

Введение в язык Python



На что похоже создание программ?



Зачем нужны языки?



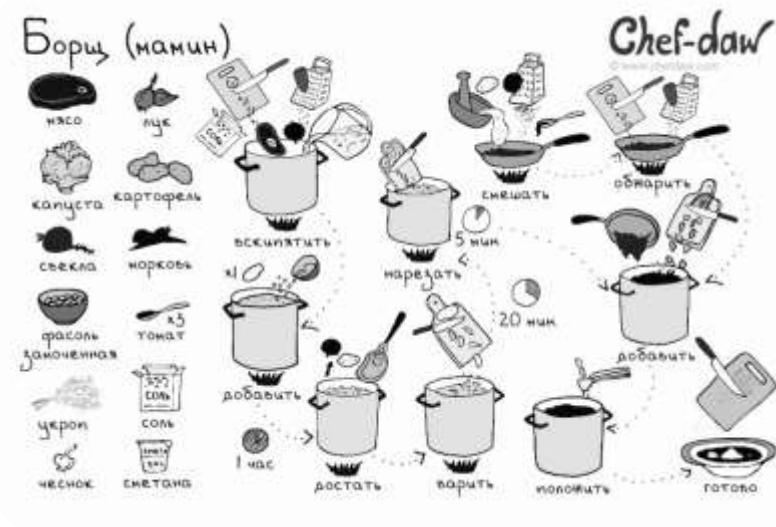
«Язык —
инструмент
мышления.

Ноам Хомский

Фото: Duncan Rawlinson; Цитата: chomsky.info



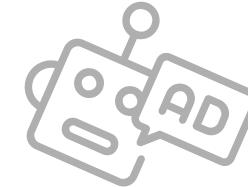
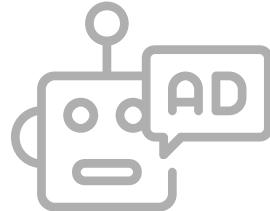
Зачем нужны языки?



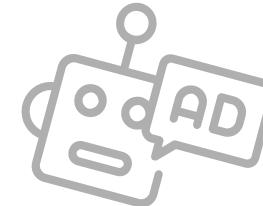
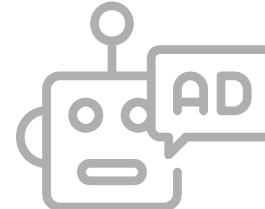
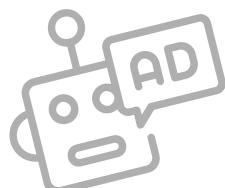
«Эксмо»; chefdaw.livejournal.com



Python – искусственный язык



- 1 Очень мало слов (около 50 на весь наш курс)
- 2 Строгие правила написания и оформления текстов



Как видит происходящее компьютер?

> разноцветный каподастр перерезал тонер

- 1 Оставить по 3 первые буквы в каждом слове
- 2 В полученных трёхбуквенных кусочках слов заменить все буквы «а» на символ @
- 3 Все буквы «о» заменить на цифру 0
- 4 Записать получившиеся трёхбуквенные комбинации через дефис «-»



Как видит происходящее компьютер?

> разноцветный каподастр перерезал тонер

1 Оставить по 3 первые буквы в каждом слове

2 раз кап пер тон

В полученных трёхбуквенных кусочках слов
заменить все буквы «а» на символ @

3

Все буквы «о» заменить на цифру 0

4

Записать получившиеся трёхбуквенные
комбинации через дефис «-»



Как видит происходящее компьютер?

> разноцветный каподастр перерезал тонер

- 1 Оставить по 3 первые буквы в каждом слове
раз кап пер тон
- 2 В полученных трёхбуквенных кусочках
слов заменить все буквы «а» на символ @
р@з к@п пер тон
- 3 Все буквы «о» заменить на цифру 0
- 4 Записать получившиеся трёхбуквенные
комбинации через дефис «-»



Как видит происходящее компьютер?

> разноцветный каподастр перерезал тонер

- 1 Оставить по 3 первые буквы в каждом слове
раз кап пер тон
- 2 В полученных трёхбуквенных кусочках слов
заменить все буквы «а» на символ @
р@з к@п пер тон
- 3 **Все буквы «о» заменить на цифру 0**
р@з к@п пер т0н
- 4 Записать получившиеся трёхбуквенные
комбинации через дефис «-»



Как видит происходящее компьютер?

> разноцветный каподастр перерезал тонер

- 1 Оставить по 3 первые буквы в каждом слове
раз кап пер тон
- 2 В полученных трёхбуквенных кусочках слов
заменить все буквы «а» на символ @
р@з к@п пер тон
- 3 Все буквы «о» заменить на цифру 0
р@з к@п пер т0н
- 4 Записать получившиеся трёхбуквенные
комбинации через дефис «-»
р@з-к@п-пер-т0н



Как видит происходящее компьютер?

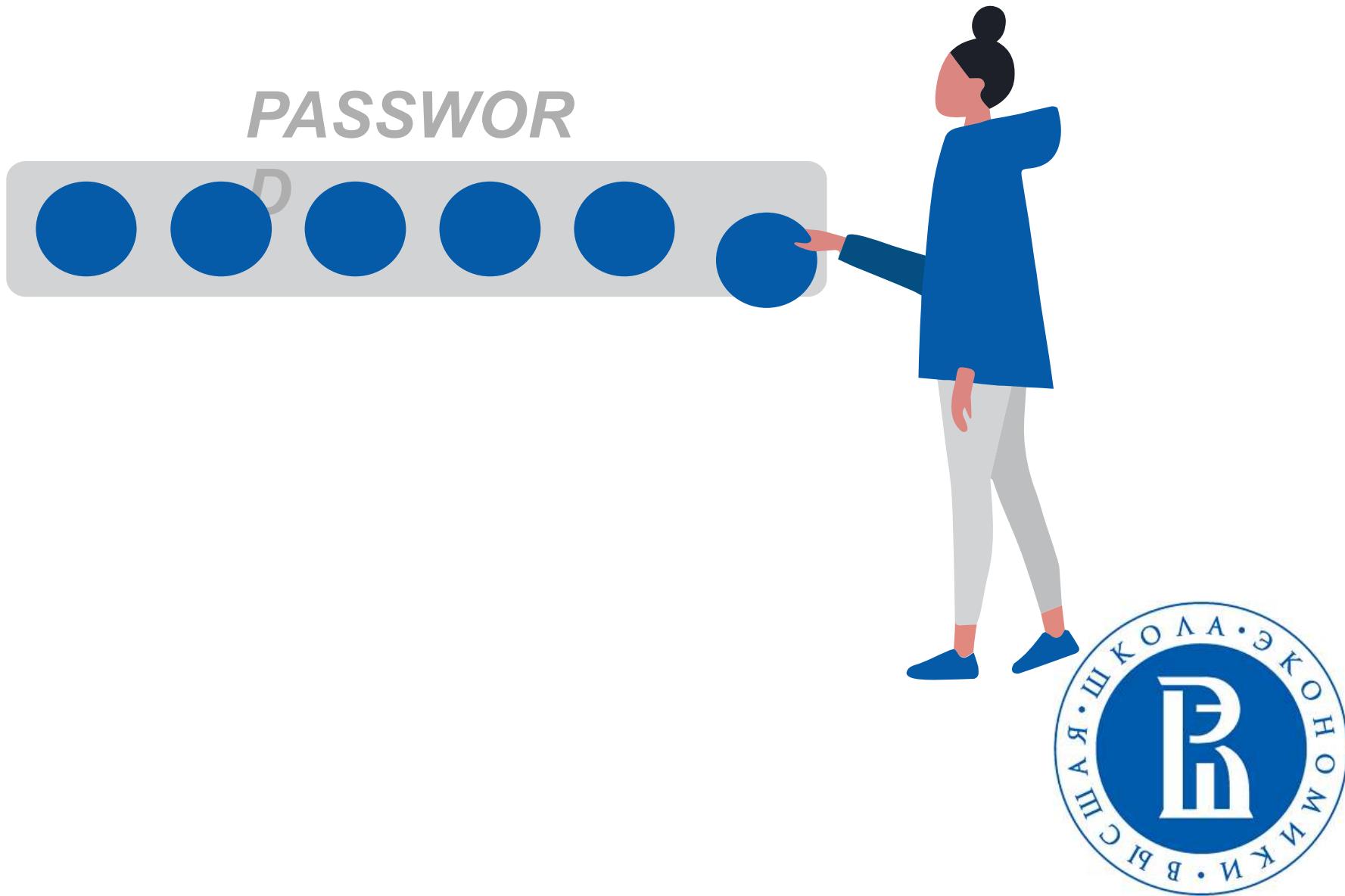
> разноцветный каподастр перерезал тонер

- 1 Оставить по 3 первые буквы в каждом слове
раз кап пер тон
- 2 В полученных трёхбуквенных кусочках слов
заменить все буквы «а» на символ @
р@з к@п пер тон
- 3 Все буквы «о» заменить на цифру 0
р@з к@п пер т0н
- 4 Записать получившиеся трёхбуквенные
комбинации через дефис «-»
р@з-к@п-пер-т0н

РЕЗУЛЬТАТ



Как видит происходящее компьютер?



Как видит происходящее компьютер?

Верхний удар рукой,
Нижний удар ногой,
Верхний удар ногой,
Нижний удар рукой,
Верхний удар рукой,
Верхний удар ногой,
Верхний удар ногой,
Верхний удар рукой,
Верхний удар рукой,
Нижний удар рукой,
Верхний удар рукой.



mortalkombat.fandom.com

«Brutality» у Саб-Зиро
в UMК3 для Sega
Genesis



Что будем делать мы?

1

Думать, как
бы я решил
этую задачу

2

Писать
решение
«для себя»

3

Переводить
решение
на
понятный
компьютеру
язык

4

Проверять
получившийся
исходный
код



Введение в язык Python



Подготовка рабочего места и первая программа



Из чего будет состоять рабочее место?



