

Лекция 1. Основы интеллектуального анализа данных и программирования на Python в среде Anaconda

Дисциплина: Интеллектуальный анализ данных, текстов и изображений

Лектор: доцент кафедры бизнес-информатики
к.т.н. Буров Сергей Александрович, burov-sa@ranepa.ru

20.02.2023



Вопросы лекции

- | | |
|--------------------------------------------|------|
| 1. Введение | - 3 |
| 2. Основы интеллектуального анализа данных | - 14 |
| 3. Основы работы в среде Anaconda | - 22 |
| 4. Основы языка Python | - 29 |

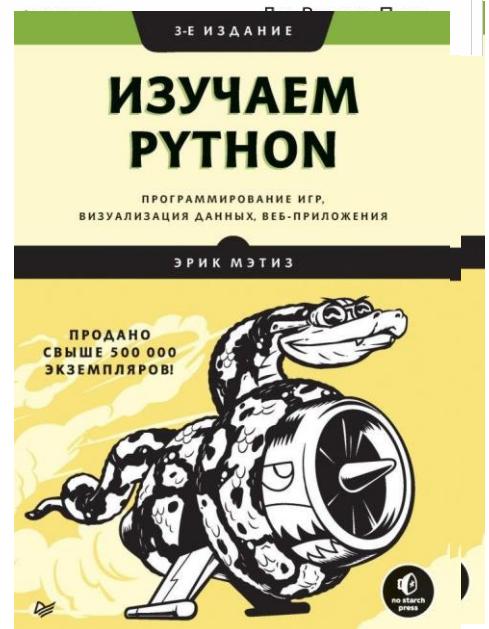
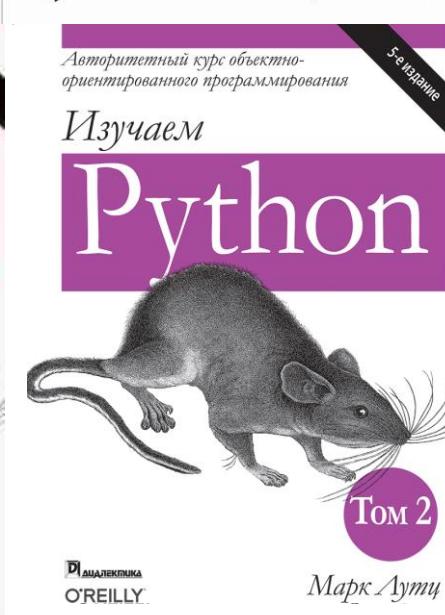
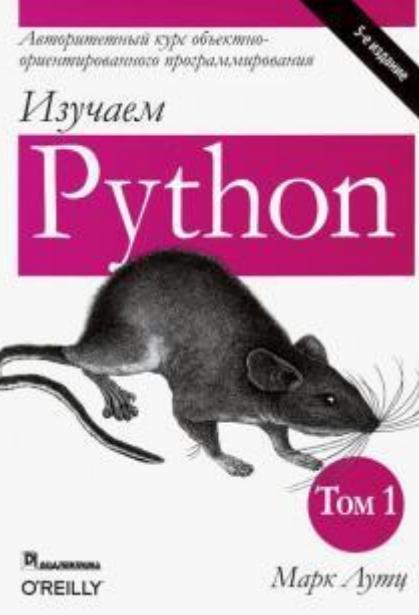
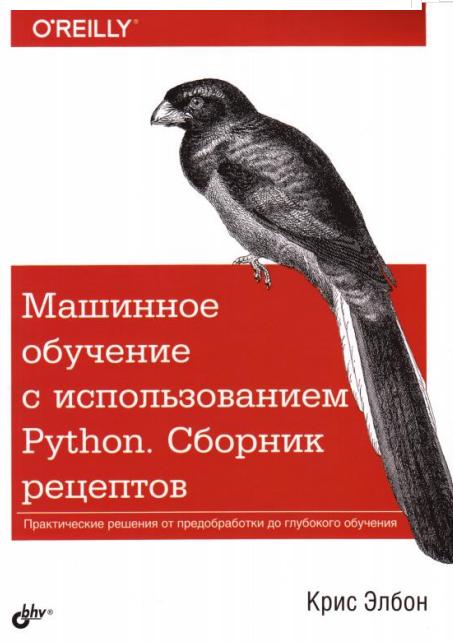
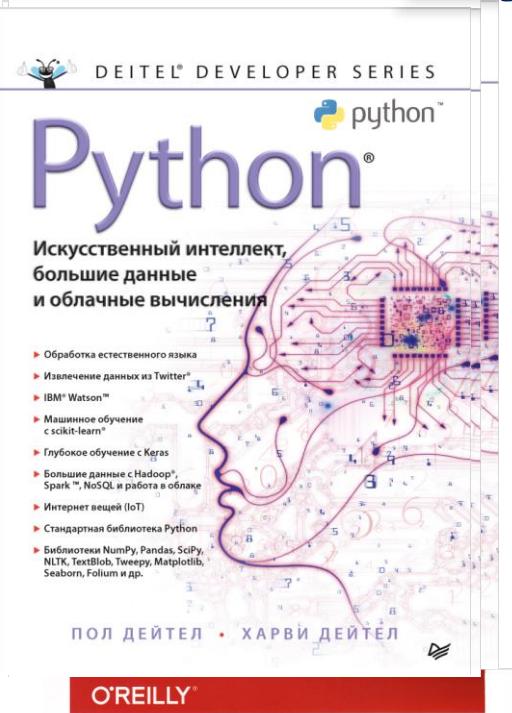
Структура курса

№ п/п	Наименование тем	Всего	Объем дисциплины, час.					Форма текущего контроля успеваемости**, промежуточной аттестации***
			Контактная работа обучающихся с преподавателем по видам учебных занятий			СР		
			Л	ПЗ	КСР	СРО	СП	
Тема 1	Основы глубокого обучения. Математические основы нейронных сетей	10	4	4		8		УО/Зад/Т
Тема 2	Машинное обучение на Python	14	4	4		10		УО/Зад/Т
Тема 3	Нейронные сети на Python	12	4	4		10		УО/Зад
Тема 4	Глубокое обучение и нейронные сети. Анализ текстов и изображений		4	4		12		УО/Зад
Промежуточная аттестация					2*			Зачет 05.06.2023
Всего (акад./астр. часы):		72/54	16/12	16/12	2/1,5	40/30		

Даты занятий:

20.02.23
 27.02.23
 06.03.23
 13.03.23
 20.03.23
 27.03.23
 03.04.23
 10.04.23
05.06.23

Рекомендуемая литература



Как изучать материал

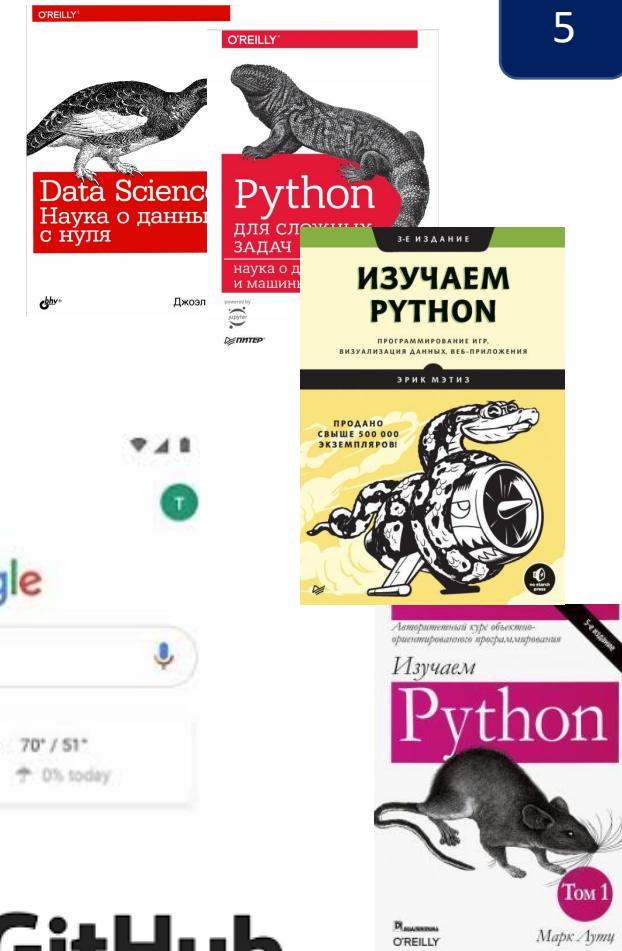
Самостоятельна работа

Занятия в РАНХиГС



kaggle

stack overflow



Введение

1. «Данные никогда не спят»

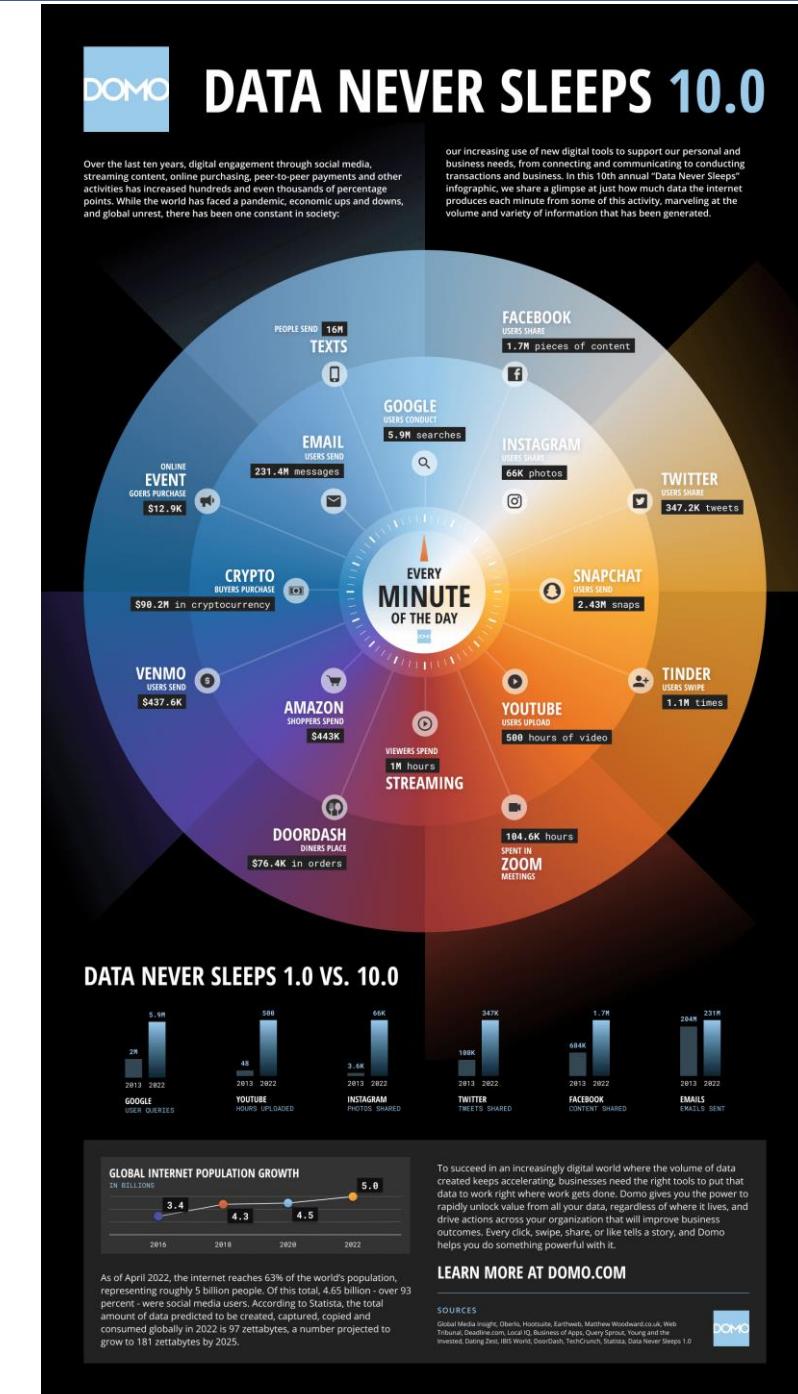
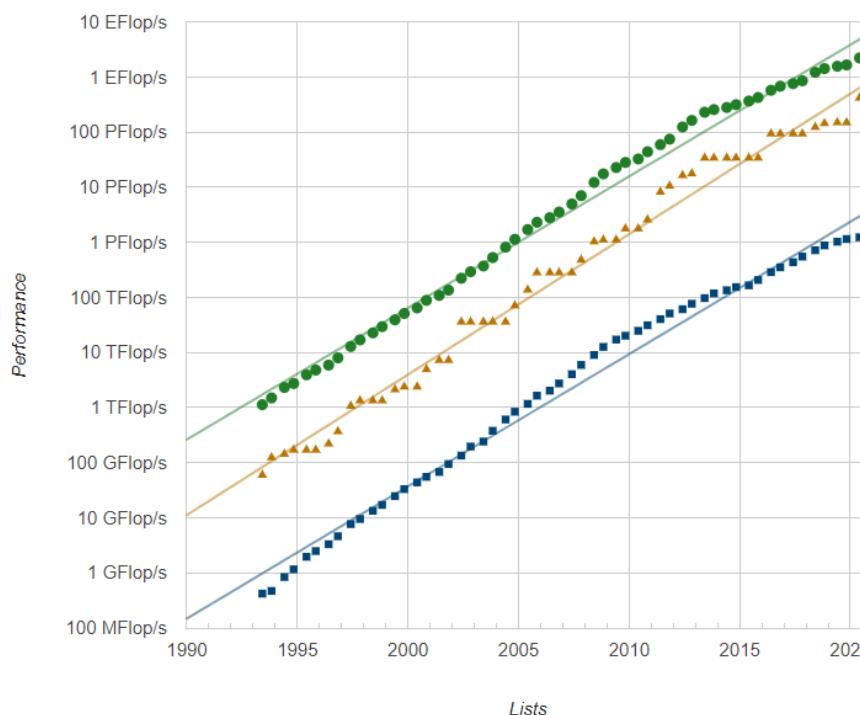
Факт. На каждого человека производится 1,7 мегабайта в секунду

<https://analyticsweek.com/content/big-data-facts/>

Численность населения – 7,8 млрд. человек – 13 петабайт новых данных каждую секунду

Сколько байтов в килобайте?

