Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

Программная инженерия. Разработка ПО (Python для продвинутых специалистов)

Модуль: Введение в Python для анализа: извлечение данных, подготовка и визуализация.

Лекция 1: Основы программирования на Python

Дата: 25.03.2025

Приветствую вас!

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Сергей Ильницкий
@seriln

Преподаватель очной школы Python/DA/DE.
Много говорит, часто по делу.
Показывает как надо и как не надо.

ПОЛИТЕХ
Санкт-Петербургский политехнический университет Петра Великого

HYUNDAI

HYUNDAI GLOVIS

HYUNDAI AutoEver

BIA Technologies

СБЕР УНИВЕРСИТЕТ

Правила курса и фокус

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

- Посещение вебинаров;
- Опоздания;
- Не стесняться задавать вопросы;
- Непонятные слова;
- Интерактив;

- Сохранение кода по ходу занятия;
- Много информации -> нужен порядок;

- Обратная связь.

Программное обеспечение:

- Python: основной язык программирования курса.
- Google Collab / VS Code Jupyter Notebook: интерактивная среда для выполнения кода на Python и ведения заметок.

Опционально:

- WSL: для работы с Unix.
- Docker: для экспериментов с базами данных и Unix.
- DBeaver: для работы с базами данных.

Обзор курса

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Структура курса:

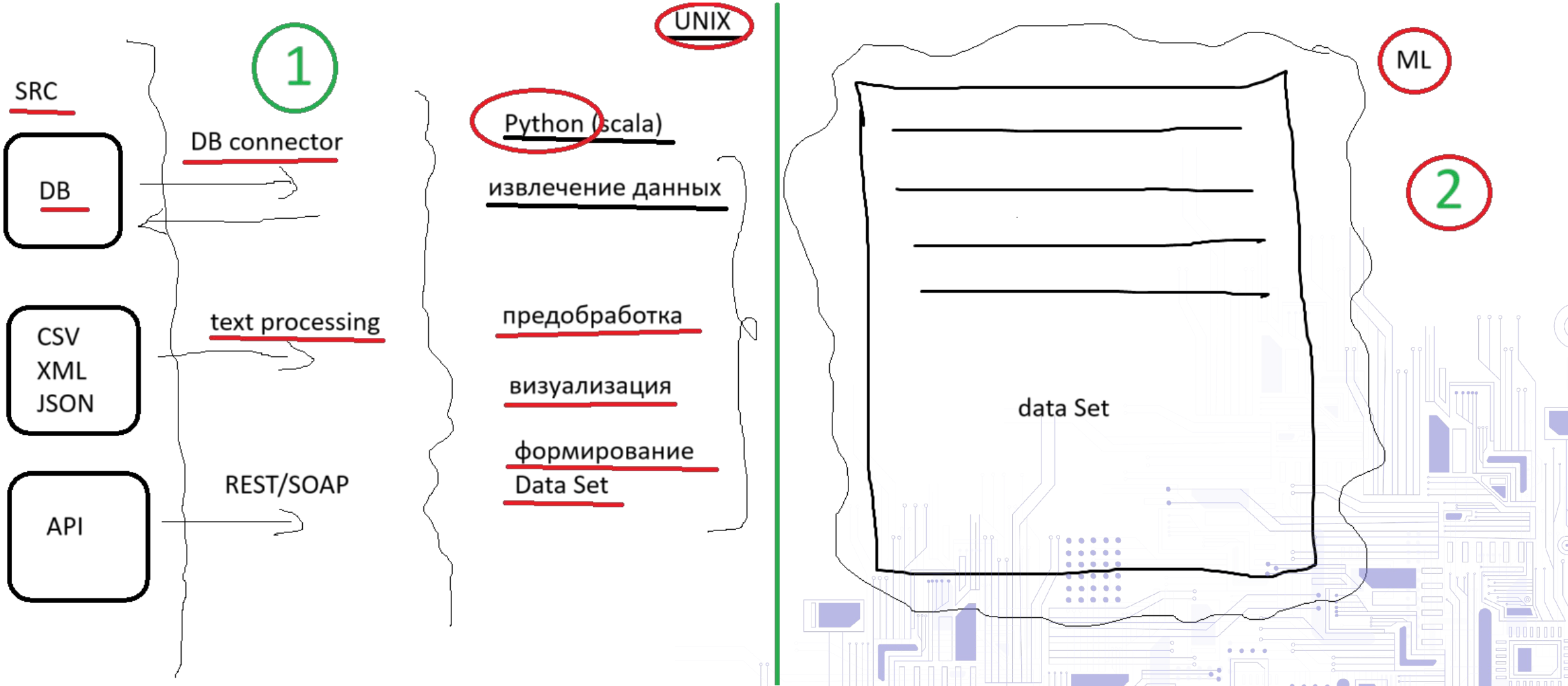
- Блок 1 - Введение в Python для анализа: извлечение данных, подготовка и визуализация
  - 1 - Основы программирования на Python
  - 2 - Структуры данных в Python
  - 3 - Библиотеки: NumPy, Pandas
  - 4 - Обзор средств визуализации в Python
  - 5 - Unix. База для уверенной работы
  - 6 - Работа с базой данных из Python
  - 7 - SQLalchemy для работы с базой данных
  - 8 - SQL. Коротко о самом важном при работе с данными
  - 9 - Регулярные выражения для очистки и нормализации данных
  - 10 - Как собрать DataSet без потерь из разных источников
  - 11 - RT.DataVision
  - 12 - Структура DS проекта и разведочный анализ данных
- Блок 2 - Предобработка данных и машинное обучение
  - ...