Для работы во время этого курса мы будем использовать комплект программ «Анаконда»



Из чего будет состоять рабочее место?



Для работы во время этого курса мы будем использовать комплект программ «Анаконда»



anaconda.com/products/individual#Downloads



Установка: скачивание программы



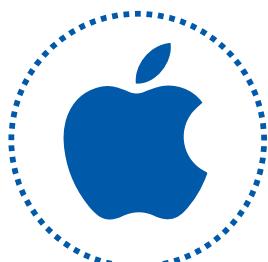
Windows

Python 3.7

- 64-Bit Graphical Installer (466 MB)
- 32-Bit Graphical Installer (423 MB)

Python 2.7

- 64-Bit Graphical Installer (413 MB)
- 32-Bit Graphical Installer (356 MB)



MacOS

Python 3.7

- 64-Bit Graphical Installer (442 MB)
- 64-Bit Command Line Installer (430 MB)

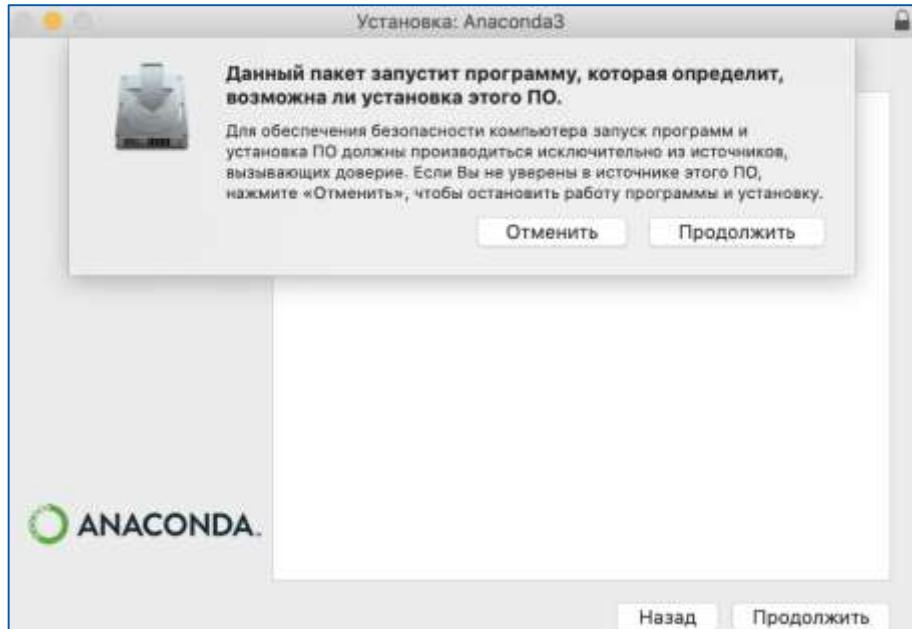
Python 2.7

- 64-Bit Graphical Installer (637 MB)
- 64-Bit Command Line Installer (409 MB)



ANACONDA INSTALLERS

Установка: запуск мастера установки



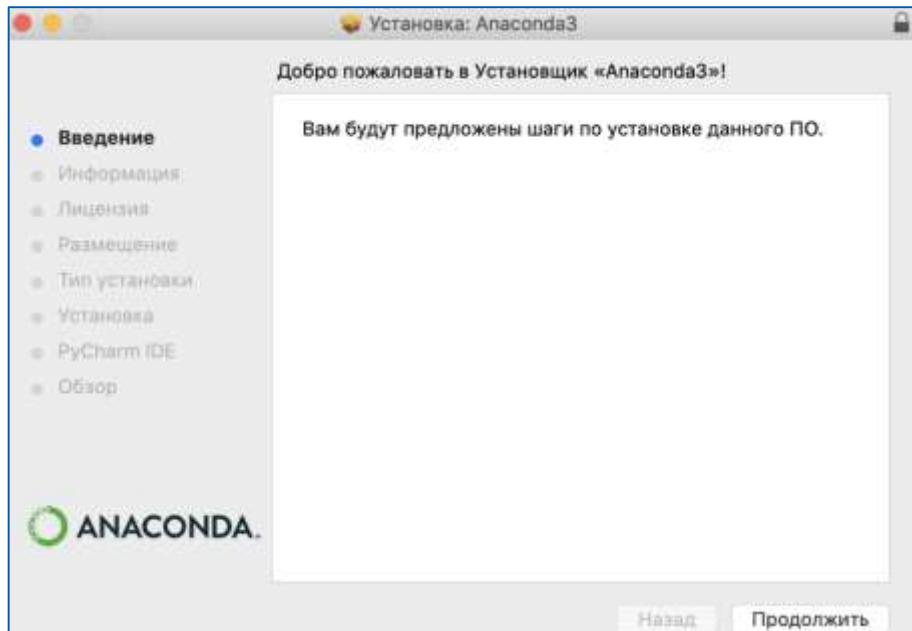
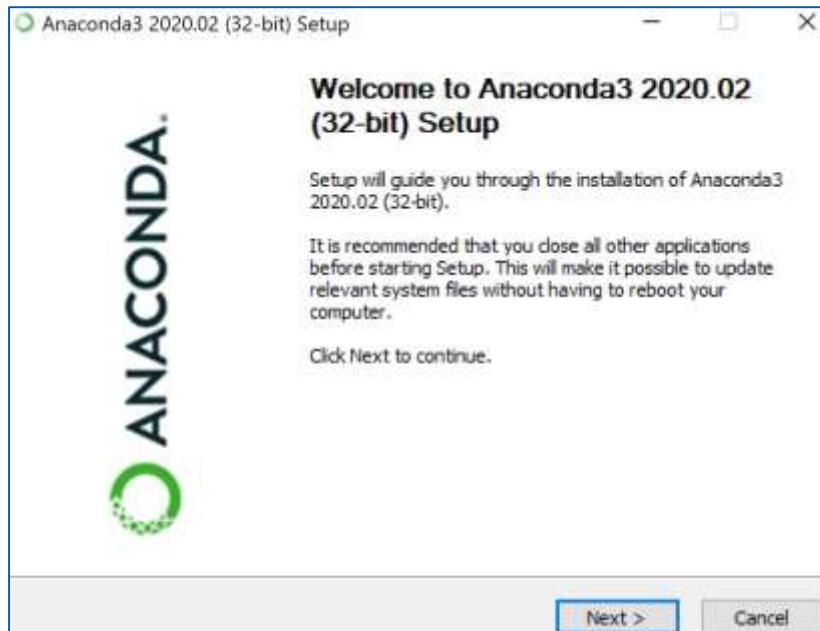
Для компьютеров
Windows



Для компьютеров
Apple Mac



Установка: запуск мастера установки



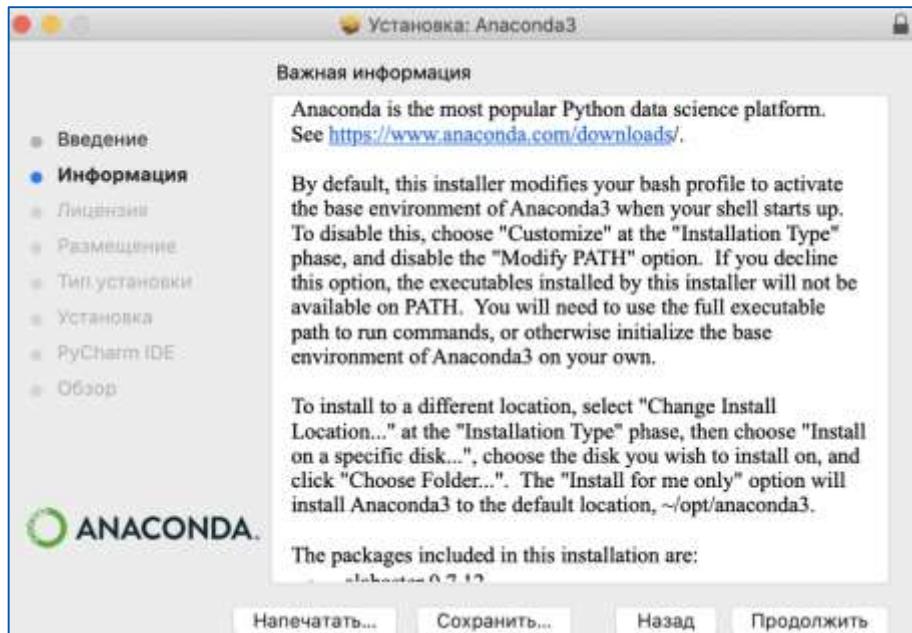
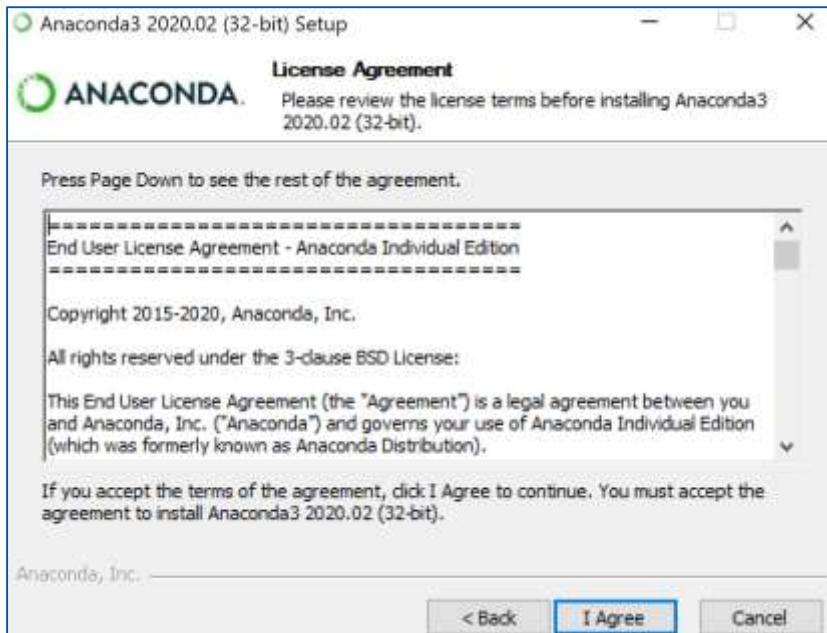
Для компьютеров
Windows



