

---

# Язык программирования Python

---

С.В. Лемешевский  
([sergey.lemeshevsky@gmail.com](mailto:sergey.lemeshevsky@gmail.com))

Институт математики НАН Беларуси

Feb 20, 2020



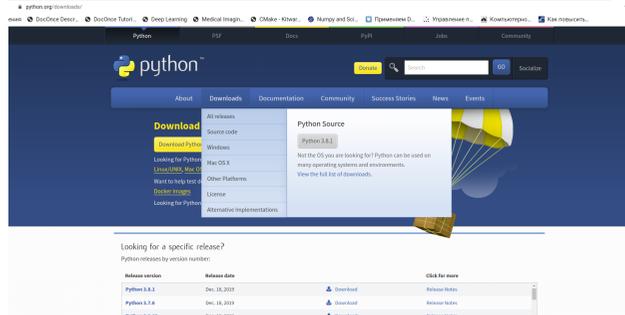
## 1.1. Установка

### 1.1.1. Версии Python

На сегодняшний день существуют две версии Python — это Python 2 и Python 3, у них отсутствует полная совместимость друг с другом. На данный момент вторая версия Python ещё широко используется, но, судя по изменениям, которые происходят, со временем, она останется только для запуска старого кода. Мы будем Python 3, и, в дальнейшем, если где-то будет встречаться слово Python, то под ним следует понимать Python 3. Случаи применения Python 2 будут специально оговариваться.

### 1.1.2. Установка Python

Для установки интерпретатора Python на ваш компьютер, первое, что нужно сделать — это скачать дистрибутив. Загрузить его можно с официального сайта, перейдя по ссылке <https://www.python.org/downloads/>



**Установка Python в Windows.** Для операционной системы Windows дистрибутив распространяется либо в виде исполняемого файла, либо в виде архивного файла.

Version	Operating System	Description	MD5 Sum	File Size	CPY
Colpined source tarball	Source release		c215a27f5a7784773017817c502b2ed	23978369	SG
XZ compressed source tarball	Source release		138b59647f9b0f950c026480ca0c57	17823408	SG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	4150f96531246da16c118030a4c550a3	20951411	SG
Windows help file	Windows		fb2b04c238f16c38b9f1825c464c2	8486993	SG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	4489185712153d4649300c522b211861	8015540	SG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	3e4c420f880be0e428c912674b061	2754380	SG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	662961733cc947638a7332789b8c145	1363800	SG
Windows x86 embeddable zip file	Windows		9805745a7e623b0c546f4644a00734	7143308	SG
Windows x86 executable installer	Windows		294c70e746bc082313c3e0c0bbf79	2646128	SG
Windows x86 web-based installer	Windows		d2170b0dca544e7a768e32b6b020951	1325432	SG

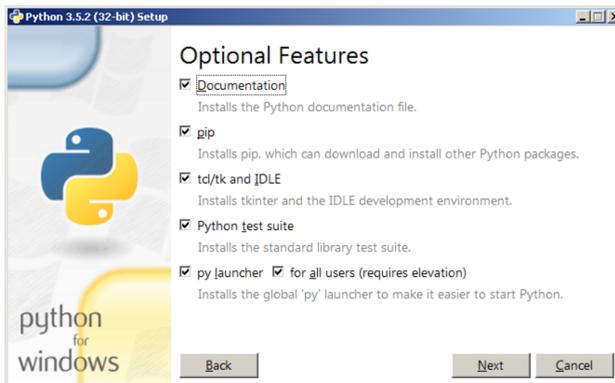
## Порядок установки.

1. Запустите скачанный установочный файл.
2. Выберите способ установки.



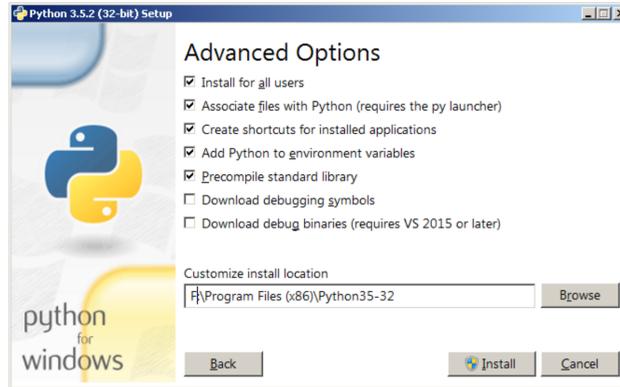
В данном окне предлагается два варианта *Install Now* и *Customize installation*. При выборе *Install Now*, Python установится в папку по указанному пути. Помимо самого интерпретатора будет установлен IDLE (интегрированная среда разработки), pip (пакетный менеджер) и документация, а также будут созданы соответствующие ярлыки и установлены связи файлов, имеющие расширение .py с интерпретатором Python. *Customize installation* – это вариант настраиваемой установки. Опция *Add python 3. to PATH* нужна для того, чтобы появилась возможность запускать интерпретатор без указания полного пути до исполняемого файла при работе в командной строке.

1. Отметьте необходимые опции установки (доступно при выборе *Customize installation*)



На этом шаге нам предлагается отметить дополнения, устанавливаемые вместе с интерпретатором Python . Рекомендуем выбрать все опции:

- Documentation – установка документаций.
- pip – установка пакетного менеджера pip.
- tcl/tk and IDLE – установка интегрированной среды разработки (IDLE) и библиотеки для построения графического интерфейса (tkinter).
- Выберите место установки (доступно при выборе *Customize installation* )

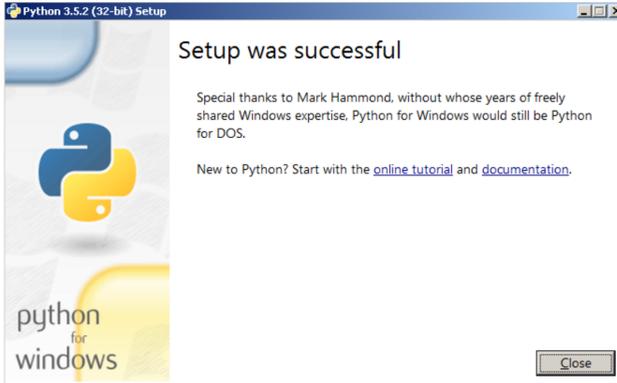


Помимо указания пути, данное окно позволяет внести дополнительные изменения в процесс установки с помощью опций:

- **Install for all users** – Установить для всех пользователей. Если не выбрать данную опцию, то будет предложен вариант инсталляции в папку пользователя, устанавливающего интерпретатор.
- **Associate files with Python** – Связать файлы, имеющие расширение `.py`, с Python. При выборе данной опции будут внесены изменения в Windows, позволяющие запускать Python скрипты по двойному щелчку мыши.
- **Create shortcuts for installed applications** – Создать ярлыки для запуска приложений.
- **Add Python to environment variables** – Добавить пути до интерпретатора Python в переменную PATH.
- **Precompile standard library** – Провести прекомпиляцию стандартной библиотеки.

