Язык программирования Питон (Python)

Введение

Создатель:

голландский математик Гвидо ван Россум (Guido van Rossum) в 1991 году

Особенности

- интерпретируемый
- объектно-ориентированный
- высокоуровневый язык
- встроенные высокоуровневые структуры данных
- динамическая типизация
- синтаксис прост в изучении
- поддержка модулей и пакетов (большинство библиотек бесплатны)
- универсальный
- интеграция с другими языками (C, C++, Java)

Ветки (несовмесимые) языка:

- Python 2.x
- Python 3.x

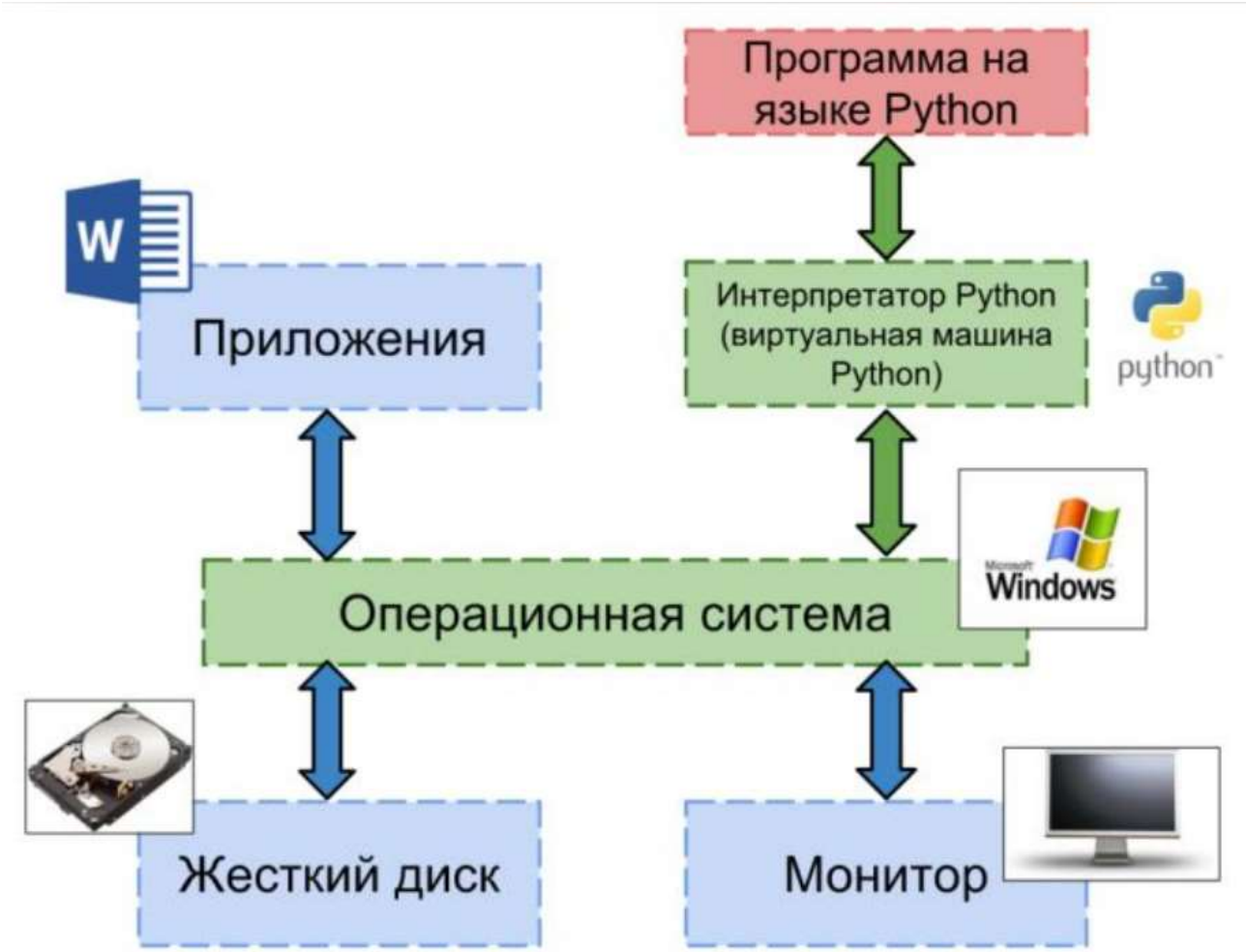
Поддерживаемые парадигмы:

- императивное программирование (процедурный, структурный, модульный подходы)
- объектно-ориентированное программирование
- функциональное программирование

Области активного использования:

1. Системное программирование.
2. Разработка программ с графическим интерфейсом.
3. Разработка динамических веб-сайтов.
4. Интеграция компонентов.
5. Разработка программ для работы с базами данных.
6. Быстрое создание прототипов.
7. Разработка программ для научных вычислений.
8. Разработка игр

Запуск программ на языке Python



Переносимость программ: программа-интерпретатор (виртуальная машина) Python скрывает от Python-программиста все особенности операционной системы, поэтому, написав программу на Python в системе Windows, ее можно запустить, например, в GNU/Linux и получить такой же результат.