Для компьютеров
Apple Mac



Установка: лицензия



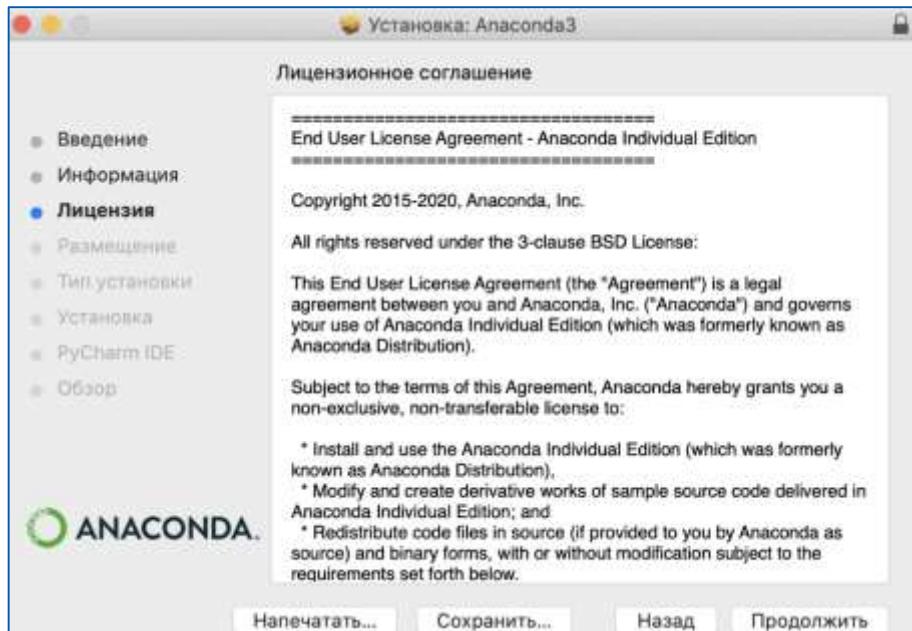
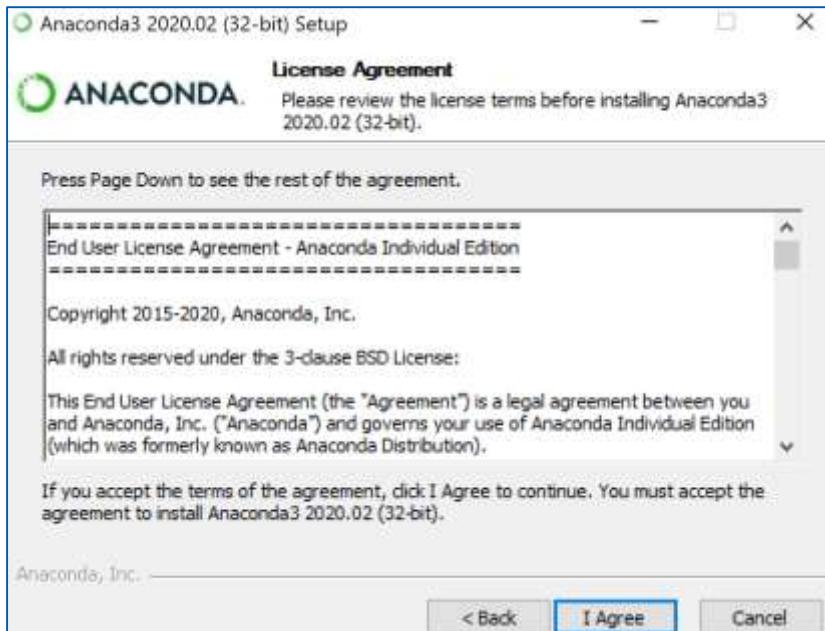
Для компьютеров
Windows



Для компьютеров
Apple Mac



Установка: лицензия



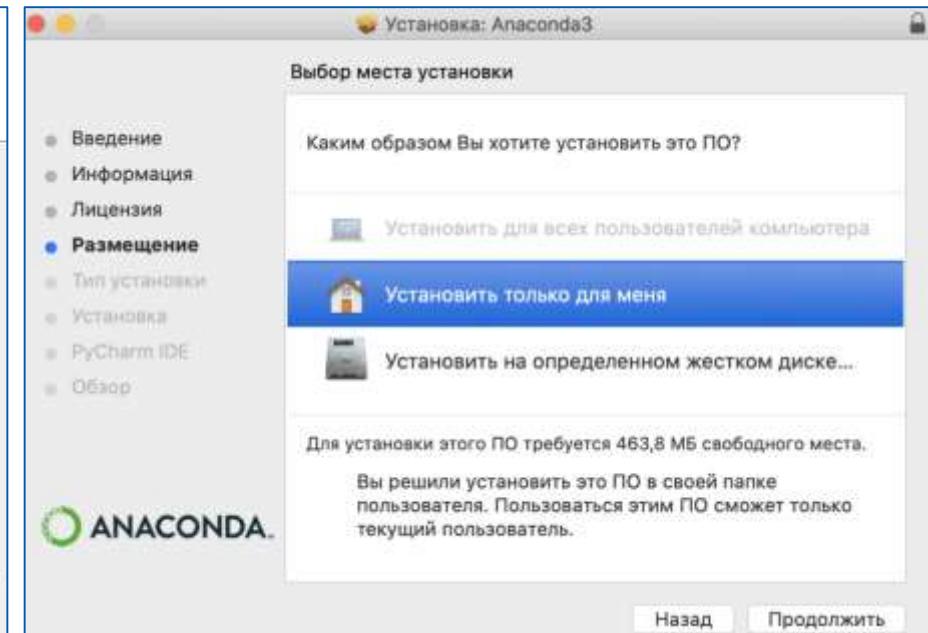
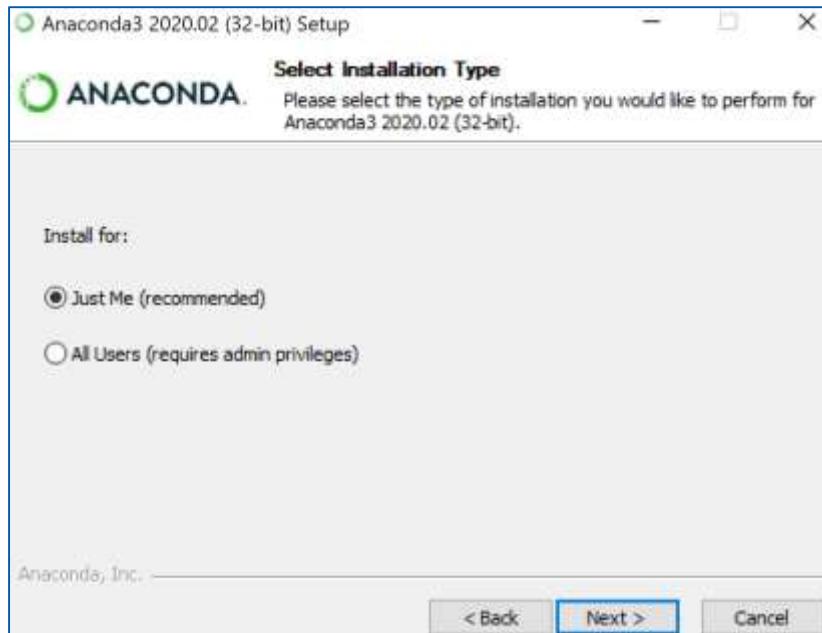
Для компьютеров
Windows



Для компьютеров
Apple Mac



Установка: размещение программы



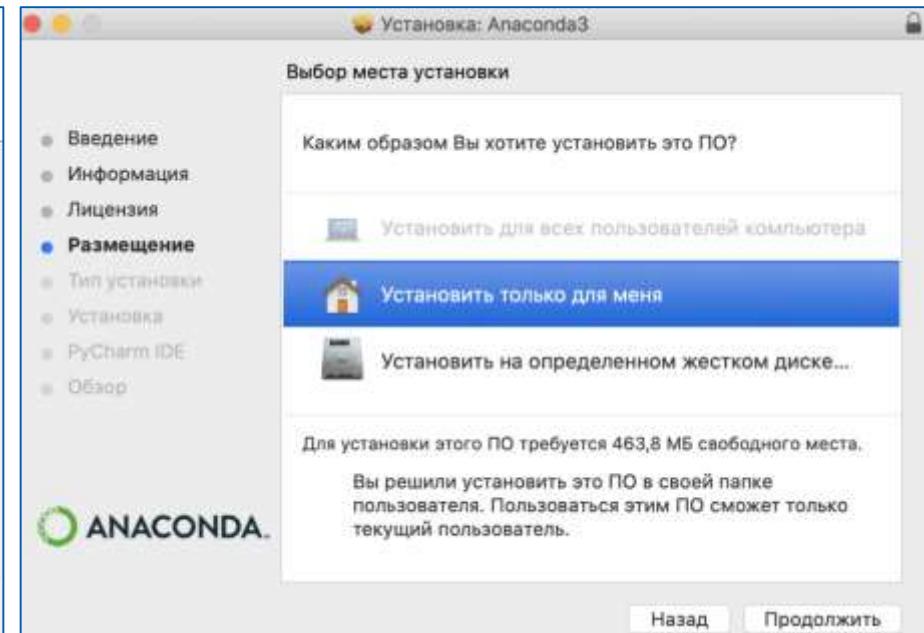
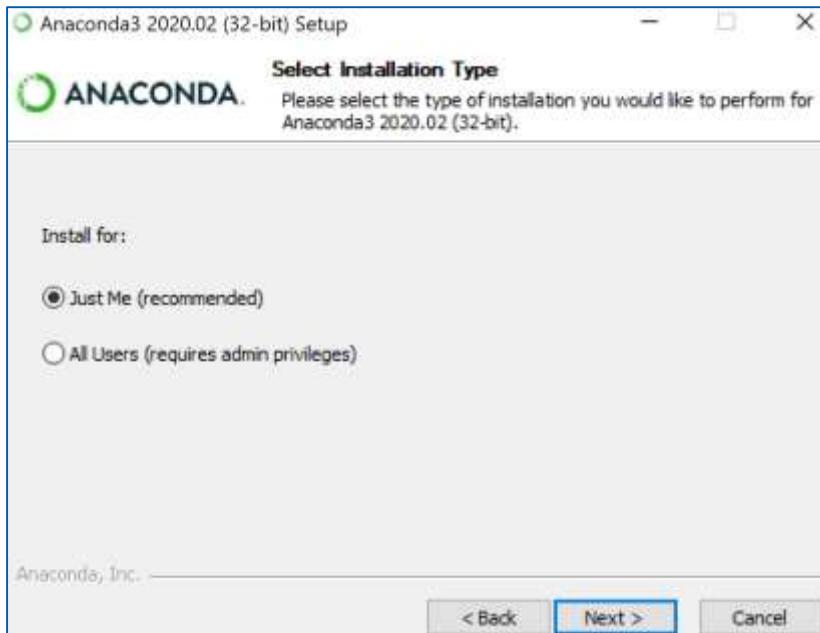
Для компьютеров
Windows