Активности курса:

- Вебинары – 2 раза в неделю
- Консультации – 1 раз в неделю
- Домашние задания – 1 раз в неделю
- Аттестации

Почему курс собран именно так?

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

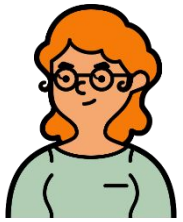


Проектный/продуктовый бестиарий

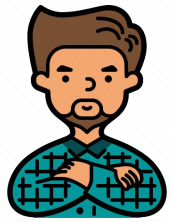
ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Data Scientist
(ML Engineer)

Architect



Project
Manager

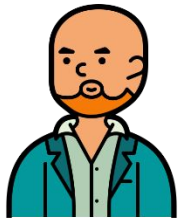


Product
Manager

Information
Security Officer

Data Engineer

Data Analyst



DevOps

Business
Intelligence

Support

ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Содержание лекции

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

- **Знакомство**
- Основы Python
- Практическая часть

https://pythontutor.ru/ - теория по Python

https://leetcode.com/ - для решения практических задач

ЯП: Компилируемые VS Интерпритируемые

Python — интерпретируемый язык программирования:

- CPython,
- Jython,
- ...

Python: стандарты

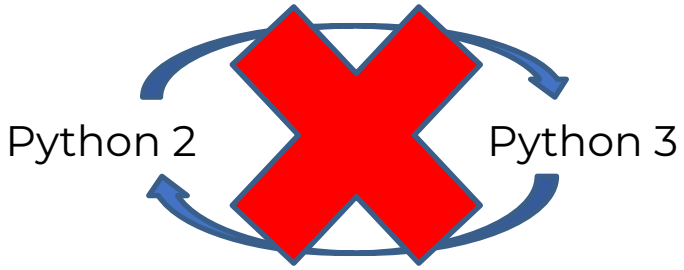
ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

PEP – **Python Enhancement Proposal**, предложения по развитию Python.

«Код читается намного больше раз, чем пишется»

PEP0 - перечень всех PEP
PEP8 - правила оформления кода
PEP20 - философия Python в афоризмах:

- Явное лучше, чем неявное.
- Простое лучше, чем сложное.
- Сложное лучше, чем запутанное.
- Плоское лучше, чем вложенное.
- Читаемость имеет значение.
- Ошибки никогда не должны замалчиваться.
- Если они не замалчиваются явно.
- Встретив двусмысленность, отбрось искушение угадать.
- Должен существовать один и, желательно, только один очевидный способ сделать это.
- Если реализацию сложно объяснить, то это плохая идея.
- Если реализацию легко объяснить, то идея, возможно хороша.
- ...