Попутные замечания

Первые программы писались **на машинном языке**, т.к. для ЭВМ того времени еще не существовало развитого программного обеспечения, а машинный язык – это единственный способ взаимодействия с аппаратным обеспечением компьютера.

Каждую команду машинного языка напрямую выполняет то или иное электронное устройство. Данные и команды программисты записывали в цифровом виде (например, в шестнадцатеричной или двоичной системах счисления).

Стремление человека оперировать словами и не цифрами привело к появлению **ассемблеров**. Это языки, в которых вместо численного обозначения команд и областей памяти используются словесно-буквенные.

После ассемблеров наступил рассвет **языков** так называемого **высокого уровня**.

Под каждый язык программирования создаются **трансляторы** – специальные программы, преобразующие программный код с языка программирования в машинный код.

В отличие от ассемблеров, которые остаются привязанными к своим типам машин, языки высоко уровня обладают переносимостью.

Выделяют два основных способа трансляции – компиляция программы или ее интерпретация.

При **компиляции** весь исходный программный код (тот, который пишет программист) сразу переводится в машинный. Создается так называемый отдельный **исполняемый файл**, который никак не связан с исходным кодом. Выполнение исполняемого файла обеспечивается операционной системой.

При **интерпретации** выполнение кода происходит последовательно (можно сказать, строка за строкой). Операционная система взаимодействует с интерпретатором, а не исходным кодом.

Выполнение откомпилированной программы происходит быстрее, т.к. она представляет собой готовый машинный код. Однако на современных компьютерах снижение скорости выполнения при интерпретации обычно не заметно.

При использовании Python для анализа данных и машинного обучения необходимо:

– **Python** версии 3.4.3 или более поздней
– ключевые библиотеки Python для научных вычислений:
- **SciPy** – набор функций для научных вычислений в Python. Содержит продвинутые процедуры линейной алгебры, математическую оптимизацию функций, обработку сигналов, специальные математические функции и статистические функции.
- **NumPy** – основополагающая библиотека, необходимая для научных вычислений на Python. Содержит функциональные возможности для работы с многомерными массивами, высокоуровневыми математическими функциями (операции линейной алгебры, преобразование Фурье, генератор псевдослучайных чисел). Базовый функционал NumPy – это класс ndarray, многомерный (n-мерный) массив. Все элементы массива должны быть одного и того же типа.
- **scikit-learn** – интегратор классических алгоритмов машинного обучения. Требует наличия NumPy и SciPy. Массив NumPy – это основная структура данных scikit-learn. Любые используемые данные должны быть преобразованы в массив NumPy.
- **matplotlib** – библиотека для работы с двумерными графиками. Включает функции для создания высококачественных визуализаций типа линейных диаграмм, гистограмм, диаграмм разброса и т.д.
- **pandas** – инструмент для анализа структурных данных и временных рядов (надстройка над NumPy). Построена на основе структуры данных, называемой DataFrame, представляющей собой таблицу, похожую на электронную таблицу Excel. В отличие от NumPy, который требует, чтобы все записи в массиве были одного и того же типа, в pandas каждый столбец может иметь отдельный тип.

– **Jupyter Notebook** – интерактивная среда программирования на основе браузера.
– **Spyder** – инструментальная среда программирования на Python.

Официальный сайт языка Python - http://python.org.

Python ***распространяется свободно*** на основании лицензии подобной GNU General Public License.
Это пример свободного и открытого программного обеспечения – FLOSS (Free/Libré and Open Source Software).

Установка

Anaconda - Дистрибутив Python, предназначенный для крупномасштабной обработки данных, прогнозной аналитики и научных вычислений от компании Continuum Analytics.

Это ***бесплатный, включая коммерческое использование***, и ***готовый к использованию*** в среде предприятия дистрибутив Python, который объединяет все ключевые библиотеки, необходимые для работы в области науки о данных, математики и разработки, в одном удобном для пользователя кросс-платформенном дистрибутиве.

Anaconda уже включает NumPy, SciPy, matplotlib, pandas, IPython, Jupyter Notebook и scikit-learn.

Есть версии для Mac OS, Windows и Linux.

https://www.anaconda.com/download/ – ссылка для загрузки.