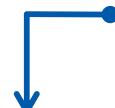
Для компьютеров
Apple Mac



Установка: размещение программы



Для компьютеров
Windows

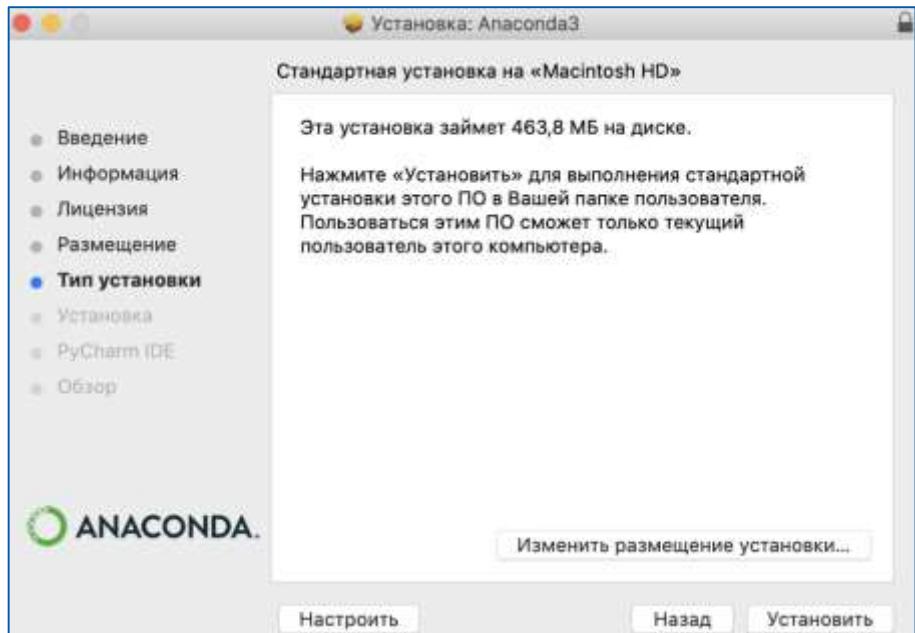
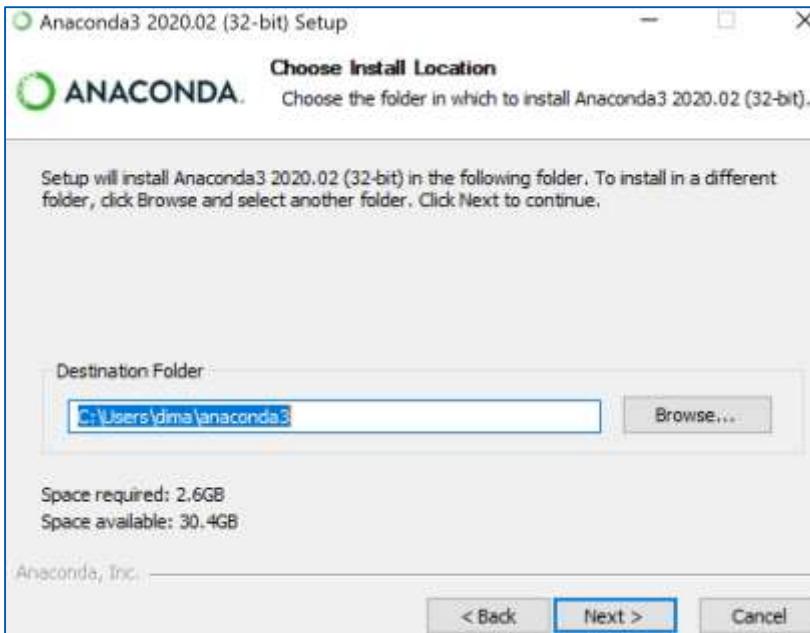


Для компьютеров
Apple Mac

Возможно, нужно будет нажать кнопку
«Установить только для меня»



Установка: размещение программы



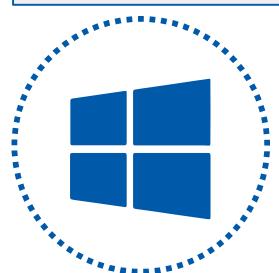
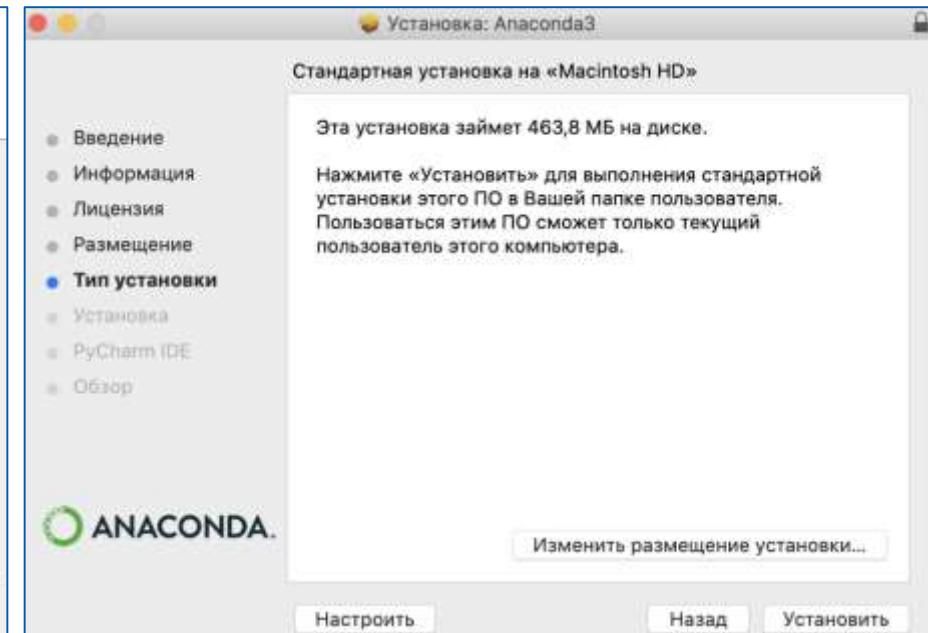
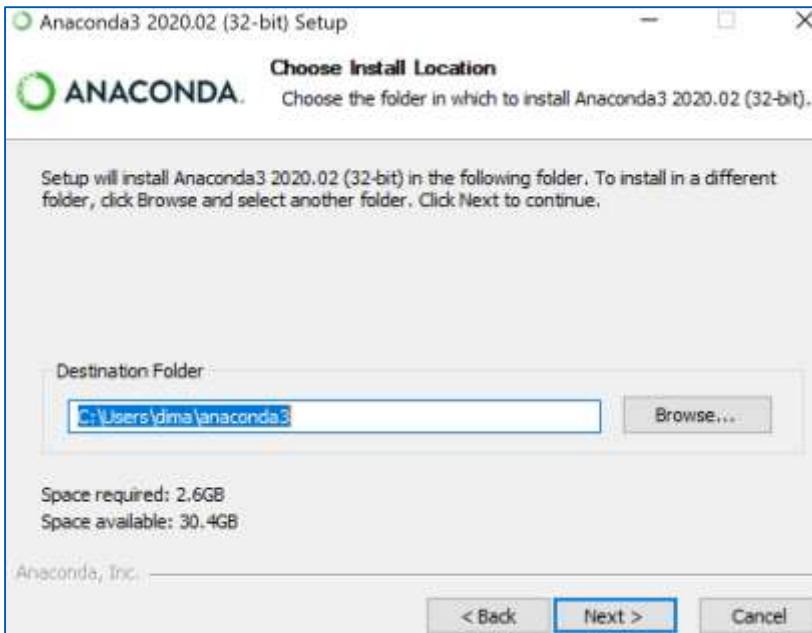
Для компьютеров
Windows