2. Непрерывный рост вычислительной мощности



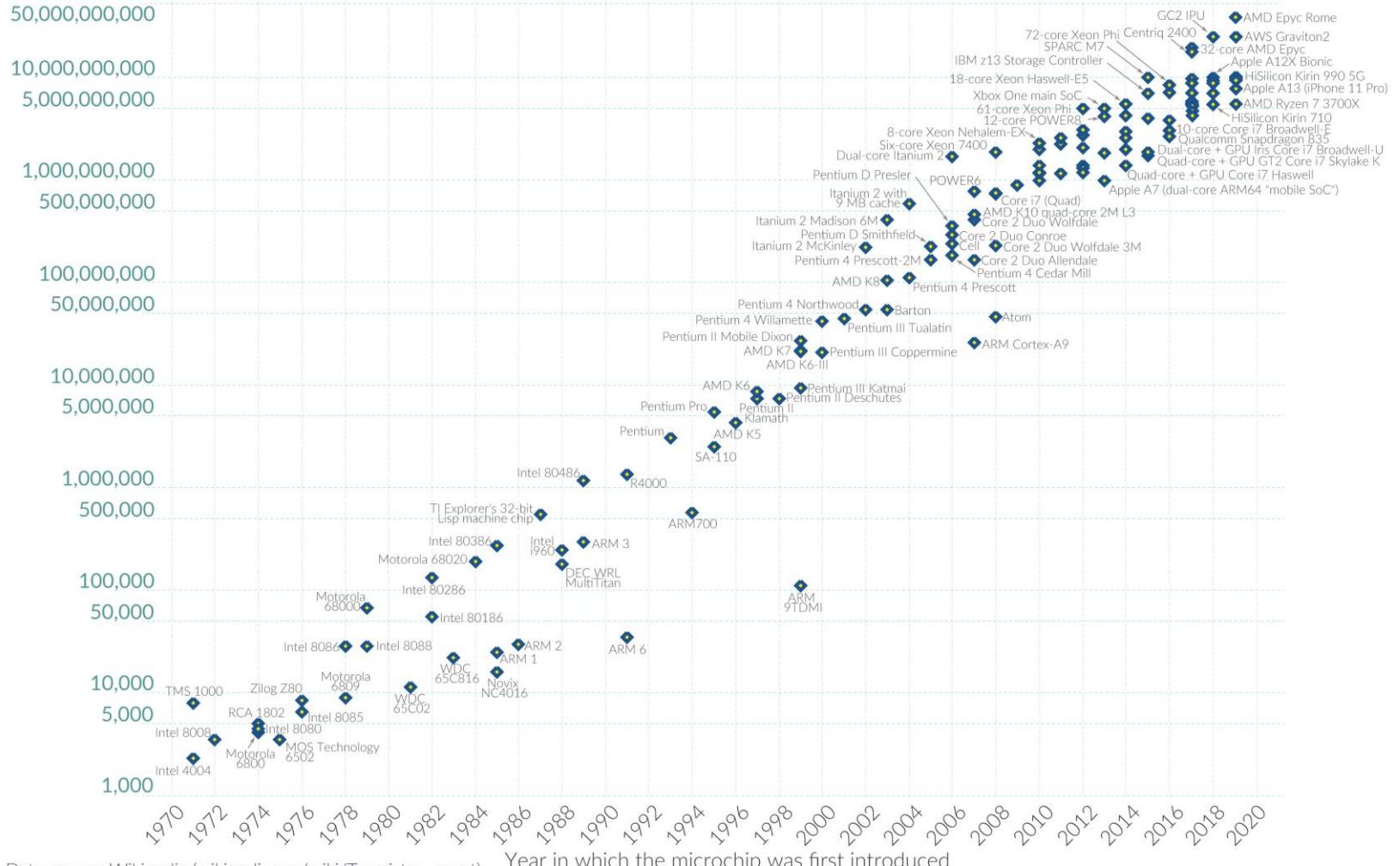
Увеличение количества данных (Закон Мура)

Our World
in Data

Moore's Law: The number of transistors on microchips doubles every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Transistor count



Data source: Wikipedia ([wikipedia.org/wiki/Transistor_count](https://en.wikipedia.org/wiki/Transistor_count))

OurWorldinData.org – Research and data to make progress against the world's largest problems.

Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

Закон Мура - Википедия

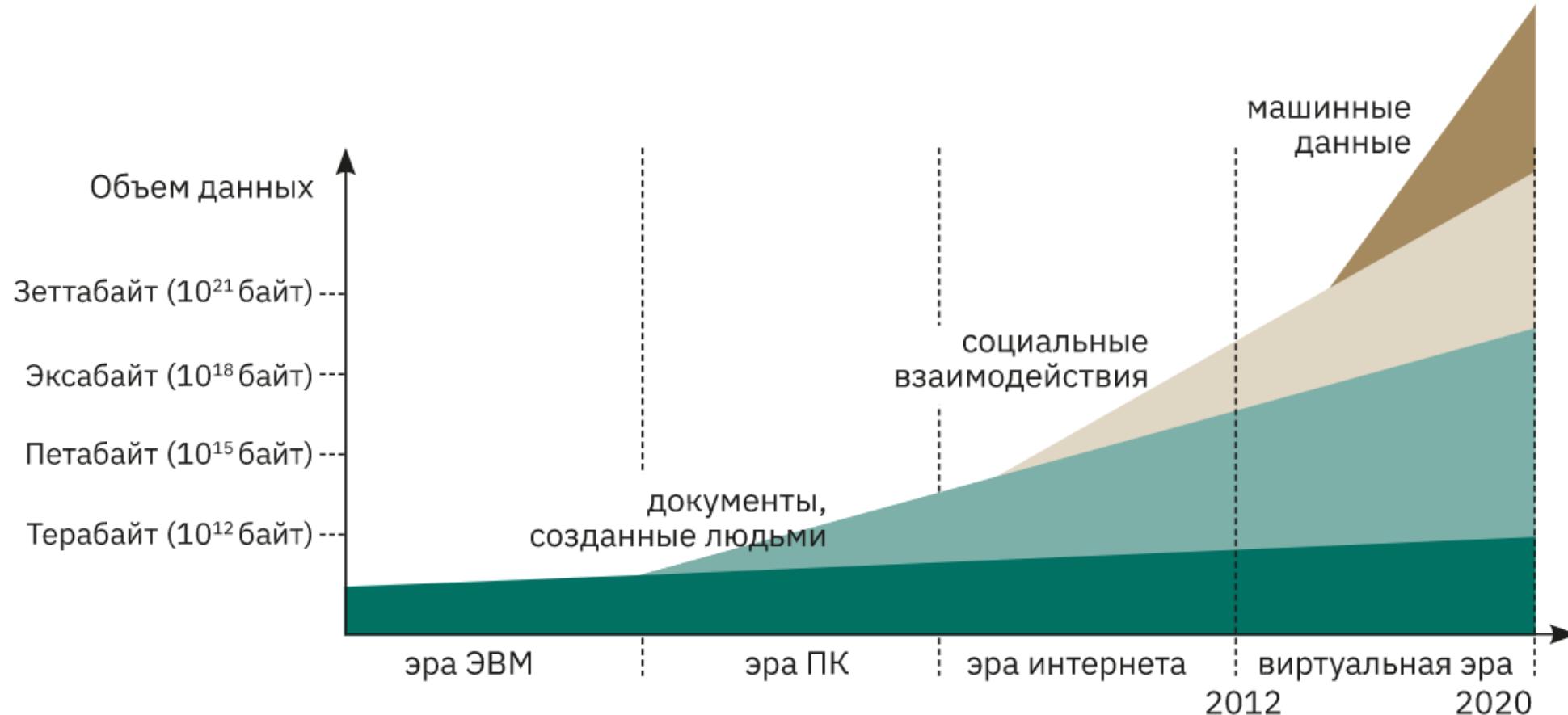
Информационные революции



Big Data – новая глобальная проблема 21-го века

Быстрый рост объемов информации: в 150 раз за 10 лет. На каждого жителя Земли приходится 130 ГБ.

Ключевые проблемы: структуризация, поиск, анализ и сжатие информации;



Инновации влияющие на цифровую экономику



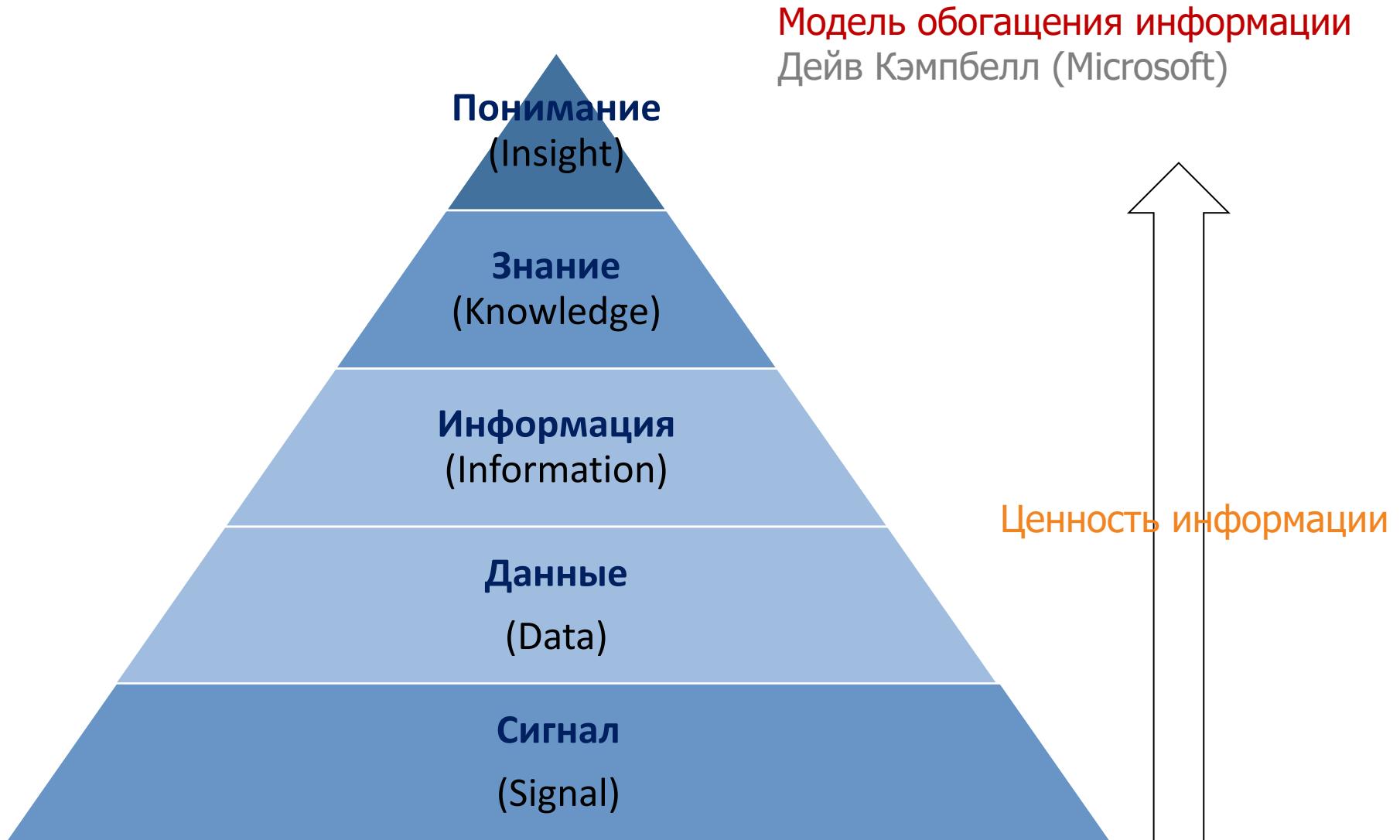
«Целью вычислений является понимание сути,
а не числа»

Hamming, R. W., Numerical Methods for Scientists and Engineers
(New York, NY., McGraw Hill, 1962)

Основные характеристики больших данных (четыре V)

- 1. Объем (Volume)** — количество данных, производимых в мире, растет с экспоненциальной скоростью.
- 2. Скорость (Velocity)** — темпы производства данных, их перемещения между организациями и изменения данных стремительно растут
- 3. Разнообразие (Variety)** — некогда данные в основном были алфавитно-цифровыми (то есть состояли из символов алфавита, цифр, знаков препинания и некоторых специальных знаков). В наши дни они также включают графику, аудио, видео и данные от стремительно растущего числа датчиков «интернета вещей», установленных в жилищах, коммерческих организациях, транспортных средствах, городах и т. д.
- 4. Достоверность (Veracity)** — характеристика, указывающая на то, являются ли данными полными и точными, и, как следствие, позволяющая ответить на вопросы вроде: «Можно ли на них полагаться при принятии критических решений?», «Насколько реальна информация?» и пр.