Последние два пункта (**Download debugging symbols**, **Download debug binaries**) связаны с загрузкой компонентов для отладки, их мы устанавливать не будем.

1. После успешной установки вас ждет следующее сообщение.



**Установка Python в Linux.** Чаще всего интерпретатор Python уже входит в состав дистрибутива. Это можно проверить набрав в терминале

---

```
Terminal
```

---

```
Terminal> python
```

---

ИЛИ

---

```
Terminal
```

---

```
Terminal> python3
```

---

В первом случае, вы запустите Python 2 во втором – Python 3. В будущем, скорее всего, во всех дистрибутивах Linux, включающих Python, будет входить только третья версия. Если у вас, при попытке запустить Python, выдается сообщение о том, что он не установлен, или установлен, но не тот, что вы хотите, то у вас есть два пути: а) собрать Python из исходников; б) взять из репозитория.

Например, для установки из репозитория в Ubuntu воспользуйтесь командой:

---

```
Terminal
```

---

```
Terminal> sudo apt-get install python3
```

---

### 1.1.3. Установка Anaconda

Для удобства запуска примеров и изучения языка Python , советуем установить на свой ПК пакет Anaconda . Этот пакет включает в себя интерпретатор языка Python (есть версии 2 и 3), набор наиболее часто используемых библиотек и удобную среду разработки и исполнения, запускаемую в браузере.

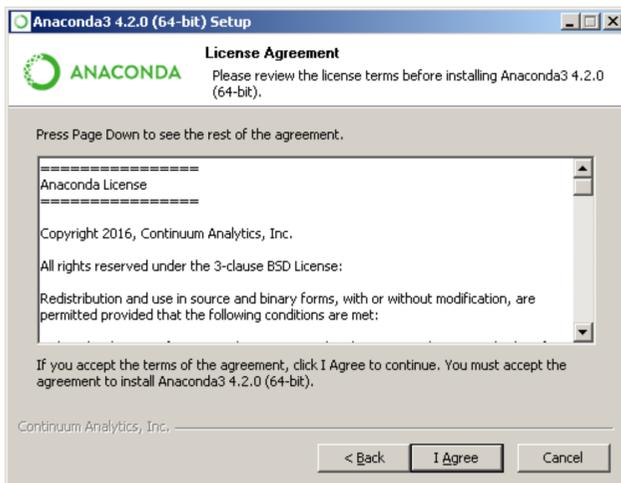
Для установки этого пакета, предварительно нужно скачать дистрибутив <https://www.continuum.io/downloads>.

Есть варианты под Windows , Linux и Mac OS .

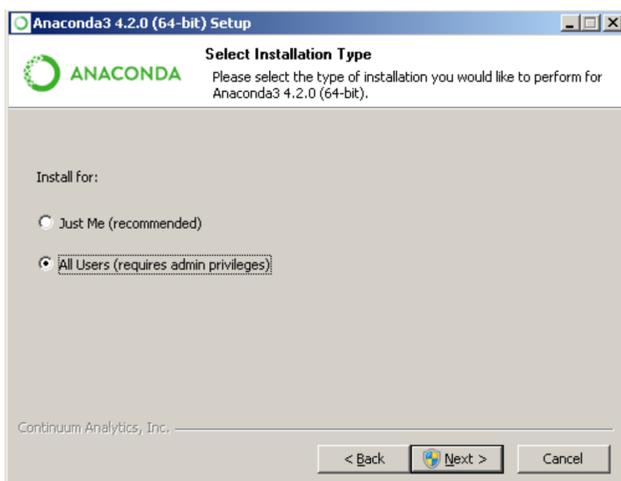
**Установка Anaconda в Windows.** 1. Запустите скачанный инсталлятор. В первом появившемся окне необходимо нажать «Next».



2. Далее следует принять лицензионное соглашение.

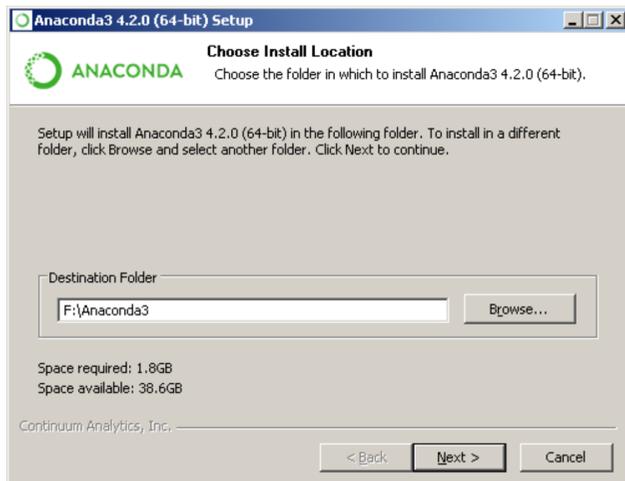


### 3. Выберите одну из опций установки:

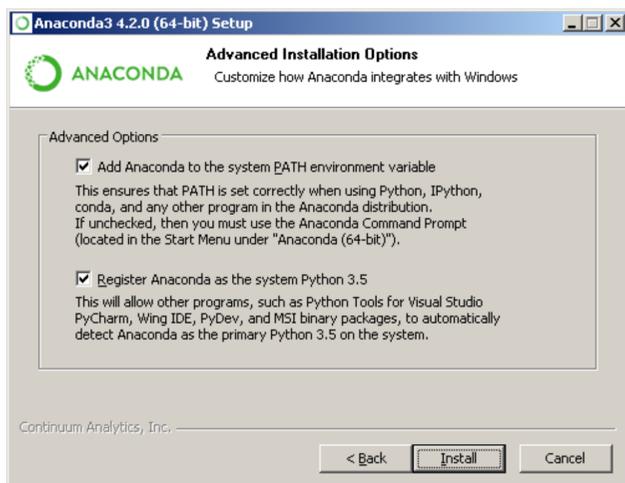


- Just Me – только для пользователя, запустившего установку;
- All Users – для всех пользователей.

