

# Python



# Основы языка программирования Python

- Python – один из самых используемых языков программирования. Благодаря простоте синтаксиса и большому числу встроенных инструментов и библиотек, он быстро завоевал популярность даже за пределами мира IT. Сегодня это основной язык, который используется в сфере Data Science.
- Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.
- Язык является полностью объектно-ориентированным — всё является объектами.
- Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации.
- Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов.
- Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как Си или C++.

Python — популярный высокоуровневый язык программирования, который предназначен для создания приложений различных типов.

Это и веб-приложения, и программы для работы с базами данных, и настольные программы, и компьютерные игры.

Довольно большое распространение язык программирования Python получил в таком направлении искусственного интеллекта как машинное обучение.

Python — достаточно простой для освоения язык программирования, он имеет лаконичный и понятный синтаксис.

## Области применения языка программирования Python

- создание приложений различных типов:
  - Web-приложения
  - программы для работы с базами данных
  - настольные программы
  - игры
- разработки в области искусственного интеллекта, в частности — машинного обучения

Стандартная библиотека включает большой набор полезных переносимых функций, начиная от функционала для работы с текстом и заканчивая средствами для написания сетевых приложений. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений или разработка игр, могут реализовываться посредством обширного количества сторонних библиотек, а также интеграцией библиотек, написанных на Си или C++, при этом и сам интерпретатор Python может интегрироваться в проекты, написанные на этих языках. Существует и специализированный репозиторий программного обеспечения, написанного на Python, — PyPI. Данный репозиторий предоставляет средства для простой установки пакетов в операционную систему и стал стандартом де-факто для Python. По состоянию на 2019 год в нём содержалось более 175 тысяч пакетов.



Python стал одним из самых популярных языков, он используется **в анализе данных, машинном обучении**, DevOps и др.

За счёт читабельности, простого синтаксиса и отсутствия необходимости в компиляции язык хорошо подходит для обучения программированию, позволяя концентрироваться на изучении алгоритмов, концептов и парадигм. Отладка же и экспериментирование в значительной степени облегчаются тем фактом, что язык является интерпретируемым. Применяется язык многими крупными компаниями, такими как Google или Facebook. По состоянию на апрель 2021 года Python занимает третье место в рейтинге TIOBE (на основе подсчёта результатов поисковых запросов, содержащих название языка) популярности языков программирования с показателем 11,03 %. «Языком года» по версии TIOBE Python объявлялся в 2007, 2010, 2018 и 2020 году.

## Введение / история

- Синтаксис Python позволяет создавать ясный и выразительный исходный код.
- Python имеет "батарейки в комплекте" - обширную стандартную библиотеку, охватывающую ключевые задачи разработки ПО.
- Python поддерживает различные парадигмы программирования: императивное, объектно-ориентированное, и, частично, функциональное программирование.
- Основные характеристики языка: динамическая неявная типизация, переменные ссылаются на значения и передаются по ссылке, объекты базовых типов (числа и строки) неизменяемые, автоматическое управление памятью.
- Python рассчитан на исполнение с помощью интерпретатора. На данный момент доступны интерпретаторы Python для всех основных платформ и программы, написанные на Python, могут без изменений запускаться под разными операционными системами.

## Краткая история Питона:

Автор - Гвидо ван Россум (Guido van Rossum), нидерландский программист, автор языка программирования Python. До разработки Python участвовал в проекте по написанию языка для обучения программированию — ABC. Ван Россум родился и вырос в Нидерландах, где он закончил Амстердамский университет по специальности «математика и информатика» в 1982 году. Затем он работал в различных исследовательских институтах, в том числе Голландском центре математики и информатики (Амстердам), Национальном Институте Стандартов (США). В 2001 году ван Россум получил премию Free Software Award, а в 2002 году — премию Нидерландской ассоциации профессионалов UNIX (NLUUG). В 2006 он был признан Ассоциацией вычислительной техники «выдающимся инженером».



Покинув в декабре 2012 года корпорацию Google, с 2013 года работал в компании и Dropbox, выйдя на пенсию в 2019. В ноябре 2020 года на своём twitter-канале заявил о своём решении присоединиться к подразделению разработки компании Microsoft.



## Программисты отмечают следующие преимущества языка Python:

Во-первых. Единообразие оформления программного кода на языке Python способствует лучшему его пониманию.

Во-вторых. Использование этого языка (например, по сравнению C/C++, Java) повышает производительность труда разработчиков, во-многом, за счет уменьшения объема программного кода, вводимого с клавиатуры.

### *Преимущества языка Python*

1. Единообразие оформления программного кода
2. Небольшой объем программного кода
3. Возможность портирования программного кода
4. Большая библиотека функциональных возможностей
5. Интеграция с другими языками программирования



В-третьих. Возможность перенесения программного кода из одной операционной системы в другую простым копированием файлов (например, из OS Linux в OS Windows).

В-четвертых. Наличие большого количества собранных и переносимых функциональных возможностей, называемых стандартной библиотекой. Эта библиотека может расширяться как за счет собственных модулей, так и за счет модулей, созданных другими программистами.

## *Преимущества языка Python*

1. Единообразие оформления программного кода
2. Небольшой объем программного кода
3. Возможность портирования программного кода
4. Большая библиотека функциональных возможностей
5. Интеграция с другими языками программирования

Стандартная библиотека в составе Python, включает множество модулей, которые позволяют достаточно просто разрабатывать средства искусственного интеллекта, например, чат-боты. В-пятых. Возможность интеграции программного кода на языке Python с программными компонентами, написанными на других языках программирования. Так, программный код на языке Python может вызывать функции из библиотек в составе C/C++, интегрироваться с программными компонентами на языке Java. Такая интеграция позволяет использовать язык Python для настройки и расширения функциональных возможностей программных продуктов.

## *Преимущества языка Python*

1. Единообразие оформления программного кода
2. Небольшой объем программного кода
3. Возможность портирования программного кода
4. Большая библиотека функциональных возможностей
5. Интеграция с другими языками программирования



Среди **недостатков** языка Python можно отметить, что скорость выполнения его программ может быть ниже, чем программ, написанных на некоторых языках программирования (например, C/C++). Причина заключается в следующем. В современной реализации язык Python транслирует инструкции исходного программного кода в промежуточное представление, известное как байт-код, который является платформо-независимым форматом и обеспечивает переносимость программ. Затем байт-код интерпретируется. Поэтому некоторые программы на интерпретируемом языке Python могут работать медленнее своих аналогов, написанных на компилируемых языках программирования, которые сразу же создают двоичный машинный код.

## Недостатки языка Python

1. Python – интерпретируемый язык
2. Невысокая скорость выполнения программ



Язык Python изначально является объектно-ориентированным языком программирования. В то же время он поддерживает структурный стиль программирования.

Начнем изучение языка Python с точки зрения структурного программирования. Это позволит, базируясь на сформированных знаниях и умениях школьников в обязательном курсе информатики, рассмотреть особенности реализации алгоритмов на языке программирования Python. Возможности объектно-ориентированного программирования языка Python будем использовать по мере необходимости.

## Язык программирования Python

1. Объектно-ориентированный
2. Структурный

В соответствии с парадигмой структурного программирования при разработке алгоритмов используются три базовые управляющие структуры: следование, ветвление и цикл (повторение).

Реализация этих структур и их название в разных языках программирования могут различаться.

Разработка таких алгоритмов для решения задачи ведётся пошагово, по принципу «сверху вниз», что предполагает разделение задачи на подзадачи. И если для какой-либо подзадачи уже создан алгоритм, то его можно использовать в качестве вспомогательного при создании новых.

Вспомогательные алгоритмы, записанные на языках программирования, называют подпрограммами (а именно, процедурами или функциями).