Для компьютеров
Apple Mac



Установка: размещение программы



Для компьютеров
Windows

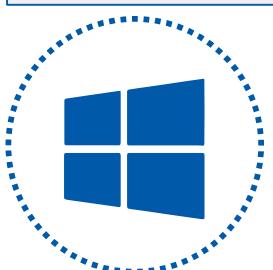
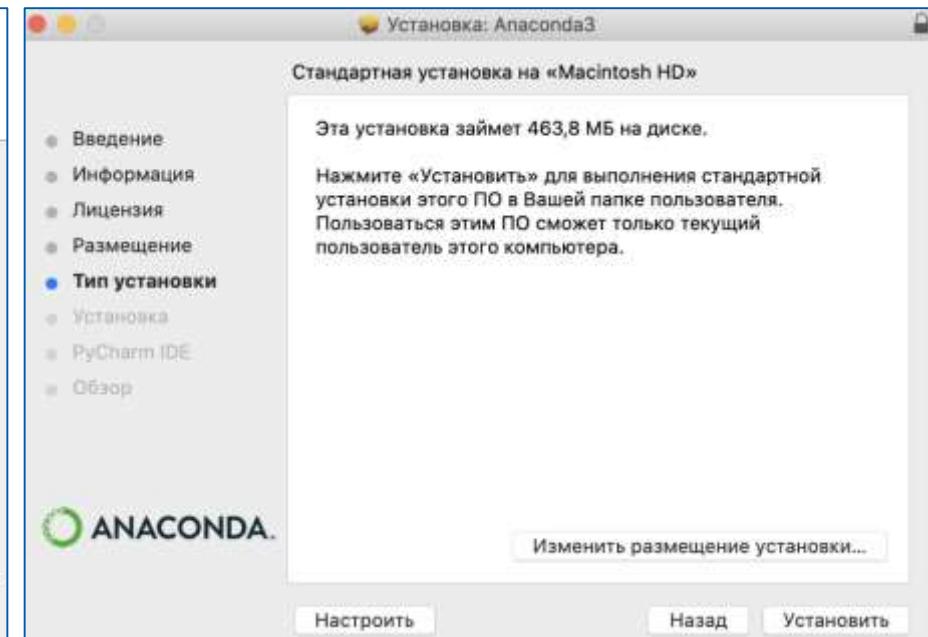
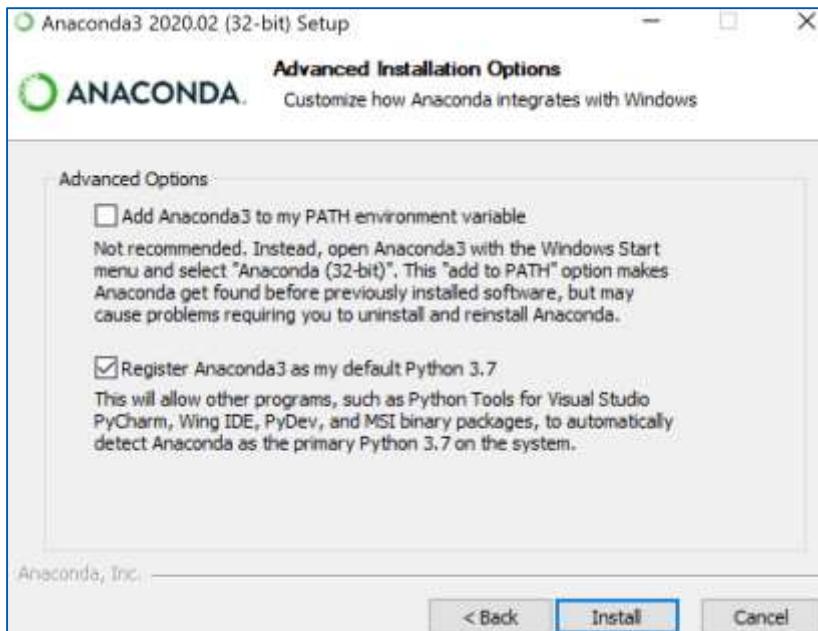


Для компьютеров
Apple Mac

Если высакивает ошибка, проверьте,
что в Destination Folder нет русских букв и пробелов



Установка: размещение программы



Для компьютеров
Windows



Для компьютеров
Apple Mac

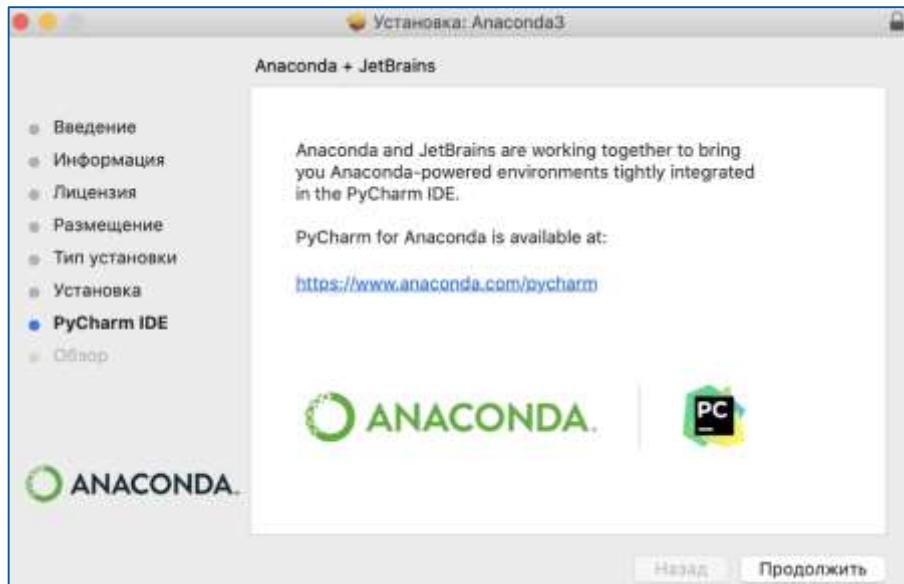
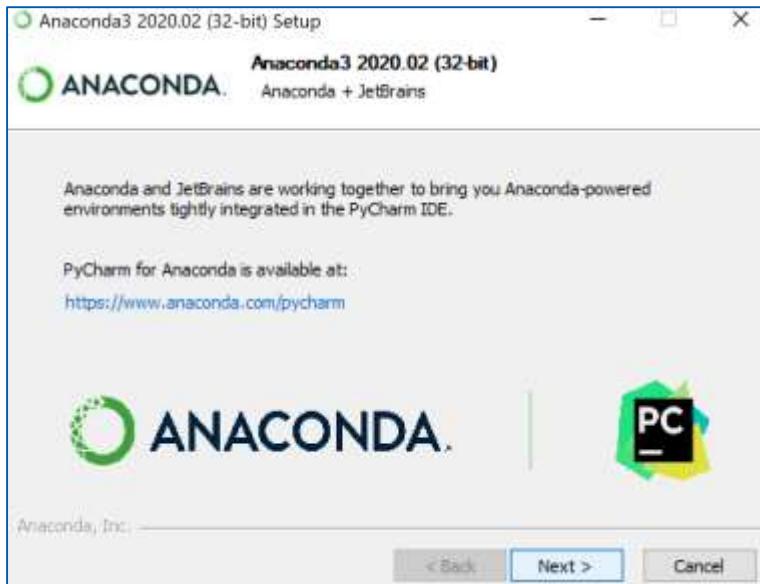


Что внутри?

- 1 Интерпретатор языка Python 3;
- 2 Специальная программа для написания текстов на Python 3 — **Jupyter Notebook**;
- 3 Много дополнительных программ для использования Python в анализе данных и машинном обучении.



Установка: завершение установки



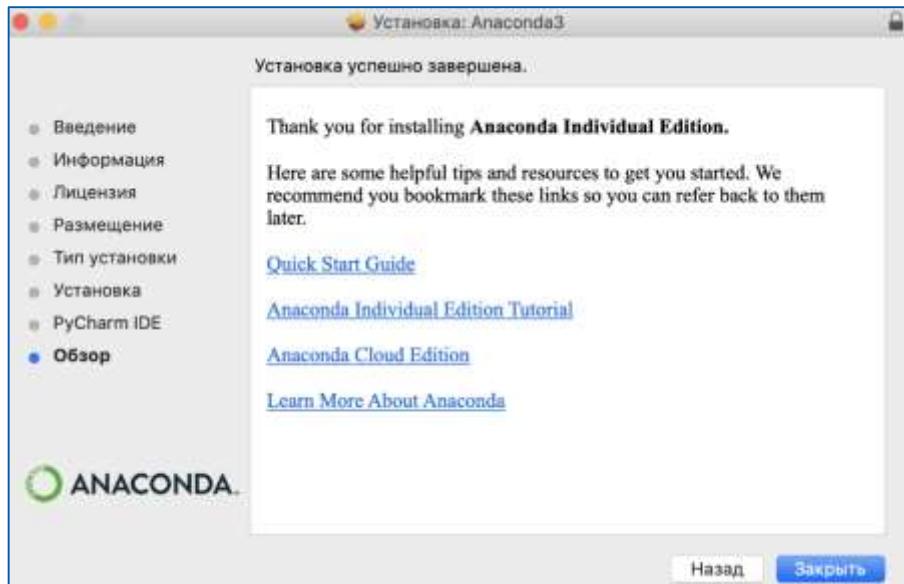
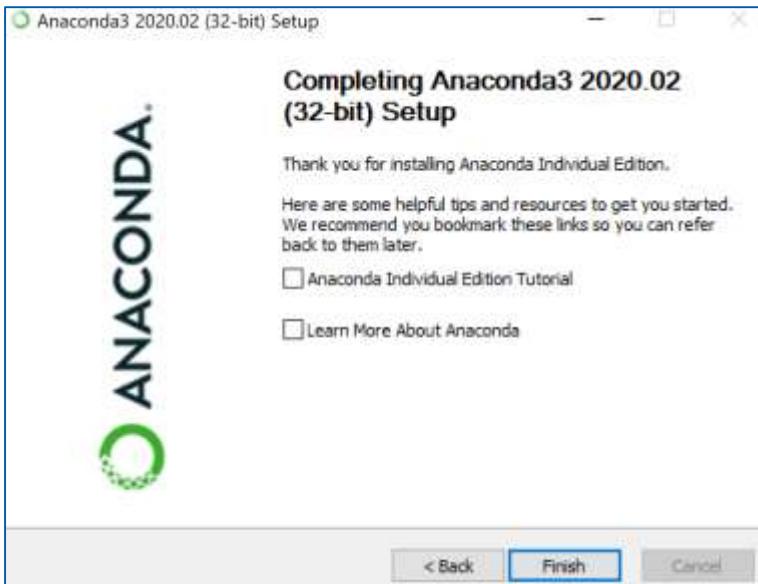
Для компьютеров
Windows