### 4. Укажите путь, по которому будет установлена Anaconda.



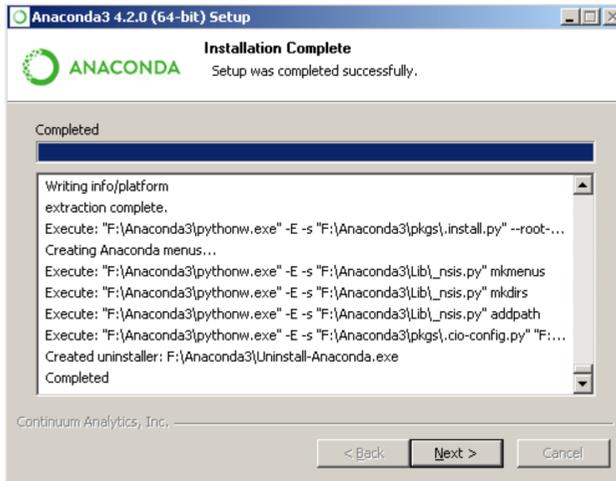
## 5. Укажите дополнительные опции:



- Add Anaconda to the system PATH environment variable – добавить Anaconda в системную переменную PATH;
- Register Anaconda as the system Python 3 – использовать Anaconda, как интерпретатор Python 3 по умолчанию.

Для начала установки нажмите на кнопку «Install».

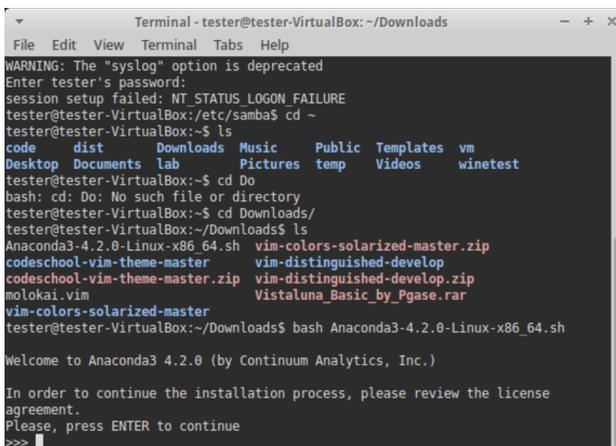
5. После этого будет произведена установка Anaconda на ваш компьютер.



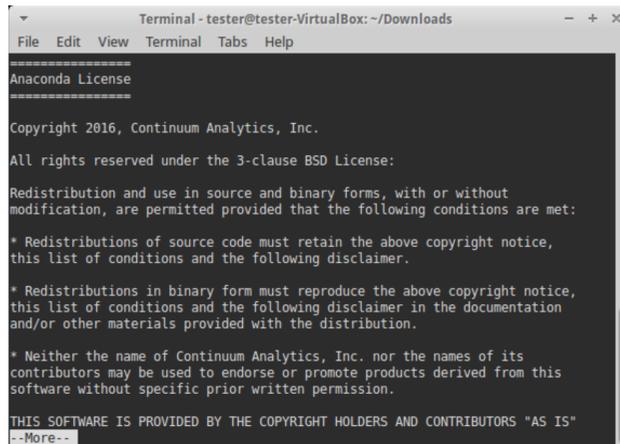
**Установка Anaconda в Linux.** Скачайте дистрибутив Anaconda для Linux, он будет иметь расширение `.sh`, и запустите установку командой:

```
Terminal
Terminal> bash имя_дистрибутива.sh
```

В результате вы увидите приглашение к установке. Для продолжения процессе нажмите «Enter».



2. Прочитайте лицензионное соглашение, его нужно пролистать до конца.



```
Terminal - tester@tester-VirtualBox: ~/Downloads
File Edit View Terminal Tabs Help
=====
Anaconda License
=====
Copyright 2016, Continuum Analytics, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

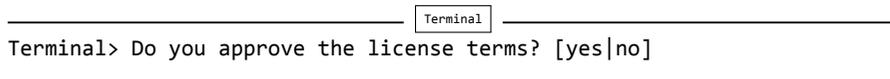
* Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

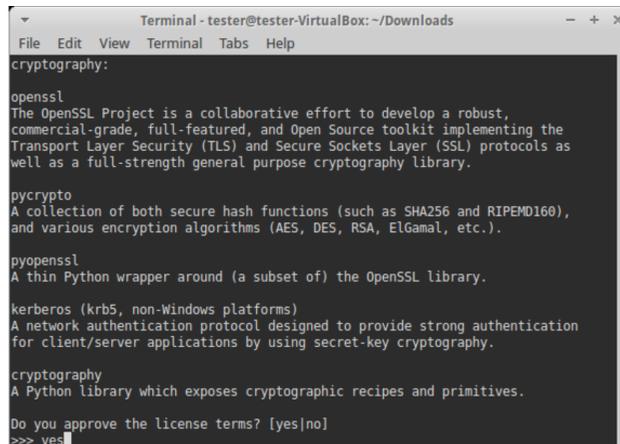
* Neither the name of Continuum Analytics, Inc. nor the names of its
contributors may be used to endorse or promote products derived from this
software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
--More--
```

Согласитесь с ним, для этого требуется набрать в командной строке `yes`, в ответе на вопрос инсталлятора:



```
Terminal
Terminal> Do you approve the license terms? [yes|no]
```



```
Terminal - tester@tester-VirtualBox: ~/Downloads
File Edit View Terminal Tabs Help
cryptography:
openssl
The OpenSSL Project is a collaborative effort to develop a robust,
commercial-grade, full-featured, and Open Source toolkit implementing the
Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols as
well as a full-strength general purpose cryptography library.

pycrypto
A collection of both secure hash functions (such as SHA256 and RIPEMD160),
and various encryption algorithms (AES, DES, RSA, ElGamal, etc.).

pyopenssl
A thin Python wrapper around (a subset of) the OpenSSL library.

kerberos (krb5, non-Windows platforms)
A network authentication protocol designed to provide strong authentication
for client/server applications by using secret-key cryptography.

cryptography
A Python library which exposes cryptographic recipes and primitives.

Do you approve the license terms? [yes|no]
>>> yes
```

3. Выберите место установки. Можно выбрать один из следующих вариантов:

```

Terminal - tester@tester-VirtualBox: ~/Downloads
File Edit View Terminal Tabs Help
A collection of both secure hash functions (such as SHA256 and RIPEMD160),
and various encryption algorithms (AES, DES, RSA, ElGamal, etc.).

pyopenssl
A thin Python wrapper around (a subset of) the OpenSSL library.

kerberos (krb5, non-Windows platforms)
A network authentication protocol designed to provide strong authentication
for client/server applications by using secret-key cryptography.

cryptography
A Python library which exposes cryptographic recipes and primitives.

Do you approve the license terms? [yes|no]
>>> yes

Anaconda3 will now be installed into this location:
/home/tester/anaconda3

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[/home/tester/anaconda3] >>>

```

- Press ENTER to confirm the location - нажмите ENTER для принятия предложенного пути установки. Путь по умолчанию для моей машины: /home/tester/anaconda3, он представлен чуть выше данного меню.
- Press CTRL-C to abort the installation - нажмите CTRL-C для отмены установки.
- Or specify a different location below - или укажите другой путь в строке ниже.