## Язык программирования Python

1. Объектно-ориентированный
2. Структурный



В соответствии с парадигмой структурного программирования при разработке алгоритмов используются три базовые управляющие структуры: следование, ветвление и цикл (повторение).

Реализация этих структур и их название в разных языках программирования могут различаться.

Разработка таких алгоритмов для решения задачи ведётся пошагово, по принципу «сверху вниз», что предполагает разделение задачи на подзадачи. И если для какой-либо подзадачи уже создан алгоритм, то его можно использовать в качестве вспомогательного при создании новых. Вспомогательные алгоритмы, записанные на языках программирования, называют подпрограммами (а именно, процедурами или функциями).

## Принципы структурного программирования

- использование трёх базовых управляющих структур: следование, ветвление и повторение
- разработка программ методом «сверху вниз»
- использование подпрограмм



Программа на языке Python, как и на других языках структурного программирования, представляет собой последовательность команд, а точнее, последовательность инструкций. В некоторых языках программирования инструкции называются операторами.

## Принципы структурного программирования

- использование трёх базовых управляющих структур: следование, ветвление и повторение
- разработка программ методом «сверху вниз»
- использование подпрограмм

## Код на Python - первый взгляд

- В исходном коде не объявляется тип переменных, параметров функций и их возвращаемых значений. Это делает код более компактным и гибким, но приводит к потере возможности эффективной проверки кода на этапе компиляции.
- Python отслеживает тип всех переменных во время исполнения программы и сообщает об ошибках в момент, когда встречается их при попытке исполнить код.
- Обычно самым простым и эффективным способом понять как будет вести себя код на Python - запустить его.
- Для работы с кодом на Python часто используется REPL (Read Eval Print Loop) - форма организации интерактивной среды программирования, которая получает введенное пользователем выражение, исполняет его и возвращает пользователю полученный результат. Программа, написанная в среде REPL, исполняется по частям.

# Переменные

Для доступа к объекту предназначены переменные.

**В языке Python все переменные являются ссылками на объекты.**

В Python при инициализации в переменной сохраняется ссылка (адрес объекта в памяти компьютера) на объект.

Благодаря этой ссылке можно в дальнейшем использовать объект из программы.

В Python для присвоения используется `=` (для сравнения значений на равенство используется `==`). Первое присвоение значения переменной создает ее. Нет необходимости декларировать тип переменной (он определяется автоматически - по типу присваиваемого объекта).



## Явная и неявная типизация

Тип новых переменных / функций / их аргументов в ЯП с явной типизацией нужно задавать явно.

Достоинства явной типизации:

- Наличие у каждой функции сигнатуры (например `int add(int, int)`) позволяет без проблем определить, что функция делает.
- Программист сразу записывает, какого типа значения могут храниться в конкретной переменной, что снимает необходимость запоминать это.

Достоинства неявной типизации

- Сокращение записи — `def add(x, y)` короче, чем `int add( int x, int y)`.
- Устойчивость к изменениям. Например если в функции временная переменная была того же типа, что и входной аргумент, то в явно типизированном языке при изменении типа входного аргумента нужно будет изменить еще и тип временной переменной.

**В Python используется неявная типизация.**

Рассмотрим инструкцию, осуществляющая вывод информации на экран. Она записывается в следующем формате: `print (<список вывода>)`.

Например, в круглых скобках этой инструкции в апострофах или двойных кавычках можно написать текст (т.е. текстовую константу). При запуске программы этот текст будет выведен на экран без кавычек.

Каждая инструкция Python-программы помещается на новую строку. Строка заканчивается нажатием клавиши «Enter».

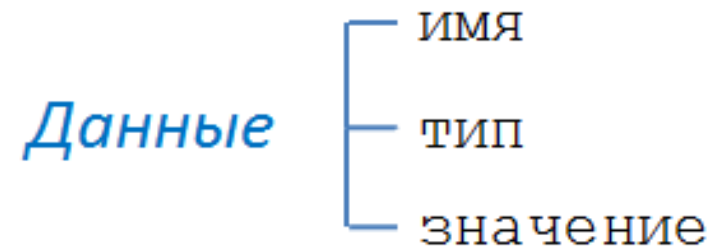
## *Инструкция вывода*

```
print (<список вывода>)
```

*Пример:*

```
print ('Hello World')
```

При создании программ для работы с данными используют переменные и константы, которые, как известно, имеют три характеристики: **ИМЯ, ТИП И значение**. Если значение константы неизменно, то значение переменной может меняться в процессе исполнения программы.





Рассмотрим использование переменных в языке программирования Python.

Имя (другими словами, идентификатор) переменной в языке Python может начинаться с заглавной или строчной буквы, а также со знака подчеркивания и может содержать алфавитно-цифровые символы и знаки подчеркивания. Кроме того, имя переменной не должно совпадать с названием ключевых (или служебных) слов языка Python. Полный перечень ключевых слов вы видите на экране. Большую часть этих слов мы будем использовать при программировании.

Также надо учитывать регистрозависимость языка: переменные, имена которых различаются заглавными или строчными буквами, представляют разные объекты.

В языке Python не требуется объявлять переменные, т.е. не требуется описывать имя и тип переменной в начале программы.

Переменные: `name_Var`, `_name`,  
`Name`, `NAME`, `name`

*Ключевые слова языка Python:*

<code>and</code>	<code>False</code>	<code>nonlocal</code>
<code>as</code>	<code>finally</code>	<code>not</code>
<code>assert</code>	<code>for</code>	<code>or</code>
<code>break</code>	<code>from</code>	<code>pass</code>
<code>class</code>	<code>global</code>	<code>raise</code>
<code>continue</code>	<code>if</code>	<code>return</code>
<code>def</code>	<code>import</code>	<code>True</code>
<code>del</code>	<code>in</code>	<code>try</code>
<code>elif</code>	<code>is</code>	<code>while</code>
<code>else</code>	<code>lambda</code>	<code>with</code>
<code>except</code>	<code>None</code>	<code>yield</code>

Как известно, переменные предназначены для возможности осуществления доступа к данным. Данные могут быть различных типов. Тип данных указывает, какие значения может принимать переменная, а также, какие операции можно выполнять с переменными.

Язык программирования Python поддерживает 12 типов данных. Перечислим те из них, с которыми нам предстоит работать:

- `boolean` — тип, которому принадлежат два логических значения: `True` (истина) и `False` (ложь)
- тип `int` — представляет целые числа

## Основные типы данных

Тип данных	Описание	Примеры		
<code>boolean</code>	логические значения	<code>True</code>	<code>False</code>	
<code>int</code>	целые числа	<code>5</code>	<code>-4</code>	<code>0</code>
<code>float</code>	вещественные числа	<code>-0.7</code>	<code>1.67e+23</code>	
<code>str</code>	строки (литералы)	<code>'hello'</code>	<code>"Python"</code>	<code>'a'</code>
<code>list</code>	списки	<code>[2, 'a', 4.8]</code>		
<code>dict</code>	словари	<code>{3: 'hello', 2: "Python"}</code>		



- float — представляет вещественные числа. Этот тип еще называют - число с плавающей точкой. Естественно, язык Python поддерживает экспоненциальную форму записи вещественных чисел.

- str — тип, представляющий строки. Записывается строка в апострофах или двойных кавычках. Заметим, что отдельного типа данных для символов в языке Python нет. Одиночный символ — это тоже строка.

- list — тип, представляющий списки, а именно, упорядоченные наборы объектов произвольных типов. Элементы списка указываются в квадратных скобках, через запятую.

- dict — тип, представляющий словари. Словари – это составные объекты, где каждый элемент имеет ключ и значение. Записываются словари в фигурных скобках. Более подробно синтаксис и способы работы с каждым из этих типов данных обсудим несколько позже.

