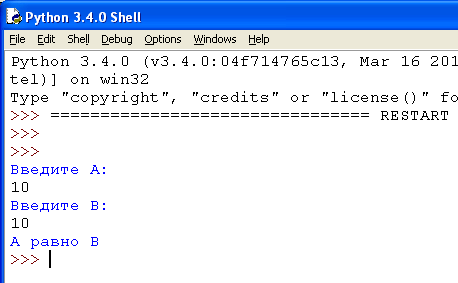
Если это выражение истинно, то выполняются инструкции, определяемые данным оператором. Выражение является истинным, если его результатом является число не равное нулю, непустой объект, либо логическое True. После выражения нужно поставить двоеточие “:”.

**ВАЖНО:** блок кода, который необходимо выполнить, в случае истинности выражения, отделяется **четырьмя**пробелами слева!

Программа запрашивает у пользователя два числа, затем сравнивает их и если числа равны, то есть логическое выражение A==B истинно, то выводится соответствующее сообщение.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if.png?attredirects=0)

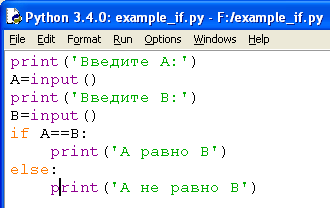
Пример программы на **Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if_res.png?attredirects=0)

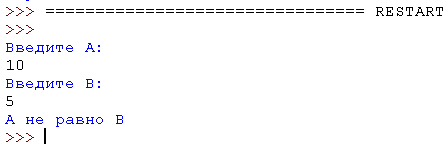
Результат выполнения программы с использованием условного оператора if

**2. Конструкция if – else**  
Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т.е. при истинном условии нужно выполнить один набор инструкций, при ложном – другой. Для этого используется конструкция if – else.  
  
Синтаксис оператора if – else выглядит так:  
**if** **логическое выражение:**  
    команда\_1  
    команда\_2  
    ...  
    команда\_n  
**else:**  
    команда\_1      
    команда\_2  
    ...  
    команда\_n

Программа запрашивает у пользователя два числа, затем сравнивает их и если числа равны, то есть логическое выражение A==B истинно, то выводится соответствующее сообщение. В противном случае выводится сообщение, что числа не равны.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if-else.png?attredirects=0)

Пример программы на **Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if-else_res.png?attredirects=0)

Результат выполнения программы с использованием условного оператора if-else

**3. Конструкция if – elif – else**  
    Для реализации выбора из нескольких альтернатив можно использовать конструкцию if – elif – else.  
Синтаксис оператора if – elif – else выглядит так:

**if  логическое выражение\_1:**    команда\_1

    команда\_2

    ...

    команда\_n  
**elif логическое выражение\_2:**    команда\_1

    команда\_2

    ...

    команда\_n  
**elif** **логическое выражение\_3:**

    команда\_1

    команда\_2

    ...

    команда\_n

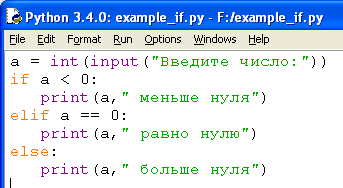
**else:**  
    команда\_1

    команда\_2

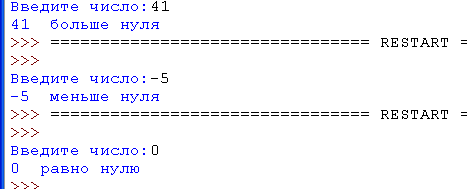
    ...

    команда\_n

Программа запрашивает число у пользователя и сравнивает его с нулём a<0. Если оно меньше нуля, то выводится сообщение об этом. Если первое логическое выражение не истинно, то программа переходит ко второму - a==0. Если оно истинно, то программа выведет сообщение, что число равно нулю, в противном случае, если оба вышеуказанных логических выражения оказались ложными, то программа выведет сообщение, что введённое число больше нуля.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if-elif-else.png?attredirects=0)

Пример программы на **Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/if-elif-else_res.png?attredirects=0)

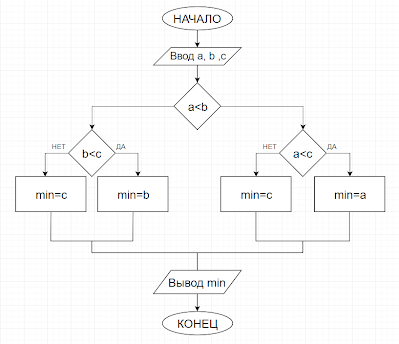
Результат выполнения программы с использованием условного оператора **if-elif-else**

**Пример**

Дано 3 числа. Найти минимальное среди них и вывести на экран.

**Решение**

Для простоты построим блок-схему задачи.

**[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/block.png?attredirects=0)**

Командами

a=**input**('')   
b=**input**('')  
c=**input**('')

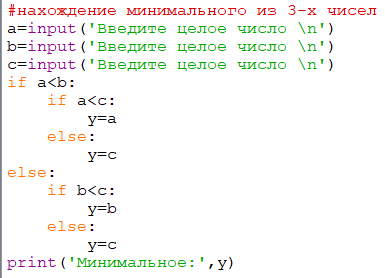
введём три числа, присвоив значения переменным a, b, c.

Условной конструкцией if-else проверим на истинность логическое выражение a <b. Если оно истинно, то переходим на проверку логического выражения a <c. Если оно истинно, то переменной "y" присвоим значение переменной "а", т.е. "а" будет минимальным, а иначе "y" присвоится значение переменной "с".

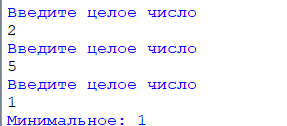
Если в начале логическое выражение a <b оказалось ложным, то переходим на проверку другого логического выражения b <c.

Если оно истинно, то "у" присвоится значение переменной "b", иначе "c".

Командой **print()** выводим минимальное значение.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/ex_ifelse_prog_var0.png?attredirects=0)

Пример программы

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-4/ex_ifelse_prog_var0_res.png?attredirects=0)

Результат выполнения программы

**Пример**

Напишите программу расчета индекса массы тела человека по его росту, весу и интерпретировать результаты в соответствии с рекомендациями ВОЗ.

|  |  |
| --- | --- |
| Индекс массы тела | Описание |
| Меньше 18,5 | Недостаточная масса тела |
| 18,5-24,99 | Нормальная масса тела |
| 25-29,99 | Избыточная масса теал |
| >=30 | Ожирение |

Методические рекомендации по решению задачи.

Постановка задачи:

* Входные данные;
* Выходные данные;
* Формулы и алгоритмы;
* Функциональные возможности

Входные:

* Вес, кг
* Рост, см
* Возраст, лет (полных лет)
* Имя

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Пример | Недопустимые значения | Тип |
| Имя пользователя | Андрей, Иван, Екатерина | - | str |
| Возраст | 23, 20, 35 | < 10 | Int |
| Рост | 175, 181,165 | 0 <рост <300 | float |
| Вес | 65, 78, 54 | 0 <вес <500 | float |

**Выходные данные:**

|  |  |
| --- | --- |
| **Описание** | **Пример** |
| Индекс массы тела | 18.45, 24.12, 27.1 |
| Заключение | Недостаточная масса тела, Нормальная масса тела, Избыточная масса тела, Ожирение |

**Описание переменных:**

|  |  |  |
| --- | --- | --- |
| **Описание** | **Имя** | **Тип** |
| Имя пользователя | **name** | **string** |
| Возраст (количество полных лет) | **age** | **int** |
| Рост (м) | **height** | **float** |
| Вес (кг) | **weight** | **float** |
| Индекс массы тела | **bmi** | **float** |
| Заключение | **description** | **string** |

**Используемые формулы:**

Индекс массы тела рассчитывается по формуле:

где: *bmi–*индекс массы тела в кг/м²,  
*m* *–* масса тела в килограммах,  
*h* *–* рост в метрах.