Нажмите «ENTER».

4. После этого начнется установка.

```

Terminal - tester@tester-VirtualBox: ~/Downloads
File Edit View Terminal Tabs Help
installing: xz-5.2.2-0 ...
installing: yaml-0.1.6-0 ...
installing: zeromq-4.1.4-0 ...
installing: zlib-1.2.8-3 ...
installing: anaconda-4.2.0-mp11py35_0 ...
installing: ruamel_yaml-0.11.14-py35_0 ...
installing: conda-4.2.9-py35_0 ...
installing: conda-build-2.0.2-py35_0 ...
Python 3.5.2 :: Continuum Analytics, Inc.
creating default environment...
Installation finished.
Do you wish the installer to prepend the Anaconda3 install location
to PATH in your /home/tester/.bashrc ? [yes|no]
[no] >>>
You may wish to edit your .bashrc or prepend the Anaconda3 install location:

$ export PATH=/home/tester/anaconda3/bin:$PATH

Thank you for installing Anaconda3!

Share your notebooks and packages on Anaconda Cloud!
Sign up for free: https://anaconda.org

tester@tester-VirtualBox: ~/Downloads$

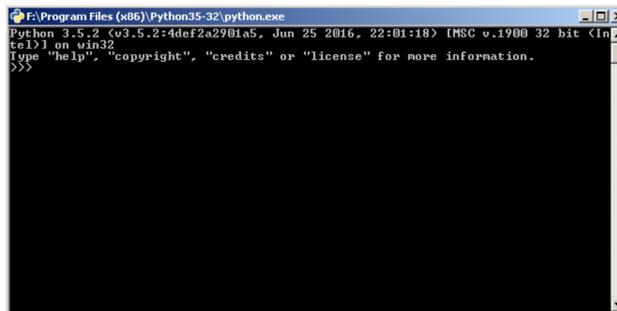
```

### 1.1.4. Проверка работоспособности

Теперь проверим работоспособность всего того, что мы установили.

**Проверка интерпретатора Python.** Для начала протестируем интерпретатор в командном режиме. Если вы работаете в Windows, то нажмите сочетание Win+R и в появившемся окне введите python. В Linux откройте окно терминала и в нем введите python3 (или python).

В результате Python запустится в командном режиме, выглядеть это будет примерно так (картинка приведена для Windows, в Linux результат будет аналогичным):



```
F:\Program Files (x86)\Python35-32\python.exe
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

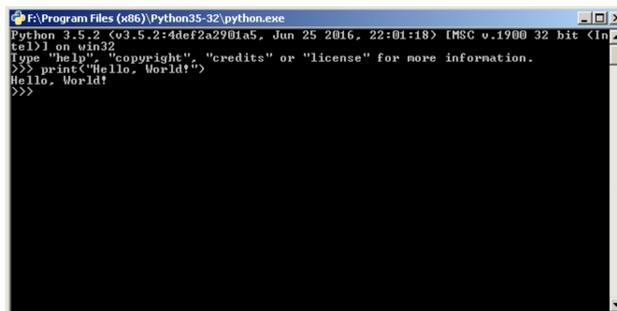
В окне введите:

---

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

---

Результат должен быть следующий:

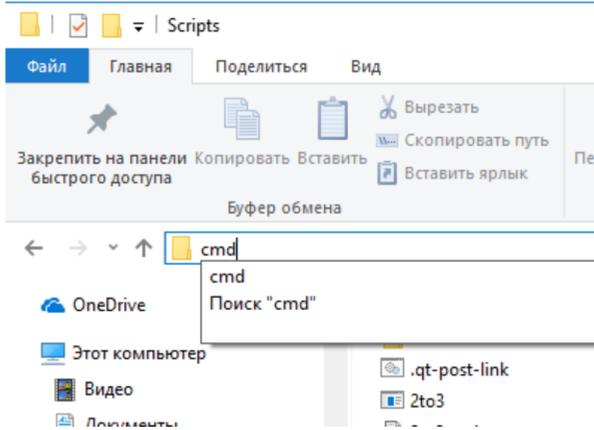


```
F:\Program Files (x86)\Python35-32\python.exe
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>>
```

**Проверка Anaconda.** Здесь и далее будем считать, что пакет Anaconda установлен в Windows, в папку `C:\Anaconda3`, в Linux, вы его можете найти в каталоге, который выбрали при установке.

Перейдите в папку `Scripts` и введите в командной строке:

```
Terminal  
Terminal> ipython notebook
```



Если вы находитесь в Windows и открыли папку `C:\Anaconda3\Scripts` через проводник, то для запуска интерпретатора командной строки для этой папки в поле адреса введите `cmd`.

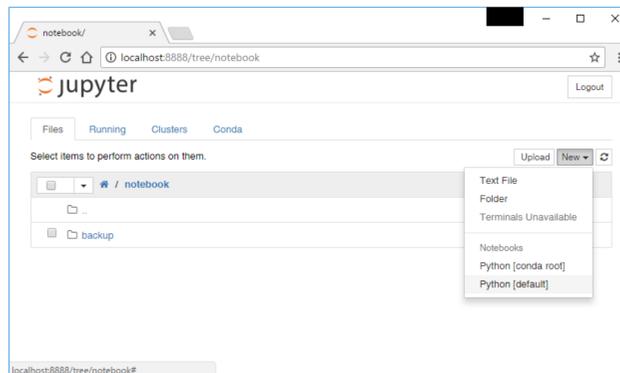
```