## Основные типы данных

Тип данных	Описание	Примеры		
boolean	логические значения	True	False	
int	целые числа	5	-4	0
float	вещественные числа	-0.7	1.67e+23	
str	строки (литералы)	'hello'	"Python"	'a'
list	списки	[2, 'a', 4.8]		
dict	словари	{3: 'hello', 2: "Python"}		

Как уже было ранее отмечено, объявлять типы переменных в Python-программах не требуется, поскольку язык Python автоматически определяет тип данных переменной, исходя из ее значения. Значение переменной хранится в ячейке памяти, соответствующей этой переменной.

## Основные типы данных

Тип данных	Описание	Примеры		
boolean	логические значения	True	False	
int	целые числа	5	-4	0
float	вещественные числа	-0.7	1.67e+23	
str	строки (литералы)	'hello'	"Python"	'a'
list	списки	[2, 'a', 4.8]		
dict	словари	{3: 'hello', 2: "Python"}		



Одним из способов сохранения (по-другому, помещения) значения переменной в ячейке памяти является использование инструкции присваивания, которая имеет следующий формат:

<имя переменной> = <выражение>

Знак присваивания в языке Python обозначается знаком равенства. На экране мы видим примеры таких записей ...

Так, если присвоить переменной значение строки в двойных кавычках или апострофах, то переменная получит строковый тип. При присвоении переменной значения целого числа, язык Python автоматически определяет тип переменной как целочисленный.

Кроме того, язык программирования Python является языком с динамической типизацией, поэтому в процессе работы программы тип переменной можно изменить, присвоив ей значение другого типа.

Оператор = - это не оператор присваивания значения переменной, как в некоторых других языках программирования.

Оператор = связывает ссылку на объект (переменную) с объектом, находящимся в памяти.

- Если ссылка на объект (переменная) уже существует, ее легко можно связать с другим объектом, указав этот объект справа от оператора = .
- Если ссылка на объект еще не существует, она будет создана оператором = .

## Оператор присваивания

<имя\_переменной> = <выражение>

*Примеры:* a = 'Привет'  
b = -5  
a = 3.6

## Статическая и динамическая типизация

- Статическая типизация - переменная, параметр подпрограммы, возвращаемое значение функции связывается с типом в момент объявления и тип не может быть изменен позже.

### Достоинства:

- хороша для написания сложного, но быстрого кода;
- хорошо работает автодополнение в IDE;
- многие ошибки исключаются на стадии компиляции.

### Недостатки:

- многословный код;
- сложность написания обобщенных алгоритмов и универсальных коллекций;
- сложности с работой с данными из внешних источников.

**В языке Python используется динамическая типизация**, т.е. "тип переменной" (являющейся по сути ссылкой на объект) может меняться во время ее жизни.



Чтобы значение некоторой переменной было выведено на экран, используется уже знакомая нам инструкция print(). Для этого имя переменной указывается внутри скобок инструкции print(), но в кавычки не заключается, поскольку нужно вывести на экран значение переменной, а не ее имя.

Инструкция print позволяет выводить на экран не только текст и значения переменных, заданных или вычисленных в программе ранее.

### *Инструкция вывода*

```
print (<имя_переменной>)
```

*Пример:*     print(a)

В круглых скобках инструкции `print` может быть и выражение. Если на экран необходимо вывести значение этого выражения, естественно, при условии, что это значение определено, то тогда это выражение записывается без кавычек.

Если на экран хотим вывести саму запись выражения, то тогда это выражение следует написать в кавычках.

### Инструкция вывода

```
print(<имя_переменной>)
```

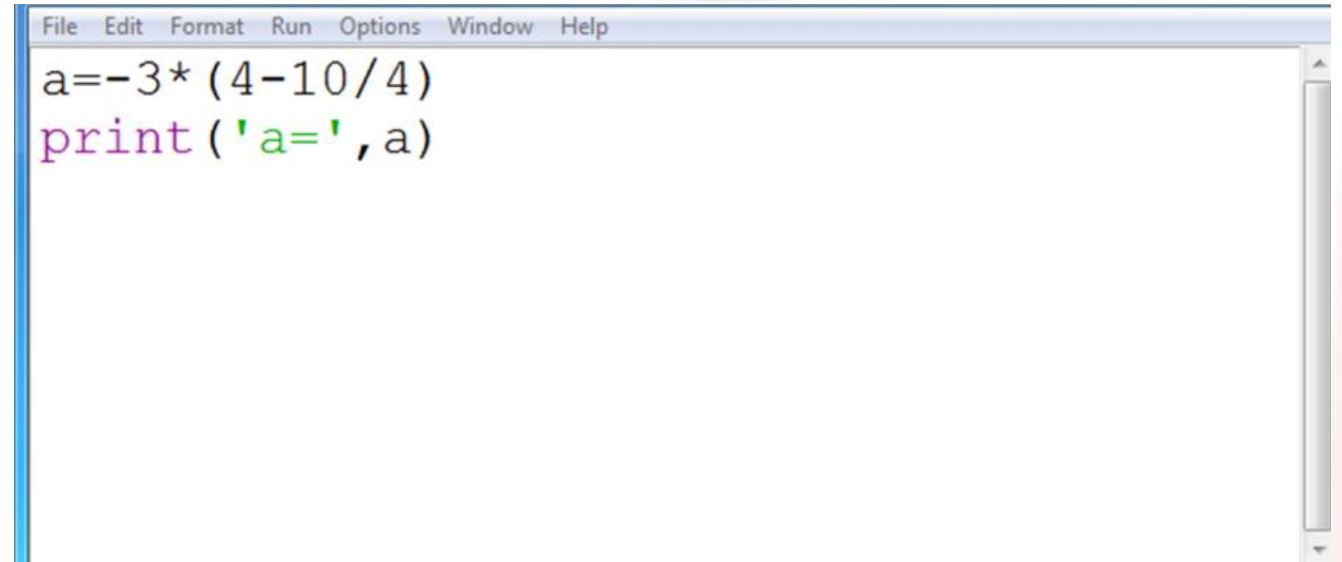
*Пример:*     `print(a)`

```
print(<выражение>)
```

*Пример:*     `print(a-2*b)`

Напишем программу, состоящую из двух инструкций, а именно, переменной A присвоить значение некоторого арифметического выражения и вывести значение переменной A на экран...

Заметим, что в круглых скобках инструкции `print()`, в качестве списка вывода, можно написать произвольное количество элементов, разделенных запятыми. Эти элементы могут быть как строками, заключенными в кавычки, так и именами переменных, констант, а также выражениями.