Соотношение понятий информация-данные-знания



«Целью вычислений является понимание сути,
а не числа»

Hamming, R. W., Numerical Methods for Scientists and Engineers
(New York, NY., McGraw Hill, 1962)

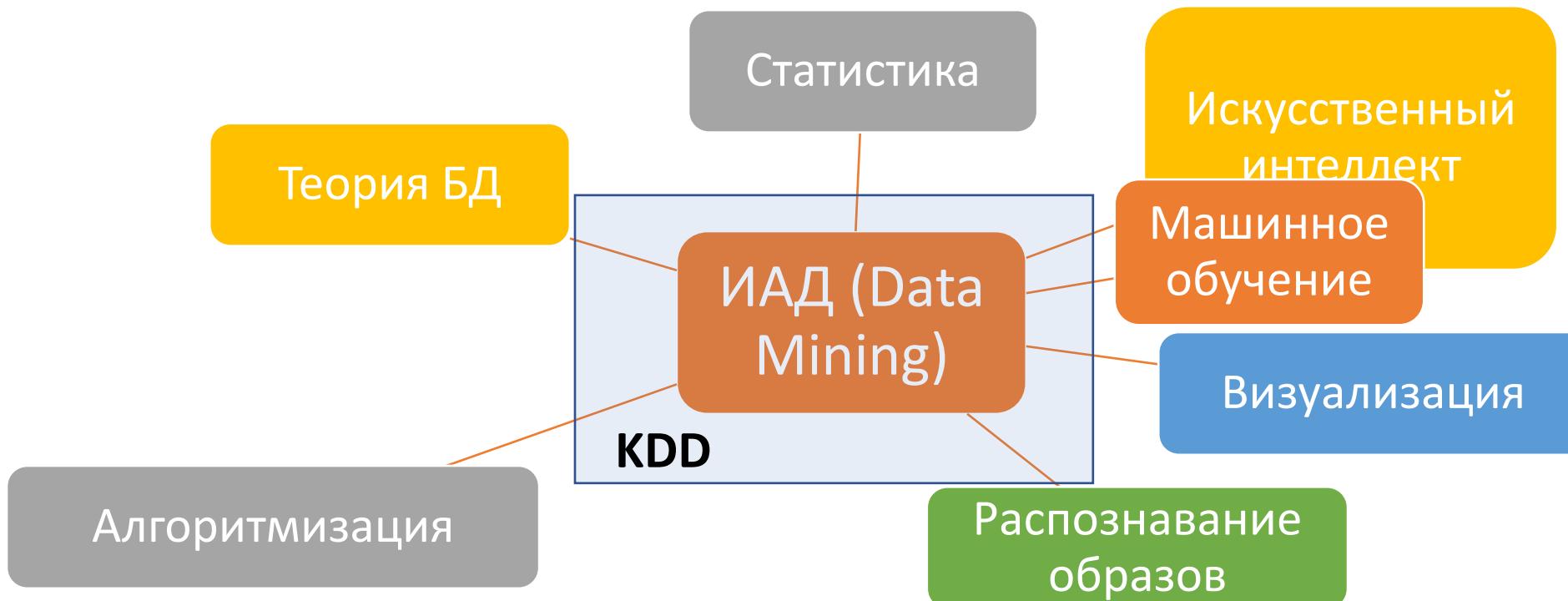
Введение

Основные характеристики больших данных (четыре V)

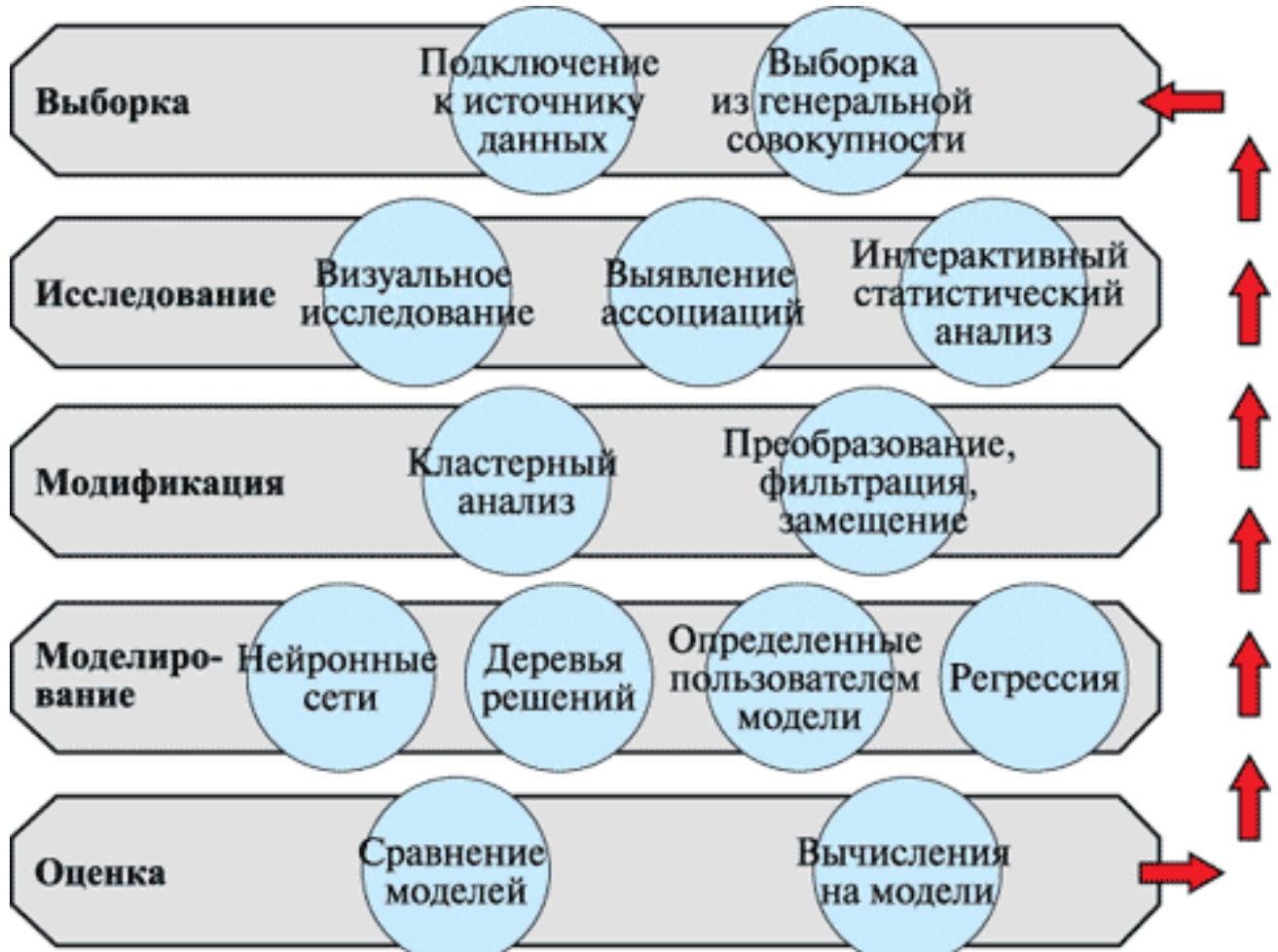
- 1. Объем (Volume)** — количество данных, производимых в мире, растет с экспоненциальной скоростью.
- 2. Скорость (Velocity)** — темпы производства данных, их перемещения между организациями и изменения данных стремительно растут
- 3. Разнообразие (Variety)** — некогда данные в основном были алфавитно-цифровыми (то есть состояли из символов алфавита, цифр, знаков препинания и некоторых специальных знаков). В наши дни они также включают графику, аудио, видео и данные от стремительно растущего числа датчиков «интернета вещей», установленных в жилищах, коммерческих организациях, транспортных средствах, городах и т. д.
- 4. Достоверность (Veracity)** — характеристика, указывающая на то, являются ли данными полными и точными, и, как следствие, позволяющая ответить на вопросы вроде: «Можно ли на них полагаться при принятии критических решений?», «Насколько реальна информация?» и пр.

Интеллектуальный анализ данных (Data Mining)

ИАД (Data Mining) – это методология и процесс обнаружения в больших массивах данных, накапливающихся в информационных системах компаний, ранее неизвестных, нетривиальных, практически полезных и доступных для интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. Data Mining является одним из этапов более масштабной методологии KDD (Knowledge Discovery in Databases) – обнаружения знаний в базах данных.

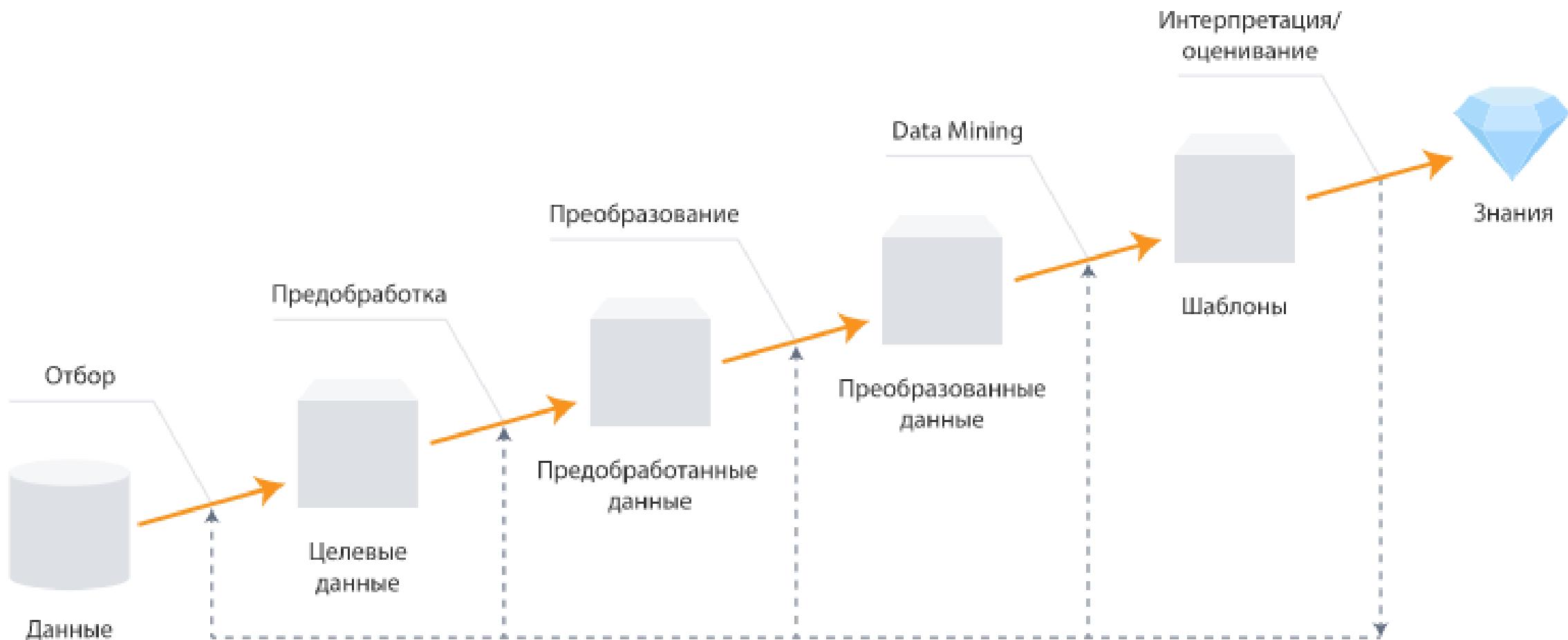


Этапы ИАД по стандарту SEMA компании SAS



- 1. Выборка данных** – формирование начального набора данных для моделирования (dataset), который должен быть достаточно большим, чтобы содержать достаточную информацию для извлечения, и в то же время ограниченным, чтобы его можно было эффективно использовать.
- 2. Исследование** – выявление ассоциаций, визуальный и интерактивный статистический анализ, понимание данных путем обнаружения ожидаемых и непредвиденных связей между переменными, а также отклонений с помощью визуализации данных.
- 3. Модификация** – применение методов выбора, создания и преобразования переменных при подготовке к моделированию: кластерный анализ, преобразование, фильтрация и замещение информации.
- 4. Моделирование** – применение методов построения и обработки моделей интеллектуального анализа данных: искусственные нейронные сети, деревья принятия решений, регрессионный анализ и т.д.
- 5. Оценка** – сравнение результатов моделирования между собой и с планируемыми показателями, анализ надежности и полезности созданных моделей.