Anaconda 2019.10 For Windows Installer
Python 3.7 version
64-Bit Graphical Installer (462 MB)
32-Bit Graphical Installer (410 MB)

После успешной установки дистрибутива Anaconda можно

– ***установить и обновлять пакеты*** Python;

– ***создавать виртуальные среды*** и переключаться между ними

при помощи системы управления пакетами и средой **Conda.**

Для того, чтобы использовать команды **`conda`** через командную строку (**`cmd`**), необходимо запустить программу **Anaconda Prompt (Anaconda3)**

Примечание:

необходимо запускать терминал от имени администратора:

Пуск/Anaconda3(64-bit)/Anaconda Prompt/ -**правая кнопка мыши** –

запуск от имени администратора/

Использование командной консоли

pip – менеджер пакетов для Python. Устанавливает только пакеты Python из **PyPI**.
Используется при установке чистых акетов Python, недоступных на каналах Conda.

Предварительно нужно активировать требуемую виртуальную среду:

```
conda activate newproject
```

и установить **pip**:

```
conda install pip
```

после чего можно его использовать для установки пакетов. Например для пакета **unipath**:

```
pip install unipath
```

venv – менеджер среды для Python.

conda – система управления пакетами и средой

- помогает управлять зависимостями и изолировать проекты:
- поддерживает языки, отличные от Python.

С помощью **conda** можно:

- установить и обновлять пакеты и их зависимости (написанные на любом языке) из репозиториев, таких как **Anaconda Repository** и **Anaconda Cloud**;
- установить пакеты из **PyPI**, используя **pip** в активной среде **conda**.
- создавать, сохранять, загружать и переключается между виртуальными средами на локальном компьютере.

Основные команды Conda

conda info – проверка версии **Conda**
conda --version

conda info –e – получить список всех окружений. Активная среда обозн. ***

conda list – список установленных пакетов и версий в активной среде

conda search pack_name – поиск пакета в репозиториях **channels by Conda**

conda install pack_name – установка пакета **pack_name**
conda install – установка всего стандартного набора пакетов (>150, ~3 Гб)
conda install -c chanel pack_name – установить пакет **pack_name** с канала **chanel**

conda update conda – обновление **Conda**
conda update anaconda – обновление метапакета анаконды (**anaconda**)
conda update pack_name – обновить пакет **pack_name** в текущей среде
conda update --all – обновление всех пакетов

conda remove pack_name – удаление пакета. Все зависимые от него пакеты тоже удалятся

conda clean –t – удаление кеша – архивов .tar.bz2

conda env list – список доступных виртуальных сред **Conda**

conda create --name otherenv – создать виртуальную среду с именем **otherenv**

conda create --name py2 python=2.7 – создать среду с именем **py2** с Python 2.7

conda activate otherenv – активировать новую среду для её использования

conda deactivate – дезактивировать (отключить) текущую среду и вернуться в корневую среду

conda remove --name env_name --all – удаление среды с именем **env_name**

Примеры:

```
conda install tensorflow
conda install -c conda-forge keras
conda config --append channels conda-forge
conda install scikit-learn=0.19.2
```

Для запуска программ ***Python из командной строки Windows*** (т.е. приглашения DOS) необходимо установить должным образом ***переменную PATH***.

Windows 10 и Windows 8

1. В строке "Поиск" выполните поиск: Система (Панель управления)
2. Нажмите на ссылку **Дополнительные параметры системы**.
3. Нажмите **Переменные среды**. В разделе **Переменные среды** выберите переменную среды PATH. Нажмите **Изменить**. Если переменной PATH не существует, нажмите Создать.
4. В окне **Изменение системной переменной** (или **Новая системная переменная**) укажите значение переменной среды PATH:

```
c:\ProgramData\Anaconda3\
```

Нажмите **ОК**. Закройте остальные открытые окна, нажимая **ОК**.