Для компьютеров
Apple Mac



Установка: завершение установки



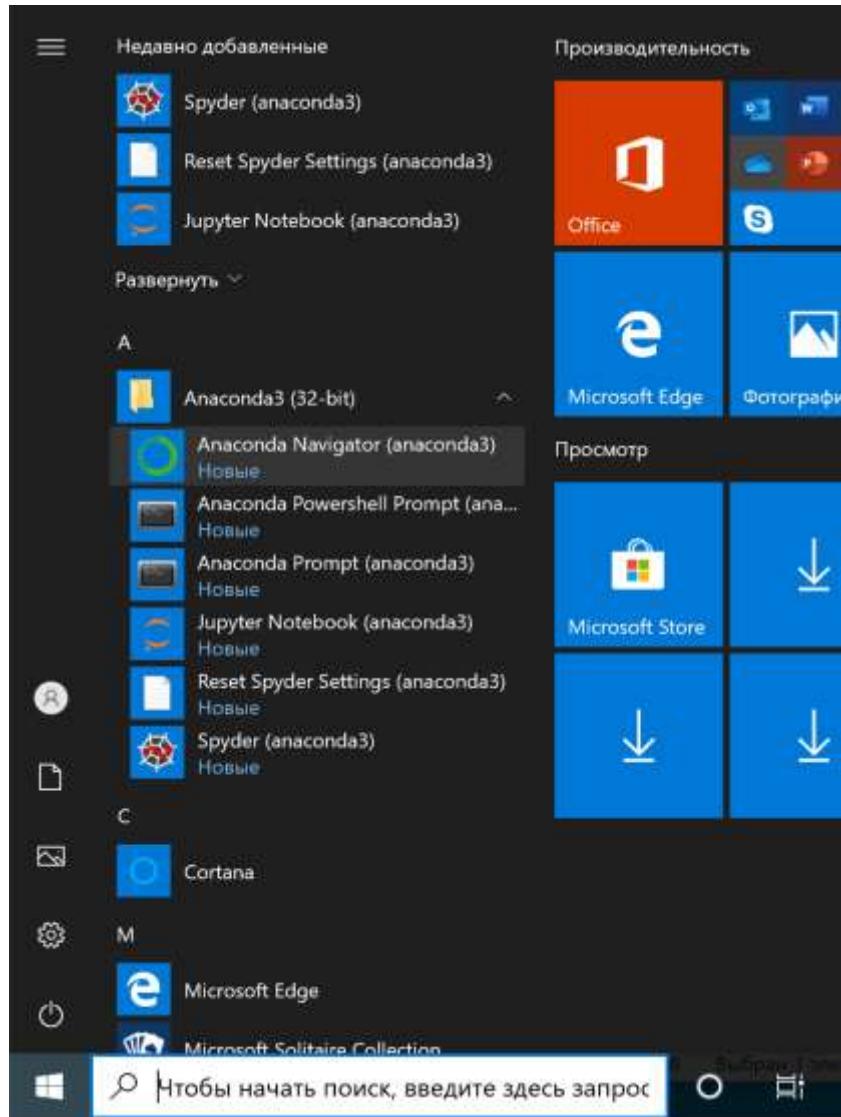
Для компьютеров
Windows



Для компьютеров
Apple Mac



Запуск Anaconda Navigator



Календарь



Локатор



Диктофон



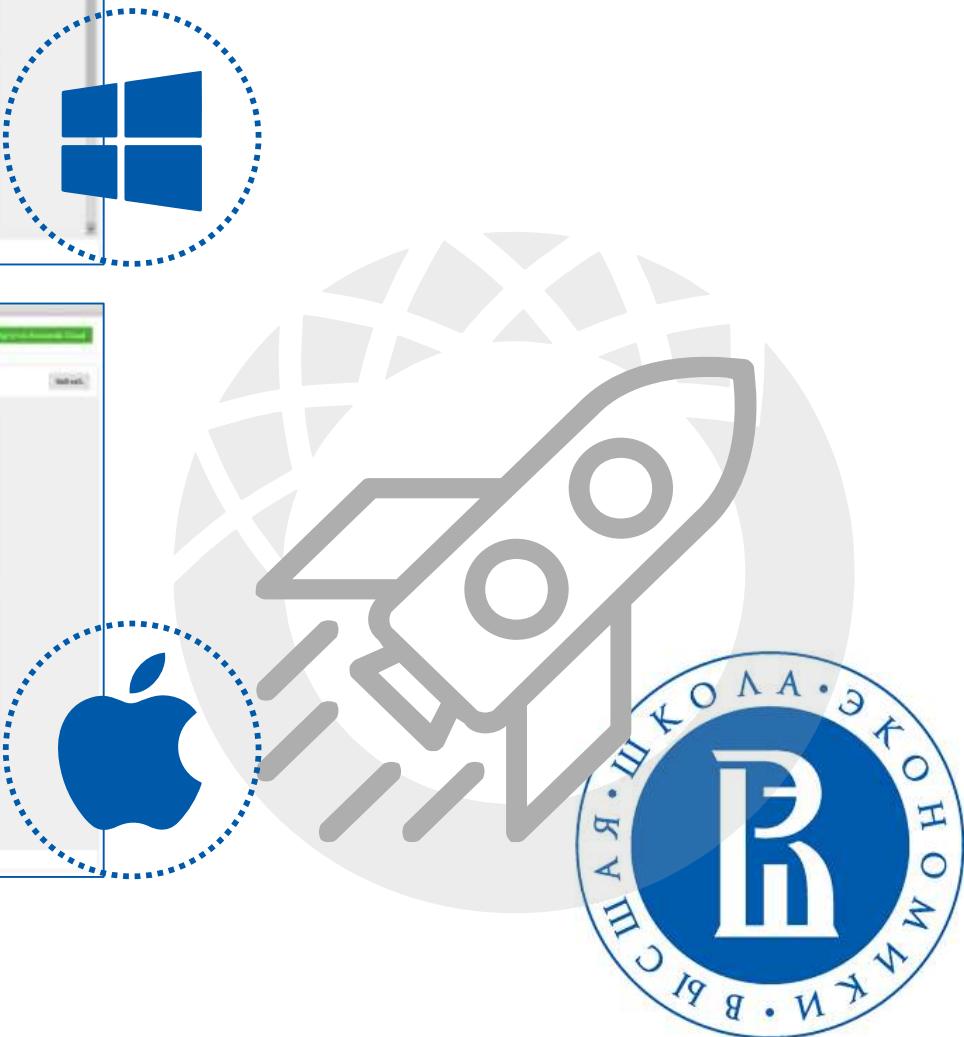
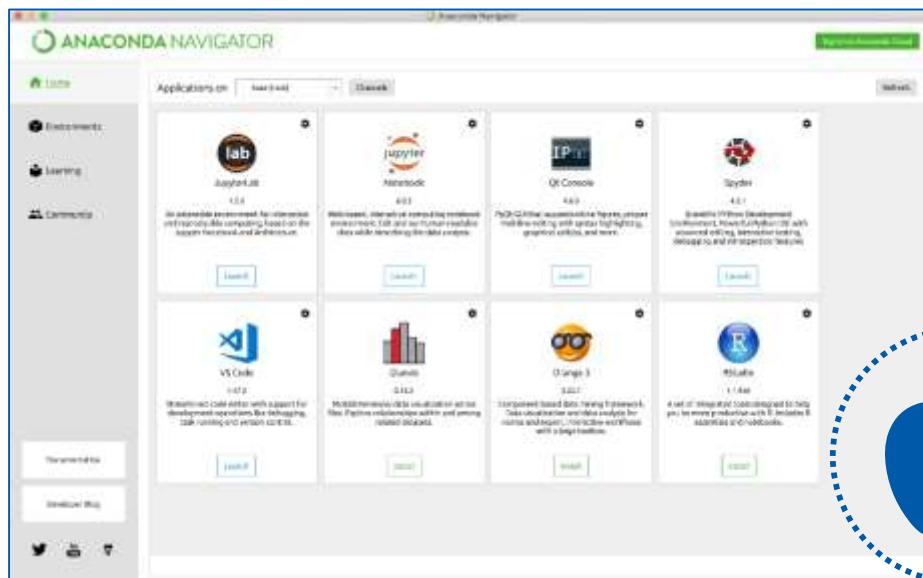
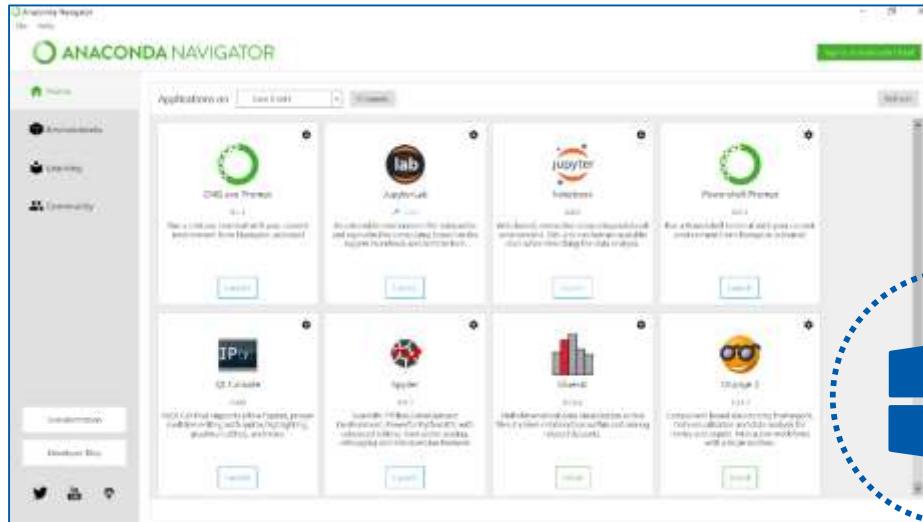
Терминал