C:\Windows\System32\cmd.exe - ipython notebook
Microsoft Windows [Version 6.1.7601]
(C) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Anaconda3\Scripts>ipython notebook
[TerminalPythonApp] WARNING | Subcommand 'ipython notebook' is deprecated and will
be removed in future versions.
[TerminalPythonApp] WARNING | You likely want to use 'jupyter notebook' in the
future
[W 14:59:24.199 NotebookApp] Unrecognized JSON config file version, assuming ver
sion 1
[I 14:59:29.233 NotebookApp] [nb_conda_kernels] enabled, 1 kernels found
[I 14:59:35.048 NotebookApp] [nb_conda] enabled
[I 14:59:36.811 NotebookApp] \u2713 nbpresent HTML export ENABLED
[W 14:59:36.832 NotebookApp] \u2717 nbpresent PDF export DISABLED: No module nam
ed 'nbpresentpdf'
[I 14:59:38.850 NotebookApp] [nb_anacondacloud] enabled
[I 14:59:38.985 NotebookApp] Serving notebooks from local directory: C:\Anaconda
3\Scripts
[I 14:59:38.985 NotebookApp] 0 active kernels
[I 14:59:38.985 NotebookApp] The Jupyter Notebook is running at: http://localhos
t:8888/
[I 14:59:38.985 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).

```

В результате запустится веб-сервер и среда разработки в браузере.



Создайте ноутбук для разработки, для этого нажмите на кнопку «New» (в правом углу окна) и в появившемся списке выберите Python.

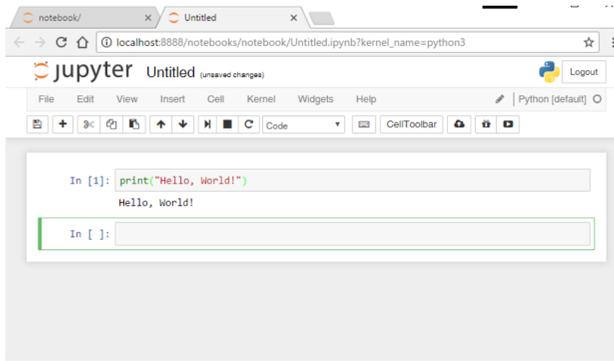
В результате будет создана новая страница в браузере с ноутбуком. Введите в первой ячейке команду

---

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

---

и нажмите Alt+Enter на клавиатуре. Ниже ячейки должна появиться соответствующая надпись.



## 1.2. Запуск программ на Python

Программный код на языке Python можно записать с помощью любого простого текстового редактора, который способен загружать и сохранять текст либо в кодировке ASCII, либо UTF-8. По умолчанию предполагается, что файлы с программным кодом на языке Python сохраняются в кодировке UTF-8, надмножестве кодировки ASCII, с помощью которой можно представить практически любой символ любого национального алфавита. Файлы с программным кодом на языке Python обычно имеют расширение `.py`, хотя в некоторых UNIX-подобных системах (таких как Linux и Mac OS X) некоторые приложения на языке Python не имеют расширения, а программы на языке Python с графическим интерфейсом, в частности в Windows и Mac OS X, обычно имеют расширение `.pyw`. В этой книге все время будет использоваться расширение `.py` для обозначения консольных программ и модулей Python и расширение `.pyw` – для программ с графическим интерфейсом. Все примеры, представленные в книге, не требуют изменений для запуска в любой из платформ, поддерживаемых Python 3.

Язык Python – это *интерпретируемый* язык. Это означает, что помимо непосредственно самой программы, вам необходим специальный инструмент для её запуска. Напомним, что существуют компилируемые и интерпрети-

руемые языки программирования. В первом случае, программа с языка высокого уровня переводится в машинный код для конкретной платформы. В дальнейшем, среди пользователей, она, как правило, распространяется в виде бинарного файла. Для запуска такой программы не нужны дополнительные программные средства (за исключением необходимых библиотек, но эти тонкости выходят за рамки нашего обсуждения). Самыми распространенными языками такого типа являются C++ и C. Программы на интерпретируемых языках, выполняются интерпретатором и распространяются в виде исходного кода.

Python может работать в двух режимах:

- интерактивный;
- пакетный.

### 1.2.1. Интерактивный режим работы

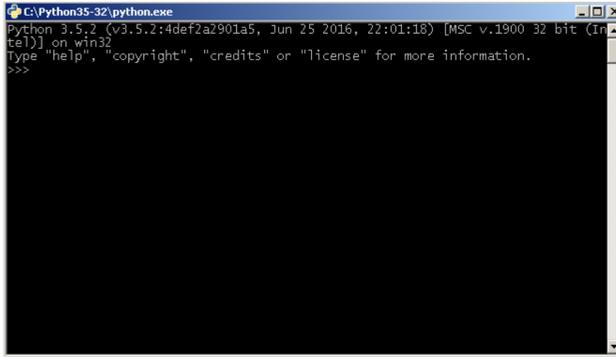
В интерактивный режим можно войти, набрав в командной строке

```
Terminal  
Terminal> python
```

или

```
Terminal  
Terminal> python3
```

В результате Python запустится в интерактивном режиме и будет ожидать ввод команд пользователя.



Если же у вас есть файл с исходным кодом на Python , и вы его хотите запустить, то для этого нужно в командной строке вызвать интерпретатор Python и в качестве аргумента передать ваш файл. Например, для файла с именем test.py процедура запуска будет выглядеть так:

---