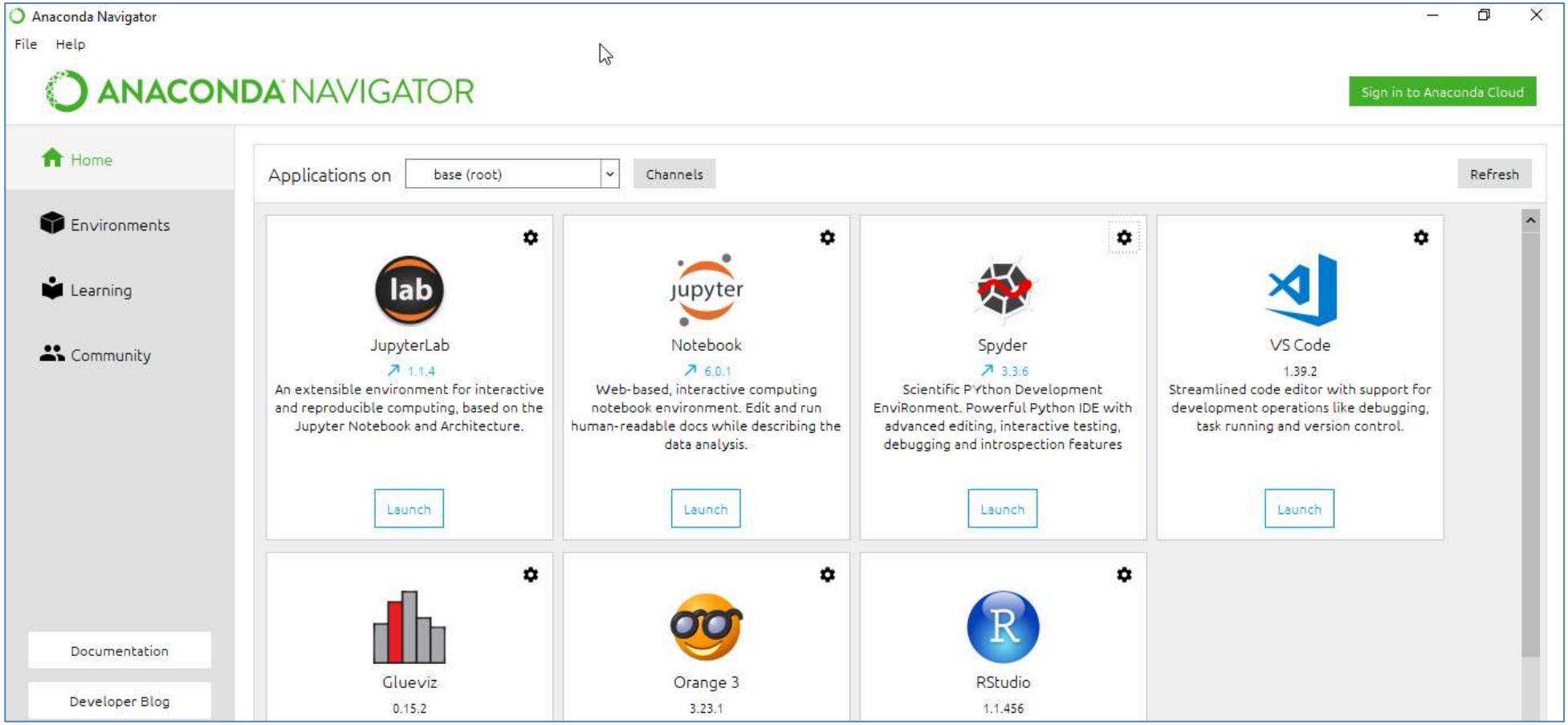
Что бы узнать ***параметры среды***, в которой работает Spyder:

```
import sys
    # рабочая среда (версия Python)
print(sys.version)
print(sys.base_prefix)
    # Список доступных модулей в текущем окружении
print('\n'.join(sys.modules.keys()))
```

Использование Anaconda Navigator

Anaconda Navigator– это графический интерфейс пользователя (GUI), включенный в дистрибутив **Anaconda**, который позволяет запускать приложения и легко управлять пакетами, средами и каналами **conda** без использования команд командной строки.

Навигатор может искать пакеты в **Anaconda Cloud** или в локальном репозитории **Anaconda**.



- **JupyterLab –** интерактивная среда разработки для работы с блокнотами, кодом и данными.

- **Jupyter Notebok –** удобный инструмент для создания красивых аналитических отчетов, позволяет хранить вместе код, изображения, комментарии, формулы и графики. Работа ведется в браузере.

- **Spyder –** интерактивной IDE для научных расчетов на языке Python. Позволяет писать, редактировать и тестировать код. Предлагает просмотр и редактирование переменных с помощью GUI, динамическую интроспекцию кода, нахождение ошибок на лету и многое другое.

При необходимости, можно интегрировать Anaconda с другими Python IDE, включая PyCharm и Atom.

- **VS Code –** оптимизированный редактор кода с поддержкой таких операций разработки, как отладка, запуск задач и контроль версий.

- **Glueviz –** используется для визуализации многомерных данных в файлах. Он исследует отношения внутри и между связанными наборами данных.

- **Orange 3 –** основанная на компонентах структура интеллектуального анализа данных. Может быть использована для визуализации и анализа данных. Рабочие процессы в Orange 3 очень интерактивны и предоставляют большой набор инструментов.

- **RStudio –** это набор интегрированных инструментов, предназначенных для повышения продуктивности работы с R.

Разделение виртуальных сред для использования различных версий пакетов

Многие научные пакеты зависят от конкретных версий других пакетов.

Иногда разработанное приложение перестаёт работать из-за того, что какой-то из пакетов (библиотек) больше несовместим с другими частями программы вследствие критических изменений.

Возможное решение состоит в настройке новой виртуальной среды для данного приложения, которая содержит версию Python и версии пакетов, полностью совместимые с этим приложением.

При исследовании данных часто используют несколько версий множества пакетов и несколько сред для разделения этих разных версий.