Трансформация данных в процессе ИАД



Основные методы ИАД

Наименование	Суть	Примеры задач
1. Классификация	Отнесение объекта по его признакам (характеристикам) к одному из заранее определённых классов	Оценка кредитоспособности заёмщика Обнаружение фродов (мошенничества) Определение лояльности клиентов Медицинская диагностика
2. Регрессия	Восстановление регрессии – функциональной зависимости одной величины от других величин, с целью прогнозирования	Кредитный скоринг Оценка стоимости недвижимости
3. Кластеризация	Разбиение исходной выборки данных на кластеры – группы объектов со схожими характеристиками. Классификация без заранее определённых классов.	Экономическая география (разбиение стран по экономическим показателям) Маркетинг (разбиение потребителей на группы)
4. Ассоциация	Поиск ассоциативных правил – закономерностей и зависимостей в потоке данных	Маркетинг (изучение поведения покупателей)
5. Последовательные шаблоны	Обнаружение связей между последовательными шаблонами	Маркетинг (оценка востребованности нового продукта, тарифного плана)
6. Анализ отклонений	Поиск в данных аномалий – нетипичных, редких данных, не соответствующих логике бизнес-процесса	Обнаружение фродов (мошенничества) Обнаружение вирусов

Качество данных

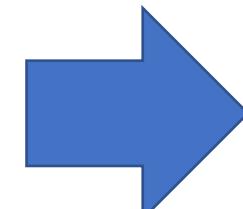
GIGO Мусор на входе – мусор на выходе!

«что посеешь, то и пожнёшь».

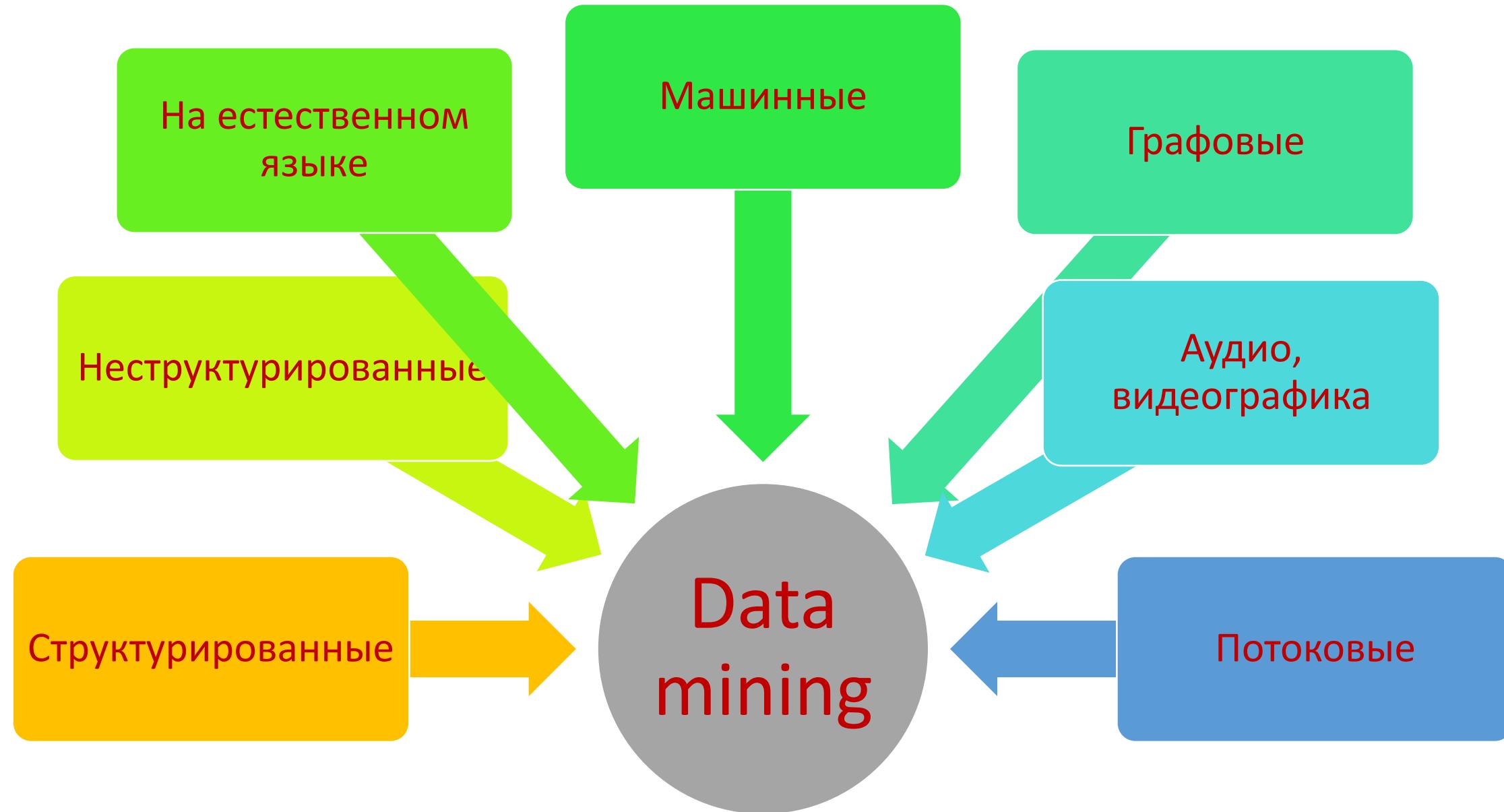
Качество данных (Data quality) – критерий, определяющий полноту, точность, своевременность и возможность интерпретации данных

Данные низкого качества (грязные данные) – отсутствующие, неточные или бесполезные данные с точки зрения практического применения (например, представленные в неверном формате, не соответствующем стандарту) :

пропущенные значения (Missing Values),
дублирование данные (Duplicate Data),
шумы и выбросы



Категории данных



Обработка отсутствующих данных

Метод	Достоинства	Недостатки
Исключение пропущенных значений	Простота	Потеря информации, полученной в ходе наблюдений
Присваивание null	Простота	Не все методы моделирования и (или) реализации корректно обрабатывают null
Присваивание статического значения (например, 0 или среднего арифметического)	Простота Предотвращение потери информации от других переменных	Возможность формирования ложных оценок на основе модели
Вычисление значения на основании предполагаемого или теоретического распределения	Незначительное влияние на модель	Относительная сложность Необходимость допущений относительно данных
Моделирование значения (независимое)	Незначительное влияние на модель	Может потребовать слишком большой уверенности в модели Может создать искусственные зависимости между переменными Относительная сложность Необходимость допущений относительно данных



Обработка выбросов



Выброс (outlier) – результат наблюдений, заметно отклоняющийся от других результатов.

Выброс обусловлен иной логикой или иным порождающим процессом, чем другие результаты.



Среда Anaconda



<https://anaconda.org/>

Anaconda – интегрированная среда разработки для научного программирования на Python и R

Conda - менеджер пакетов и система управления средой разработки языков программирования Python и R, которая устанавливает, запускает и обновляет пакеты и их зависимости

Anaconda Navigator - это графический пользовательский интерфейс (GUI) среды разработки python, включенный в дистрибутив **Anaconda**, который позволяет пользователям запускать приложения и управлять пакетами **conda**, средами и каналами без использования командной строки



`pip` — это менеджер пакетов только для Python.

`venv` — является менеджером среды только для Python.

Работает с виртуальными средами Python.

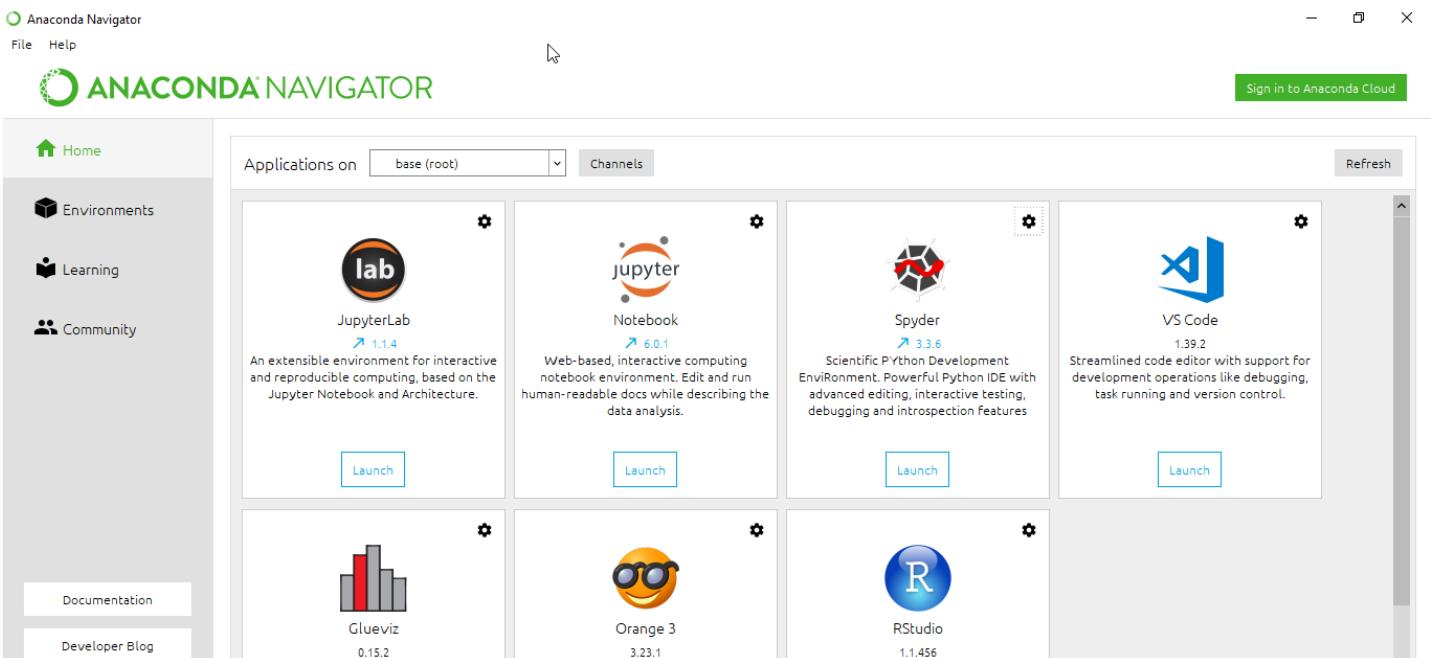
`conda` — является одновременно менеджером пакетов и среды и не зависит от языка. Возможно создание изолированных виртуальных сред для любых языков программирования.

Работает с виртуальными системными средами.

`conda build` — создаёт виртуальные среды

`conda install` — выполняет установку из пакетов сборки Conda

Anaconda Navigator



VS Code — это 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 — это основанная на компонентах структура интеллектуального анализа данных. Это может быть использовано для визуализации данных 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. Он включает в себя основы R и Notebooks.

JupyterLab — это интерактивная среда разработки для работы с блокнотами (notebooks), кодом и данными.

Jupyter Notebook — 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. Данная IDE позволяет писать, редактировать и тестировать код.