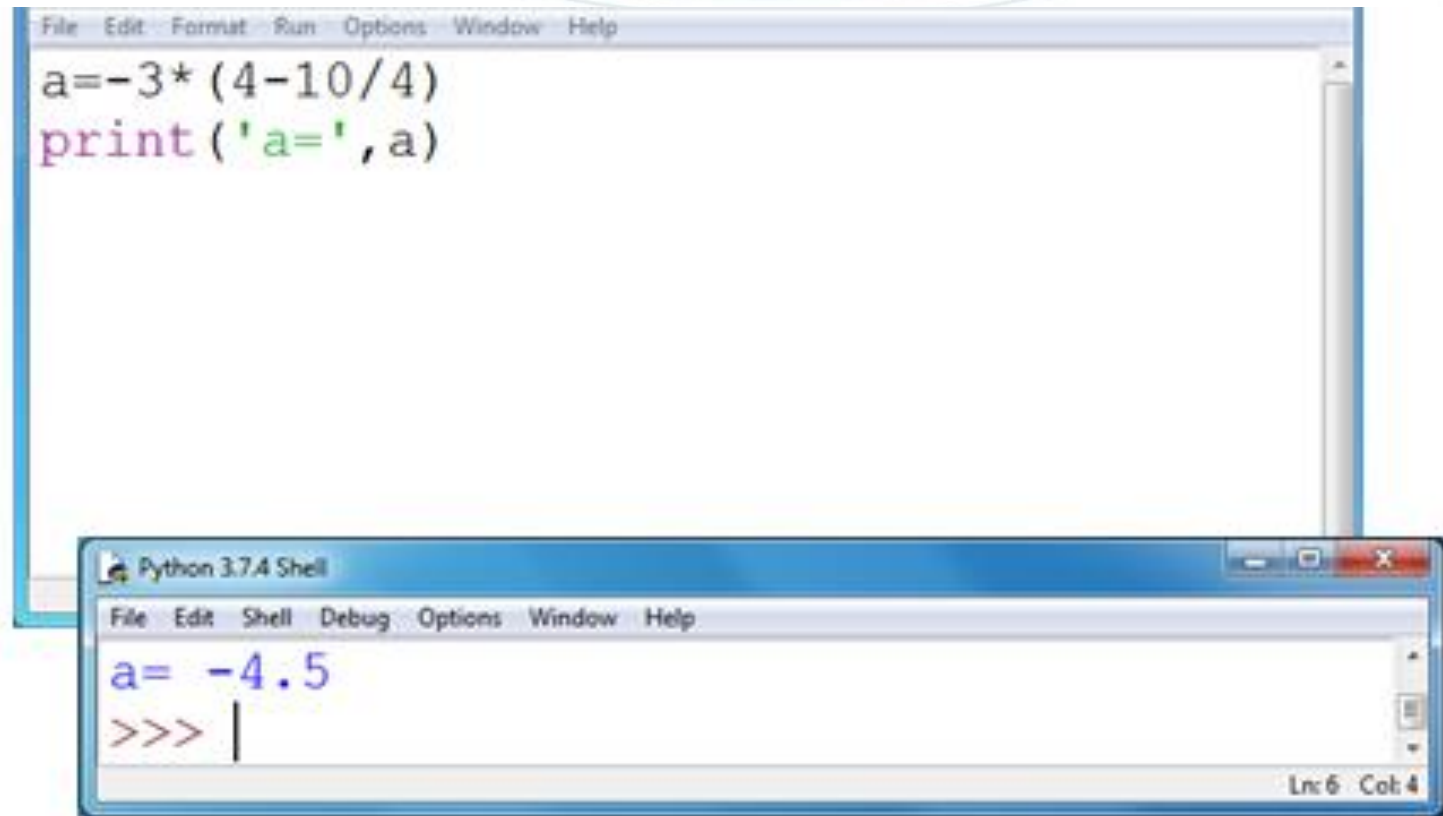


```
File Edit Format Run Options Window Help
a=-3*(4-10/4)
print('a=', a)
```



В итоге при выполнении инструкции, а именно при выводе на экран, все переданные значения будут расположены через пробелы в одну строку. Это используют для того, чтобы вывод на экран был более наглядным и информативным ...

Итак, мы узнали, что объявлять переменные и их типы в Python-программах не требуется; выяснили основные типы данных языка программирования Python, рассмотрели синтаксис инструкции присваивания и синтаксис инструкции вывода.



The image shows a screenshot of a Python IDE with two windows. The top window contains the following code:

```
File Edit Format Run Options Window Help
a=-3*(4-10/4)
print('a=', a)
```

The bottom window, titled "Python 3.7.4 Shell", shows the execution of the code:

```
File Edit Shell Debug Options Window Help
a= -4.5
>>> |
```

The status bar at the bottom right of the shell window indicates "Ln: 6 Col: 4".

Теперь мы рассмотрим инструкцию, отвечающую за ввод информации, обсудим приемы работы с числами и числовыми выражениями, а также выясним возможности использования различных функций.

## *Содержание*

- инструкция ввода
- работа с числами
- встроенные функции для работы с числами

Если инструкция `print()` отвечает за вывод данных, то за ввод данных отвечает инструкция `input()`.

Инструкция `input()` является вторым способом, который позволяет задать значение переменной. Эта инструкция осуществляет ввод данных с клавиатуры в память компьютера и имеет следующий формат:  
`<имя переменной> = input()`

### *Инструкция ввода*

```
<имя_переменной> = input()
```

*Примеры:*     `name = input()`



Инструкция `input()` ожидает ввода с клавиатуры последовательности символов. После их набора и нажатия клавиши Enter из этих символов формируется упорядоченная последовательность символов, называемая **строкой**. При выполнении этой инструкции значения, вводимые с помощью клавиатуры компьютера, будут сохранены в ячейках памяти, соответствующих указанной переменной.

## Инструкция ввода

```
<имя_переменной> = input()
```

*Примеры:*     `name = input()`

Это и отражено в форме записи, которую вы видите на экране. Так, например, можно сохранить введенную последовательность символов под именем name. Переменная name при этом автоматически получит строковый тип.

В скобках инструкции input() можно ничего не писать. Тогда при запуске программы система просто будет ожидать ввода с клавиатуры.

На экране в это время будет мигать текстовый курсор. Но при такой ситуации пользователю непонятно, что именно от него требуется.

## Инструкция ввода

```
<имя_переменной> = input()
```

*Примеры:*      name = input()

Поэтому внутри скобок в апострофах или двойных кавычках желательно написать вопрос, ответ на который должен быть введен с клавиатуры. Он отобразится на экране в процессе работы этой инструкции, и пользователь сможет ответить на него.