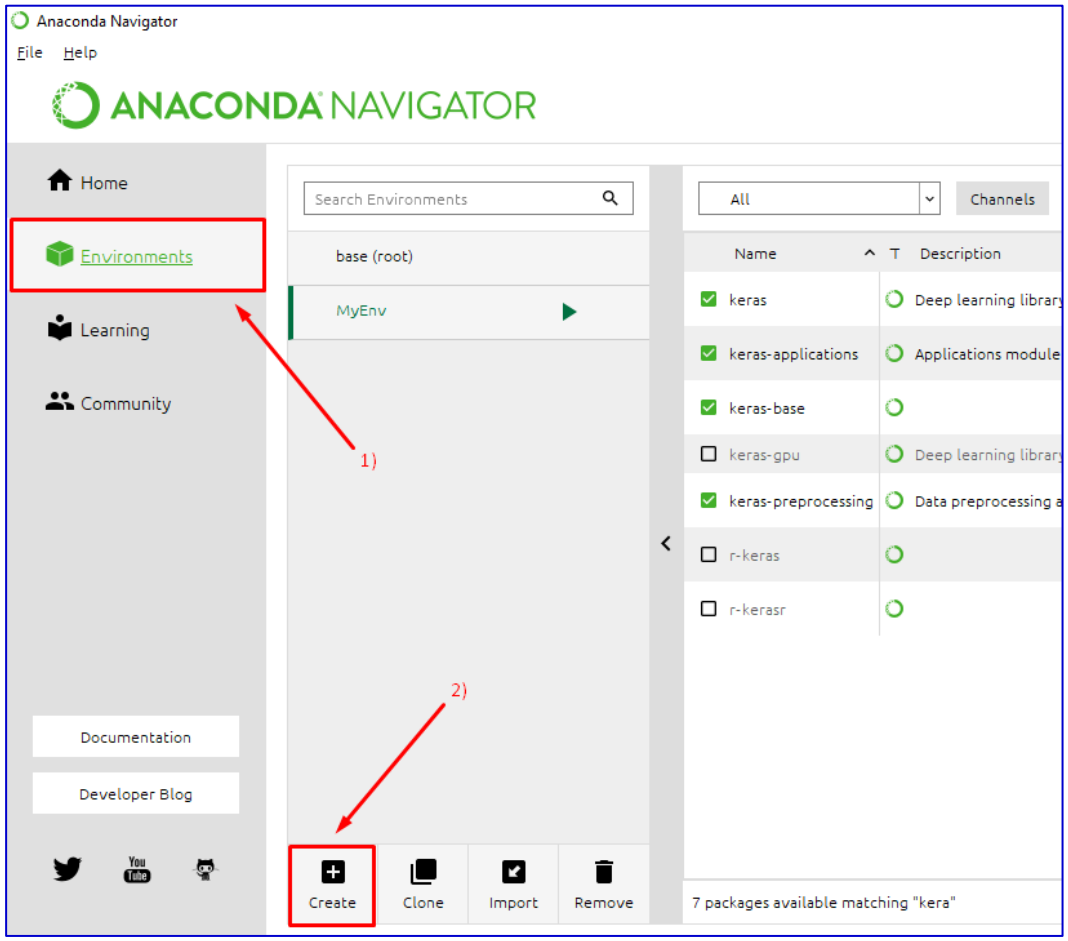
Программа командной строки **conda** является одновременно менеджером пакетов и менеджером среды. Это создает гарантию, что каждая версия каждого пакета имеет все необходимые зависимости и работает правильно.

Navigator – это простой и удобный способ работы с пакетами и средами без необходимости вводить команды **conda** в окне терминала.

