

Лекция 1

Введение в компьютерные вычисления

Задачи курса

- Уверенное умение «заставлять компьютер делать то, что ты хочешь»
- Развитие алгоритмического мышления
- Владение искусством решения основных вычислительных задач

А что умеет делать компьютер?

- Главным образом, компьютер:
 - Выполняет вычисления
 - Помнит их результаты
- Какие вычисления?
 - Встроенные «примитивные» операции
 - Созданные на их основе собственные вычислительные методы
- И это всё?

А что умеет делать компьютер?

- Главным образом, компьютер:
 - Выполняет вычисления
 - Помнит их результаты
- Какие вычисления?
 - Встроенные «примитивные» операции
 - Созданные на их основе собственные вычислительные методы
- И это всё?

ДА!

А что умеет делать компьютер?

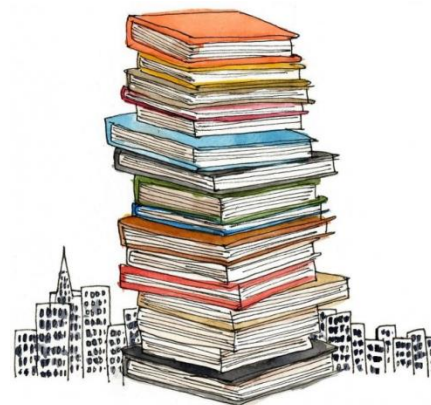
- **Да, но дело в скорости и объемах...**

- 2 ГГц



???

- Хранилище
объемом в сотни и тысячи Гб



1 Тб = ? ??

Устный (?) счет

1. Свет от лампы до стола

Скорость света = 300 000 км/с = 3×10^8 м/с

Расстояние = 30 см = 3×10^{-1} м

Время = расстояние/скорость = 10^{-9} с

Гигагерц = миллиард Герц = 10^9 (раз в секунду). **Итог: 2 операции**

2. Падение мячика

$$s = gt^2/2$$

$$s = 0,3 \text{ м} \quad g = 9,8 \text{ м/с}^2$$

$$t = \sqrt{(2 s / g)} = \sqrt{(0.061)} = 0.247 = (\text{грубо}) 0.25$$

$$0.25 \cdot 2 \cdot 10^9 = 5 \cdot 10^8$$

3. Стопка книг

500 страниц = 5 см, 1000 страниц = 10 см, 10 000 страниц = 1 м

1 страница = 40 строк*80 букв = 3 Кб. Ну ок, 5 Кб.

1 м = 50 000 Кб = (грубо) 50 Мб

1 Гб = 20*50 Мб = 20 м

1 Тб = 20 000 м

А что умеет делать компьютер?

- Да, но дело в скорости и объемах...
 - 2 ГГц

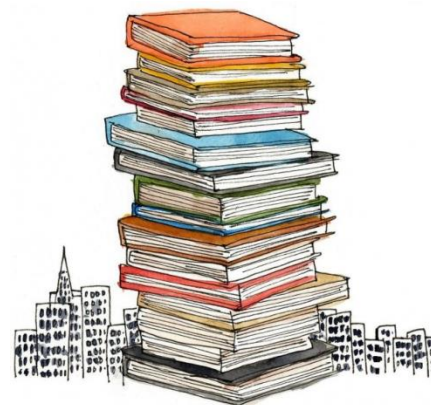


2 операции



500 000 000
операций

- Хранилище
объемом в сотни и тысячи Тб



1 Тб = 20 км

Достаточно ли этих простых вычислений?

- Веб-поиск
 - 45 000 000 000 индексируемых страниц (2015 год)
 - ~ 1000 слов на странице
 - 20 элементарных операций по проверке слова (реально – гораздо больше)
- **Попробуйте оценить время поиска нужных страниц простым перебором всех слов**

Достаточно ли этих простых вычислений?

- Веб-поиск
 - 45 000 000 000 индексируемых страниц (2015 год)
 - ~ 1000 слов на странице
 - 20 элементарных операций по проверке слова (реально – гораздо больше)
- **Попробуйте оценить время поиска нужных страниц простым перебором всех слов**

$$45 \cdot 10^9 \cdot 10^3 \cdot 20 = 9 \cdot 10^{14} \text{ операций}$$

«Супер-сервер Гугл» (1000 процессоров по 2ГГц)

$$2 \cdot 10^9 \cdot 10^3 = 2 \cdot 10^{12} \text{ операций в секунду}$$

Делим, получаем $4,5 \cdot 10^2$ секунд = 450 : 60 = 7.5 минуты

Достаточно ли этих простых вычислений?