**Функциональные возможности:**

Создаваемая программа должна:

* спросить у пользователя его имя;
* спросить у пользователя, сколько полных лет ему исполнилось;
* вывести приветствие на экран;
* спросить у пользователя его рост и вес;
* вычислить индекс массы тела;
* вывести индекс массы тела на экран;
* вывести заключение по посчитанному индексу.

**Реализация задачи:**

1. Пишем код программы, которая спрашивает у пользователя его имя.

**print** ("Ваше имя?")

name = **input**()

**print** ("Привет, ", name, "!")

2. Далее пишем код так, чтобы пользователь, кроме имени, вводил свой возраст, рост и вес.

age = int(**input**("Сколько Вам полных лет? "))

height = float(**input**("Ваш рост? "))

weight = float(**input**("Ваш вес? "))

3. Вычислить индекс массы тела и вывести его значение на экран.

bmi = weight / height \*\* 2

**print**("Ваш индекс массы тела: ", bmi)

4. Округлить результат до 2-х знаков после запятой (количество знаков может быть любым, в таблице для интерпретации результатов нашей задачи используется вещественные числа с двумя знаками после запятой, значит более точные результаты не нужны).

bmi = weight / height \*\* 2

**print**("Ваш индекс массы тела: ",round(bmi, 2))

5. Объяснить пользователю результат вычисления в соответствии с таблицей Всемирной Организации Здравоохранения. При этом на экран предполагается вывести сообщение типа «Вы относитесь к группе с нормальной массой тела», поэтому в переменную **description** словосочетание занесено в творительном падеже.

if bmi < 18.5:

description = "недостаточной массой тела."

elif bmi < 25:

description = "нормальной массой тела."

elif bmi < 30:

description = "избыточной массой тела."

else:

description = "ожирением."

**Код полной программы:**

**print**("Ваше имя?")

name = **input**()

**print**("Привет, ", name, "!")

age = int(**input**("Сколько Вам полных лет? "))

height = float(**input**("Ваш рост? "))

weight = float(**input**("Ваш вес? "))

if age < 10 or height <= 0 or height > 3 or weight <= 0 or weight > 500:

**print**("Ошибочные входные данные")

else:

bmi = weight / height \*\* 2

bmi = round(bmi, 2)

**print**("Ваш индекс массы тела: ", str(bmi))

if bmi < 18.5:

description = "недостаточной массой тела."

elif bmi < 25:

description = "нормальной массой тела."

elif bmi < 30:

description = "избыточной массой тела."

else:

description = "ожирением."

**print**("Вы относитесь к группе людей с", description)

## Варианты заданий

В данной лабораторной работе варианты заданий выдаются преподавателем.

## Порядок выполнения лабораторной работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги.

3. Разработать алгоритм решения задачи, взяв за основу пример из теоретической части.

4. Изучить примеры, приведенные в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.

5. Подготовить свой вариант решения и отладить его с помощью компьютера.

## Контрольные вопросы

1. Приведите пример минимально возможной конструкции условного выражения в Python.
2. Назовите все возможные типовые варианты написания условных инструкций.
3. В чем разница между двумя нижеследующими конструкциями? Приведите примеры, когда правильнее использовать первый вариант инструкции (данные замените на свои), а когда второй.

Первая конструкция:

if x % 5 == 0:

print('Х делится на 5')

if x % 2 == 0:

print('Х – четное число')

Вторая конструкция:

if x % 2 == 0:

print('Х – четное число')

elif x % 5 == 0:

print('Х делится на 5')

# ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА № 4

**Тема:** работа с циклами.

**Цель работы:**познакомиться с циклическими конструкциями

## Теоретическое введение

В **Python** существуют два типа цикличных выражений:

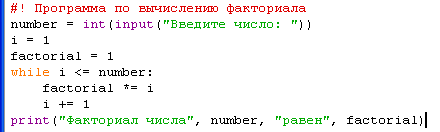
* Цикл while
* Цикл for
  1. **Цикл while в Python**

    Инструкция while в **Python** повторяет указанный блок кода до тех пор, пока указанное в цикле логическое выражение будет оставаться истинным.  
Синтаксис цикла while:

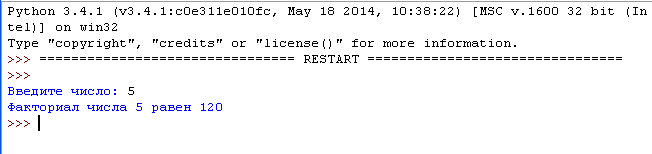
**while логическое выражение**:

команда 1  
команда 2  
...  
команда n

    После ключевого слова **while** указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций, который идет далее.  
    Все инструкции, которые относятся к циклу **while**, располагаются на последующих строках и должны иметь отступ от начала строки (4 пробела).

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-5/while.png?attredirects=0)

**Пример программы на Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-5/while_res.png?attredirects=0)

Результат выполнения программы с использованием циклического оператора **while**

* 1. **Цикл for в Python:**  
         Цикл **for** в **Python** обладает способностью перебирать элементы любого комплексного типа данных (например, строки или списка).   
         Синтаксис цикла **for**:  
     **for** i**in**range():  
         команда 1

    команда 2

    ...

   команда n  
  
Переменной «i» присваивается значение первого элемента функции **range()**, после чего выполняются команды. Затем переменной «i» присваивается следующее по порядку значение и так далее до тех пор, пока не будут перебраны все элементы функции **range().**

Функция **range()** является универсальной функцией **Python** для создания списков (**list**) содержащих арифметическую прогрессию. Чаще всего она используется в циклах **for.**

**range** (старт, стоп, шаг) - так выглядит стандартный вызов функции **range()** в **Python**. По умолчанию старт равняется нулю, шаг единице.

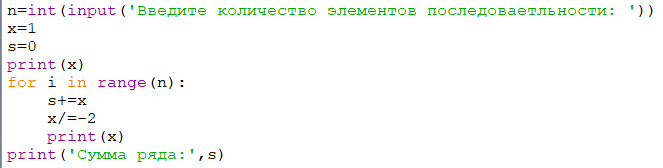
**Пример**

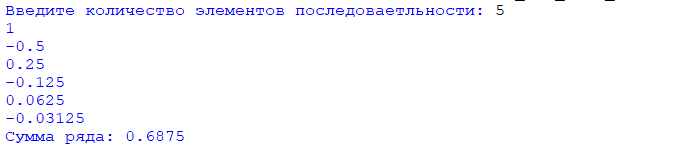
1. Найти сумму n элементов следующего ряда чисел: 1 -0.5 0.25 -0.125 ... n. Количество элементов (n) вводится с клавиатуры. Вывести на экран каждый член ряда и его сумму. Решить задачу используя циклическую конструкцию for.

**Решение:**

В данном случае ряд чисел состоит из элементов, где каждый следующий меньше предыдущего в два раза по модулю и имеет обратный знак. Значит, чтобы получить следующий элемент, надо предыдущий разделить на -2.

Какой-либо переменной надо присвоить значение первого элемента ряда (в данном случае это 1). Далее в цикле добавлять ее значение к переменной, в которой накапливается сумма, после чего присваивать ей значение следующего элемента ряда, разделив текущее значение на -2. Цикл должен выполняться n раз.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-5/ex_for_var0_prog.png?attredirects=0)

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-5/ex_for_var0_res.png?attredirects=0)

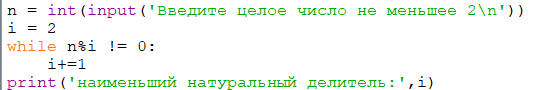
2. Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.