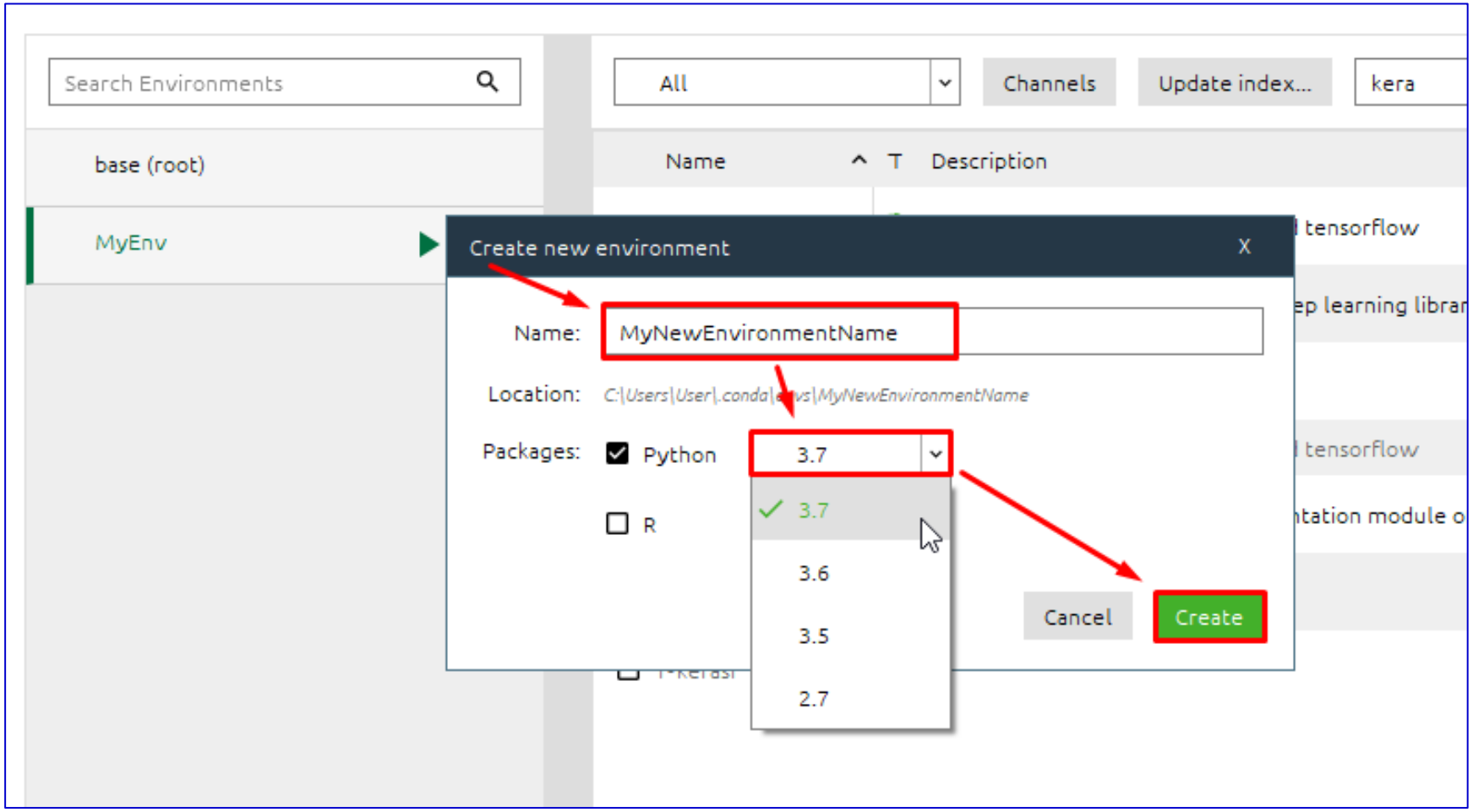
Его можно использовать, чтобы найти нужные пакеты, установить их в среде, запустить пакеты и обновить их.

Создание новой среды в Anaconda Navigator

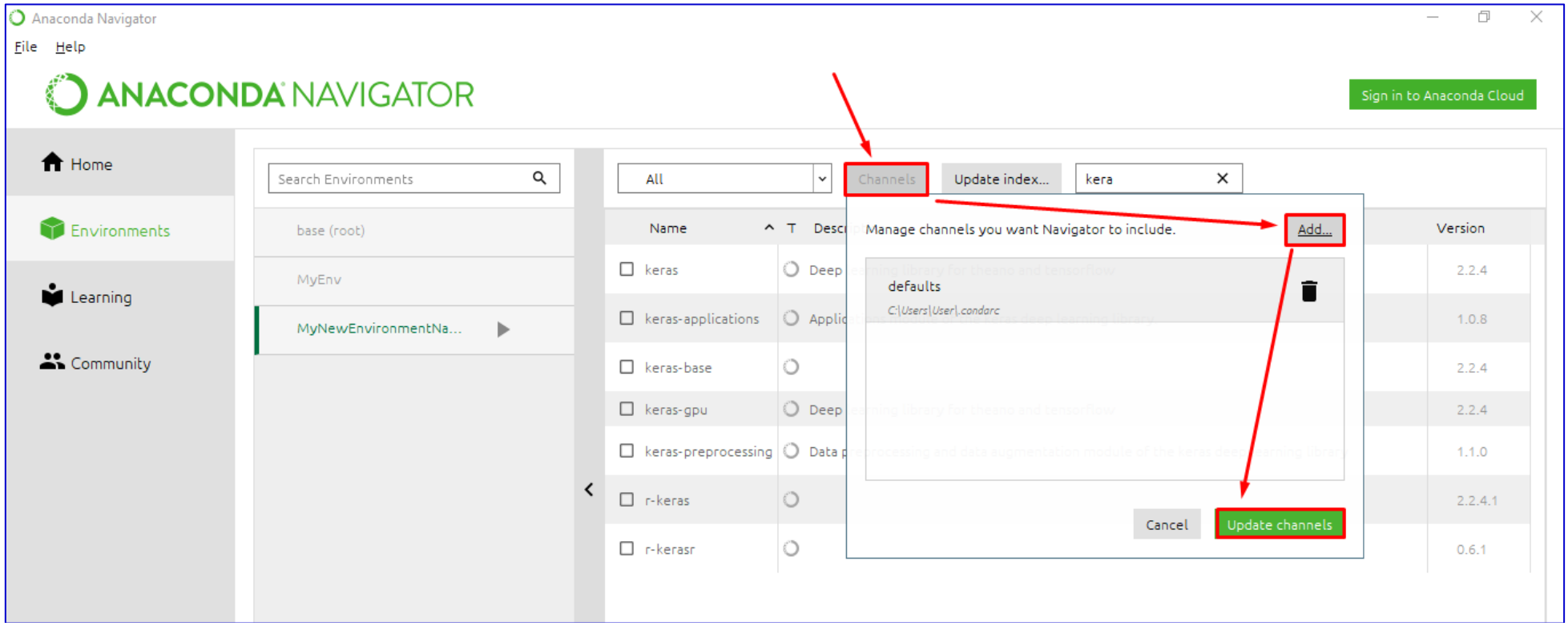
– выбираем пункт **Environments**, а затем **Create**:



– указываем **наименование среды** и выбираем **версию Python**:



– добавление нового канала:

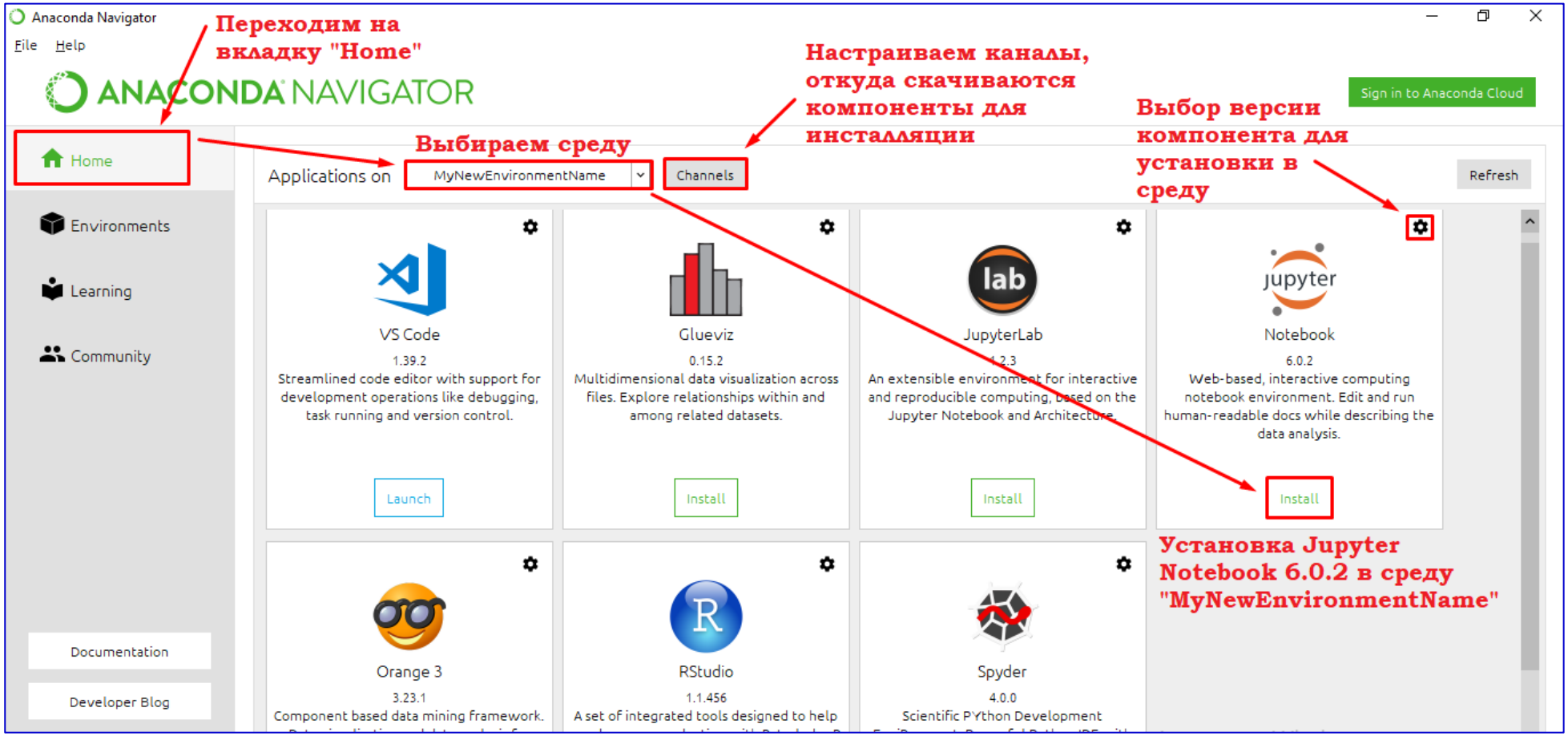


Каналы – это места хранилищ, где **Conda** ищет пакеты.