- Веб-поиск
 - 45 000 000 000 индексируемых страниц (2015 год)
 - ~ 1000 слов на странице
 - 20 элементарных операций по проверке слова (реально – гораздо больше)
- **Попробуйте оценить время поиска нужных страниц простым перебором всех слов**

Делим, получаем $4,5 \cdot 10^2$ секунд = 450 : 60 = 7.5 минуты

Google (если верить Википедии) обрабатывает 41,345 млрд запросов в месяц

То есть, примерно 15 тысяч запросов в секунду

Достаточно ли этих простых вычислений?

- Можно ли в памяти (супер)компьютера сохранить информацию о положениях и скоростях атомов одного моля вещества?

$$N_A = 6,022\ 140\ 76 \cdot 10^{23} \text{ моль}^{-1}$$

Достаточно ли этих простых вычислений?

- Можно ли в памяти (супер)компьютера сохранить информацию о положениях и скоростях атомов одного моля вещества?

$$N_A = 6,022\ 140\ 76 \cdot 10^{23} \text{ моль}^{-1}$$

$$6 \cdot 10^{23} \cdot 6 \text{ (координаты и скорости)} \cdot 8 \text{ (байт на одно число)} = 2.88 \cdot 10^{25} \text{ (байт)}$$

$$\text{Терабайт (грубо)} = 10^3 \text{ (kilo)} \cdot 10^3 \text{ (mega)} \cdot 10^3 \text{ (giga)} \cdot 10^3 \text{ (tera)} = 10^{12} \text{ (байт)}$$

$$\text{Делим одно на другое: } 2 \cdot 10^{13} \text{ терабайт}$$

Будем все хранить на флешках, объемом 10 тб, вес 50 грамм

$$20 \text{ Тб} = 100 \text{ грамм}$$

$$200 \text{ Тб} = 1 \text{ кг}$$

$$2 \cdot 10^{13} / 200 = 10^{11} \text{ кг} = 10^8 \text{ тонн}$$

$$\text{Грузовой поезд} = 5000 \text{ тонн}$$

Нужно 20 000 грузовых поездов с «флешками»

Достаточно ли этих простых вычислений?

- **Вывод:** для того чтобы выполнить задачу, нужен еще хороший алгоритм и его реализация.

Пределы все-таки существуют?

- Несмотря на скорость и объемы хранения компьютер имеет ограничения двух типов:
 - Некоторые задачи до сих пор слишком сложные:
 - Точное предсказание погоды в заданной точке
 - Взлом схем шифрования
 - Некоторые задачи неразрешимы в принципе
 - Алгоритм определения, успешно ли закончится выполнение заданного программного кода для любого набора входных данных.
Посмотрим иллюстрацию (найдено в [YouTube](#))

Решение вычислительных задач

- Что такое вычисление?

Вычислѐние — математическое преобразование, позволяющее преобразовывать входящий поток информации в выходной, с отличной от первого структурой. Если смотреть с точки зрения теории информации, вычисление — это получение из входных данных нового знания.



Википедия
Свободная энциклопедия

- Что такое «знание»?

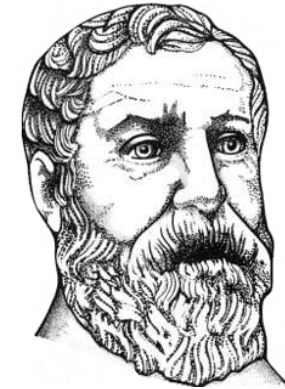
- Декларативное знание
 - Факты, утверждения
- Императивное знание
 - Методы, рецепты

Декларативное знание

- «Арифметическим квадратным корнем из неотрицательного числа X называется такое неотрицательное число Y , что $Y^2 = X$ »
- Можно ли использовать этот факт, чтобы найти квадратный корень из любого конкретного числа X ?