Jupyter Notebook



25

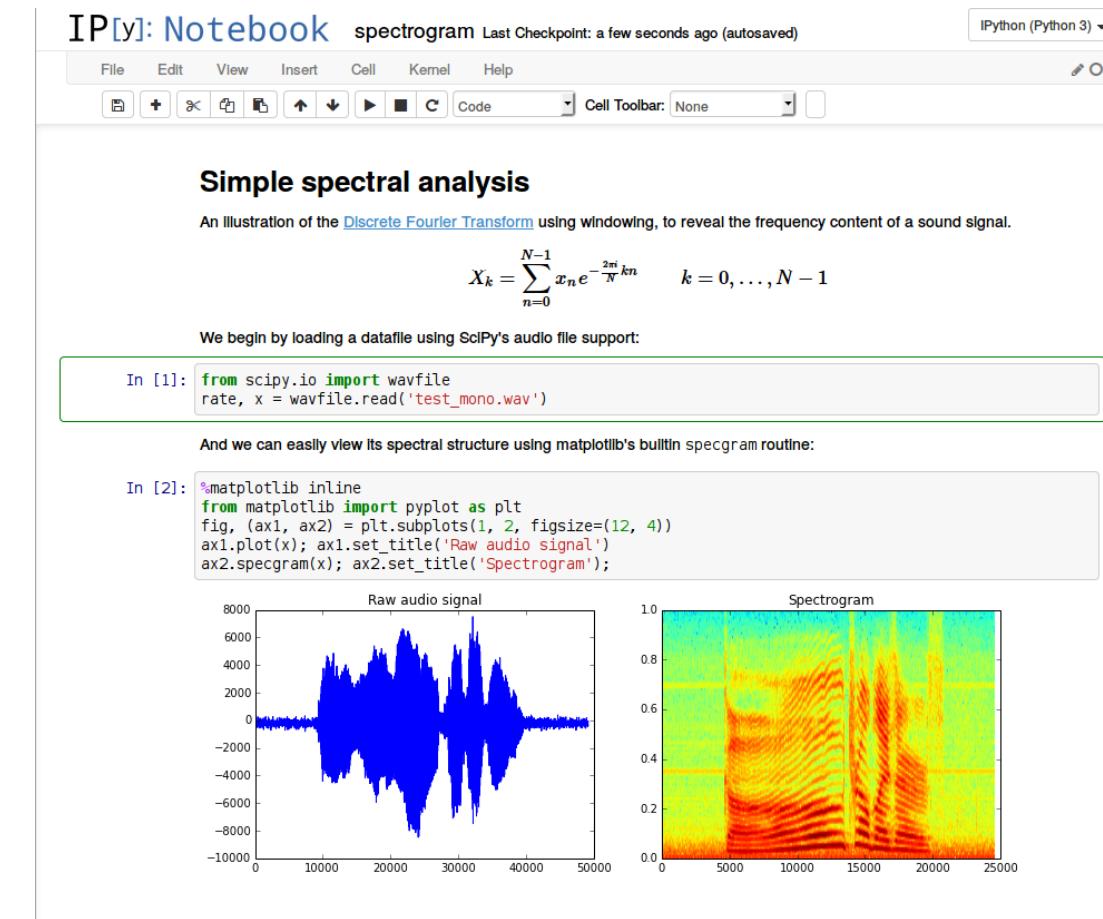
Jupyter Notebook — это мощный инструмент для разработки и представления проектов [Data Science](#) в интерактивном виде. Он объединяет код и вывод все в виде одного документа, содержащего текст, математические уравнения и визуализации.

.jupyter — текстовый файл, который описывает содержимое вашей записной книжки в формате JSON.

kernel (Ядро) — это «вычислительный движок», который выполняет код, содержащийся в документе ноутбука.

cell (Ячейка) — это контейнер для текста, который будет отображаться в записной книжке, или код, который будет выполняться ядром записной книжки.

Markdown (произносится маркдаун) — [облегчённый язык разметки](#), созданный с целью обозначения форматирования в [простом тексте](#), с максимальным сохранением его читаемости человеком, и пригодный для машинного преобразования в языки для продвинутых публикаций ([HTML](#), [Rich Text](#) и других).



Язык Markdown + LaTex

Справочник по разметке Markdown для docs.microsoft.com – Contributor Guide | Microsoft Docs



The screenshot shows the Upmath: Markdown & LaTeX Online editor. It displays a LaTeX snippet $p=\frac{1}{q}$ with a note about placing big equations on separate lines. Below it, there's a snippet of LaTeX code for solving a quadratic equation. The interface includes a toolbar with buttons for preview, HTML, and LaTeX conversion. The right side of the interface shows the rendered output of the LaTeX code, which is a mathematical formula for the roots of a quadratic equation. There are also notes about writing Cyrillic symbols and using matrices.

Особенности оболочки IPython

Функции	Выполняемые действия	Пример
?	Доступ к документации	<i>len?</i>
??	Доступ к исходному коду с помощью символов ??	<i>square??</i>
<TAB>	Автодополнение кода	
*	Подбор по джокерному символу	<i>*Warning?</i>
%Магические команды		
%timeit %%timeit	Подсчёт времени многократно выполняемого фрагмента кода	<i>%timeit sum(range(100))</i>
%time	Подсчёт времени операции	
%inline		
и др.		

Горячие клавиши IPython

Комбинация клавиш	Действие
Ctrl+A	Перемещает курсор в начало строки
Ctrl+E	Перемещает курсор в конец строки
Ctrl+B (или стрелка «влево»)	Перемещает курсор назад на один символ
Ctrl+F (или стрелка «вправо»)	Перемещает курсор вперед на один символ

Комбинация клавиш	Действие
Backspace	Удаляет предыдущий символ в строке
Ctrl+D	Удаляет следующий символ в строке
Ctrl+K	Вырезает текст, начиная от курсора и до конца строки
Ctrl+U	Вырезает текст с начала строки до курсора
Ctrl+Y	Вставляет предварительно вырезанный текст
Ctrl+T	Меняет местами предыдущие два символа

Комбинация клавиш	Действие
Ctrl+P (или стрелка «вверх»)	Доступ к предыдущей команде в истории
Ctrl+N (или стрелка «вниз»)	Доступ к следующей команде в истории
Ctrl+R	Поиск в обратном направлении по истории команд

Комбинация клавиш	Действие
Ctrl+L	Очистить экран терминала
Ctrl+C (или стрелка «вниз»)	Прервать выполнение текущей команды Python
Ctrl+D	Выйти из сеанса IPython ¹



Общая характеристика языка Python

Python — объектно-ориентированный сценарный язык, один из наиболее популярных языков программирования для научных вычислений в мире.

Парадигмы программирования: процедурная, функциональная, объектно-ориентированная, используется динамическая типизация с автоматической уборкой мусора»

Достоинства Python и причины его популярности:

- 1. Общедоступность.** Python бесплатный общедоступный проект с открытым кодом, имеющий огромное сообщество пользователей
- 2. Относительная простота.** Он проще в изучении, чем такие языки, как C, C++, C# и Java, что позволяет быстро включиться в работу как новичкам, так и профессиональным разработчикам.
- 3. Эффективность.** Python повышает эффективность труда разработчика за счет обширной подборки стандартных и сторонних библиотек с открытым кодом. Существует множество бесплатных приложений Python с открытым исходным кодом

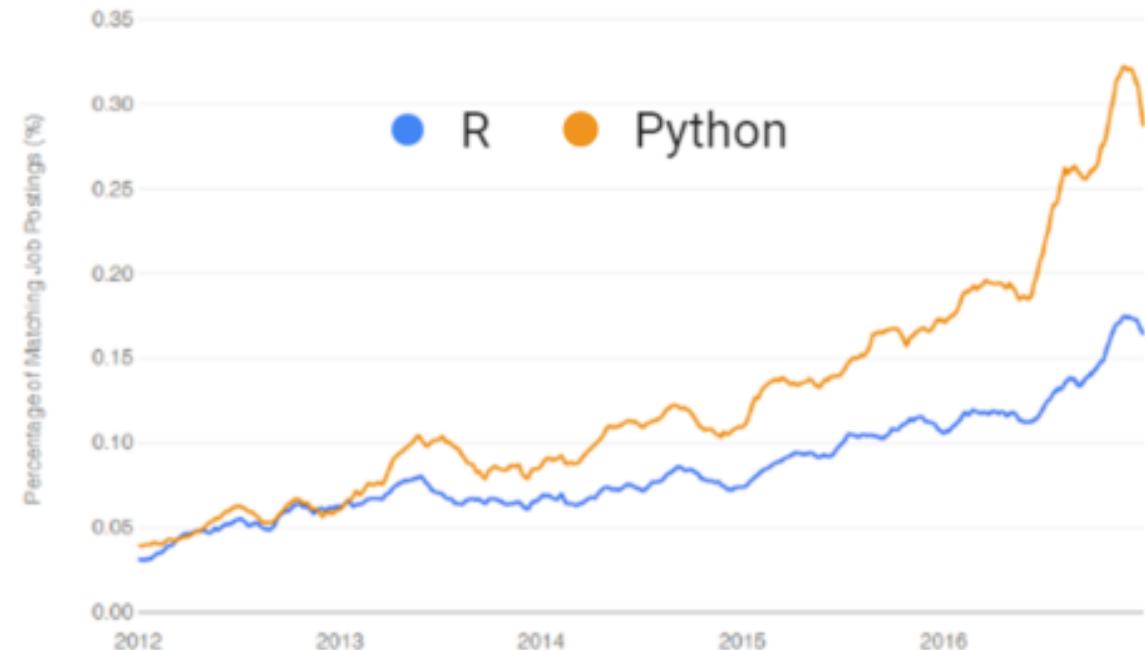
Python – популярный язык веб-разработки (Django, Flask)



Сравнительный анализ Python и R

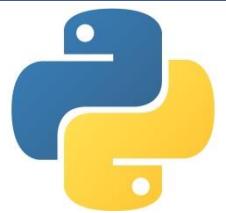


Comparison Factors	R	Python
Ease of Learning	✗	✓
Speed	✗	✓
Data Handling Capabilities	✓	✓
Graphics & Visualization	✓	✗
Flexibility	✓	✓
Popularity	✗	✓
Job Scenario	✗	✓
Community Support	✓	✓



Сравнительный анализ Python и R

R	Python
R коды нуждаются в дополнительном обслуживании.	Коды Python более надежны и просты в обслуживании.
R является более статистическим языком и также используется для графических методов.	Python используется как язык общего назначения для разработки и развертывания.
R лучше использовать для визуализации данных.	Python лучше для глубокого изучения.
R имеет сотни пакетов или способов выполнить одну и ту же задачу. Он имеет несколько пакетов для одной задачи.	Python разработан на основе философии, согласно которой «должен быть один, а желательно только один очевидный способ сделать это». Следовательно, он имеет несколько основных пакетов для выполнения задачи.
Более простые библиотеки и графики.	Изучение библиотек Python может быть немного сложным.
R поддерживает только процедурное программирование для некоторых функций и объектно-ориентированное программирование для других функций.	Python - это мультипарадигмальный язык. Это означает, что Python поддерживает несколько парадигм, таких как объектно-ориентированное, структурированное, функциональное, аспектно-ориентированное программирование.
R - интерпретируемый язык командной строки.	Python стремится к простому синтаксису. Это имеет сходство с английским языком.
R разработан для анализа данных, следовательно, он имеет более мощные статистические пакеты.	Статистические пакеты Python менее мощны.
R медленнее, чем Python, но не намного.	Python быстрее.
R облегчает использование сложных математических расчетов и статистических тестов.	Python хорош для создания чего-то нового с нуля. Он также используется для разработки приложений.
R менее популярен, но, тем не менее, у него много пользователей.	Python более популярен, чем R



История Python



1980-е –
задуман
Python

1989 –
начало
разработки

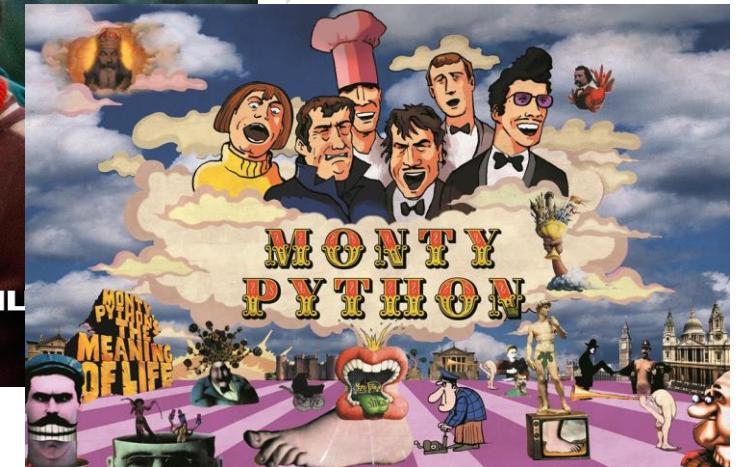
1991 –
первая
публикация
кода

16.10.2000
– версия 2.0

3.12.2008 –
версия 3.0

2010 –
Python 2.7

04.10.2021
– Python
3.10



Гвидо Ван Россум

«Дзен Python»

1. Красивое лучше, чем уродливое.
2. Явное лучше, чем неявное.
3. Простое лучше, чем сложное.
4. Сложное лучше, чем запутанное.
5. Плоское лучше, чем вложенное.
6. Разреженное лучше, чем плотное.
7. Читаемость имеет значение.
8. Особые случаи не настолько особые, чтобы нарушать правила.
9. При этом практичность важнее безупречности.
10. Ошибки никогда не должны замалчиваться.
11. Если не замалчиваются явно.
12. Встретив двусмысленность, отбрось искушение угадать.
13. Должен существовать один — и, желательно, только один — очевидный способ сделать это.
14. Хотя он поначалу может быть и не очевиден, если вы не голландец
15. Сейчас лучше, чем никогда.
16. Хотя никогда зачастую лучше, чем прямо сейчас.
17. Если реализацию сложно объяснить — идея плоха.
18. Если реализацию легко объяснить — идея, возможно, хороша.
19. Пространства имён — отличная штука! Будем делать их побольше!

import this



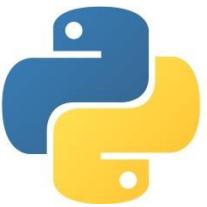
Основы синтаксиса Python



1. Конец строки является концом инструкции (не нужна ;)
2. Структура программы выстраивается с помощью отступов (пробелы, табуляция) .