**Решение:**

Для начала введём целое число командой **int**(**input**(текст сообщения)).

Затем зададим переменной i значение 2. Переменная i выполняет роль счётчика. Если задать ей значение 1, то условие задачи не будет выполнено, а результатом всегда будет 1.

В цикле while в качестве логического выражения используется команда n%i сравниваемая с нулём. Таким образом, если остаток от деления введённого числа на текущее значение i не равно нулю, то счётчик увеличивается на 1, а если равно нулю цикл заканчивается и командой **print**() выводится сообщение и значение i.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-5/ex_while_var0_prog.png?attredirects=0)

[https://www.sites.google.com/site/moiboarkin/_/rsrc/1547276081707/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-5/ex_while_var0_res.png](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-5/ex_while_var0_res.png?attredirects=0)

## Варианты заданий

В данной лабораторной работе варианты заданий выдаются преподавателем.

## Порядок выполнения работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги.

3. Разработать алгоритм решения задачи, взяв за основу пример из теоретической части.

4. Изучить примеры, приведенные в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.

5. Подготовить свой вариант решения и отладить его с помощью компьютера.

## Контрольные вопросы

1. В чем основное отличие циклов **for**и **while**?
2. Приведите пример бесконечного цикла. Применяется ли он на практике?

# ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА № 5

**Тема:** работа со строками.

**Цель работы:**познакомится с методами работы со строками.

## Теоретическое введение

Строки в Python - упорядоченные неизменяемые последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

Строки можно создать несколькими способами:

1. С помощью одинарных и двойных кавычек.

Например:

first\_string = 'Я текст в одинарных кавычках'

second\_string = "Я текст в двойных кавычках"

Строки в одинарных и двойных кавычках - одно и то же. Причина наличия двух вариантов в том, чтобы позволить вставлять в строки символы кавычек, не используя экранирование. Например, вот так (обратите внимание на кавычки внутри строки):

first\_string = 'Слово "Python" обычно подразумевает змею'

second\_string = "I'm learning Python"

1. С помощью тройных кавычек.  
   Главное достоинство строк в тройных кавычках в том, что их можно использовать для записи многострочных блоков текста. Внутри такой строки возможно присутствие кавычек и апострофов, главное, чтобы не было трех кавычек подряд. Пример:

**Python**

my\_string = '''Это очень длинная

строка, ей нужно

много места'''

1. С помощью метода str().

Как это работает:

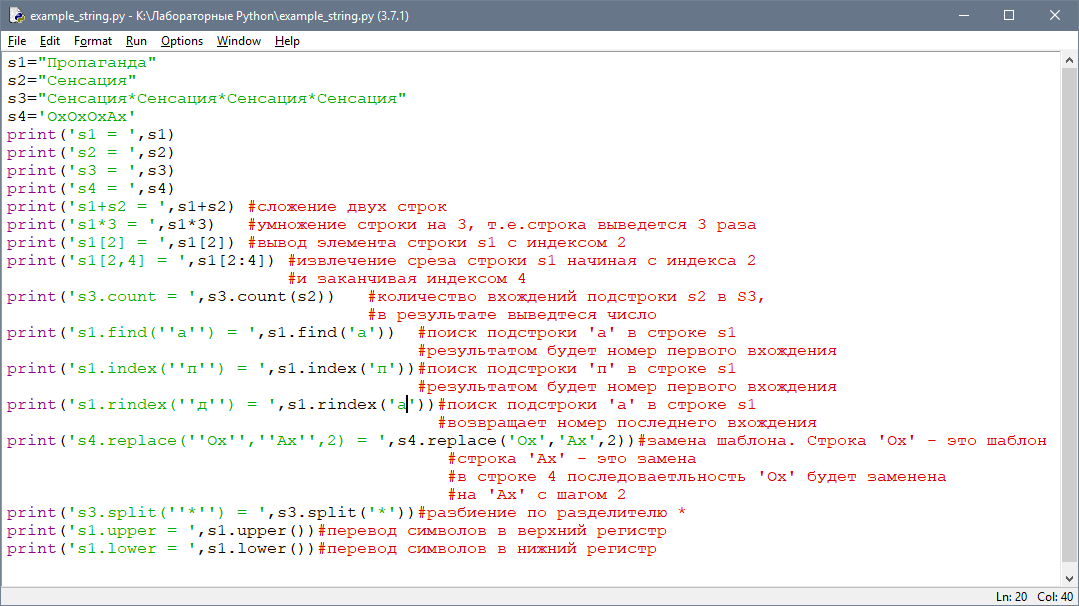
my\_num = 12345

my\_str = str(my\_num)

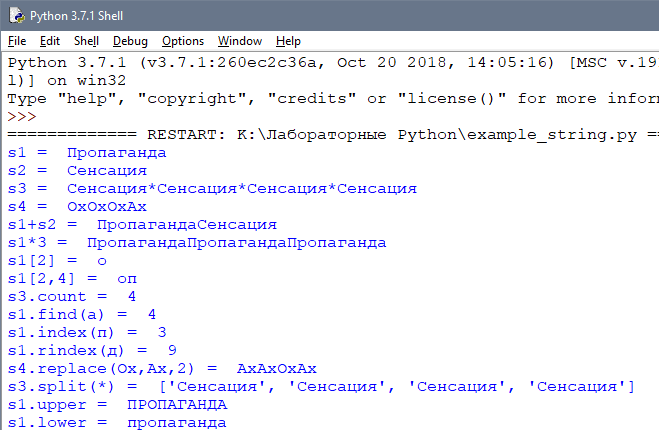
**Функции и методы работы со строками**

|  |  |
| --- | --- |
| **Функция или метод** | **Назначение** |
| **S1 + S2** | Конкатенация (сложение строк) |
| **S1 \* 3** | Повторение строки |
| **S[i]** | Обращение по индексу |
| **S[i:j:step]** | Извлечение среза |
| **len**(S) | Длина строки |
| **S.join**(список) | Соединение строк из последовательности **str** через разделитель, заданный строкой |
| **S1.count**(S[, i, j]) | количество вхождений подстроки s в строку s1. Результатом является число. Можно указать позицию начала поиска i и окончания поиска |
| **S.find**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер первого вхождения или -1 |
| **S.index**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError |
| **S.rindex**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError |
| **S.replace**(шаблон, замена) | Замена шаблона |
| **S.split**(символ) | Разбиение строки по разделителю |
| **S.upper**() | Преобразование строки к верхнему регистру |
| **S.lower**() | Преобразование строки к нижнему регистру |

Ниже приведена программа, демонстрирующая использование функций и методов работы со строками.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-3/ex_string.png?attredirects=0)

Пример программы на **Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-3/ex_string_res.png?attredirects=0)

Результат выполнения программы с использованием функций и методов работы со строками

**Пример**

Проверить, будет ли строка читаться одинаково справа налево и слева направо (т. е. является ли она палиндромом).

**Решение**

Сначала введём строку командой: s=**input**('Введите строку ').

Затем определим логическую переменную flag и присвоим ей значение 1: flag=1.

Для начала в введённой строке нужно удалить пробелы. Для этого воспользуемся циклической конструкцией for, которая выполнится столько раз, какую имеет длину строка. Длину строки определим функцией len(s).

В теле цикла будем проверять следующее условие: s[i] !=' '. Данное логическое выражение будет истинно в том случае, если i-ый элемент строки не будет равен пробелу, тогда выполнится команда, следующая после двоеточия: string+=s[i].

К сроку string, которая была объявлена в начале программы, будет добавляться посимвольно строка s, но уже без пробелов.

Для проверки строки на "палиндром" воспользуемся циклической конструкцией **for.**

Длина половины строки находится делением нацело на 2.  Если количество символов нечетно, то стоящий в середине не учитывается, т.к. его сравниваемая пара - он сам.