```
Terminal> python test.py
```

---

Откройте Python в интерактивном режиме и наберите в нем следующее:

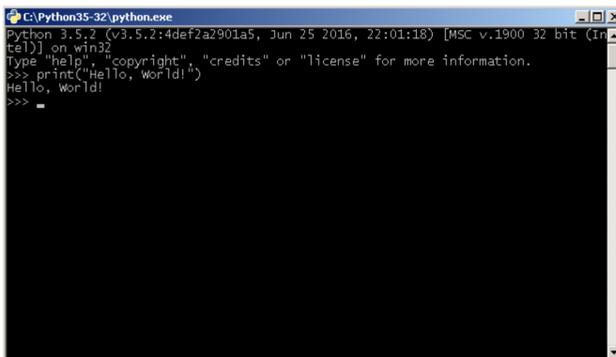
---

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

---

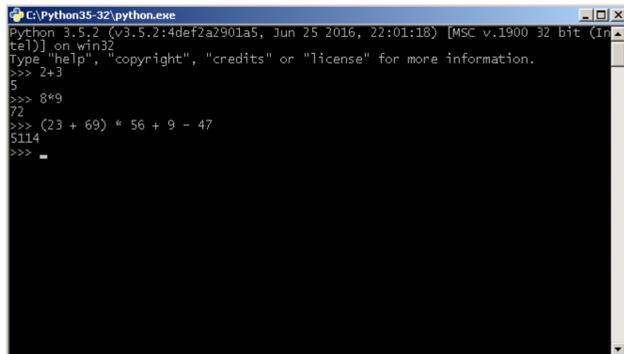
И нажмите ENTER .

В ответ на это интерпретатор выполнит данную строку и отобразит строкой ниже результат своей работы.



Python можно использовать как калькулятор для различных вычислений, а если дополнительно подключить необходимые математические библиотеки, то по своим возможностям он становится практически равным таким пакетам как Matlab, Octave и т.п.

Различные примеры вычислений приведены ниже. Более подробно об арифметических операциях будет рассказано далее.

A screenshot of a Windows command prompt window titled "C:\Python35-32\python.exe". The window shows the Python 3.5.2 shell interface. The prompt is ">>>". The user has entered several commands and their outputs: ">>> 2+3" results in "5"; ">>> 8\*9" results in "72"; ">>> (23 + 69) \* 56 + 9 - 47" results in "5114". The prompt ">>> ." is shown at the bottom, indicating the end of the session.

```
C:\Python35-32\python.exe
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 2+3
5
>>> 8*9
72
>>> (23 + 69) * 56 + 9 - 47
5114
>>> .
```

Для выхода из интерактивного режима, наберите команду

---

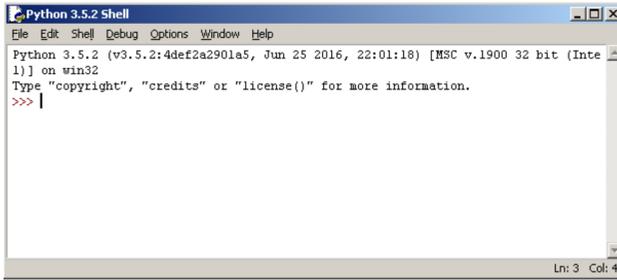
```
exit()
```

---

и нажмите ENTER .

В комплекте вместе с интерпретатором Python идет IDLE (интегрированная среда разработки). По своей сути она подобна интерпретатору, запущенному в интерактивном режиме с расширенным набором возможностей (подсветка синтаксиса, просмотр объектов, отладка и т.п.).

Для запуска IDLE в Windows необходимо перейти в папку Python в меню "Пуск" и найти там ярлык с именем «IDLE (Python 3 XX-bit)».



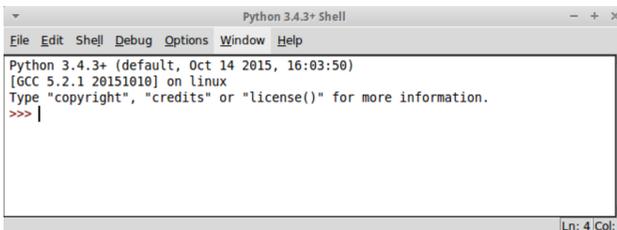
В Linux оболочка IDLE по умолчанию отсутствует, поэтому ее предварительно нужно установить. Для этого, если у вас Ubuntu, введите в командной строке (для Python 3.4):

```
Terminal  
Terminal> sudo apt-get install idle-python3
```

В результате IDLE будет установлен на ваш компьютер. Для запуска оболочки, введите:

```
Terminal  
Terminal> idle-python3.4
```

Ниже представлен внешний вид IDLE в ОС Linux.



### 1.2.2. Пакетный режим работы

Теперь запустим Python в режиме интерпретации файлов с исходным кодом (пакетный режим). Создайте файл с именем `test.py`, откройте его с помощью любого текстового редактора и введите следующий код:

---

```
a = int(input())  
print(a**2)
```

---

Эта программа принимает целое число на вход и выводит его квадрат. Для запуска, наберите в командной строке

---

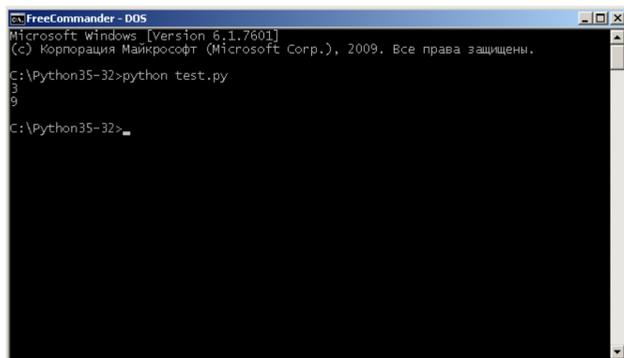
Terminal

---

```
Terminal> python test.py
```

---

Пример работы программы приведен в окне ниже.



```
FreeCommander - DOS  
Microsoft Windows [Version 6.1.7601]  
(C) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.  
C:\Python35-32>python test.py  
3  
9  
C:\Python35-32>
```

Здесь разберем как Python работает с переменными и определим, какие типы данных можно использовать в рамках этого языка. Подробно рассмотрим модель данных Python, а также механизмы создания и изменения значения переменных.

## 2.1. Кратко о типизации языков программирования

Если достаточно формально подходить к вопросу о типизации языка Python, то можно сказать, что он относится к языкам с неявной сильной динамической типизацией.

Неявная типизация означает, что при объявлении переменной вам не нужно указывать её тип, при явной – это делать необходимо. В качестве примера языков с явной типизацией можно привести Java, C++ . Вот как будет выглядеть объявление целочисленной переменной в Java и Python.

- Java:

---

```
int a = 1 ;
```

---

- Python:

---

```
a = 1
```

---

## 2.2. Типы данных в Python

В Python типы данных можно разделить на встроенные в интерпретатор (built-in) и не встроенные, которые можно использовать при импортировании соответствующих модулей.

К основным встроенным типам относятся:

1. None (неопределенное значение переменной)
2. Логические переменные (Boolean Type)
3. Числа (Numeric Type)
  - a. int - целое число
  - b. float - число с плавающей точкой
  - c. complex - комплексное число
4. Списки (Sequence Type)
  - a. list - список
  - b. tuple - кортеж
  - c. range - диапазон
5. Строки (Text Sequence Type)
  - a. str
6. Бинарные списки ( Binary Sequence Types )
  - a. bytes - байты
  - b. bytearray - массивы байт
  - c. memoryview - специальные объекты для доступа к внутренним данным объекта через protocol buffer
7. Множества (Set Types)
  - a. set - множество
  - b. frozenset - неизменяемое множество
8. Словари (Mapping Types)
  - a. dict - словарь

## 2.3. Модель данных

Рассмотрим как создаются объекты в памяти, их устройство, процесс объявления новых переменных и работу операции присваивания.

Для того, чтобы объявить и сразу инициализировать переменную необходимо написать её имя, потом поставить знак равенства и значение, с которым эта переменная будет создана.

Например строка:

---