Вложенные инструкции объединяются в блоки по величине отступов .

3. Основная инструкция завершается двоеточием : в след за которым располагается код вложенный блок кода



Ключевые слова Python

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

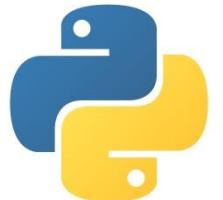
`keyword.kwlist` – список всех доступных ключевых слов

`keyword.iskeyword(строка)` – проверка, является ли строка ключевым словом



Управляющие последовательности Python

Управляющая последовательность	Описание
\n	Вставляет в строку символ новой строки. При выводе для каждого символа новой строки экранный курсор перемещается в начало следующей строки
\t	Вставляет символ горизонтальной табуляции. При выводе для каждого символа табуляции экранный курсор перемещается к следующей позиции табуляции
\\	Вставляет символ обратного слеша
\"	Вставляет символ двойной кавычки
'	Вставляет символ одиночной кавычки



Операторы сравнения

Алгебраический оператор	Оператор Python	Пример условия	Смысл
>	>	$x > y$	x больше y
<	<	$x < y$	x меньше y
\geq	\geq	$x \geq y$	x больше или равно y
\leq	\leq	$x \leq y$	x меньше или равно y
=	$==$	$x == y$	x равно y
\neq	$!=$	$x != y$	x не равно y



Правила приоритета операторов

Операторы	Группировка	Тип
<code>()</code>	слева направо	круглые скобки
<code>**</code>	справа налево	возведение в степень
<code>* / // %</code>	слева направо	умножение, деление, целочисленное деление, остаток
<code>+ -</code>	слева направо	сложение, вычитание
<code>> <= < >=</code>	слева направо	меньше, меньше или равно, больше, больше или равно
<code>== !=</code>	слева направо	равно, не равно

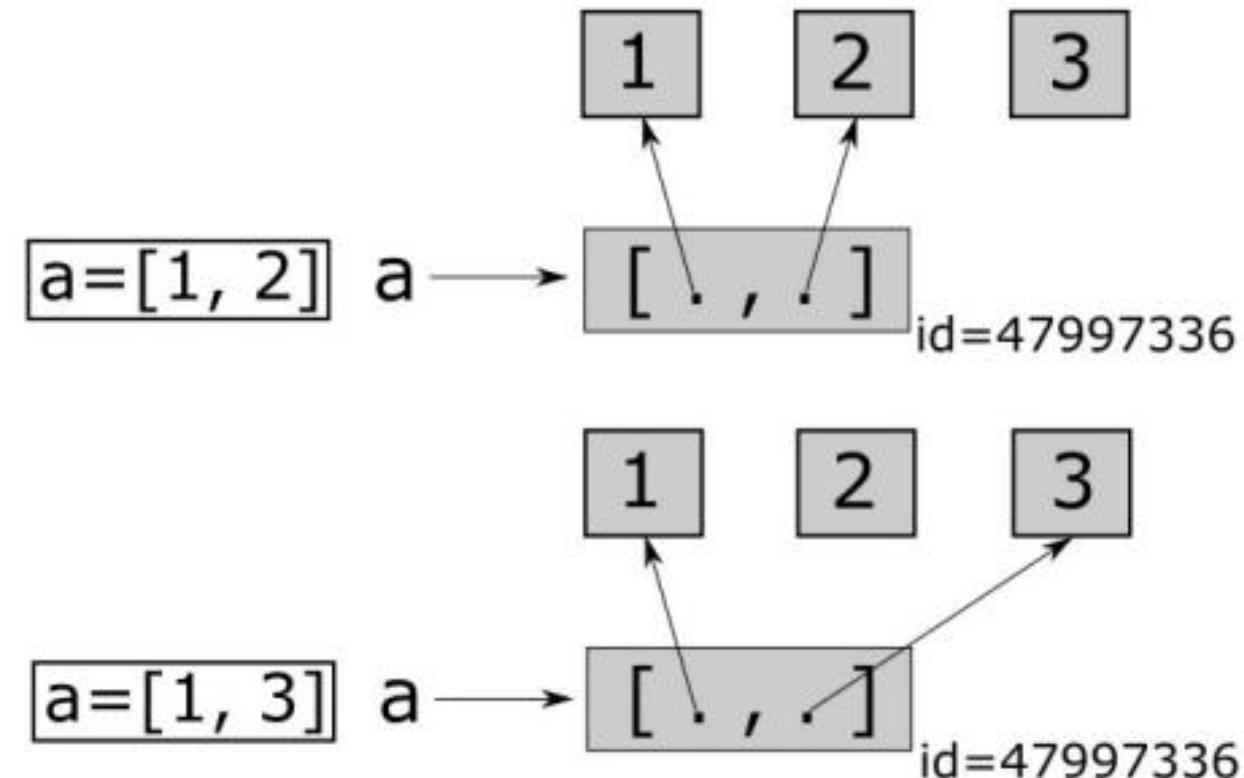
Типы данных питон

Любой элемент данных, используемый в программе на языке Python, является объектом. Каждый объект имеет свою идентичность, тип (или класс) и значение.

Имя	Тип	Описание	
Целые числа	int	1, 2, 3	К неизменяемым (<i>immutable</i>) типам относятся:
Числа с плавающей точкой	float	1.1, 2.4, 3.0	<ul style="list-style-type: none"> целые числа (<i>int</i>); числа с плавающей точкой (<i>float</i>); комплексные числа (<i>complex</i>); логические переменные (<i>bool</i>); кортежи (<i>tuple</i>); строки (<i>str</i>); неизменяемые множества (<i>frozen set</i>).
Строки	str	последовательность символов: "Hello", 'hello', "42"	
Списки	list	последовательность объектов: [1, 2.0, "hello"]	
Словари	dict	Список пар «ключ-значение»: {"john" : "+1-23-45", "bob" : "+1-32-65"}	
Кортежи	tup	Последовательность неизменяемых объектов: [1, 2.0, "hello"]	К изменяемым (<i>mutable</i>) типам относятся
Множества	set	последовательность уникальных объектов: {"a", "b"}	<ul style="list-style-type: none"> списки (<i>list</i>); множества (<i>set</i>); словари (<i>dict</i>).
Булевые значения	bool	логические значения: True или False	

Изменяемые данные

```
>>> a = [1, 2]  
>>> id(a)  
47997336  
>>> a[1] = 3  
>>> a  
[1, 3]  
>>> id(a)  
47997336
```



Числа

```
>>> type(1)          ①
<class 'int'>
>>> isinstance(1, int) ②
True
>>> 1 + 1.0          ③
2.0
>>> type(2.0)
<class 'float'>
```

- ① Можно использовать функцию `type()` для проверки типа любого значения или переменной.
- ② Функцию `isinstance()` тоже можно использовать для проверки принадлежности значения или переменной определенному типу.
- ③ Сложение значений типа `int` и `float` дает в результате `float`. Для выполнения операции сложения Python преобразует значение типа `int` в значение типа `float`, и в результате возвращает `float`.

Логический операции

Python - Logical Operators

- not

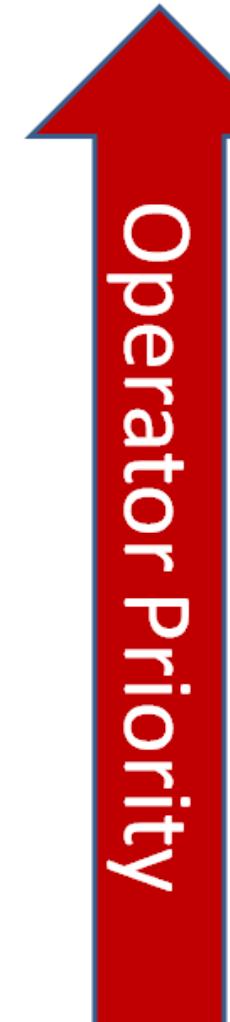
x	not x
False	True
True	False

- and

x	y	x and y
False	False	False
False	True	False
True	False	False
True	True	True

- or

x	y	x or y
False	False	False
False	True	True
True	False	True
True	True	True



Строки

Строка (string) – тип данных для хранения последовательности символов.

Для строк можно использовать одинарные, двойные и тройные кавычки. Для того, чтобы указать кавычки в самой строке, нужно заключить строку в одинарные кавычки, а внутри неё использовать двойные кавычки.

Метод строк	Описание
isalnum()	Возвращает True, если строка состоит только из <i>алфавитно-цифровых</i> символов (то есть только из букв и цифр)
isalpha()	Возвращает True, если строка состоит только из <i>алфавитных</i> символов (то есть только из букв)
isdecimal()	Возвращает True, если строка состоит только из <i>десятичных</i> цифр (то есть цифр десятичной системы счисления) и не содержит знаков + или -
isdigit()	Возвращает True, если строка состоит только из цифр (например, '0', '1', '2')
isidentifier()	Возвращает True, если строка представляет действительный <i>идентификатор</i>
islower()	Возвращает True, если все алфавитные символы в строке относятся к <i>нижнему регистру</i> (например, 'a', 'b', 'c')
isnumeric()	Возвращает True, если символы в строке представляют <i>числовое значение</i> без знака + или - и без точки-разделителя дробной части
isspace()	Возвращает True, если строка содержит только символы- <i>пропуски</i>
istitle()	Возвращает True, если символами <i>верхнего регистра</i> в строке являются только первые символы каждого слова
isupper()	Возвращает True, если все алфавитные символы в строке относятся к <i>верхнему регистру</i> (например, 'A', 'B', 'C')

Создание строки:

`x = str()`

`x = ''`

`x = ""`

`x = """"""`

Некоторые операции над строками

Операция	Пример кода
Конкатенация (сложение)	<code>>>> S1 = 'spam' >>> S2 = 'eggs' >>> print(S1 + S2) 'spameggs'</code>
Дублирование строки	<code>>>> print('spam' * 3) spamspamspam</code>
Длина строки	<code>>>> len('spam') 4</code>
Доступ по индексу	<code>>>> S = 'spam' >>> S[0] 's'</code>
Извлечение среза	<code>>>> s = 'spameggs' >>> s[3:5] 'me'</code>

Строки

Разбиение строк

Метод `split` без аргументов разбивает строку на подстроки по символам-пропускам, возвращая список лексем. Чтобы выполнить разбиение строки по нестандартному ограничителю (например, по параметрам «запятая и пробел»),

укажите строку-ограничитель (например, `' , '`), которая должна использоваться при разбиении строки:

```
In [1]: letters = 'A, B, C, D'
In [2]: letters.split(',')
Out[2]: ['A', 'B', 'C', 'D']
```

Объединение строк

Метод `join` выполняет конкатенацию строк в своем аргументе, в котором должен передаваться итерируемый объект, содержащий только строковые значения; в противном случае происходит ошибка `TypeError`. Разделителем между объединяемыми строками становится строка, для которой вызывается `join`. В следующем коде создаются строки со списками значений, разделенных запятыми:

```
In [4]: letters_list = ['A', 'B', 'C', 'D']
In [5]: ','.join(letters_list)
Out[5]: 'A,B,C,D'
```

Метод `format` вызывается для *форматной строки*, содержащей *заполнители* в фигурных скобках `{}`, — возможно, с форматными спецификаторами. Методу передаются форматируемые значения. Отформатируем значение с плавающей точкой `17.489` до двух знаков в дробной части:

```
In [1]: '{:.2f}'.format(17.489)
Out[1]: '17.49'
```

Несколько заполнителей

Форматная строка может содержать сразу несколько заполнителей; в этом случае аргументы метода `format` сопоставляются с заполнителями слева направо:

```
In [2]: '{} {}'.format('Amanda', 'Cyan')
Out[2]: 'Amanda Cyan'
```

Ссылка на аргументы по позиции

Форматная строка может ссылаться на аргументы по их позиции в списке аргументов метода `format`; первому аргументу соответствует позиция 0:

```
In [3]: '{0} {0} {1}'.format('Happy', 'Birthday')
Out[3]: 'Happy Happy Birthday'
```

Ссылка на ключевые аргументы

К ключевым аргументам можно обращаться в заполнителях по их ключам:

```
In [4]: '{first} {last}'.format(first='Amanda', last='Gray')
Out[4]: 'Amanda Gray'
```

```
In [5]: '{last} {first}'.format(first='Amanda', last='Gray')
Out[5]: 'Gray Amanda'
```

Список (list) – тип последовательности для представления упорядоченных коллекций элементов.

Коллекция – структура данных, состоящая из взаимосвязанных элементов данных.