Количество повторов цикла равно длине половины строки. Длину строки определим функцией **len(s),** где аргумент введённая нами строка s. Зная длину строки, можно вычислить количество повторов цикла. Для этого целочисленно разделим длину строки на 2: **len(s)**//2.

Для задания диапазона для цикла используем функцию **range()**, в которой аргументом будет являться половина длины строки: range(len(s//2 )).

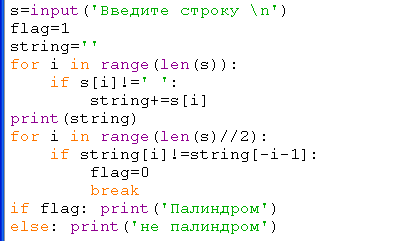
**for** i **in** **range(len(s//2 ))**.

Если символ с индексом i не равен "симметричному" символу с конца строки (который находится путем индексации с конца)

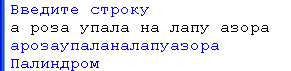
if s[i] != s[-1-i],

то переменной flag присваивается значение 0 и происходит выход из цикла командой **break**.

Далее, при помощи условной конструкции if-else в зависимости от значения flag либо - 0, либо -1 выводится сообщение, что строка палиндром, либо нет.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-3/ex_string_var0_prog.png?attredirects=0)

Пример программы на **Python**

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-3/ex_string_var0_res.png?attredirects=0)

## Варианты заданий

В данной лабораторной работе варианты заданий выдаются преподавателем.

## Порядок выполнения работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги.

3. Разработать алгоритм решения задачи, взяв за основу пример из теоретической части.

4. Изучить примеры, приведенные в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.

5. Подготовить свой вариант решения и отладить его с помощью компьютера.

## Контрольные вопросы

1. Строки в python обозначаются кавычками. Приведите все способы.
2. Как применяют операции сложения и умножения к строкам?
3. Какие типы данных можно преобразовать в строку?

# ЛАБОРАТОРНАЯ(ПРАКТИЧЕСКАЯ) РАБОТА № 6

**Тема:** Работа со списками. Операции над списками в Python.

**Цель работы:** Изучение одномерных массивов в **Python**.

## Теоретическое введение

Массивы (списки) в **Python** — это определенное количество элементов одного типа, которые имеют общее имя, и каждый элемент имеет свой индекс — порядковый номер.

Часто для работы с массивами используются списки.

Список (**list**) – это структура данных для хранения объектов различных типов.

Списки являются упорядоченными последовательностями, которые состоят из различных типов данных, заключающихся в квадратные скобки [ ] и отделяющиеся друг от друга с помощью запятой.

**Создание списков на Python.**

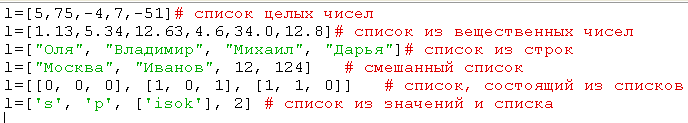
Создать список можно несколькими способами

**1. Получение списка через присваивание конкретных значений.**

Так выглядит в коде **Python** пустой список:

s = []  # Пустой список

Примеры создания списков со значениями:

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list_prog.png?attredirects=0)

Списки можно складывать (конкатенировать) с помощью знака «+»:

[https://www.sites.google.com/site/moiboarkin/_/rsrc/1547172263718/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list_prog1_res.png](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list_prog1_res.png?attredirects=0)

Результат:

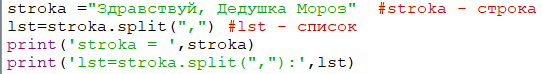
[https://www.sites.google.com/site/moiboarkin/_/rsrc/1547172301822/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list_prog1.png](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list_prog1.png?attredirects=0)

**2. Создание списка при помощи функции split().**

Используя функцию split в **Python** можно получить из строки список.

stroka ="Привет, страна"

lst=stroka.split(",")

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_split_prog.png?attredirects=0)

Результат:

[https://www.sites.google.com/site/moiboarkin/_/rsrc/1547547471226/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_split_res.png](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_split_res.png?attredirects=0)

**3. Генераторы списков.**

В **Python** создать список можно также при помощи генераторов.

**Первый способ.**

Сложение одинаковых списков заменяется умножением:

Список из 10 элементов, заполненный единицами

l = [1]\*10

**Второй способ.**

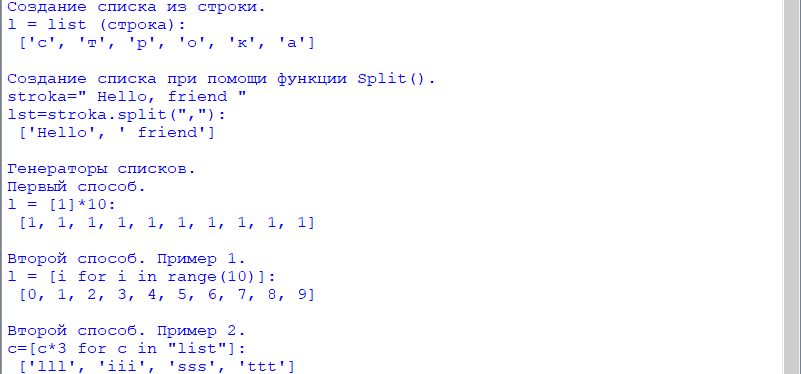
**Пример 1.**

l = [i for i in range(10)]

**Пример 2.**

c = [c \* 3 for c in 'list']

**print** (c) # ['lll', 'iii', 'sss', 'ttt']

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list.png?attredirects=0)

**Примеры использования генераторов списка.**

**Пример 1.**

Заполнить список квадратами чисел от 0 до 9, используя генератор списка.

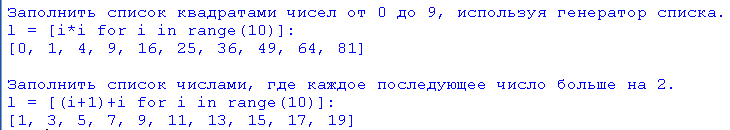
Решение:

l = [i\*i for i in range(10)]  
  
**Пример 2.**

Заполнить список числами, где каждое последующее число больше на 2.

l = [(i+1)+i for i in range(10)]

**print**(l)

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list4.png?attredirects=0)

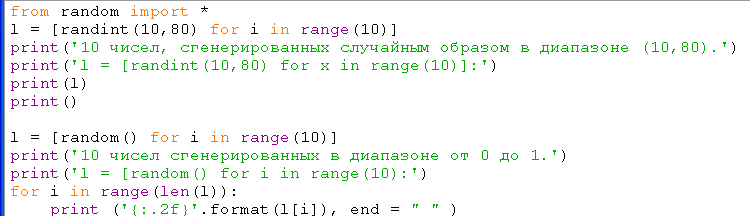
Модуль **random** предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.  
**random.randint**(A, B) - случайное целое число N, A ≤ N ≤ B.  
**random.random**() - случайное число от 0 до 1.