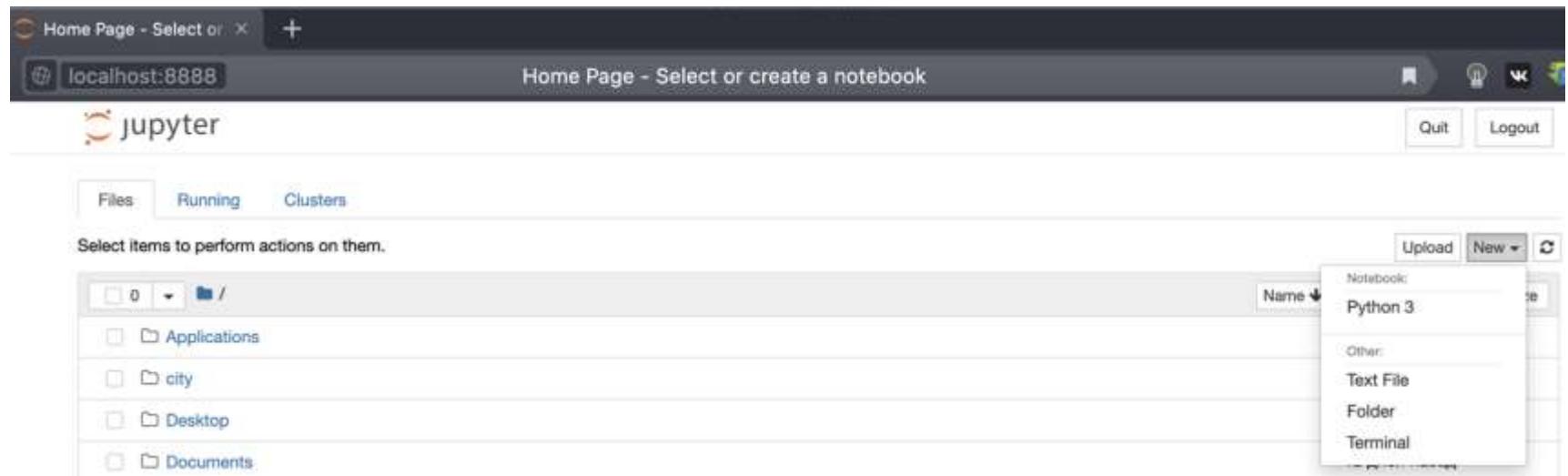
Анаconda-Navigator



Запуск Anaconda Navigator



Создадим первую программу



Создадим файл для своей первой программы:



New



Python 3



Закроем рабочее место

- 1 Закроем две вкладки Jupyter Notebook;
- 2 Закроем Anaconda Navigator;
- 3 Если на компьютере Apple Mac осталось чёрное окошко, закроем его.



```
lari — jupyter_mac.command — 80x24
[I 22:49:47.782 NotebookApp] JupyterLab application directory is /Users/lari/opt/anaconda3/share/jupyter/lab
[I 22:49:48.153 NotebookApp] Serving notebooks from local directory: /Users/lari
[I 22:49:48.153 NotebookApp] The Jupyter Notebook is running at:
[I 22:49:48.153 NotebookApp] http://localhost:8888/?token=fc062960b727a038813d08
3862f92c3f45fb7d2a38905398
[I 22:49:48.153 NotebookApp] or http://127.0.0.1:8888/?token=fc062960b727a03881
3d083862f92c3f45fb7d2a38905398
[I 22:49:48.153 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[IC 22:49:48.167 NotebookApp]

To access the notebook, open this file in a browser:
  file:///Users/lari/Library/Jupyter/runtime/nbserver-18034-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=fc062960b727a038813d083862f92c3f45fb7d2a389
05398
  or http://127.0.0.1:8888/?token=fc062960b727a038813d083862f92c3f45fb7d2a389
05398
[I 22:49:52.117 NotebookApp] Shutting down on /api/shutdown request.
[I 22:49:52.119 NotebookApp] Shutting down 0 kernels

[Процесс завершен]
```



Переменные и арифметические действия



Ввод-вывод текста в программу



Разберём первую программу

```
print('Hello, world!')
```

Компьютер выведет на экран:
Hello, World!



Разберём первую программу

Имя
команды

```
print('Hello, world!')
```



Скобки, благодаря которым компьютер понимает,
что **print** — название команды

Компьютер выведет на экран:
Hello, World!



Разберём первую программу

Имя
команды

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

Аргумент команды

Скобки, благодаря которым компьютер понимает, что `print` — название команды, и к команде `print()` относится аргумент '`Hello, World!`'

Компьютер выведет на экран:
`Hello, World!`



Аргументов может быть несколько

Имя
команды

Разделитель

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

Аргумент
команды

Второй
аргумент

Компьютер выведет на экран:

Второй
аргумент

Hello, World!

Первый
аргумент



Переменные

Иногда нам нужно работать с вещью, о которой мы знаем только название. В этой ситуации нам помогут **переменные** — такие слова внутри программы, значение которых может меняться.

Переменная

Значение переменной

name = 'Аня'

print('Привет, ', name)

Компьютер выведет на экран:

Привет, Аня



Переменные

Поменяем значение переменной, сохранив имя:

Переменная

Значение переменной

name = 'дима'

print('Привет, ', name)

Компьютер выведет на экран:

Привет, Дима



Ввод переменных с клавиатуры

Но главный смысл переменных в том, что мы можем попросить компьютер узнать их значения у человека:

Переменная

Функция, запрашивающая
значение для переменной *name*

```
name = input()
print('Привет, ', name)
```

Компьютер выведет на экран:

Привет, <введённое имя>



Переменные и арифметические действия



Простые типы данных: числа и строки



Зачем разделять информацию по типам?



Зачем разделять информацию по типам?



Есть двигатель
Нужно управлять

Нет двигателя
Нужно управлять
Нужно прикладывать усилия



Зачем разделять информацию по типам?



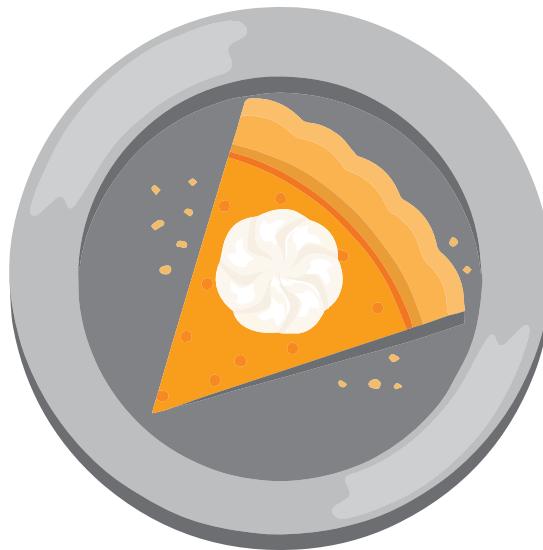
Есть двигатель
Нужно управлять

Нет двигателя
Нужно управлять
Нужно прикладывать усилия



Какие типы данных есть в Python

1 Строки

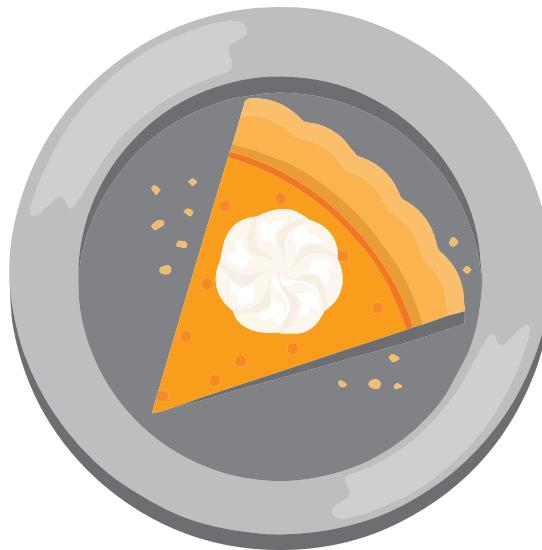


«Мы купили пирог»



Какие типы данных есть в Python

- 1 Строки
- 2 Целые числа



«Мы купили пирог»
Разрежем пирог на 4 куска



Какие типы данных есть в Python

- 1 Строки
- 2 Целые числа
- 3 Вещественные числа



«Мы купили пирог»
Разрежем пирог на 4 куска
Пирог весит 0,5 килограмма



Что можно делать с разными данными?

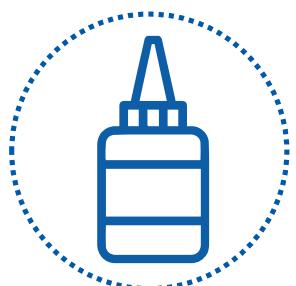
С числами:

+	Складывать	$5 + 7 = 12$
-	Вычитать	$5 - 7 = -2$
*	Умножать	$5 * 7 = 35$
/	Делить	$6 / 12 = 0,5$
**	Возводить в степень	$2 ** 3 = 8$
//	Делить нацело	$11 // 2 = 5$
%	Находить остаток от деления	$11 \% 2 = 1$



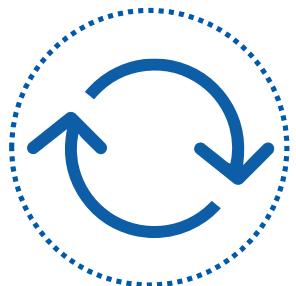
Что можно делать с разными данными?

Со строками:



Склейвать:

`'Hello, ' + 'World!' = 'Hello, World!'`



Повторять:

`'Hello' * 2 = 'HelloHello'`



Переменные и арифметические действия



Строки



Работа с конкретными символами строки

Как и люди, компьютеры могут увидеть отдельный символ из текста:

```
print('Hello, World!' [0]) # Выведет букву H
```

Номер, указанный в квадратных скобках, мы будем называть *индексом*.

0	1	2	3	4	5	6	7	8	9	10	11	12
H	e	l	l	o	,		W	o	r	l	d	!

```
print('Hello, World!' [7]) # Выведет букву W  
print('Hello, World!' [0]) # Выведет букву H
```



Длина строки

Чтобы узнать длину нашей строки, можно использовать функцию `len()`:

```
print(len('Hello, World!')) # Выведет 13
```

0	1	2	3	4	5	6	7	8	9	10	11	12
H	e	l	l	o	,	W	o	r	l	d	!	