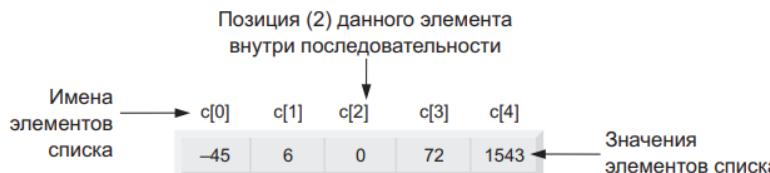
Списки изменяе́мы, в них часто хранятся однородные данные, но данные также могут быть разнородными.

Создание списка:

```
x = list()
```

```
X = []
```

Чтобы обратиться к элементу списка, укажите имя списка, за которым следует индекс элемента (то есть *номер его позиции*) в квадратных скобках ([]); эта конструкция называется *оператором индексирования*. На следующей диаграмме изображен список с именами его элементов:



Некоторые методы списков

<code>list.append(x)</code>	Добавляет элемент в конец списка
<code>list.extend(L)</code>	Расширяет список list, добавляя в конец все элементы списка L
<code>list.insert(i, x)</code>	Вставляет на i-ый элемент значение x
<code>list.remove(x)</code>	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
<code>list.pop([i])</code>	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
<code>list.index(x, [start [, end]])</code>	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
<code>list.count(x)</code>	Возвращает количество элементов со значением x
<code>list.sort([key=функция])</code>	Сортирует список на основе функции
<code>list.reverse()</code>	Разворачивает список
<code>list.copy()</code>	Поверхностная копия списка
<code>list.clear()</code>	Очищает список

Кортежи

Кортеж (tuple) – тип последовательности для представления упорядоченных коллекций элементов.

Кортежи неизменяемы, в них часто хранятся разнородные данные, но данные также могут быть однородными. Длина кортежа определяется количеством элементов в кортеже и не может изменяться во время выполнения программы.

Некоторые методы кортежей

1. **cmp(tuple1, tuple2)** - сравнение элементов двух кортежей;
2. **len(tuple)** - количество элементов в кортеже;
3. **max(tuple)** - получить наибольший элемент кортежа;
4. **min(tuple)** - получить наименьший элемент кортежа;
5. **sorted(tuple)** – отсортировать кортеж;
6. **tuple.index()** – получить индекс указанного элемента кортежа;
7. **tuple.count()** - получить количество вхождений в кортеж указанного элемента

Создание кортежа:

```
x = tuple()
```

```
X = ()
```

Словари

Словарь (dictionary) – неупорядоченная коллекция для хранения пар «ключ-значение», связывающих неизменяемые ключи со значениями (по аналогии с тем, как в традиционных словарях слова связываются с определениями).

Создание словаря:

```
x = dict()
X = {}
days_per_month = { 'January': 31, 'February': 28, 'March': 31}
```

Словарь связывает ключи со значениями. Каждому ключу соответствует конкретное значение

Ключи	Типы ключей	Значения	Типы значений
Названия стран	str	Коды стран в интернете	str
Целые числа	int	Римские числа	str
Штаты	str	Сельскохозяйственные продукты	список str
Пациенты в больнице	str	Физиологические параметры	кортеж int и float
Игроки в бейсбольной команде	str	Средняя результативность	float
Единицы измерения	str	Сокращения	str
Коды складского учета	str	Запасы на складе	int

Некоторые методы словарей

Метод	Значение
dict()	создание словаря
len()	возвращает число пар
clear()	удаляет все значения из словаря
copy()	создает псевдокопию словаря
deepcopy()	создает полную копию словаря
fromkeys()	создание словаря
get()	получить значение по ключу
has_key()	проверка значения по ключу
items()	возвращает список значений
iteritems()	возвращает итератор
keys()	возвращает список ключей
iterkeys()	возвращает итератор ключей
pop()	извлекает значение по ключу

Множества

Множество (set) – неупорядоченная коллекция для хранения пар уникальных неизменяемых элементов.

Множества изменяе́мы – вы можете добавлять и удалять элементы, но при этом элементы множеств должны оставаться неизменяе́мыми. Таким образом, элементами множеств не могут быть другие множества. Фиксированное множество неизменяе́мо – его невозможно изменить после создания.

Операции над множествами

Создание множества:

```
x = set()
```

Создание фиксируванного множества:

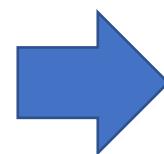
```
x = frozenset()
```

Методы и операции	Значение
<code>A B</code> <code>A.union(B)</code>	Возвращает множество, являющееся объединением множеств A и B.
<code>A = B</code> <code>A.update(B)</code>	Добавляет в множество A все элементы из множества B.
<code>A & B</code> <code>A.intersection(B)</code>	Возвращает множество, являющееся пересечением множеств A и B.
<code>A &= B</code> <code>A.intersection_update(B)</code>	Оставляет в множестве A только те элементы, которые есть в множестве B.
<code>A - B</code> <code>A.difference(B)</code>	Возвращает разность множеств A и B (элементы, входящие в A, но не входящие в B).
<code>A -= B</code> <code>A.difference_update(B)</code>	Удаляет из множества A все элементы, входящие в B.

Условный оператор if

```
if test1:  
    state1  
elif test2:  
    state2  
else:  
    state3
```

```
if X:  
    A = Y  
else:  
    A = Z
```



```
A = Y if X else Z
```

Цикл While и For

while выражение:

инструкция_1

инструкция_2

...

инструкция_n

a = 0

while a >= 0:

на выход **if** a == 7:

из цикла

break

a += 1

print("A")

a = 0

while a < 7:

print("A")

a += 1

a = -1

while a < 10:

a += 1

if a >= 7:

на
очередную
итерацию **print**("A")

for i **in** range(5):

print("Hello")

range() – создание серии целых числе

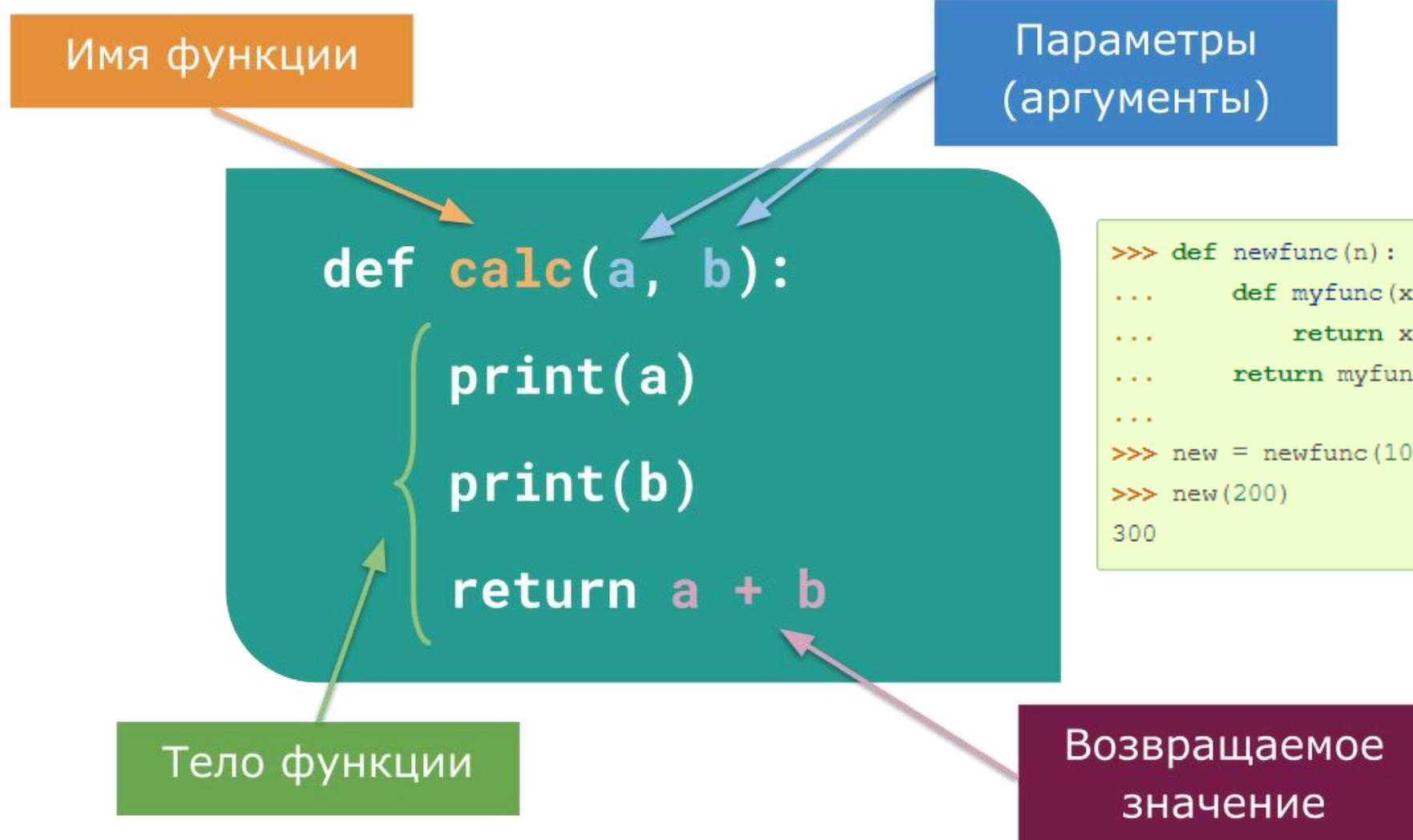
- 1 **range(стоп)** берет один аргумент [0, стоп)
- 2 **range(старт, стоп)** берет два аргумента [старт, стоп)
- 3 **range(старт, стоп, шаг)** берет три аргумента

word_str = "Hello, world!"

for l **in** word_str:

print(l)

Функции в Python



```
>>> def newfunc(n):
...     def myfunc(x):
...         return x + n
...     return myfunc
...
>>> new = newfunc(100)  # new - это функция
>>> new(200)
300
```

Анонимные функции (лямда-функции)

Анонимные функции могут содержать лишь одно выражение, но и выполняются они быстрее. Анонимные функции создаются с помощью инструкции `lambda`. Кроме этого, их не обязательно присваивать переменной, как делали мы инструкцией `def func():`

>>>

```
>>> func = lambda x, y: x + y
>>> func(1, 2)
3
>>> func('a', 'b')
'ab'
>>> (lambda x, y: x + y)(1, 2)
3
>>> (lambda x, y: x + y)('a', 'b')
'ab'
```

```
>>> func = lambda *args: args
>>> func(1, 2, 3, 4)
(1, 2, 3, 4)
```

Стандартные модули (библиотеки, пакеты) Python

`collections` — дополнительные структуры данных помимо списков, кортежей, словарей и множеств.

`csv` — обработка файлов с данными, разделенными запятыми.

`datetime, time` — операции с датой и временем.

`decimal` — вычисления с фиксированной и плавающей точкой, включая финансовые вычисления.

`doctest` — простое модульное тестирование с использованием проверочных тестов и ожидаемых результатов, закодированных в doc-строках.

`json` — обработка формата JSON (JavaScript Object Notation) для использования с веб-сервисами и базами данных документов NoSQL.

`math` — распространенные математические константы и операции.

`os` — взаимодействие с операционной системой.

`queue` — структура данных, работающая по принципу «первым зашел, первым вышел» (FIFO).

`random` — псевдослучайные числа.

`re` — регулярные выражения для поиска по шаблону.

`sqlite3` — работа с реляционной базой данных SQLite.

`statistics` — функции математической статистики (среднее, медиана, дисперсия, moda и т. д.).

`string` — обработка строк.

`sys` — обработка аргументов командной строки; потоки стандартного ввода, стандартного вывода; стандартный поток ошибок.

`timeit` — анализ быстродействия.

Библиотеки для Data Science

Научные вычисления и статистика

NumPy (Numerical Python) — в Python нет встроенной структуры данных массива. В нем используются списки — удобные, но относительно медленные. NumPy предоставляет высокопроизводительную структуру данных ndarray для представления списков и матриц, а также функции для обработки таких структур данных.

SciPy (Scientific Python) — библиотека SciPy, встроенная в NumPy, добавляет функции для научных вычислений: интегралы, дифференциальные уравнения, расширенная обработка матриц и т. д. Сайт scipy.org обеспечивает поддержку SciPy и NumPy.

StatsModels — библиотека, предоставляющая функциональность оценки статистических моделей, статистических критериев и статистического анализа данных.

Машинное обучение, глубокое обучение и обучение с подкреплением

scikit-learn — ведущая библиотека машинного обучения. Машинное обучение является подмножеством области искусственного интеллекта, а глубокое обучение — подмножеством машинного обучения, ориентированным на нейронные сети.

Keras — одна из самых простых библиотек глубокого обучения. Keras работает на базе TensorFlow (Google), CNTK (когнитивный инструментарий Microsoft для глубокого обучения) или Theano (Монреальский университет).

TensorFlow — самая популярная библиотека глубокого обучения от Google. TensorFlow использует графические процессоры или тензорные процессоры Google для повышения быстродействия. TensorFlow играет важную роль в областях искусственного интеллекта и аналитики больших данных с их огромными требованиями к вычислительным мощностям. Мы будем использовать версию Keras, встроенную в TensorFlow.