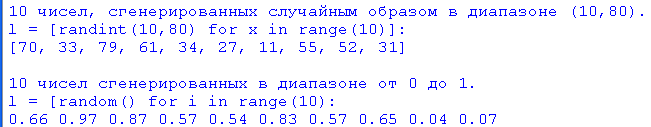
Случайные числа в списке:

10 чисел, сгенерированных случайным образом в диапазоне (10,80)  
from random import randint

l = [randint(10,80) for x in range(10)]

10 чисел, сгенерированных случайным образом в диапазоне (0,1)  
l = [random() for i in range(10)]

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list5_res.png?attredirects=0)

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list5.png?attredirects=0)

**4. Ввод списка (массива) в языке Python.**

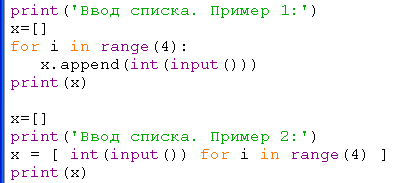
Для ввода элементов списка используется цикл **for** и команда **range** ():

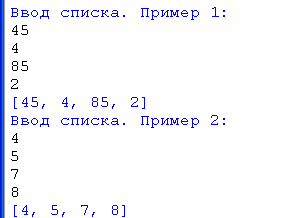
for i in range(N):

   x[i] = int( **input**() )

**Более простой вариант ввода списка:**

x = [ int(**input**()) for i in range(N) ]

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_array_prog.png?attredirects=0)

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_array_res.png?attredirects=0)

Функция **int** здесь используется для того, чтобы строка, введенная пользователем, преобразовывалась в целые числа.

**5.Вывод списка (массива) в языке Python.**

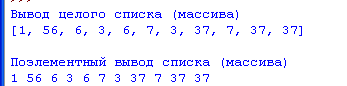
Вывод целого списка (массива):

**print** (L)

**Поэлементный вывод списка (массива):**

for i in range(N):

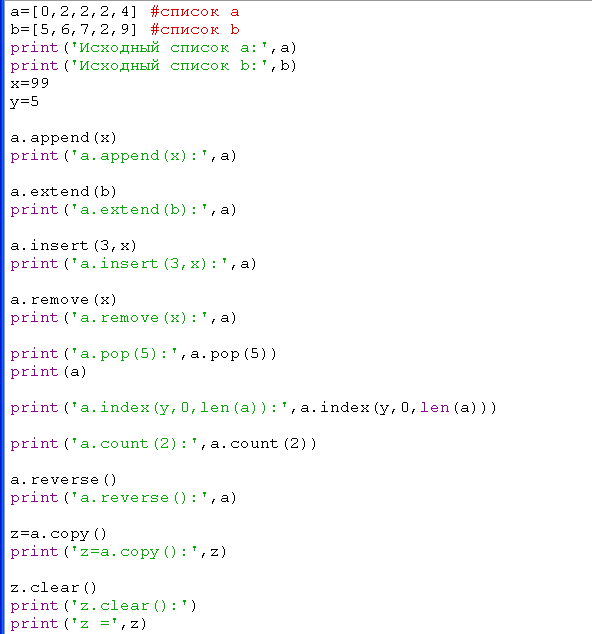
**print** ( L[i], end = " " )

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list_prog2_res.png?attredirects=0)

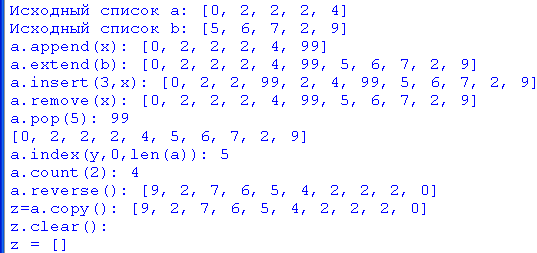
**2. Методы списков.**

| **Метод** |  | **Что делает** |
| --- | --- | --- |
| **list.append**(x) |  | Добавляет элемент в конец списка |
| **list.extend**(L) |  | Расширяет список list, добавляя в конец все элементы списка L |
| **list.insert**(i, x) |  | Вставляет перед i-ым элементом значение x |
| **list.remove**(x) |  | Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует |
| **list.pop**([i]) |  | Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент |
| **list.index**(x, [start [, end]]) |  | Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end) |
| **list.count**(x) |  | Возвращает количество элементов со значением x |
| **list.reverse**() |  | Разворачивает список |
| **list.copy**() |  | Поверхностная копия списка |
| **list.clear**() |  | Очищает список |

Ниже приведена программа, демонстрирующая методы работы списков.

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list_prog2.png?attredirects=0)

Пример программы на **Python**

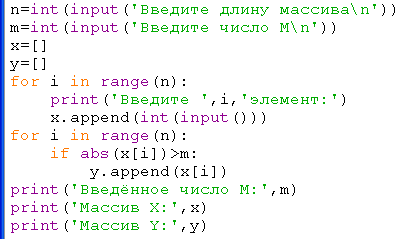
[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_list2.png?attredirects=0)

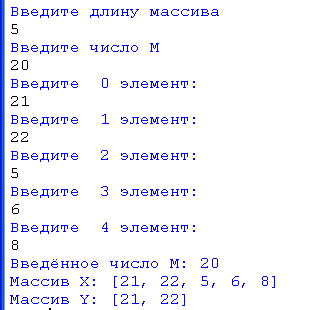
Результат выполнения программы

**Пример**

1. Из  массива X  длиной n, среди элементов которого  есть положительные, отрицательные и равные нулю, сформировать новый массив Y, взяв в него только те элементы из X,  которые больше по модулю заданного числа M. Вывести на  экран число M, данный и полученные массивы.

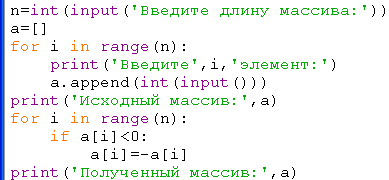
**Решение:**

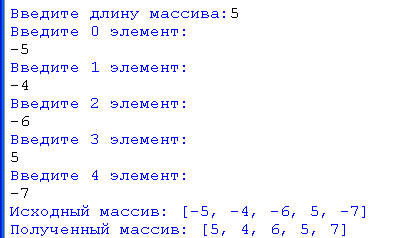
[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_array_prog_var0.png?attredirects=0)

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_array_prog_var0_res.png?attredirects=0)

2. В массиве целых чисел все отрицательные элементы заменить на положительные. Вывести исходный массив и полученный.

**Решение:**

**[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_array_var0_2_prog.png?attredirects=0)**

**[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_array_var0_2_res.png?attredirects=0)**

## Варианты заданий

В данной лабораторной работе варианты заданий выдаются преподавателем.

## Порядок выполнения лабораторной работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги.

3. Разработать алгоритм решения задачи, взяв за основу пример из теоретической части.

4. Изучить примеры, приведенные в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.

5. Подготовить свой вариант решения и отладить его с помощью компьютера.

## Контрольные вопросы

1. Перечислите характеристики типа данных «список», которые вы знаете.
2. Как проверить наличие элемента в списке?
3. Чем отличаются методы append() и extend()?
4. Какие параметры можно передавать при срезах списков?

# ЛАБОРАТОРНАЯ (ПРАКТИЧЕСКАЯ) РАБОТА № 7

**Тема:** функции и процедуры в **Python**.

**Цель работы:**изучение процедур и функций в **Python**.

## Теоретическое введение

знать *-*синтаксис процедур и функций, процедура с параметром, локальные и глобальные переменные;

уметь *-*применять синтаксис процедур и функций при составлении программы;

владеть *-*основными навыками работы с функциями и процедурами.

**Подпрограмма** - это именованный фрагмент программы, к которому можно обратиться из другого места программы  
Подпрограммы делятся на две категории: процедуры и функции.

1. Процедуры.

Рассмотрим синтаксис процедуры:

def имя процедуры (список параметров):

  Система команд

Для определения процедуры используется ключевое слово def, затем указывается имя процедуры и в скобках её формальные параметры, если они присутствуют. После ставится двоеточие и со следующей строки с отступом в 4 пробела указываются команды.

Процедура — вспомогательный алгоритм, выполняющий некоторые действия.

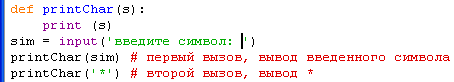
Процедура должна быть определена к моменту её вызова. Определение процедуры начинается со служебного слова def.  
Вызов процедуры осуществляется по ее имени, за которым следуют круглые скобки, например, Err().  
В одной программе может быть сколько угодно много вызовов одной и той же процедуры.  
Использование процедур сокращает код и повышает удобочитаемость.