Императивное знание

- «Рецепт» вычисления квадратного корня с нужной точностью:
 - Начните с любого числа, g
 - Если g^2 достаточно близко к x , остановитесь и назовите g требуемым решением
 - В противном случае замените число g средним арифметическим чисел g и x/g
 - Повторяйте процесс, начиная с п. 2, пока не получите нужное значение



Герон Александрийский — греческий математик и механик. Время жизни отнесено ко второй половине I века н. э. на том основании, что он приводит в качестве примера лунное затмение 13 марта 62 г. н. э. Подробности его жизни неизвестны

Википедия

Пример

- Вычислим квадратный корень из 25

g	$g * g$	x / g	$\frac{1}{2} (g + x / g)$

Базовые «примитивы»

- А. Тьюринг доказал, что используя шесть примитивов, можно вычислить всё*
 - Машина Тьюринга
 - Полнота по Тьюрингу
- К счастью, в современных языках программирования примитивов побольше, и они поудобнее
- Кроме того в них есть способы создавать новые «примитивы»
- Но в любом случае: все, что вычислимо с помощью одного языка программирования, может быть вычислено с помощью любого другого.

* Что может быть вычислено



Как создавать рецепты?

- Каждый язык программирования предоставляет набор базовых операций (примитивов)
- Каждый язык программирования предоставляет механизм для объединения примитивов в более сложные, но допустимые выражения
- Каждый язык программирования предоставляет механизм для выяснения смысла или значения, связанного с каждым из таких выражений

Некоторые аспекты языка

- Базовые конструкции
 - **ЯП**: числа, строки, простые операторы
 - **Русский/Английский язык**: слова
- **Синтаксис** языка определяет, корректно ли построена заданная строка из букв и символов
 - ЯП:
 - $3.2 + 6.1$ – синтаксически корректное выражение на многих языках*
 - $3.2 * 6.1$ – синтаксически корректное выражение
 - $3.2 +^* 6.1$ – синтаксически некорректное выражение
 - Русский:
 - **Нынешнее лето, я студент ЮФУ.** – некорректное предложение

*Но не в русской версии MS Excel

Некоторые аспекты языка

- **Статическая семантика** определяет, имеет ли значение данная синтаксически верная строка.
 - **Английский язык.** Допустимый формат предложения
<существ.> <глагол> <артикль> <существ.>
Проверяем:
 - I am a student ОК
 - I are the champions Неправильно
 - **ЯП.** Допустимый формат выражения:
<literal> <operator> <literal>
Проверяем
 - 5 + 2 Ок
 - 5 + "Привет" Неправильно
 - x + y

Некоторые аспекты языка

- **Семантика** определяет, какое именно значение имеет та или иная синтаксически корректная строка символов, не содержащая статических семантических ошибок

Некоторые аспекты языка

- **Семантика** определяет, какое именно значение имеет та или иная синтаксически корректная строка символов, не содержащая статических семантических ошибок
- Это значение должно определяться **однозначно!**

Некоторые аспекты языка

- **Семантика** определяет, какое именно значение имеет та или иная синтаксически корректная строка символов, не содержащая статических семантических ошибок
- Это значение должно определяться **однозначно!**
- В «живых» языках с этим могут быть проблемы:
 - «У меня на обед сегодня будет друг...»

Некоторые аспекты языка

- **Семантика** определяет, какое именно значение имеет та или иная синтаксически корректная строка символов, не содержащая статических семантических ошибок
- Это значение должно определяться **однозначно!**
- В «живых» языках с этим могут быть проблемы:
 - «У меня на обед сегодня будет друг...»
- А если еще учитывать интонацию...
 - «Ну да, конечно!» - может означать и Да!, и Нет!

Некоторые аспекты языка

- **Семантика** определяет, какое именно значение имеет та или иная синтаксически корректная строка символов, не содержащая статических семантических ошибок
- Это значение должно определяться **однозначно!**
- В «живых» языках с этим могут быть проблемы:
 - «У меня на обед сегодня будет друг...»
- А если еще учитывать интонацию...
 - «Ну да, конечно!» - может означать и Да!, и Нет!
- В языках программирования семантически корректная строка может иметь **только одно значение**
 - Хотя иногда и совсем не такое, как предполагал программист 😊

Какие бывают проблемы

- Синтаксические ошибки
 - Очень распространены, но легко обнаруживаются компьютером
- Статические семантические ошибки
 - Некоторые ЯП тщательно проверяют их отсутствие до выполнения программы: другие осуществляют проверку в процессе интерпретации
 - Если ее не обнаружить, выполнение программы непредсказуемо
- Семантических ошибок нет, но значение одной из строк не соответствует запланированному
 - Крах программы (перестает работать)
 - Работает бесконечно
 - Выдает ответ, но он не соответствует ожиданиям

Чем будем заниматься?