Jupyter Notebook

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

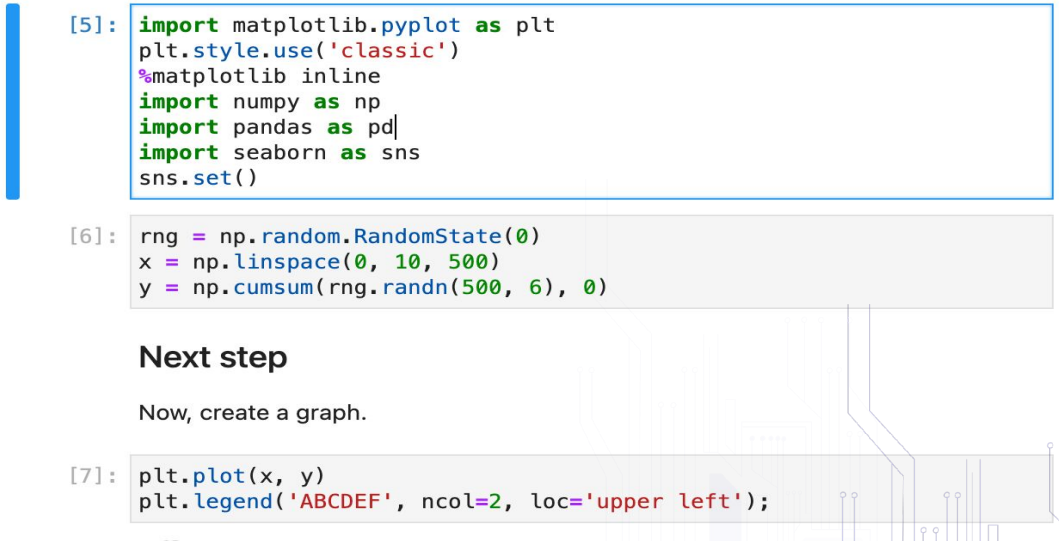
Jupyter Notebook — это интерактивная среда разработки, которая позволяет:

- выполнять код на разных языках;
- создавать заметки в Markdown;
- визуализировать данные;
- экспортировать проекты (*.py, HTML, PDF)

Такой подход удобен для анализа данных и разработки моделей машинного обучения.

Для работы мы используем:

- Google Colab
- VS Code + Jupiter Notebook
- Anaconda
- Python + Jupiter



Переменные и типы данных

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Переменные используются для хранения данных. В Python тип данных определяется автоматически при присвоении значения переменной.

```
name = "Alice"  # строка
age = 25        # целое число
height = 5.5    # число с плавающей точкой
high_math = complex(1, 2) # комплекское число
is_student = True       # логическое значение
```

Приведение типов:

```
str(<parameter>)     # для преобразования в строку
int(<parameter>)     # для преобразования в целое число
float(<parameter>)   # для преобразования в дробное число
bool(<parameter>)    # для преобразования в булево значение
```

Основные операторы

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Операторы используются для выполнения различных операций над переменными и значениями.

Арифметические операторы: +, -, *, /, %	sum_result = **10** + **3** # сложение
	difference = **10** - **3** # вычитание
	product = **10** * **3** # умножение
	quotient = **10** / **3** # деление
	remainder = **10** % **3** # остаток от деления
	remainder_ = **10** // **3** # остаток от деления (целочисленный)
	pow = **10** ** **3** # возведение в степень

Логические операторы: and, or, not	is_adult = **not** (age < **18**) # True, если возраст 18 или больше
	is_teenager = (age >= **13**) **and** (age < **20**) # True, если возраст от 13 до 19

Операторы сравнения: ==, !=, >, <, >=, <=	is_equal = (**10** == **5**) # False
	is_greater = (**10** > **5**) # True

Строки и операции над ними

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Строки в Python — это последовательности символов, заключенные в кавычки.

```
greeting = "Hello"
name = "Alice"
message = greeting + ", " + name + "!"  # Конкатенация строк
print(message)  # Выводит "Hello, Alice!"
length = len(message)  # Длина строки
```

Пример: создание приветственного сообщения

```
first_name = "John"
last_name = "Doe"

full_name = first_name + " " + last_name  # "John Doe"
print("Welcome, " + full_name + "!")

greeting = f' Welcome, {first_name} {last_name}!'  # “Welcome, John Doe!"
print(greeting)
```

Условные операторы

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Условные операторы позволяют выполнять различные действия в зависимости от истинности или ложности условий.