## *Инструкция ввода*

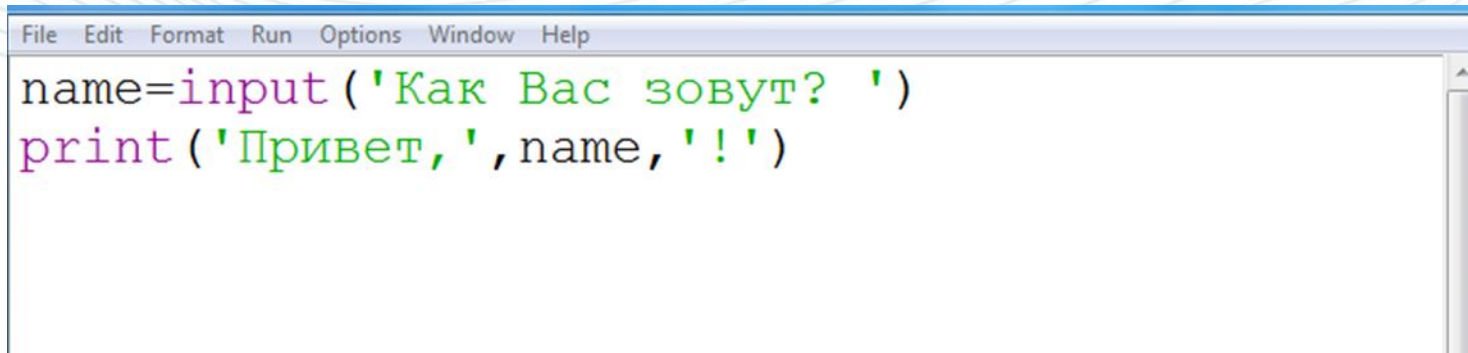
```
<имя_переменной> = input()
```

*Примеры:*     `name = input()`  
`name = input('Ваше имя: ')`

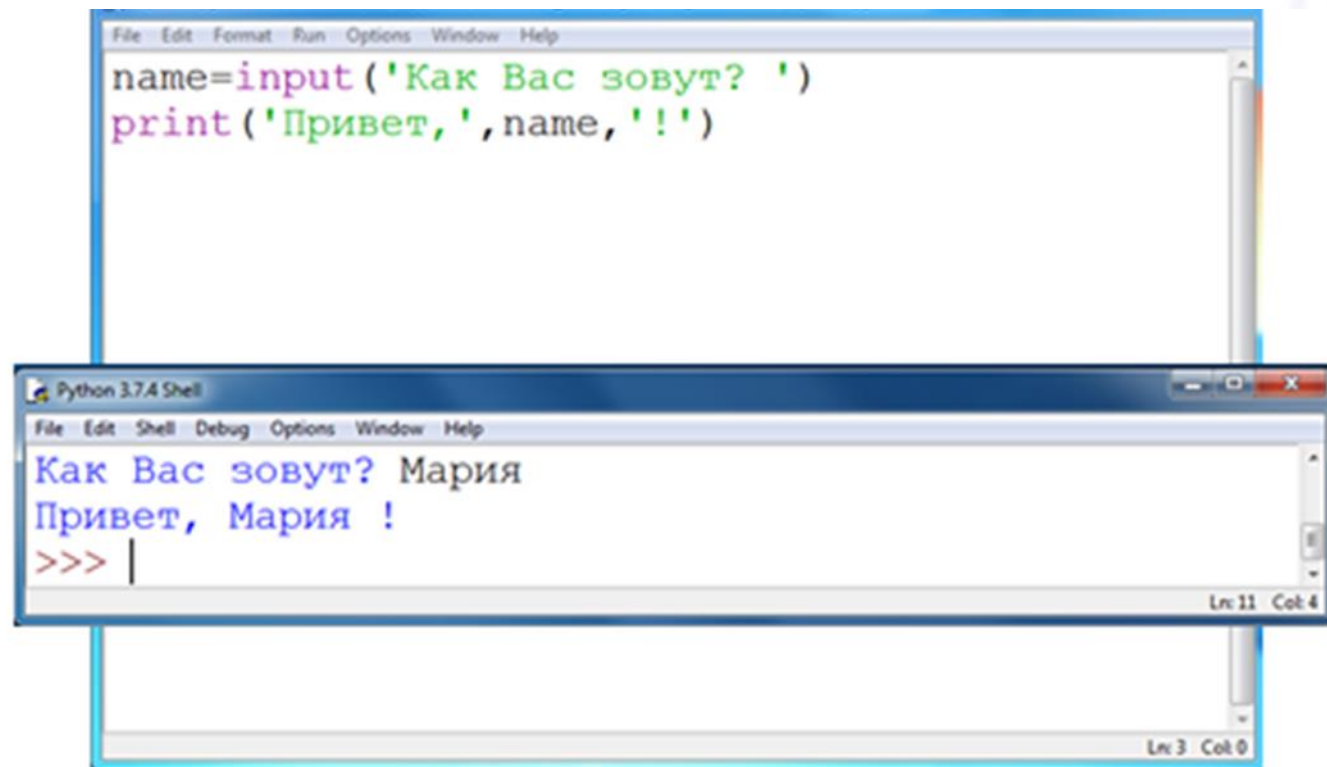


Рассмотрим в качестве примера программу, организующую начало диалога с пользователем.

В результате работы этой программы на экран выводится приветствие для пользователя с обращением к нему по имени.



```
File Edit Format Run Options Window Help
name=input('Как Вас зовут? ')
print('Привет, ', name, '!')
```



```
File Edit Format Run Options Window Help
name=input('Как Вас зовут? ')
print('Привет, ', name, '!')
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Как Вас зовут? Мария
Привет, Мария !
>>> |
Ln: 11 Col: 4
```

```
Ln: 3 Col: 0
```

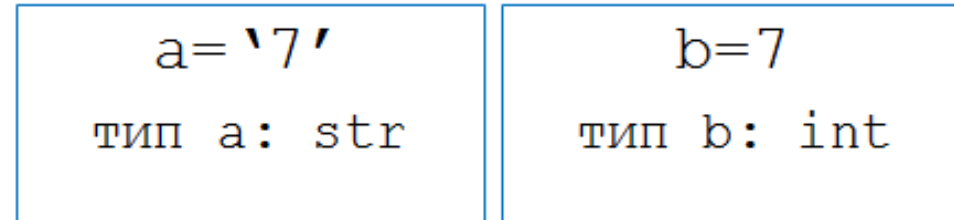
Как уже было сказано, инструкция `input()` всегда порождает строковую переменную. Но часто возникает необходимость введения пользователем числовых данных.

При этом надо понимать, что строка, содержащая символ 7, – это не то же самое, что число 7.

Поэтому возникает необходимость преобразовать строковый тип данных в числовой тип.

Сделать это можно с помощью функций `int` и `float`. Первая преобразует данные в целочисленный тип, вторая преобразует данные в вещественный тип.

*Объекты a и b – различны*



*Инструкции преобразования  
строковых объектов в числовые*

`int (<выражение>)`

`float (<выражение>)`

Для того чтобы введенные, например, с клавиатуры, данные интерпретировались системой как целое число, необходимо использовать инструкцию, которую вы видите на экране. Тип переменной N в этом случае будет целочисленный.