Процедура с параметрами.  
Как используются в **Python** параметры процедуры, рассмотрим на примере.

**Пример.**

Написать процедуру, которая печатает раз указанный символ (введенный с клавиатуры), каждый с новой строки.

def **print**Char(s):  
    **print** (s)  
sim = **input**('введите символ')  
**print**Char(sim) # первый вызов, вывод введенного символа  
**print**Char('\*') # второй вызов, вывод \*

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-7/ex_procedure1_prog.png?attredirects=0)

[https://www.sites.google.com/site/moiboarkin/_/rsrc/1547772879421/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-7/ex_procedure1_res.png](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-7/ex_procedure1_res.png?attredirects=0)

**Глобальная переменная**— если ей присвоено значение в основной программе (вне процедуры).

**Локальная переменная**(внутренняя) известна только на уровне процедуры, обратиться к ней из основной программы и из других процедур нельзя.

**Параметры процедуры**— локальные переменные.

**2. Примеры использования локальных и глобальных переменных.**

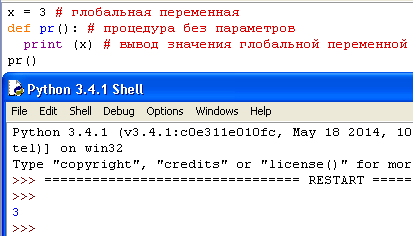
Пример 1.

x = 3 # глобальная переменная

def pr(): # процедура без параметров

**print** (x) # вывод значения глобальной переменной

pr()

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-7/ex_procedure2.png?attredirects=0)

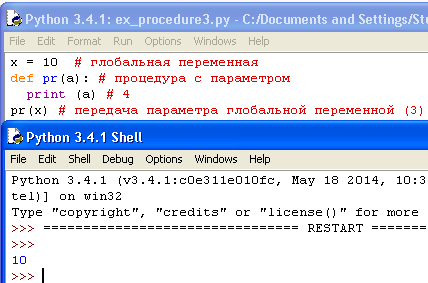
Пример 2.

x = 3  # глобальная переменная

def pr(a): # процедура с параметром

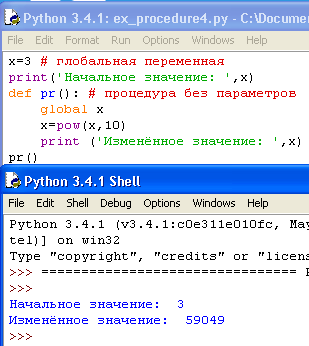
**print** (a) # 4

pr(x) # передача параметра глобальной переменной (3)



Существует возможность изменить значение глобальной переменной (не создавая локальную). В процедуре с помощью слова global:

x = 3 # глобальная переменная  
def pr(): # процедура без параметров  
    global x  
    x = pow(x,10)  
    **print** (x) # вывод измененного значения глобальной переменной   
pr()

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-7/ex_procedure4.png?attredirects=0)

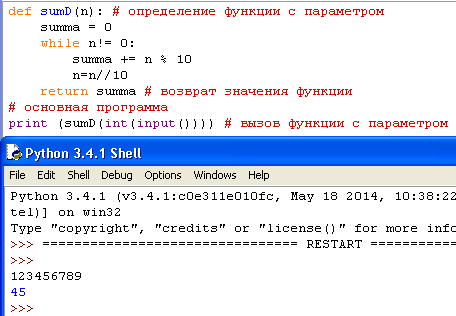
3**. Функции.**

Функция - подпрограмма, к которому можно обратиться из другого места программы.

Для создания функции используется ключевое слово def, после которого указывается имя и список аргументов в круглых скобках. Тело функции выделяется также как тело условия (или цикла): четырьмя пробелами.

Рассмотрим синтаксис функции:  
**def** имя функции(Список параметров):   
    Система команд  
    return выражение

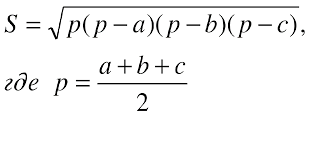
Часть функций языка **Python** являются встроенными функциями, которые обеспечены синтаксисом самого языка. Например, **int, input, randint.**  
Рассмотрим пример создания пользовательских функций.  
  
Пример 1.  
Вычислить сумму цифр числа.  
**def** sumD(n): # определение функции с параметром  
    sumD = 0  
**while** n!= 0:  
        sumD += n % 10  
        n = n // 10  
    **return** sumD # возврат значения функции  
# основная программа  
**print** (sumD(int(**input**())) # вызов функции с параметром



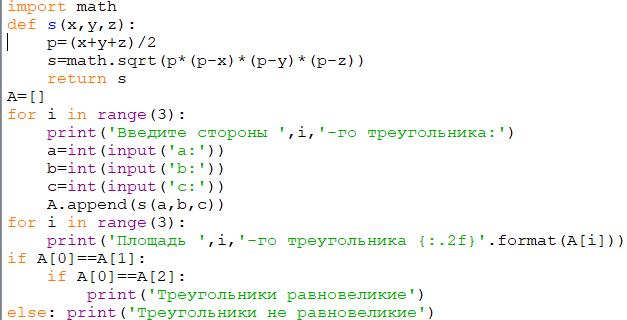
**Пример**

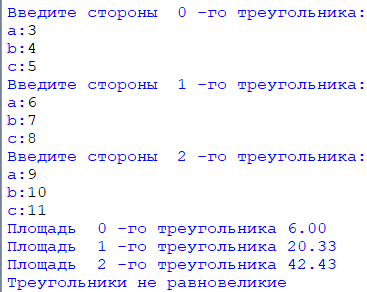
1. Определить, являются ли три треугольника равновеликими. Длины сторон вводить с клавиатуры. Для подсчёта площади треугольника использовать формулу Герона. Вычисление площади оформить в виде функции с тремя параметрами.

Формула Герона:

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-7/geron.gif?attredirects=0)

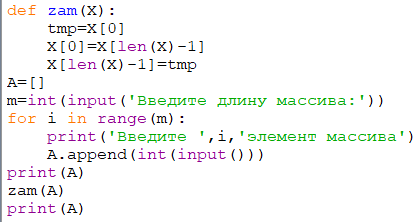
Решение:

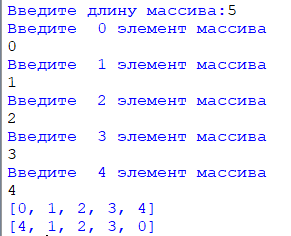




2. Ввести одномерный массив A длиной m. Поменять в нём местами первый и последний элементы. Длину массива и его элементы ввести с клавиатуры. В программе описать процедуру для замены элементов массива. Вывести исходные и полученные массивы.

**Решение:**





## Варианты заданий

В данной лабораторной работе варианты заданий выдаются преподавателем.

## Порядок выполнения лабораторной работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги.

3. Разработать алгоритм решения задачи, взяв за основу пример из теоретической части.

4. Изучить примеры, приведенные в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.

5. Подготовить свой вариант решения и отладить его с помощью компьютера.

## Контрольные вопросы

1. В чем основное различие между глобальными и локальными переменными?
2. Какая инструкция позволяет возвращать значения?
3. Назовите ключевое слово, которое определяет функцию
4. Какое количество аргументов может принимать функция?

