Python

Python - популярный язык программирования общего назначения. Мы будем рассматривать Python как инструмент для научных вычислений и анализа данных.

Python 2 и 3

На данные момент активно используются две «ветки» языка: Python 2 и Python 3. Это связано с тем, что около 2008 года авторы языка решили очистить Python 2 от ошибочных архитектурных решений. К сожалению, для этого пришлось отказаться от обратной совместимости, то есть во многих случаях код на Python 2 нужно модифицировать, чтобы запустить его на Python 3.

Версия Python 2 считается устаревшей, и больше не дорабатывается. На данный момент большинство популярных пакетов (включая NumPy, SciPy и scikit-learn) перенесены на Python 3, и как правило лучше использовать Python 3. **Мы будем использовать Python 3.**

Реализации Python

Язык Python хорошо стандартизирован. Эталонной и наиболее распространённой реализацией является CPython, https://www.python.org/ . Именно ей нужно пользоваться в большинстве случаев. Это связано с тем, что модули, содержащие код на C (к примеру, NumPy), должны учитывать «внутренности» интерпретатора CPython. Код на «чистом» Python обычно переносится на другие реализации без каких-либо проблем.

Кратко упомянем самые популярные альтернативные реализации.

- PyPy. Основная цель проекта – ускорить исполнение Python кода за счёт JIT-компиляции.
- IronPython. Реализация Python под платформу Microsoft .NET.
- JPython. Реализация Python под платформу Java.

Научные вычисления и анализ данных под Python

В отличие от MATLAB, язык Python изначально не заточен под научные вычисления. Пакет NumPy позволяет удобно работать с векторами и матрицами, при этом реализация всех операций с ними тщательно оптимизирована. NumPy можно сравнить с «ядром» языка MATLAB. Абсолютное большинство научных проектов на Python используют NumPy.

Поверх NumPy построен пакет SciPy, в котором собраны более высокоуровневые алгоритмы, к примеру, методы оптимизации.

Поверх SciPy реализовано множество модулей для различных областей науки. Один из них - Scikit-learn, содержит алгоритмы машинного обучения. Эти модули можно сравнить с MATLAB Toolbox.

Некоторые другие полезные пакеты:

- Matplotlib - рисование графиков.
- Pandas - работа с табличными данными (аналогично языку R).
- SymPy – символьные вычисления.

Учебники

Для освоения Python мы рекомендуем официальный учебник на английском. Доступен перевод на русский язык более старой версии (для Python 3.1). После этого советуем прочитать руководство Code Like a Pythonista: Idiomatic Python, особенно раздел Python has "names".

Для ознакомления с NumPy можно прочитать руководство Tentative NumPy Tutorial . Если вы знаете MATLAB, советуем документ NumPy for Matlab Users. Также есть два «словаря», показывающих, как сделать одни и те же действия на MATLAB и Python: http://mathesaurus.sourceforge.net/matlab-numpy.html, http://sebastianraschka.com/Articles/2014_matlab_vs_numpy.html

Как установить Python и модули

Под Windows

Самый простой способ – воспользоваться «научным» дистрибутивом Python, который содержит интерпретатор и большое число модулей. Мы рекомендуем бесплатный дистрибутив Anaconda (убедитесь, что скачиваете версию с Python 3).

Более сложный, но и более гибкий вариант:

1. Скачать и установить интерпретатор Python 3 с сайта www.python.org.
2. Установить нужные пакеты из подборки Unofficial Windows Binaries for Python Extension Packages. Не перепутайте версию Python и разрядность (32 или 64 бита).
   Обратите внимание, что в этой подборке NumPy собран с использованием библиотеки Intel MKL, что может в разы ускорить операции линейной алгебры (в Anaconda такая опция доступна лишь за плату).

Для самостоятельной компиляции пакетов, содержащих код на C, Вам понадобится Visual Studio 2010 и выше с компилятором C++. Не рекомендуем пытаться самостоятельно компилировать NumPy и Scipy, так как это требует ещё и компилятора Fortran.

Под Linux и MacOS

Можно воспользоваться дистрибутивом Anaconda (убедитесь, что скачиваете версию с Python 3).

Альтернативный вариант - для установки Python, NumPy и Scipy использовать системный менеджер пакетов:

- Под Linux - встроенный в систему.
- Под MacOS - MacPorts или HomeBrew.

Это нужно, чтобы установились компиляторы C и Fortran, если они отсутствуют. Для установки остальных пакетов воспользуйтесь pip.

Pip

Pip - стандартное средство для установки и управления Python-пакетами. С его помощью можно устанавливать как пакеты из центрального репозитория PyPI, так и скачанные из любого места в интернете. Документация: http://pip.readthedocs.org/en/latest/ .

Команда для установки scikit-learn: pip install scikit-learn

Conda

Conda – менеджер пакетов, поставляемый с дистрибутивом Anaconda. Имеет важное преимущество перед pip под Windows: не требуется компилятор, поскольку пакеты устанавливаются из бинарных пакетов.

Команда для установки scikit-learn: conda install scikit-learn

Разработка под Python

Среды разработки (IDE)

1. PyCharm. Свободно доступна Community Edition, которая вполне подходит для учебных и научных целей.
2. Spyder. Входит в Anaconda.

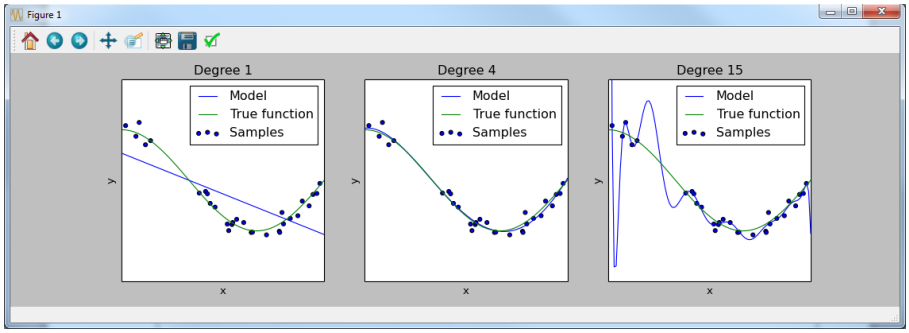
Интерактивная разработка

IPython

Пример для проверки работоспособности основных пакетов (NumPy, scikit-learn, matplotlib)

Пример взят из http://scikit-learn.org/stable/auto_examples/plot_underfitting_overfitting.html

Скопируйте текст в файл test.py, запустите при помощи команды python test.py (или python3 test.py). Если всё верно настроено, будет показана следующая картинка:




```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression


np.random.seed(0)

n_samples = 30
degrees = [1, 4, 15]

true_fun = lambda X: np.cos(1.5 * np.pi * X)
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1

plt.figure(figsize=(14, 4))
for i in range(len(degrees)):
    ax = plt.subplot(1, len(degrees), i+1)
```

```
    plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X[:, np.newaxis], y)

    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, label="Samples")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.title("Degree %d" % degrees[i])
plt.show()
```