Отрицательные индексы

Мы можем попросить компьютер показать нам не только «первый символ с начала строки», но и «первый символ с конца»:

```
print('Hello, World!' [-1]) # Выведет «!»
```

0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
H	e	l	l	o	,	W	o	r	l	d	!	

```
print('Hello, World!' [-2]) # Выведет «d»  
print('Hello, World!' [-8]) # Выведет «Hello, World!»
```



Переменные и арифметические действия



Повторение



Ввод и вывод информации

Команда *input()* позволяет спросить информацию у пользователя:

```
name = input()
```

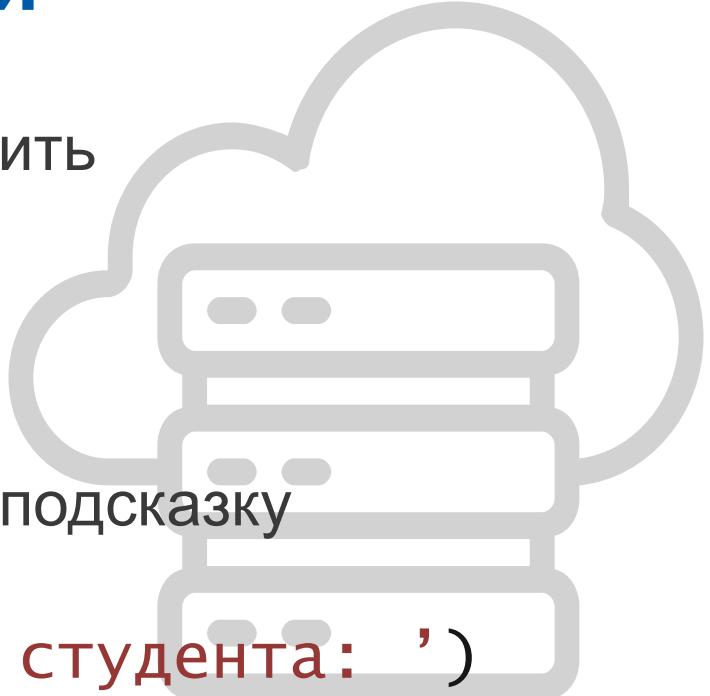
Команда *input()* может написать подсказку для пользователя:

```
name = input('Укажите имя студента: ')
```

Команды языка Python мы ещё будем называть *функциями*.

У функции вывода информации может быть несколько аргументов:

```
print('Hello, ', 'world! ')
```



Типы данных

Иногда информация об одном и том же предмете представляется по-разному: **количество кусков торта** может быть только **целым** (два куска, три куска), а вот **вес** куска может быть как целым, так и **вещественным** (0,1 кг).

Такие виды разной информации называются *типами данных*.

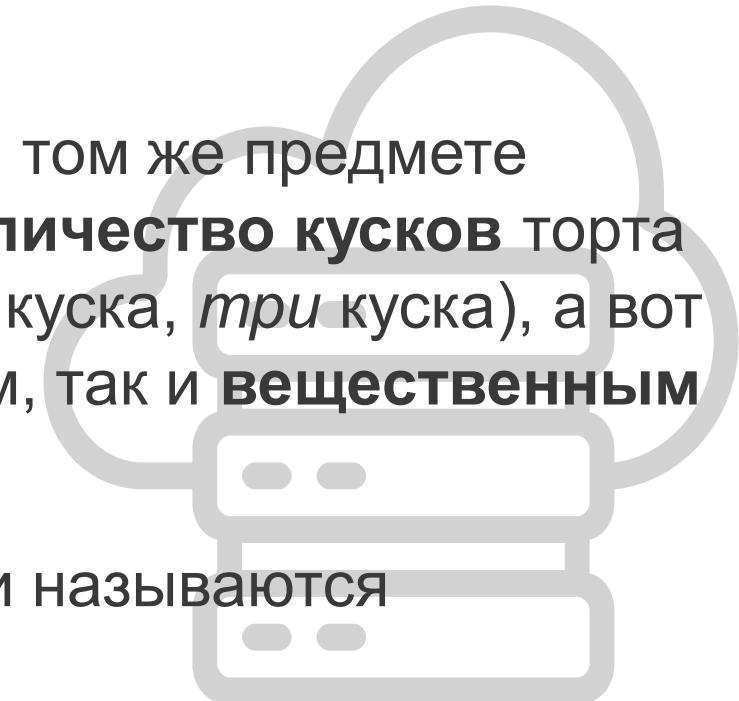
Компьютер знает много типов данных,

но нам пока нужны всего три:

1 Целые числа, **int**;

2 Вещественные числа, **float**;

3 Строки, **str**.



Что можно делать с типами данных

Числа можно складывать, вычитать, делить и умножать, а ещё делить нацело и брать остаток от деления:

$$5 + 2 = 7$$

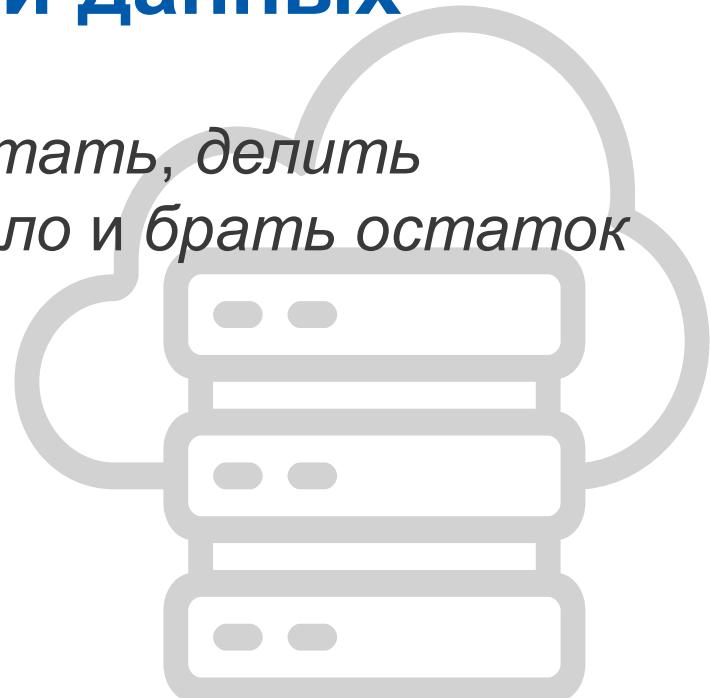
$$4.2 - 3 = 1.2$$

$$3.5 * 2 = 7$$

$$9 / 2 = 4.5$$

$$9 // 2 = 4$$

$$9 \% 2 = 1$$



Строки можно склеивать и повторять несколько раз:

‘Hello, ’ + ‘world! ’ = ‘Hello, world! ’

‘Hello’ * 4 = ‘HelloHelloHelloHello’



Оформление строк

Чтобы подставить в строчку значение переменной, нужно:

- 1 Перед строкой написать букву «f»;
- 2 Внутри строки название переменной написать в фигурных скобках:

```
name = 'Фыва Пролджэ'  
print(f'Привет, {name}!')
```

Компьютер выведет на экран:

Привет, Фыва Пролджэ!



Введение в язык Python



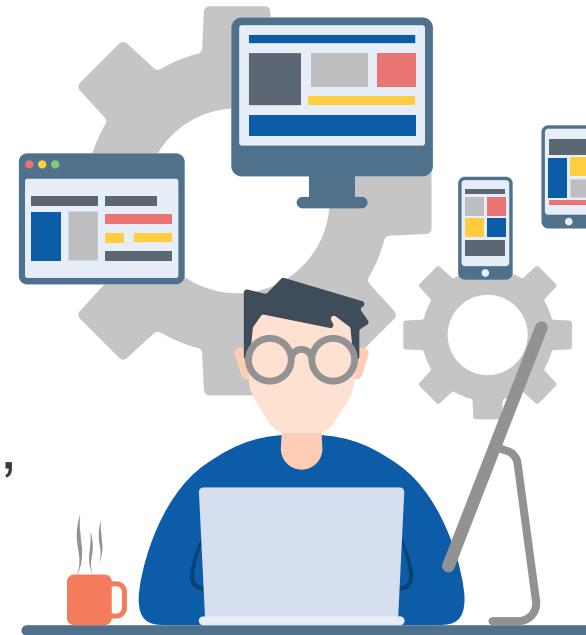
Повторение введения



Ввод и вывод информации

Программист — автор инструкций:

- 1 Создаёт максимально подробную инструкцию действий;
- 2 Переводит эту инструкцию на язык, понятный компьютеру;
- 3 Несколько раз запускает исходный код готовой программы и исправляет возникшие ошибки, о которых ему подскажет компьютер;
- 4 Отдаёт готовую программу пользователю — человеку, который не знает, как устроена программа внутри и как она работает, но которому нужен результат.



Наши инструменты

В этом курсе мы будем использовать программу **Jupyter Notebook** из комплекта программ Anaconda.

Файлы этой программы называются *блокнотами* — в них можно как писать команды для компьютера на языке Python, так и пояснения к программам на естественном языке (русском, например).

Язык Python позволяет оформлять исходный код программ по-разному, но в этом курсе мы будем придерживаться рекомендаций, созданных авторами языка Python (*PEP-8*):

`print('Hello, world!')`, но не
`print ('Hello, world')`



Первая программа: поздороваемся с миром

Чтобы вывести на экран строчку, нужно написать команду «**выведи, пожалуйста, на экран**» и указать, что именно нужно вывести:

```
print('Hello, world!')
```

- 1 Программы состоят из *команд* — английских слов со скобками после них:
`print()` — *команда*
- 2 Внутри скобок можно указать *аргументы* команды, т.е. то, что именно команда должна сделать.
Например, что именно должна вывести команда `print()`.
`'Hello, world!'` — *аргумент*