- Изучим синтаксис и семантику одного из ЯП
- Научимся использовать эти элементы для перевода «рецептов» и «методик» решения задач в форму, позволяющую компьютеру их принять и поработать за нас
- Разовьем алгоритмический стиль мышления, познакомившись с набором техник и методов решения разнообразных вычислительных задач.

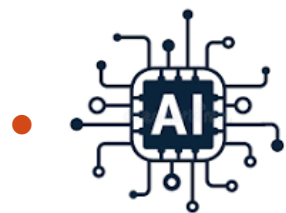
Почему Python?

Почему?

- Простой синтаксис, низкий «порог вхождения»
- Высокая скорость разработки
- Большое количество библиотек «на все случаи»
- Переносимость программ
- Доброжелательное сообщество
- Удовольствие 😊

Почему?

- Простой синтаксис, низкий «порог вхождения»
- Высокая скорость разработки
- Большое количество библиотек «на все случаи»
- Переносимость программ
- Доброжелательное сообщество
- Удовольствие 😊



TIOBE Index (август 2026)

Aug 2026	Aug 2025	Change	Programming Language		Ratings	Change
1	1			Python	18.53%	-7.61%
2	3	↑		C	11.10%	+2.07%
3	2	↓		C++	8.62%	-0.56%
4	4			Java	8.25%	-0.34%
5	5			C#	4.09%	-1.43%
6	6			JavaScript	2.63%	-0.52%
7	7			Visual Basic	2.18%	-0.15%
8	12	↑↑		SQL	1.88%	+0.16%
9	14	↑↑		R	1.56%	+0.19%
10	18	↑↑		Rust	1.45%	+0.31%
11	10	↓		Delphi/Object Pascal	1.37%	-0.46%

Не без недостатков. Но у кого их нет?

- **Низкая скорость выполнения (Производительность)**
 - Python — это интерпретируемый язык с динамической типизацией. Код выполняется построчно интерпретатором, а не компилируется в машинный код напрямую.
- **Высокое потребление памяти**
 - Каждая переменная это объект со служебной информацией
- **Проблемы с упаковкой и распространением приложений (Deployment)**
 - Здесь непросто создать единый исполняемый файл (.exe) или удобный для распространения пакет
- **Есть и другие...**