# ЛАБОРАТОРНАЯ(ПРАКТИЧЕСКАЯ) РАБОТА № 8

**Тема:** работа с двумерными массивами.

**Цель работы:** изучение двумерных массивов в **Python**.

## Теоретическое введение

**знать** - способ описания двумерного массива, способы ввода элементов двумерного массива;

**уметь** - вводить массивы, получать списки через присваивание конкретных значений, применять функции;

**владеть** -  основными навыками создания программ обработки двумерных массивов.

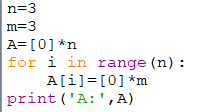
Матрицами называются массивы элементов, представленные в виде прямоугольных таблиц, для которых определены правила математических действий. Элементами матрицы могут являться числа, алгебраические символы или математические функции.

Для работы с матрицами в **Python** также используются списки. Каждый элемент списка-матрицы содержит вложенный список.

Таким образом, получается структура из вложенных списков, количество которых определяет количество столбцов матрицы, а число элементов внутри каждого вложенного списка указывает на количество строк в исходной матрице.

**1. Создание списка**  
  
Пусть даны два числа: количество строк n и количество столбцов m. Необходимо создать список размером n×m, заполненный нулями.  
Очевидное решение оказывается неверным:  
A = [ [0] \*m ]\*n  
  
В этом легко убедиться, если присвоить элементу A[0][0] значение 1, а потом вывести значение другого элемента A[1][0] — оно тоже будет равно 1! Дело в том, что [0] \* m возвращает ccылку на список из m нулей. Но последующее повторение этого элемента создает список из n элементов, которые являются ссылкой на один и тот же список (точно так же, как выполнение операции B = A для списков не создает новый список), поэтому все строки результирующего списка на самом деле являются одной и той же строкой.  
  
Таким образом, двумерный список нельзя создавать при помощи операции повторения одной строки.  
  
**Первый способ.**

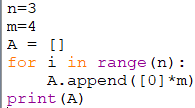
Сначала создадим список из n элементов (для начала просто из n нулей). Затем сделаем каждый элемент списка ссылкой на другой одномерный список из m элементов:  
  
A = [0] \* n  
for i in range(n):  
    A[i] = [0] \* m



[https://www.sites.google.com/site/moiboarkin/_/rsrc/1547730085561/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-8/ex_double_array_vvod1_res.png](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-8/ex_double_array_vvod1_res.png?attredirects=0)

**Второй способ.**

Создать пустой список, потом n раз добавить в него новый элемент, являющийся списком-строкой:  
  
A = []  
for i in range(n):  
    A.append([0] \* m)

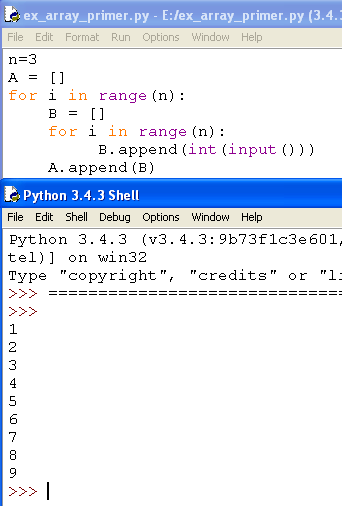


https://www.sites.google.com/site/moiboarkin/_/rsrc/1547730927754/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-8/ex_double_array_vvod2_res.png

**2. Ввод вложенного списка (двумерного массива)**

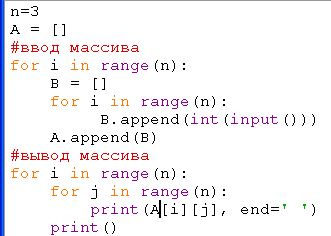
Пример:

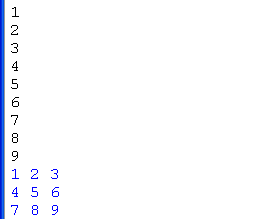
n=5  
A = []  
for i in range(n):  
    b = **input**()  
        for i in range(len(row)):  
            row[i] = int(row[i])  
A.append(row)



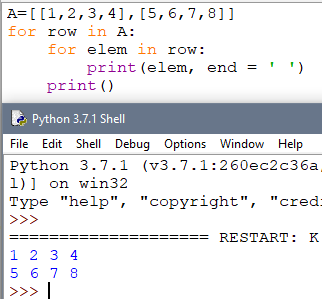
**3. Вывод вложенного списка (двумерного массива)**

Для обработки и вывода списка как правило используется два вложенных цикла. Первый цикл по номеру строки, второй цикл по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:  
  
for i in range(n):  
    for j in range(n]):  
        **print**(A[i][j], end = ' ')  
    **print**()





То же самое, но циклы не по индексу, а по значениям списка:  
  
for row in A:  
    for elem in row:  
        **print**(elem, end = ' ')  
    **print**()



Для вывода одной строки можно воспользоваться методом join. Использовав этот метод в цикле for можно

for row in A:

**print** (' '.join(list(map(str, row))))

**4. Обработка и вывод вложенных списков**  
  
Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются матрицами или двумерными массивами. В языке программирования Питон таблицу можно представить в виде списка строк, каждый элемент которого является в свою очередь списком, например, чисел. Например, создать числовую таблицу из двух строк и трех столбцов можно так:  
  
A = [ [1, 2, 3], [4, 5, 6] ]  
  
Здесь первая строка списка A[0] является списком из чисел [1, 2, 3].

То есть

A[0][0]= 1,

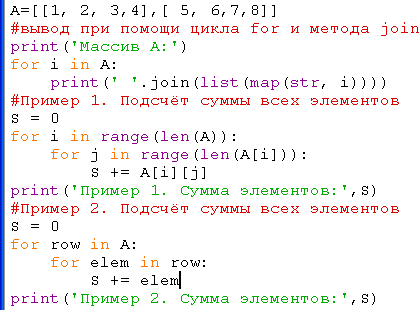
A[0][1]= 2,

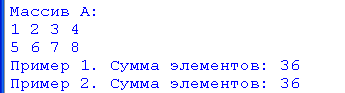
A[0][2]= 3,

A[1][0]=4,

A[1][1]=5,

A[1][2]=6.  
  
Используем два вложенных цикла для подсчета суммы всех чисел в списке:  
  
S = 0  
for i in range(len(A)):  
    for j in range(len(A[i])):  
        S += A[i][j]  
  
Или то же самое с циклом не по индексу, а по значениям строк:  
  
S = 0  
for row in A:  
    for elem in row:  
        S += elem



****

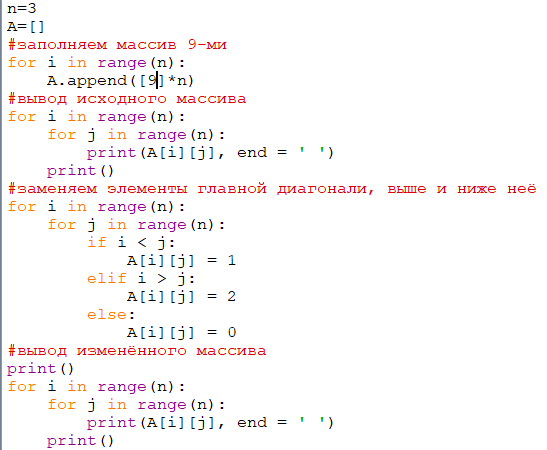
**5. Пример сложной обработки массива**  
  
Пусть дана квадратная матрица из n строк и n столбцов. Необходимо элементам, находящимся на главной диагонали, проходящей из левого верхнего угла в правый нижний (то есть тем элементам A[i][j], для которых i==j) присвоить значение 0, элементам, находящимся выше главной диагонали – значение 1, элементам, находящимся ниже главной диагонали – значение 2. То есть получить такой массив (пример для n==3):  
  
     0 1 1   
     2 0 1   
     2 2 0

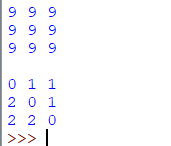
Рассмотрим несколько способов решения этой задачи.