### *Инструкции преобразования строковых объектов в числовые*

`int (<выражение>)`

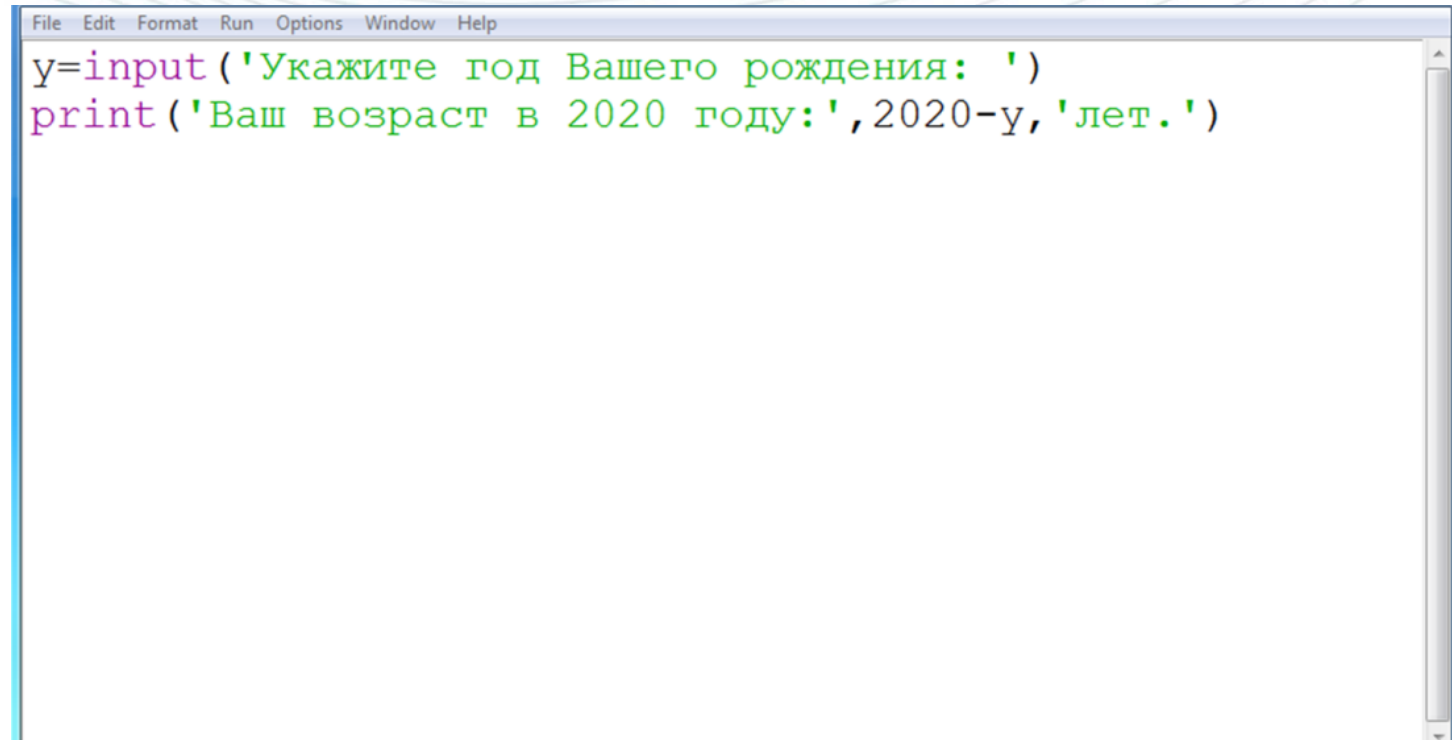
`float (<выражение>)`

#### *Пример:*

```
n=int(input('Введите число: '))
```



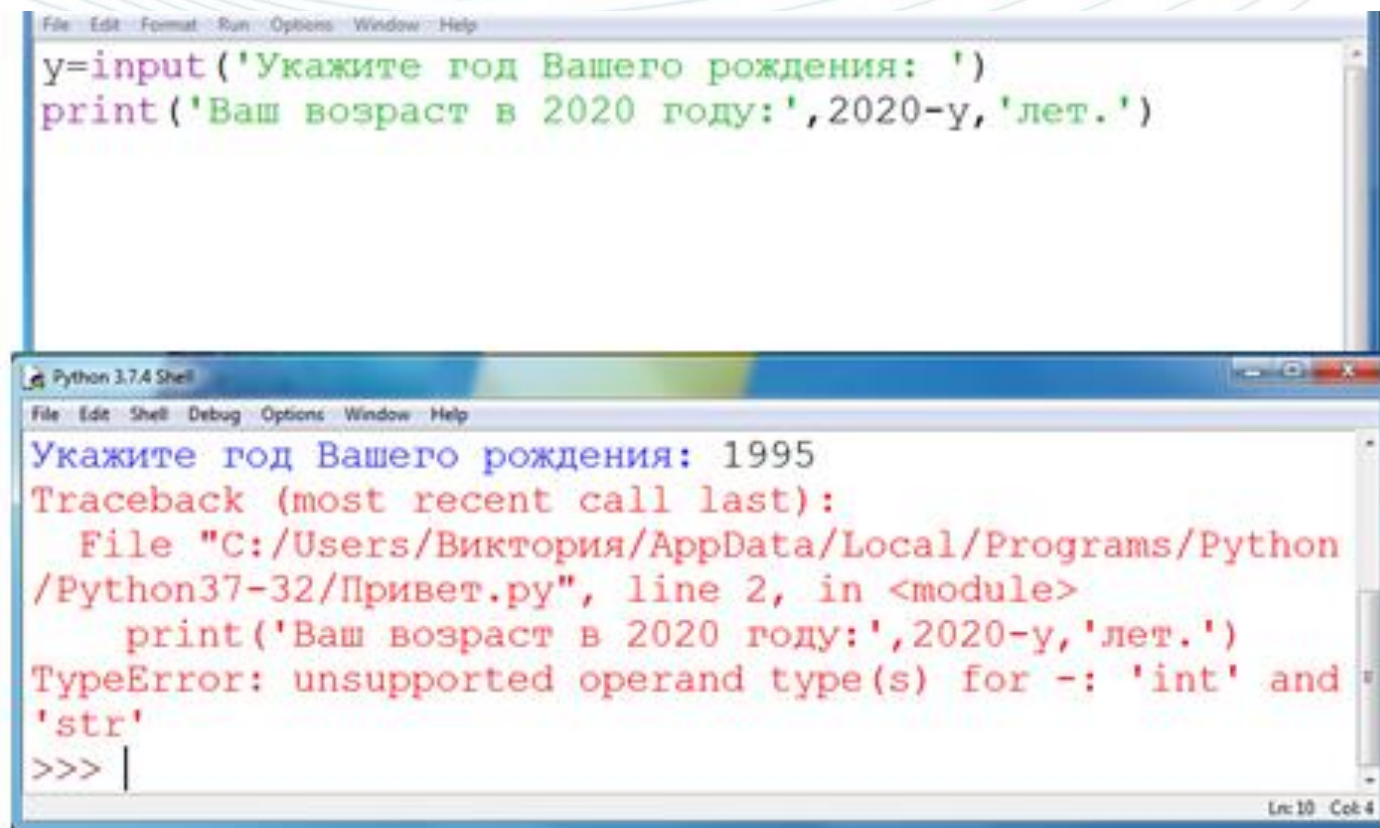
Рассмотрим программу вычисления возраста пользователя в 2020 году. После введения пользователем года своего рождения, программа должна произвести необходимые вычисления и вывести на экран сообщение и значение разности между числом 2020 и введенным значением.

A screenshot of a Python IDE window. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains two lines of Python code: `y=input('Укажите год Вашего рождения: ')` and `print('Ваш возраст в 2020 году:',2020-y,'лет.')`. The code is color-coded: 'y=' is purple, 'input' is green, and the strings are in green. The window has a vertical scrollbar on the right side.

```
File Edit Format Run Options Window Help
y=input('Укажите год Вашего рождения: ')
print('Ваш возраст в 2020 году:',2020-y,'лет.')
```

Однако, в результате работы программы интерпретатор сообщает об ошибке, которая называется «ошибка типов».

Переменная `Y` в нашей программе имеет строковый тип, а значит, выражение `2020-Y` не имеет смысла, поскольку нельзя вычесть из числа строку.



The image shows two overlapping windows from a Python 3.7.4 Shell. The top window displays the source code of a program: `y=input('Укажите год Вашего рождения: ')` and `print('Ваш возраст в 2020 году:',2020-y,'лет.')`. The bottom window shows the execution of this code. It prompts the user with 'Укажите год Вашего рождения: 1995'. A `Traceback` error is shown, indicating a `TypeError: unsupported operand type(s) for -: 'int' and 'str'`. The error message points to line 2 of the file `C:/Users/Виктория/AppData/Local/Programs/Python/Python37-32/Привет.py`. The prompt `>>> |` is visible at the bottom of the shell window.

```
File Edit Format Run Options Window Help
y=input('Укажите год Вашего рождения: ')
print('Ваш возраст в 2020 году:',2020-y,'лет.')
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Укажите год Вашего рождения: 1995
Traceback (most recent call last):
  File "C:/Users/Виктория/AppData/Local/Programs/Python
/Python37-32/Привет.py", line 2, in <module>
    print('Ваш возраст в 2020 году:',2020-y,'лет.')
```

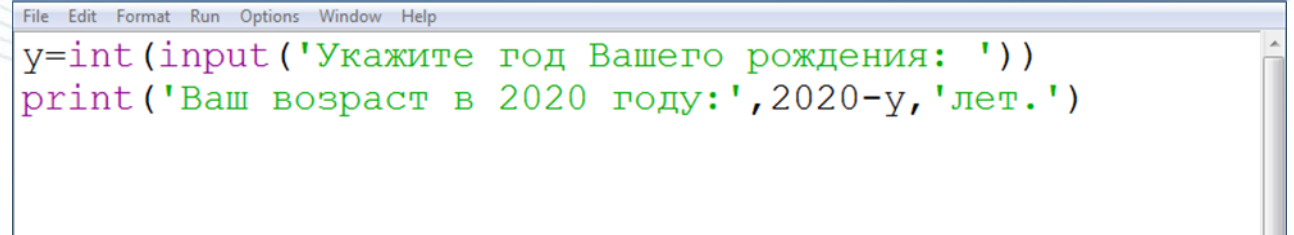
`TypeError: unsupported operand type(s) for -: 'int' and 'str'`

```
>>> |
```

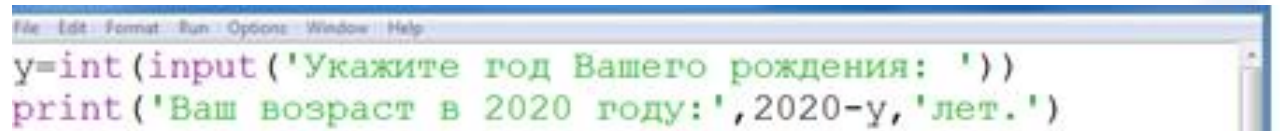
Ln: 10 Col: 4

Чтобы устранить эту ошибку, исправим первую инструкцию программы, изменив тип переменной Y со строкового на целочисленный тип...