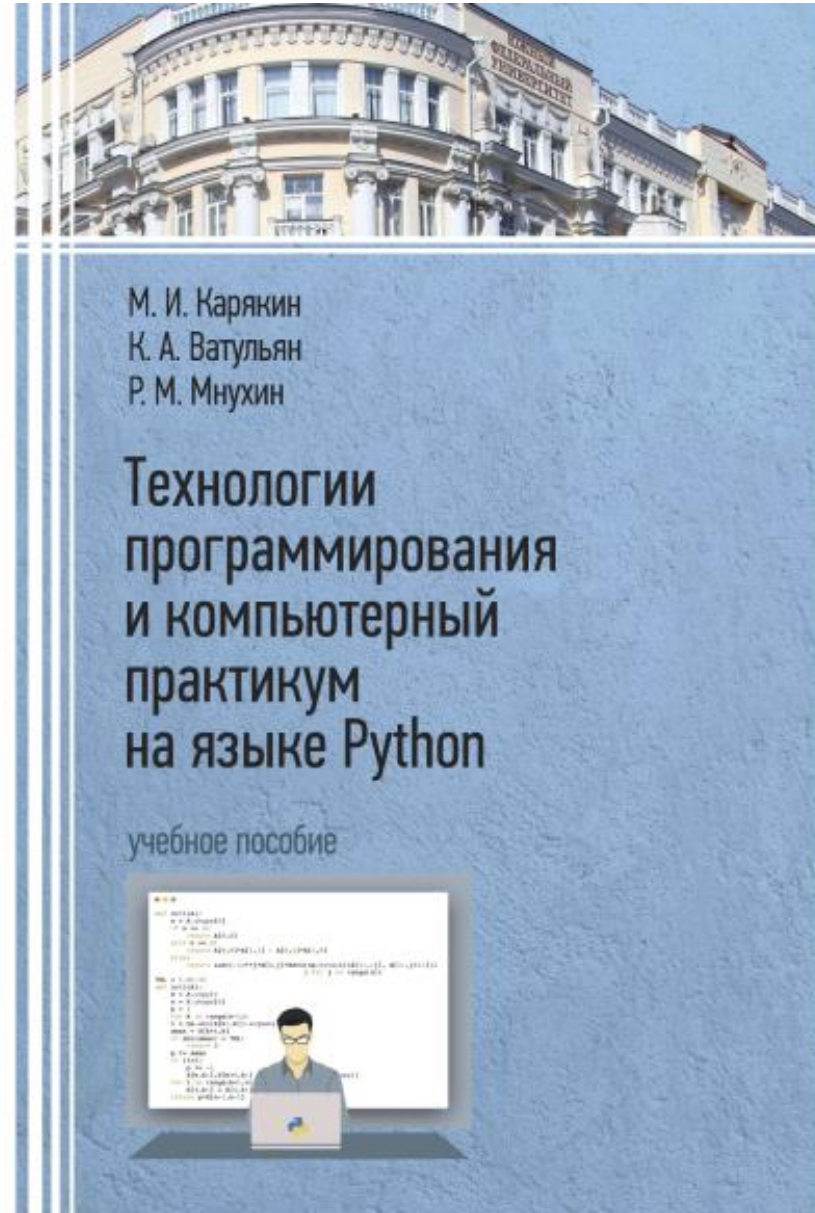
Не без недостатков. Но у кого их нет?

- **Низкая скорость выполнения (Производительность)**
 - Python — это интерпретируемый язык с динамической типизацией. Код выполняется построчно интерпретатором, а не компилируется в машинный код напрямую.
- **Высокое потребление памяти**
 - Каждая переменная это объект со служебной информацией
- **Проблемы с упаковкой и распространением приложений (Deployment)**
 - Здесь непросто создать единый исполняемый файл (.exe) или удобный для распространения пакет
- **Есть и другие...**

Для большинства областей, где Python доминирует (веб-разработка, Data Science, автоматизация, DevOps), эти недостатки либо не критичны, либо успешно нивелируются огромной экосистемой библиотек

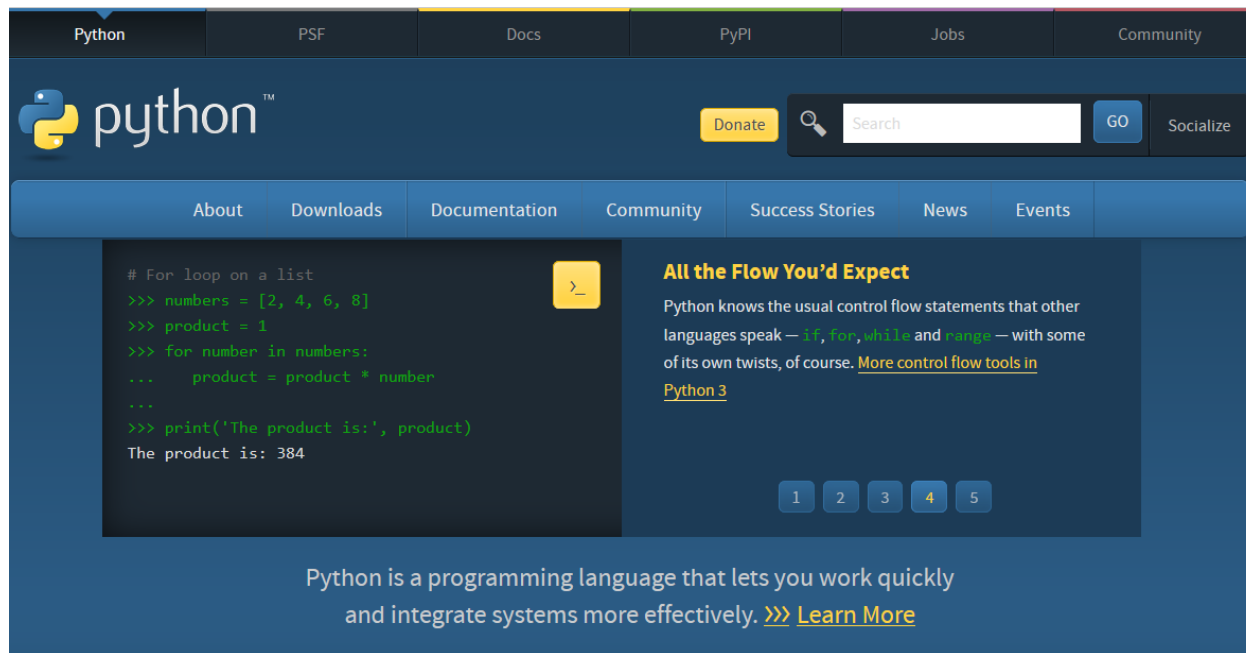
Что почитать? Что посмотреть?