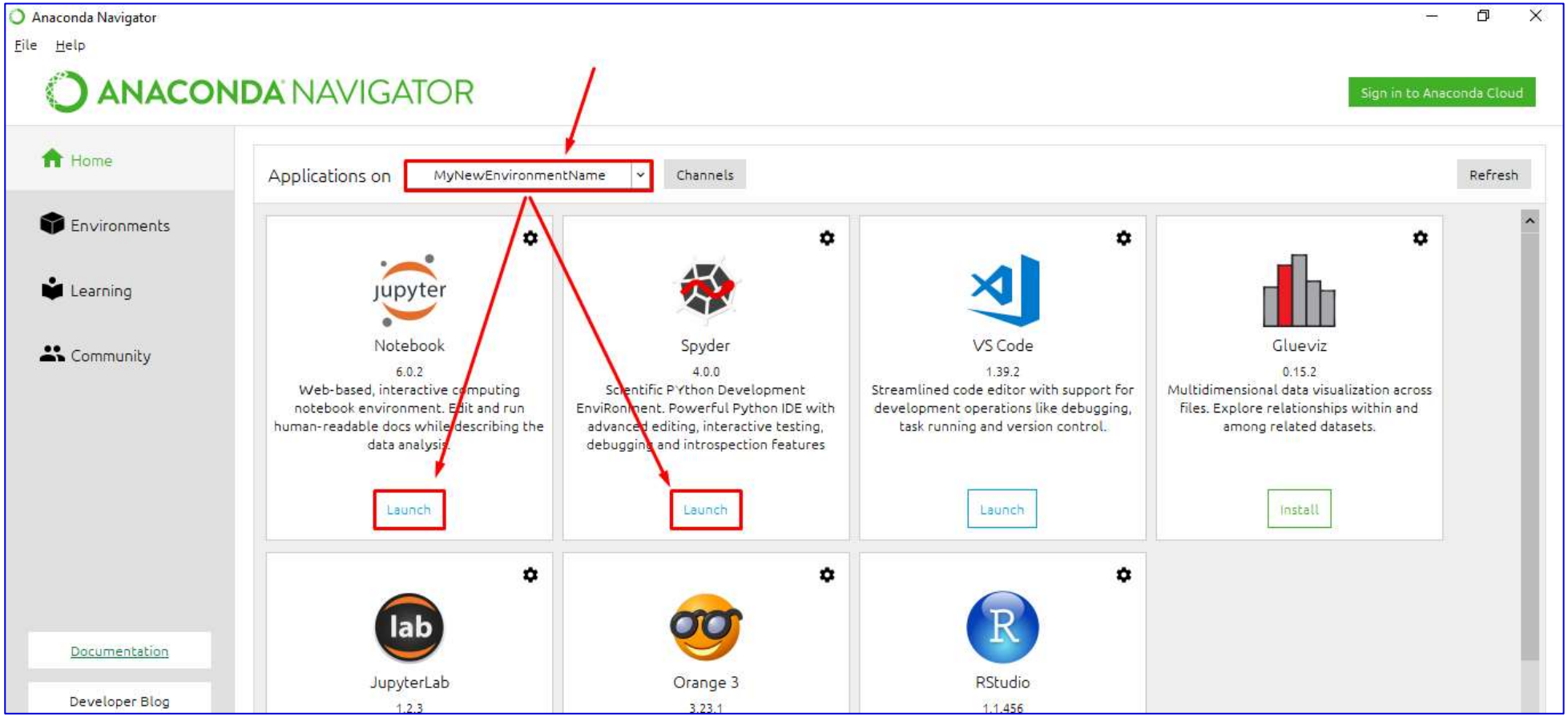
Каналы существуют **в иерархическом порядке.** Канал с наивысшим приоритетом является первым, который проверяет Conda в поисках пакета, который запрашивается. Этот порядок, а также добавить новые каналы и установить их приоритет.

Начало работы в новой среде Conda

– В Anaconda navigator переходим на вкладку **Home** и **инсталлируем в определенную среду** те компоненты, которые будем использовать.



– после инсталляции станут доступны кнопки **Launch –** запустить компонент для работы в среде.



– установка новой библиотеки (пакета) в среду
(изменения, которые вносятся в пакеты, применяются только к активной среде).