```
b = 5
```

---

Объявляет переменную `b` и присваивает ей значение 5.

Целочисленное значение 5 в рамках языка Python по сути своей является *объектом*. Объект, в данном случае – это абстракция для представления данных, данные – это числа, списки, строки и т.п. При этом, под *данными* следует понимать как непосредственно сами объекты, так и отношения между ними (об этом чуть позже). Каждый объект имеет три атрибута – это *идентификатор*, *значение* и *тип*.

*Идентификатор* – это уникальный признак объекта, позволяющий отличать объекты друг от друга, а *значение* – непосредственно информация, хранящаяся в памяти, которой управляет интерпретатор.

При инициализации переменной, на уровне интерпретатора, происходит следующее:

- создается целочисленный объект 5 (можно представить, что в этот момент создается ячейка и число 5 «кладется» в эту ячейку);
- данный объект имеет некоторый идентификатор, значение: 5, и тип: целое число;
- посредством оператора `=` создается ссылка между переменной `b` и целочисленным объектом 5 (переменная `b` ссылается на объект 5).

### Замечание

Имя переменной не должно совпадать с ключевыми словами интерпретатора Python . Список ключевых слов можно получить непосредственно в программе, для этого нужно подключить модуль `keyword` и воспользоваться командой `keyword.kwlist`.

```
import keyword
print("Python keywords: ", keyword.kwlist)
```

Проверить является ли идентификатор ключевым словом можно так:

```
>>> keyword.iskeyword("try")
True
```

```
>>> keyword.iskeyword("b")
False
```

Для того, чтобы посмотреть на объект с каким идентификатором ссылается данная переменная, можно использовать функцию `id()`.

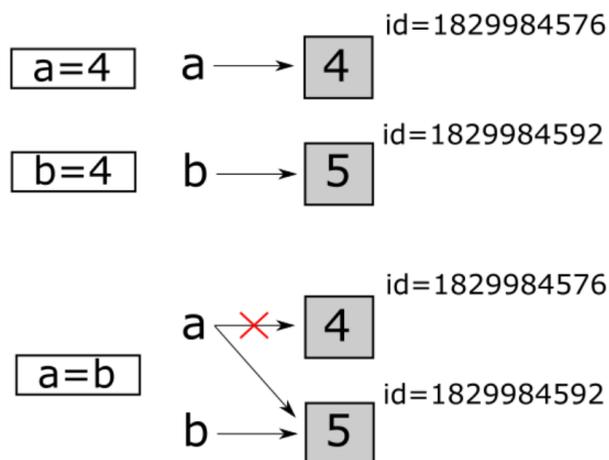
```
>>> a = 4
>>> b = 5
>>> id(a)
1829984576
```

```
>>> id(b)
1829984592
```

```
>>> a = b
>>> id(a)
1829984592
```

Как видно из примера, идентификатор – это некоторое целочисленное значение, посредством которого уникально адресуется объект. Изначально переменная `a` ссылается на объект 4 с идентификатором 1829984576, переменная

b - на объект с id = 1829984592. После выполнения операции присваивания a = b, переменная a стала ссылаться на тот же объект, что и b.



Тип переменной можно определить с помощью функции `type()`. Пример использования приведен ниже.

---

```
>>> a = 10
>>> b = "hello"
>>> c = ( 1 , 2 )
>>> type (a)
< class 'int' >
```

---

```
>>> type (b)
< class 'str' >
```

---

```
>>> type (c)
< class 'tuple' >
```

---

## 2.4. Изменяемые и неизменяемые типы данных

В Python существуют изменяемые и неизменяемые типы.

К неизменяемым (immutable) типам относятся:

- целые числа (int);
- числа с плавающей точкой (float);
- комплексные числа (complex);
- логические переменные (bool);
- кортежи (tuple);
- строки (str);
- неизменяемые множества (frozen set).

К изменяемым (mutable) типам относятся

- списки (list);
- множества (set);
- словари (dict).

Как уже было сказано ранее, при создании переменной, вначале создается объект, который имеет уникальный идентификатор, тип и значение, после этого переменная может ссылаться на созданный объект.

Неизменяемость типа данных означает, что созданный объект больше не изменяется. Например, если мы объявим переменную `k = 15`, то будет создан объект со значением 15, типа `int` и идентификатором, который можно узнать с помощью функции `id()`.

---

```
>>> k = 15
>>> id(k)
1672501744
```

---

```
>>> type(k)
< class 'int' >
```

---

Объект с `id = 1672501744` будет иметь значение 15 и изменить его уже нельзя. Если тип данных изменяемый, то можно менять значение объекта.

Например, создадим список `[1, 2]`, а потом заменим второй элемент на 3.

---

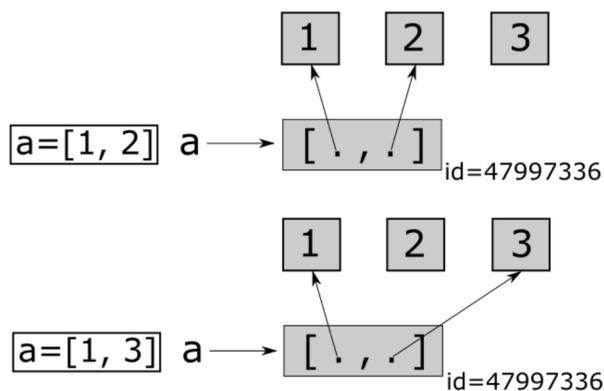
```
>>> a = [1, 2]
>>> id(a)
47997336
```

---

```
>>> a[1] = 3
>>> a
[ 1 , 3 ]
```

```
>>> id(a)
47997336
```

Как видно, объект на который ссылается переменная `a`, был изменен. Это можно проиллюстрировать следующим рисунком.



В рассмотренном случае, в качестве данных списка, выступают не объекты, а отношения между объектами. Т.е. в переменной `a` хранятся ссылки на объекты содержащие числа 1 и 3, а не непосредственно сами эти числа.



### 3.1. Арифметические операции с целыми и вещественными числами

Все эксперименты будем проводить в Python, запущенном в интерактивном режиме.

**Сложение.** Складывать можно непосредственно сами числа ...

---