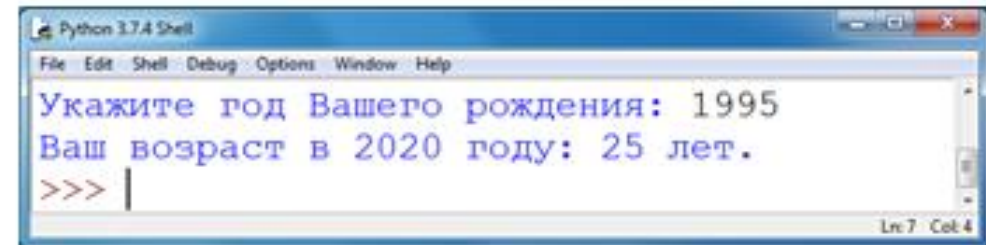
Выполнив тестирование программы, убедимся в том, что теперь она работает правильно.



```
File Edit Format Run Options Window Help
y=int(input('Укажите год Вашего рождения: '))
print('Ваш возраст в 2020 году:',2020-y,'лет.')
```



```
File Edit Format Run Options Window Help
y=int(input('Укажите год Вашего рождения: '))
print('Ваш возраст в 2020 году:',2020-y,'лет.')
```



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Укажите год Вашего рождения: 1995
Ваш возраст в 2020 году: 25 лет.
>>> |
```



Рассмотрим теперь более детально работу с числами и числовыми выражениями.

Синтаксис числовых выражений в языке Python привычен: операции сложения, вычитания, умножения и вещественного деления обозначаются так же, как и в других языках программирования; для группировки используются скобки. Возведение в степень в языке Python обозначается двумя звёздочками. Для целочисленного деления используются следующие обозначения: два знака «слэш» - целая часть от деления, знак «процент» - остаток от деления.

## Математические операции

Операция	Описание
+	сложение
-	вычитание
*	умножение
/	вещественное деление
**	возведение в степень
//	целая часть от деления
%	остаток от деления

Кроме того, есть встроенные математические функции: Abs возвращает модуль числа, указанного в скобках. Round округляет вещественное значение. В скобках функции Round указываются два аргумента: округляемое вещественное число и необходимое количество знаков после запятой.

### *Встроенные функции*

Функция	Описание
abs (<x>)	Абсолютная величина x
round (<x>, <n>)	Округление x до n знаков после запятой

Рассмотрим несколько примеров математических вычислений, выполненных для удобства в интерактивном режиме. Заметим при этом, что среду IDLE можно использовать в качестве калькулятора. Для получения значения некоторого выражения достаточно записать это выражение после приглашения системы, и на следующей строке будет выведен результат.

Посмотрите, пожалуйста, на первый пример. В этом выражении 144 возводится в степень «одна вторая», что математически равносильно извлечению квадратного корня.

## Примеры операций с числами



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>> 144**(1/2)
12.0
>>> 4-1.2
2.8
>>> 6/2
3.0
>>> abs(-7)
7
>>> round(2/3,2)
0.67
>>> |
```



Таким образом можно извлекать корни, используя дробный показатель степени, не прибегая к помощи дополнительных возможностей, о которых речь пойдет позже.

Далее, обратим внимание, что операции над числами при необходимости автоматически конвертируют целочисленный тип в вещественный, т.е., во-первых, конфликта типов не возникает, а во-вторых, выражение, в котором присутствует хотя бы одно вещественное число или есть операция деления или извлечения корня, всегда будет иметь значение вещественного типа.

И, наконец, пример работы двух встроенных функций: взятия по модулю и округления.

## Примеры операций с числами



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>> 144**(1/2)
12.0
>>> 4-1.2
2.8
>>> 6/2
3.0
>>> abs(-7)
7
>>> round(2/3,2)
0.67
>>> |
```

Кроме стандартным образом оформленных математических операций, в языке Python можно использовать краткую запись некоторых арифметических выражений.

Так, инструкция, означающая, что прежнее значение переменной  $A$  должно быть увеличено на  $N$ , может быть записана в виде  $A += N$ . Аналогичен синтаксис для всех подобных операций, если новое значение переменной вычисляется по старому значению за одно арифметическое действие.

### Специальные арифметические операции в Python

Операция	Краткая форма записи
$a = a + n$	$a += n$
$a = a - n$	$a -= n$
$a = a * n$	$a *= n$
$a = a / n$	$a /= n$
$a = a ** n$	$a **= n$
$a = a // n$	$a //= n$
$a = a \% n$	$a \% = n$

Помимо стандартных арифметических операций, при работе с числами мы можем использовать математические функции, встроенные в модули языка Python.

Богатая стандартная библиотека модулей является одной из привлекательных сторон языка программирования Python.

Для того, чтобы получить возможность использовать функции модулей, эти модули необходимо импортировать из библиотеки в свою программу.

Это действие осуществляется инструкцией `import`, помещаемой в начало программы.

## *Модули функций стандартной библиотеки Python*

## *Модули функций стандартной библиотеки Python*

```
import <имя_модуля>
```



Два основных модуля, с которыми мы будем работать, это модуль random и модуль math.

Пишем служебное слово import , далее указываем имя подключаемого модуля.

Если необходимо подключить несколько модулей в одной программе, их можно указать в одной инструкции import через запятую.

## *Модули функций стандартной библиотеки Python*

```
import <имя_модуля>
```

### *Примеры:*

```
import random  
import math
```

## *Модули функций стандартной библиотеки Python*