- По нашему курсу есть учебное пособие
- Электронная версия доступна здесь
<https://www.elibrary.ru/item.asp?id=49349781>
(требуется регистрация в системе)



Где взять Python?

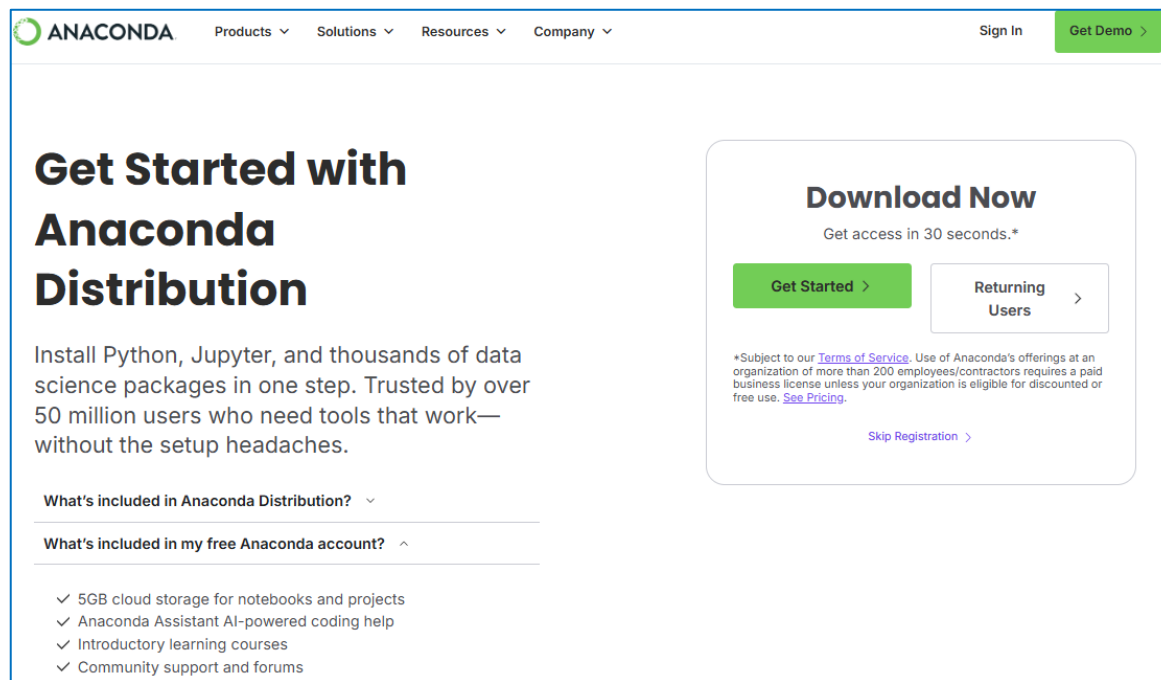
- Вариант 1. <https://www.python.org/>



- Плюсы:
 - Компактно
 - Быстро, нетребовательно к «железу»
- Минусы
 - Jupyter нужно ставить отдельно
 - Пакеты (для других курсов и нашего второго семестра) нужно ставить отдельно

Где взять Python?

- Вариант 2. <https://www.anaconda.com/download>



- Плюсы:
 - Установка «все в одном»
 - Все пакеты для нашего курса программирования уже есть
- Минусы
 - Высокие требования к «железу»
 - Нет 100% гарантии, что встроена нужная версия требуемого пакета