```
>>> 3 + 2
5
```

---

либо переменные, но они должны предварительно быть проинициализированы.

---

```
>>> a = 3
>>> b = 2
>>> a + b
5
```

---

Результат операции сложения можно присвоить другой переменной...

---

```
>>> a = 3
>>> b = 2
>>> c = a + b
```

```
>>> print (c)
5
```

---

либо ей же самой, в таком случае можно использовать полную или сокращенную запись, полная выглядит так:

---

```
>>> a = 3
>>> b = 2
>>> a = a + b
>>> print (a)
5
```

---

сокращенная так:

---

```
>>> a = 3
>>> b = 2
>>> a += b
>>> print (a)
5
```

---

Все перечисленные выше варианты использования операции сложения могут быть применены для всех ниже следующих операций.

### **Вычитание.**

```
>>> 4 - 2
2
```

---

```
>>> a = 5
>>> b = 7
>>> a - b
-2
```

---

### **Деление.**

```
>>> 9 / 3
3.0
```

---

```
>>> a = 7
>>> b = 4
>>> a / b
1.75
```

---

### Получение целой части от деления.

---

```
>>> 9 // 3
3
```

---

```
>>> a = 7
>>> b = 4
>>> a // b
1
```

---

### Получение дробной части от деления.

---

```
>>> 9 % 5
4
```

---

```
>>> a = 7
>>> b = 4
>>> a % b
7
```

---

### Возведение в степень.

---

```
>>> 5**4
625
```

---

```
>>> a = 4
>>> b = 3
>>> a**b
64
```

---

## 3.2. Работа с комплексными числами

Для создания комплексного числа можно использовать функцию `complex(a, b)`, в которую, в качестве первого аргумента, передается действительная часть, в качестве второго – мнимая. Либо записать число в виде  $a+bj$ .

Рассмотрим несколько примеров.

Создание комплексного числа.

---

```
>>> z = 1+2j
>>> print(z)
(1+2j)
```

---

---

```
>>> x = complex(3, 2)
>>> print(x)
(3+2j)
```

---

Комплексные числа можно складывать, вычитать, умножать, делить и возводить в степень.

---

```
>>> x+z
(4+4j)
```

---

---

```
>>> x-z
(2+0j)
```

---

---

```
>>> x*z
(-1+8j)
```

---

---

```
>>> x/z
(1.4-0.8j)
```

---

---

```
>>> x**z
(-1.1122722036363393-0.012635185355335208j)
```

---

---

```
>>> x**3
(- 9+46j)
```

---

У комплексного числа можно извлечь действительную и мнимую части.

---

```
>>> x = 3+2j
>>> x.real
3.0
```

---

---

```
>>> x.imag
2.0
```

---

Для получения комплексносопряженного числа необходимо использовать метод `conjugate()`.

---

```
>>> x.conjugate()
(3-2j)
```

---

### 3.3. Битовые операции

В Python доступны битовые операции, их можно производить над целыми числами.

#### Побитовое И (AND).

---

```
>>> p = 9
>>> q = 3
>>> p & q
1
```

---

#### Побитовое ИЛИ (OR).

---

```
>>> p | q
11
```

---

#### Побитовое Исключающее ИЛИ (XOR).

---

```
>>> p^q
10
```

---

#### Инверсия.

---

```
>>> ~p
-10
```

---

#### Сдвиг вправо и влево.

---

```
>>> p<<1
18
```

---

### 3.4. Представление чисел в других системах счисления

В своей повседневной жизни мы используем десятичную систему исчисления, но при программировании, очень часто, приходится работать с шестнадцатеричной, двоичной и восьмеричной.

#### Представление числа в шестнадцатеричной системе.

---

```
>>> m = 124504
>>> hex(m)
'0x1e658'
```

---

#### Представление числа в восьмеричной системе.

---

```
>>> oct(m)
'0o363130'
```

---

#### Представление числа в двоичной системе.

---

```
>>> bin(m)
'0b11110011001011000'
```

---

### 3.5. 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.

---

```
import math
```

---

Рассмотрим наиболее часто используемые функции.

**Функция `math.ceil(x)`.** Возвращает ближайшее целое число большее, чем `x`.

---

```
>>> math.ceil(3.2)
4
```

---

**Функция `math.fabs(x)`.** Возвращает абсолютное значение числа.

---

```
>>> math.fabs(-7)
7.0
```

---

**Функция `math.factorial(x)`.** Вычисляет факториал  $x$ .

---

```
>>> math.factorial(5)
120
```

---

**Функция `math.floor(x)`.** Возвращает ближайшее целое число меньше, чем  $x$ .

---

```
>>> math.floor(3.2)
3
```

---

**Функция `math.exp(x)`.** Вычисляет  $e^{**x}$ .

---

```
>>> math.exp(3)
20.08553692318766
```

---

**Функция `math.log2(x)`.** Логарифм по основанию 2.

---

```
>>> math.log2(8)
3.0
```

---

**Функция `math.log10(x)`.** Логарифм по основанию 10.

---

```
>>> math.log10(1000)
3.0
```

---

**Функция `math.log(x[, base])`.** По умолчанию вычисляет логарифм по основанию  $e$ , дополнительно можно указать основание логарифма.

---

```
>>> math.log(5)
1.609437912434100
```

---

---

```
>>> math.log(4, 8)
0.6666666666666666
```

---

**Функция `math.pow(x, y)`.** Вычисляет значение  $x$  в степени  $y$ .

---

```
>>> math.pow(3, 4)
81.0
```

---

**Функция `math.sqrt(x)`.** Корень квадратный от  $x$ .

---

```
>>> math.sqrt(25)
5.0
```

---

**Тригонометрические функции, их мы оставим без примера.**

- `math.cos(x)`
- `math.sin(x)`
- `math.tan(x)`
- `math.acos(x)`
- `math.asin(x)`
- `math.atan(x)`

И напоследок пару констант.

- `math.pi` — число  $\pi$ .
- `math.e` — число  $e$ .

Помимо перечисленных, модуль `math` содержит ещё много различных функций, за более подробной информацией можете обратиться на официальный сайт (<https://docs.python.org/3/library/math.html>).