```
import <имя_модуля>
```

### *Примеры:*

```
import random  
import math  
  
import math, random
```

Рассмотрим основные функции модулей `random` и `math`.

Модуль `random` управляет генерацией случайных чисел. Его основные функции:

- `Random` (в скобках ничего не указывается, но сами скобки написать необходимо): генерирует случайное вещественное число от 0 (включительно) до 1 (не включительно).
- `randint`: возвращает случайное целое число из диапазона с указанными в скобках границами.

### Некоторые функции модуля `random`

Функция	Функция генерирует:
<code>random()</code>	случайное вещественное число от 0.0 (включительно) до 1.0 (не включительно)
<code>randint(&lt;A&gt;, &lt;B&gt;)</code>	случайное целое число из диапазона от <A> до <B> включительно
<code>randrange(&lt;параметры&gt;)</code>	случайное целое число из набора чисел

randrange: возвращает случайное число из набора целых чисел. Набор этих чисел генерируется в соответствии с указанными параметрами по правилам функции range. Возможности функции range будем обсуждать позже.

## Некоторые функции модуля random

Функция	Функция генерирует:
<code>random()</code>	случайное вещественное число от 0.0 (включительно) до 1.0 (не включительно)
<code>randint(&lt;A&gt;, &lt;B&gt;)</code>	случайное целое число из диапазона от <A> до <B> включительно
<code>randrange(&lt;параметры&gt;)</code>	случайное целое число из набора чисел



Встроенный в стандартную библиотеку языка Python модуль `math` предоставляет набор функций для выполнения математических, тригонометрических и логарифмических операций. Например, возведение в степень, нахождение арифметического квадратного корня, факториала натурального числа, тригонометрических функций (аргументы которых должны быть указаны в радианах), логарифма, значения числа  $\pi$  и многие другие операции.

## Некоторые функции модуля `math`

Функция	Описание
<code>pow(&lt;n&gt;, &lt;p&gt;)</code>	$n^p$
<code>sqrt(&lt;a&gt;)</code>	$\sqrt{a}$
<code>factorial(&lt;n&gt;)</code>	$n!$
<code>cos(&lt;rad&gt;)</code> , <code>sin(&lt;rad&gt;)</code> и др.	Тригонометрические функции
<code>log(&lt;b&gt;, &lt;a&gt;)</code>	$\log_a b$
<code>pi</code>	$\pi$

Для того чтобы использовать в программе какую-либо функцию из модуля, нужно указать имя модуля, далее через точку имя вызываемой функции и в скобках – аргументы вызываемой функции.

На экране мы видим пример записи корня квадратного из 625.

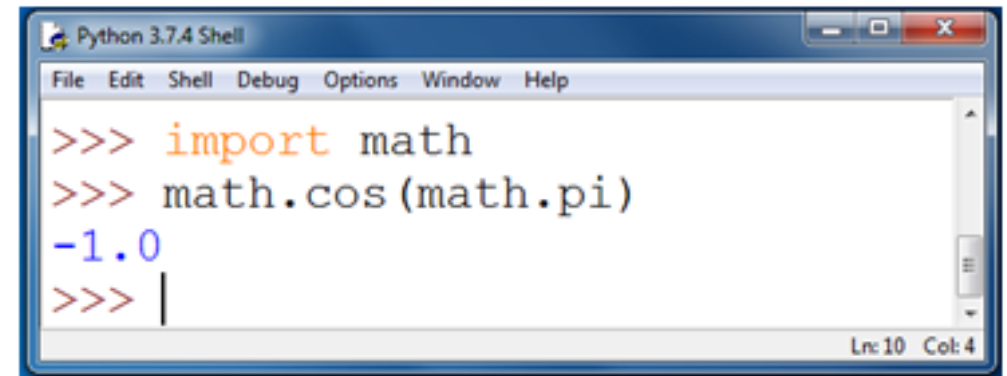
Рассмотрим пример программы для вычисления  $\cos \pi$ . Прежде всего, необходимо импортировать модуль `math`. Теперь можно обращаться к функциям этого модуля: для вызова функции косинуса и для вызова значения числа  $\pi$ . Синтаксис этих записей должен соответствовать следующему формату: `<имя модуля>.<имя функции>`. Причем функция `pi` является аргументом функции косинуса, поэтому помещается в скобки... Результат соответствует ожиданиям:  $\cos \pi = -1$ .

## Обращение к встроенной функции

`<имя_модуля>.<имя_функции> (<аргументы>)`

*Пример:* `math.sqrt(625)`

## Пример вычисления $\cos \pi$



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>> import math
>>> math.cos(math.pi)
-1.0
>>> |
```

Ln: 10 Col: 4

Рассмотрим возможности использования основной функции `random`, генерирующей вещественные числа.

Как уже было сказано, она возвращает случайные значения из интервала от 0 до 1. Однако часто возникает необходимость получить случайное число из другого интервала.

Для получения случайных чисел из интервала от A до B пользуются следующей формулой.

Эта формула масштабирует исходный интервал от 0 до 1 до интервала необходимой длины, т.е. B-A, и смещает его начало в точку A. Тем самым формула обеспечивает получение значений из интервала от A до B.

## *Случайные числа на интервале [ 0, 1 )*

```
random.random()
```

## *Случайные числа на интервале [ a, b )*

```
a+random.random()*(b-a)
```



Для иллюстрации работы генератора случайных чисел, а также указанной формулы, рассмотрим программу проверки основного тригонометрического тождества. Как известно, оно справедливо для любого произвольного вещественного числа. Но в силу периодичности тригонометрических функций достаточно, чтобы аргумент принимал случайные значения из интервала от 0 до  $2\pi$ .

Программу для вычисления суммы квадратов синуса и косинуса для случайного аргумента из интервала от 0 до  $2\pi$  вы видите на экране. Она включает в себя 5 инструкций. Прежде всего, импортируются модуль математических функций и модуль random.

Далее генерируется случайное число X в интервале от 0 до  $2\pi$ . Для этого используется функция random и формула масштабирования интервала с коэффициентом  $2\pi$ . Затем вычисляется значение Y суммы квадратов синуса и косинуса случайного аргумента X. Значения X и Y выводятся на экран.

Запустив эту программу на исполнение несколько раз, убеждаемся, что при различных значениях X значение Y всегда равно единице.

*Основное тригонометрическое тождество:*

$$\sin^2 x + \cos^2 x = 1$$

```
import math, random
```

```
x=random.random()*(2*math.pi)
y=math.sin(x)**2+math.cos(x)**2
print('x=',x)
print('y=',y)
```

```
x= 2.6455963763772203
```

```
y= 1.0
```

```
>>>
```

```
= RESTART: C:/Users/Виктория/AppData/Local/Programs/Python/Python37-32/1.py =
```

```
x= 4.30511377664887
```

```
y= 1.0
```

```
>>>
```

```
= RESTART: C:/Users/Виктория/AppData/Local/Programs/Python/Python37-32/1.py =
```

```
x= 2.120175930514672
```

```
y= 1.0
```

```
>>> |
```

# В оформлении кода на Питоне очень важны пробелы (особенно отступы) и переходы на новую строку!

- Строка кода:
  - ☐ Переход на новую строку (без точки с запятой) начинает новую строку кода.
  - ☐ Если строку кода нужно разбить на две строки, то используется символ `"""` который экранирует следующий за ним переход на следующую строку
- Выделение блока кода:
  - ☐ В Python для выделения блока кода не нужны фигурные скобки (и их аналоги).
  - ☐ Новая строка с большим отступом начинает новый блок (применяются отступы, кратные 4 пробелам, во многих средах разработки табуляция (tab) эквивалентна печати 4х пробелов).
  - ☐ Во многих случаях новый блок предваряет двоеточие (например, в циклах, ветвлении и объявлении функций).
  - ☐ Новая строка с меньшим отступом находится вне предыдущего блока и, таким образом, заканчивает его.

```
# так оформляется комментарий
# объявление целочисленной переменной:
my_value = 7 # точка с запятой в конце не нужна (но допустима)!
# объявление списка (динамического массива), состоящего из строк:
text_strings = ["one", "two", "three"]

if my_value > 10: # начало блока if
    print("Значение больше пяти") # блок оформляется отступом (фигурные скобки или их анало
else:
    print("Значение меньше пяти")
    for i in text_strings: # цикл по итерируемому объекту
        print(f"Текущее значение {i}") # вывод переменной при помощи форматированной строки
    print("Цикл окончен!")
```

Текущее значение one

Текущее значение two

Текущее значение three

Цикл окончен!



Итак, мы рассмотрели:

- возможности использования инструкции ввода;
- основные арифметические операции в языке Python;
- приемы работы с числами и числовыми выражениями;
- функции двух модулей, входящих в стандартную библиотеку Python, и способ обращения к этим функциям;
- примеры математических вычислений.

# Python