Условный оператор `if`	balance = **150** cost = **100** **if** balance >= cost: **print**("Покупка совершена")

Условный оператор `else`	weather = "rainy"
	if weather == "sunny":
	  print("Наденьте солнцезащитные очки")
	else:
	  print("Возьмите зонт")

Условный оператор `elif`	hour = **15**
	if hour < **12**:
	  print("Доброе утро")
	elif hour < **18**:
	  print("Добрый день")
	else:
	  print("Добрый вечер")

Циклы

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Циклы позволяют выполнять блок кода несколько раз

Цикл `for`:	shopping_list = ["milk", "bread", "eggs"] **for** item **in** shopping_list: **print**("Купите", item)

Цикл `while`:	savings = **0**
	goal = **1000**
	while savings < goal:
	  savings += **100** # Добавляем по 100
	  print(f"Текущие сбережения: {savings}")

Использование break и continue

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Операторы прерывания позволяют влиять на поведения цикла на заданной итерации

`break`: Прерывает выполнение цикла.	`for number in range(10):`
	`  if number == 5:`
	`  break`
	`  print(number)`
	`# Выводит числа от 0 до 4 и прерывает цикл на 5`

`continue`: Пропускает текущую итерацию и переходит к следующей	`for number in range(10):`
	`  if number % 2 == 0:`
	`  continue`
	`  print(number)`
	`# Выводит нечетные числа от 0 до 9`

Функции в Python

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Функции позволяют организовать код в блоки, которые можно многократно использовать. Они помогают сделать код более структурированным и читаемым

Функция определяется с помощью ключевого слова `def`, за которым следует имя функции и параметры в круглых скобках.	`def greet(name):`
	`  print(f"Привет, {name}!")`
	
	`greet("Алиса")`

Функция может возвращать значение с помощью ключевого слова `return`.	`def rectangle_area(length, width):`
	`  return length * width`
	
	`area = rectangle_area(5, 3)`
	`print(f"Площадь: {area}")`

Функции могут принимать позиционные и именованные аргументы	`def greet(name, age):`
	`  print(f"Привет, {name}! Вам {age} лет.")`
	
	`greet("Алиса", 20)`
	
	`def greet(name, age):`
	`  print(f"Привет, {name}! Вам {age} лет.")`
	
	`greet(name="Алиса", age=20)`
	`greet(age=20, name="Алиса")`

Области видимости

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Правило разрешения имен в Python (LEGB):

- **Область видимости local (function)**
- Область видимости enclosing (nonlocal)
- **Область видимости global**
- **Область видимости built-in**

Описывает стандартный механизм поиска “названий” (переменных) во время выполнения программы.

Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

Практическая часть

Задание 1: Странные числа

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Задание:

Давайте переберем все числа от `a` до `b` включительно и будем их выводить на экран, при этом нужно выполнить следующие условия:

1. Пропускать (не выводить) числа, которые делятся на 2 или на 3
2. Если встречаете число, кратное 100, необходимо принудительно закончить цикл, само это число не выводить

****Формат ввода****
Вводится два натуральных числа `a` и `b` в отдельных строках.

****Формат вывода****
Вывести все числа на интервале от `a` до `b` включительно согласно условиям в пунктах 1 и 2

Задание 2: Гипотеза Коллатца

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Задание:

Сиракузская последовательность, или последовательность Коллатца, строится так: возьмём натуральное число n; если оно чётное, то заменим его числом n/2; если же оно нечётное, то заменим его числом 3n+1. Получившееся число — следующее в сиракузской последовательности после числа n. Затем заменяем получившееся число по тому же правилу, и так далее.

Обычно, если проделать такую замену достаточно много раз, мы приходим к числу 1 (за которым следует снова 1). Например:

8 → 4 → 2 → 1 или 10 → 5 → 16 → 8 → 4 → 2 → 1.

Определите, сколько шагов потребуется сиракузской последовательности, стартующей с заданного числа, чтобы прийти к 1.

Если вы обнаружите число, сиракузская последовательность от которого не приходит к 1, то... вы, скорее всего, ошиблись. Но если нет, то поздравляем: вы прославитесь, ведь вопрос о том, всегда ли сиракузская последовательность приходит к 1 (независимо от начального числа), давно будоражит умы математиков.

Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

онлайн университет

Спасибо за внимание