**Первый способ.**

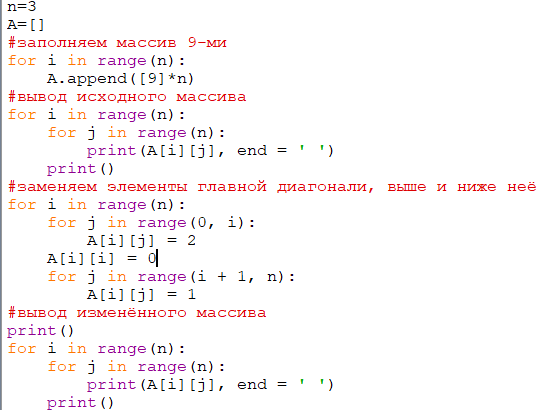
Элементы, которые лежат выше главной диагонали – это элементы A[i][j], для которых i<j, а для элементов ниже главной диагонали i>j. Таким образом, мы можем сравнивать значения i и j и по ним определять значение A[i][j]. Получаем следующий алгоритм:  
  
for i in range(n):  
    for j in range(n):  
        if i < j:  
            A[i][j] = 0  
        elif i > j:  
            A[i][j] = 2  
        else:  
            A[i][j] = 1

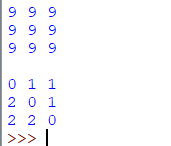
Ниже приведён пример программы, в котором квадратная матрица 3х3 заполняется элементами со значением 9, а затем элементам, находящимся на главной диагонали, проходящей из левого верхнего угла в правый нижний (то есть тем элементам A[i][j], для которых i==j) присваивается значение 0, элементам, находящимся выше главной диагонали – значение 1, элементам, находящимся ниже главной диагонали – значение 2.



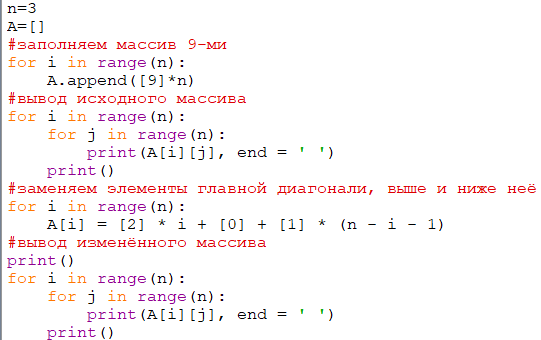


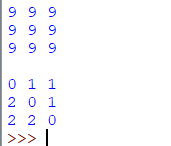
**Второй способ.**  
Данный алгоритм плох, поскольку выполняет одну или две инструкции if для обработки каждого элемента. Если мы усложним алгоритм, то мы сможем обойтись вообще без условных инструкций.  
  
Сначала заполним главную диагональ, для чего нам понадобится один цикл:  
  
for i in range(n):  
    A[i][i] = 1  
  
Затем заполним значением 0 все элементы выше главной диагонали, для чего нам понадобится в каждой из строк с номером i присвоить значение элементам A[i][j] для j=i+1, ..., n-1. Здесь нам понадобятся вложенные циклы:  
  
for i in range(n):  
    for j in range(i + 1, n):  
        A[i][j] = 0  
  
Аналогично присваиваем значение 2 элементам A[i][j] для j=0, ..., i-1:  
  
for i in range(n):  
    for j in range(0, i):  
        A[i][j] = 2  
  
Можно также внешние циклы объединить в один и получить еще одно, более компактное решение:  
  
for i in range(n):  
    for j in range(0, i):  
        A[i][j] = 2  
    A[i][i] = 1  
    for j in range(i + 1, n):  
        A[i][j] = 0





**Третий способ.**  
А вот такое решение использует операцию повторения списков для построения очередной строки списка. i-я строка списка состоит из i чисел 2, затем идет одно число 1, затем идет n-i-1 число 0:  
  
for i in range(n):  
    A[i] = [2] \* i + [1] + [0] \* (n - i - 1)

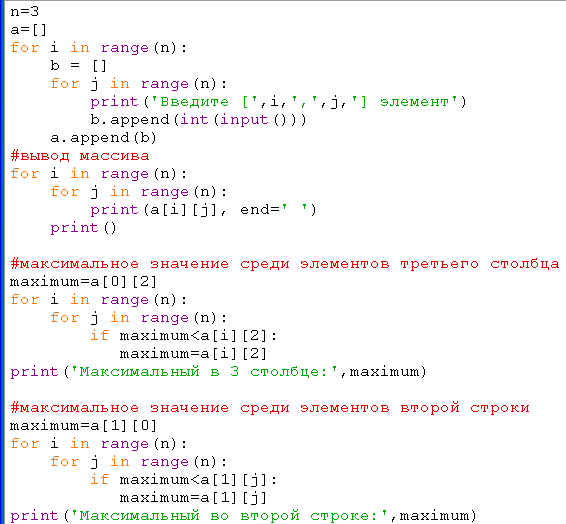


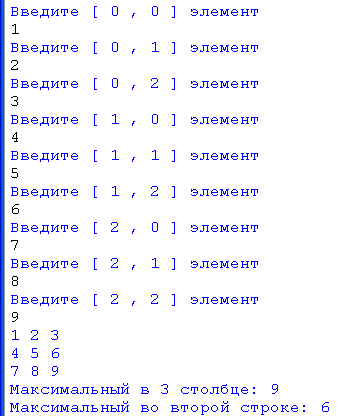


**Пример**

1. Дан двумерный массив размером 3x3. Определить максимальное значение среди элементов третьего столбца массива; максимальное значение среди элементов второй строки массива. Вывести полученные значения.

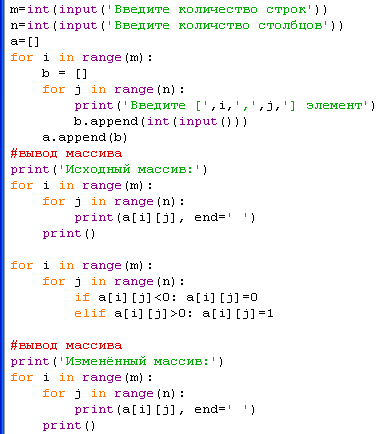
**Решение:**

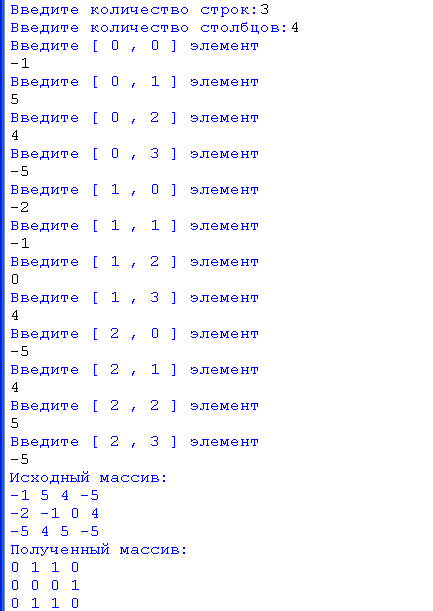
****

****

2. Дан двумерный массив размером mxn. Сформировать новый массив заменив положительные элементы единицами, а отрицательные нулями. Вывести оба массива.

**Решение:**

****

****

## Варианты заданий

В данной лабораторной работе варианты заданий выдаются преподавателем.

## Порядок выполнения лабораторной работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги.

3. Разработать алгоритм решения задачи, взяв за основу пример из теоретической части.

4. Изучить примеры, приведенные в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.

5. Подготовить свой вариант решения и отладить его с помощью компьютера.

## Контрольные вопросы

1. Что используют для создания двухмерного массива?
2. Каким образом происходит нумерация элементов массива по индексам? Приведите пример.