OpenAI Gym — библиотека и среда для разработки, тестирования и сравнения алгоритмов обучения с подкреплением.

Анализ и управление данными

pandas — чрезвычайно популярная библиотека для управления данными. В pandas широко используется структура ndarray из NumPy. Ее две ключевые структуры данных — Series (одномерная) и DataFrames (двумерная).

Визуализация

Matplotlib — библиотека визуализации и построения диаграмм с широкими возможностями настройки. Среди прочего в ней поддерживаются обычные графики, точечные диаграммы, гистограммы, контурные и секторные диаграммы, графики поля направлений, полярные системы координат, трехмерные диаграммы и текст.

Seaborn — высокоуровневая библиотека визуализации, построенная на базе Matplotlib. Seaborn добавляет более качественное оформление и дополнительные функции для работы с DataFrames.

Обработка естественного языка (NLP, Natural Language Processing)

NLTK (Natural Language Toolkit) — используется для задач обработки естественного языка (NLP).

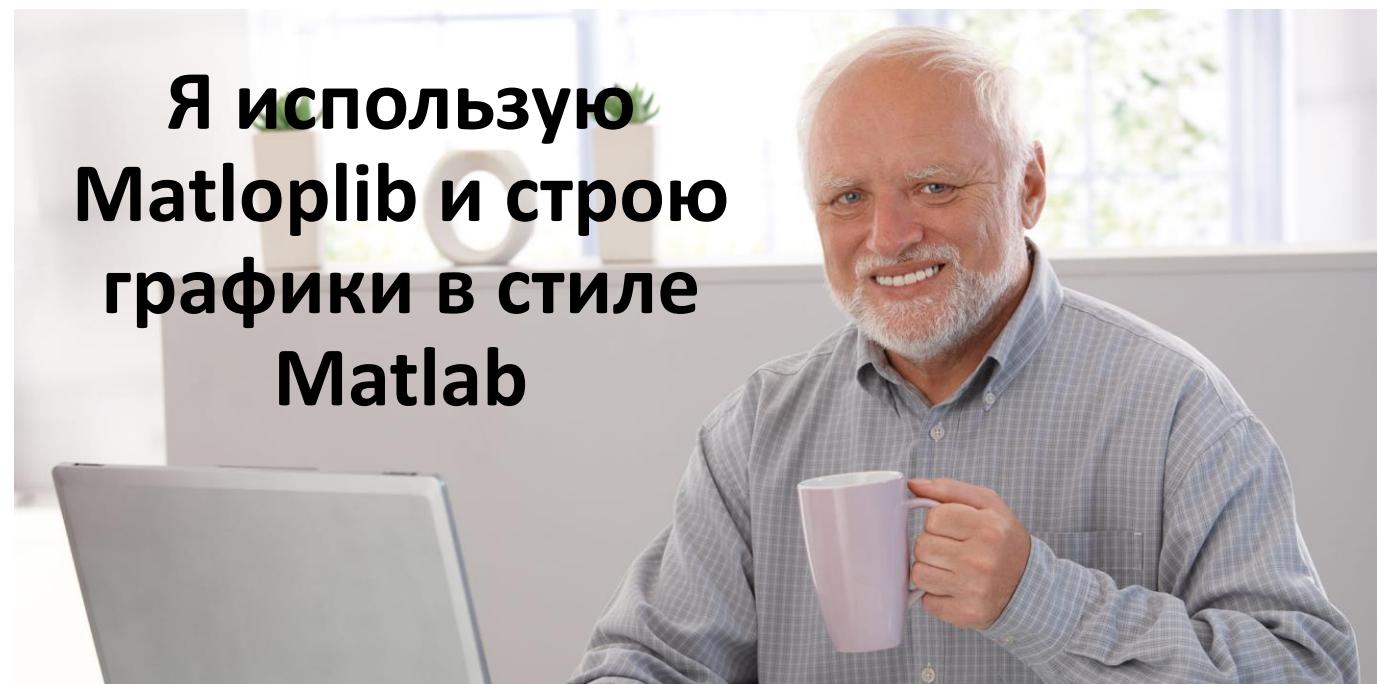
TextBlob — объектно-ориентированная NLP-библиотека обработки текста, построенная на базе библиотек NLTK и NLIP с паттернами. TextBlob упрощает многие задачи NLP.

Gensim — похожа на NLTK. Обычно используется для построения индекса для коллекции документов и последующего определения его схожести с остальными документами в индексе.

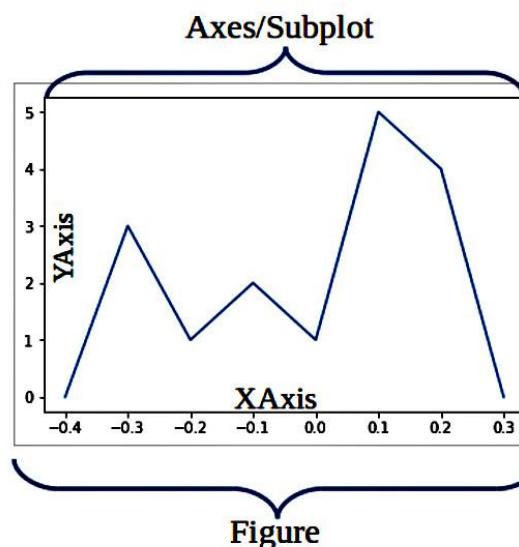
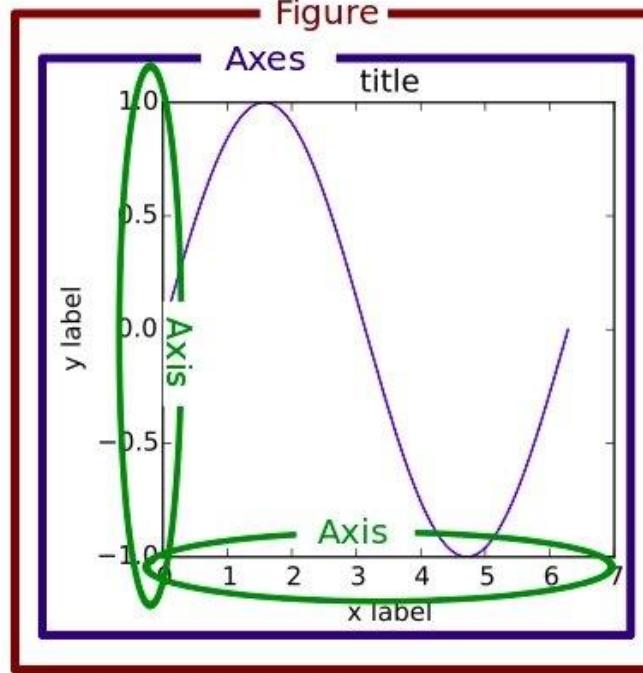
Графика в Python

Matplotlib – мультиплатформенная библиотека для визуализации данных, основанная на массивах библиотеки NumPy

Наиболее используемый модуль Matplotlib – это Pyplot, который предоставляет интерфейс, подобный MATLAB, но вместо этого он использует Python и является открытым исходным кодом.



Иерархия объектов matplotlib

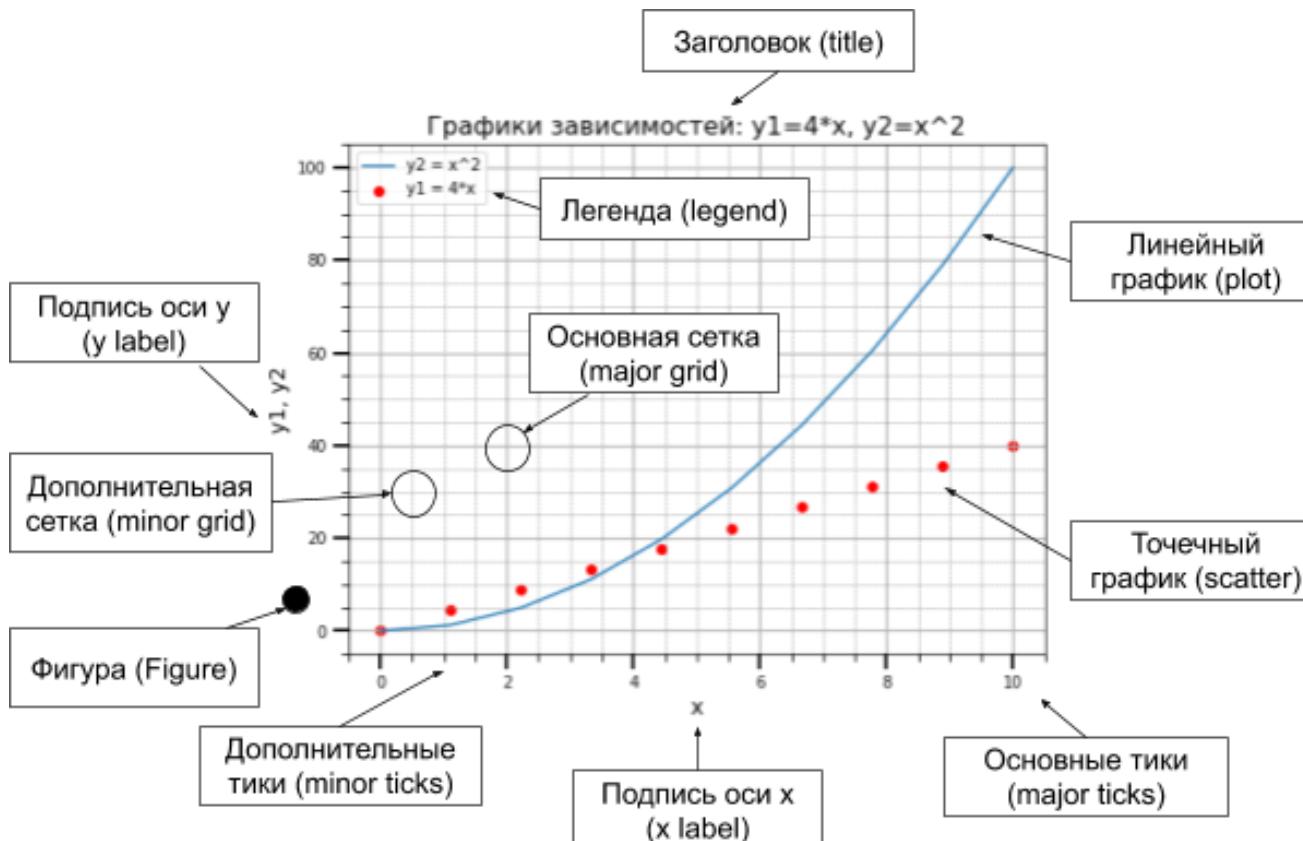


Объект **Figure** – это самый важный внешний контейнер для графики matplotlib, который может включать в себя несколько объектов **Axes**

Figure - это контейнер самого верхнего уровня, та область на которой все нарисовано. Таких областей может быть несколько, каждая из которых может содержать несколько контейнеров **Axes**.

Axes - это та область на которой чаще всего и отражаются графики (данные в виде графиков), а также все вспомогательные атрибуты (линии сетки, метки, указатели и т.д.). Часто, установка этой области сопровождается с вызовом *subplot*, который и помещает **Axes** на регулярную сетку. Поэтому, так же часто **Axes** и *Subplot* можно считать синонимами. Каждая область **Axes** содержит **XAxis** и **YAxis**. Они содержат, деления, метки и прочие вспомогательные атрибуты

Основные элементы графика



```

1. import matplotlib.pyplot as plt
2. from matplotlib.ticker import (MultipleLocator, FormatStrFormatter)
3. 
4. AutoMinorLocator()
5. 
6. import numpy as np
7. x = np.linspace(0, 10, 10)
8. y1 = 4*x
9. y2 = [i**2 for i in x]
10. 
11. fig, ax = plt.subplots(figsize=(8, 6))
12. 
13. ax.set_title("Графики зависимостей:  $y_1=4*x$ ,  $y_2=x^2$ ", fontsize=16)
14. ax.set_xlabel("x", fontsize=14)
15. ax.set_ylabel("y1, y2", fontsize=14)
16. ax.grid(which="major", linewidth=1.2)
17. ax.grid(which="minor", linestyle="--", color="gray", linewidth=0.5)
18. ax.scatter(x, y1, c="red", label="y1 = 4*x")
19. ax.plot(x, y2, label="y2 = x2")
20. ax.legend()
21. 
22. ax.xaxis.set_minor_locator(AutoMinorLocator())
23. ax.yaxis.set_minor_locator(AutoMinorLocator())
24. ax.tick_params(which='major', length=10, width=2)
25. ax.tick_params(which='minor', length=5, width=1)
26. 
27. plt.show()

```

Контрольные вопросы:

1. Назовите 4 основные характеристики больших данных, раскройте их содержание.
2. Дайте определение интеллектуального анализа данных.
3. Назовите особенности синтаксиса языка Python, основные ключевые слова
4. Назовите перечисляемые типы данных Python. Назовите их особенности и различия.
5. Дайте сравнительную характеристику Python и R.