ИНТЕЛЛЕКТУАЛЬНЫЙ МЕГАПОЛИС

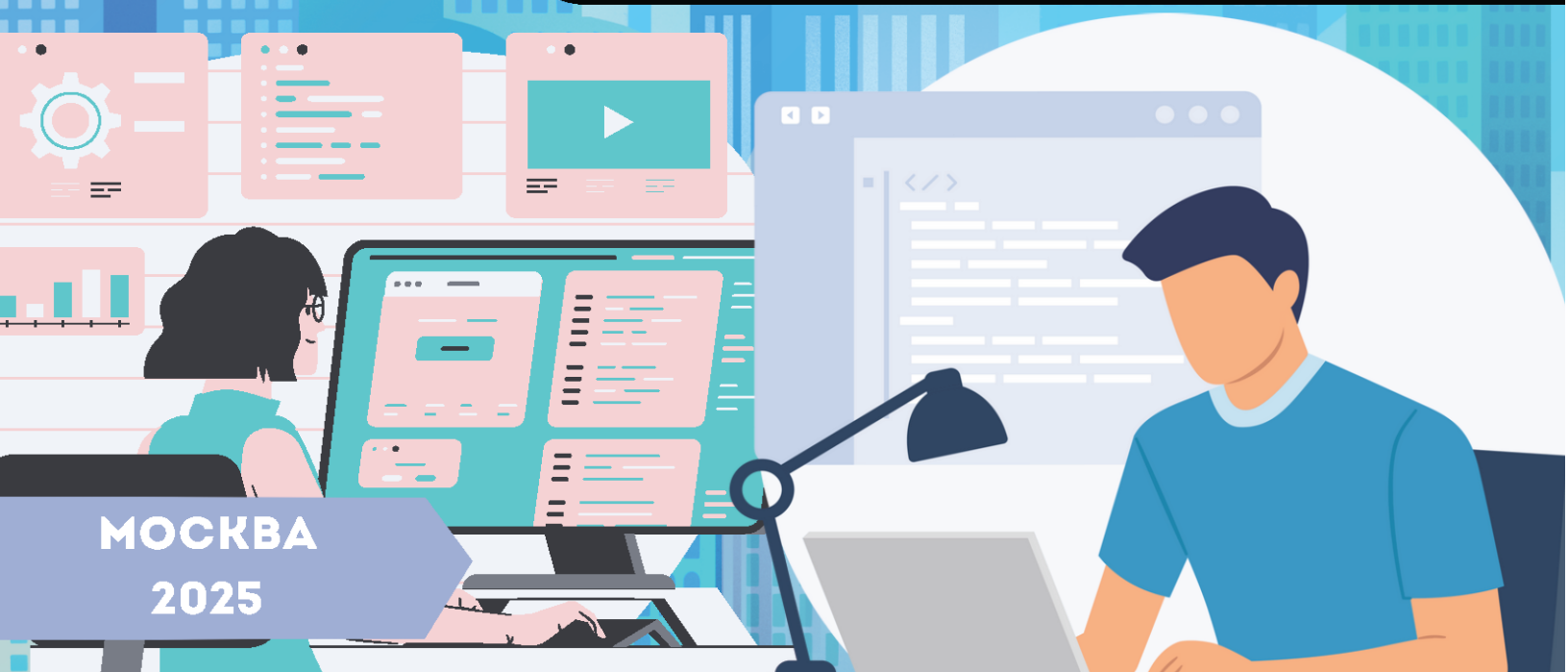
МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

ИТ-класс

в московской школе

НАПРАВЛЕНИЕ БОЛЬШИЕ ДАННЫЕ И ТЕХНОЛОГИИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ПРАКТИЧЕСКИЙ ЭТАП



МОСКВА
2025

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ РАЗРАБОТАНЫ:

Абрамешин Дмитрий Андреевич, доцент МИЭМ НИУ ВШЭ
Бубнова Мария Андреевна, ассистент ДКИ МИЭМ НИУ ВШЭ
Коваленко Алексей Анатольевич, инженер НИУ ВШЭ
Пожидаев Евгений Димитриевич, профессор-исследователь ДЭИ МИЭМ НИУ ВШЭ

Практический этап Конкурса проводится в *очной форме на базе вуза или очной дистанционной форме*. При выполнении работы обеспечивается строгое соблюдение порядка организации и проведения Конкурса.

Если этап проводится в очном дистанционном формате с использованием технологии прокторинга. Участникам необходимо иметь компьютер (ПК или ноутбук; прохождение диагностики на мобильных устройствах - невозможно) с выходом в Интернет, веб-камерой и микрофоном, а также смартфон (или планшет) со стабильным интернетом и приложением для считывания QR-кодов. Требуется предварительная настройка оборудования: https://im.mcko.ru/docs/Инструкция_для_участника_конкурса_Интеллектуальный_мегаполис_Потенциал.pdf. Браузер разрешается использовать только для прохождения заданий этапа и процедуры прокторинга.

Дополнительное ПО, разрешенное для прохождения: компьютеры. На компьютерах установлена версия Python 3.12 или выше, а также интегрированные среды разработки (IDE) PyCharm и Visual Studio Code. Участники имеют доступ к необходимым библиотекам Python, включая numpy, pandas и matplotlib, которые настроены и готовы к использованию. Для работы с базами данных на компьютерах также установлена СУБД PostgreSQL и визуальный интерфейс pgAdmin или DBeaver.

Чем пользоваться категорически нельзя (ведет к отклонению работы): веб-поиском, методическими материалами.

На выполнение заданий *практического* этапа Конкурса отводится *150* минут. Во время проведения мероприятия участник может выйти из зоны проведения мероприятия не более чем на 5 минут, предупредив *ответственного от вуза*. Мероприятие не продлевается на время отсутствия участника.

Вариант кейса содержит в себе 4 задания, спецификация которых указана в таблице.

№ задания	**Выбор задания для решения**	**Уровень сложности**	***Уникальные кодификаторы Конкурса***	**Контролируемые требования к проверяемым умениям**	**Балл**
1.	-	базовый	Библиотека pandas.	Знать: - Основные структуры данных библиотеки pandas: Series, DataFrame - Методы чтения и записи данных (CSV, Excel, JSON) - Способы фильтрации, группировки и агрегации данных Уметь: - Создавать и изменять объекты Series и DataFrame - Загружать данные из внешних файлов и сохранять их - Применять группировку (`groupby`) и агрегатные функции (`mean()`, sum() и др.)	5
2.	-	базовый	Библиотека numpy.	Знать: - Структуру и назначение массива ndarray - Основные операции с массивами: арифметика, трансформация формы, индексация - Функции генерации данных: arange, linspace, random Уметь:	5

				- Создавать одномерные и многомерные массивы - Выполнять векторные операции и логические фильтрации - Использовать встроенные функции numpy для генерации и обработки данных	
3.	-	базовый	Библиотека matplotlib.	Знать: - Основные элементы графика: фигура, оси, подписи, легенда - Команды для построения графиков (`plot`, bar, hist, `scatter`) - Параметры стилизации графиков: цвета, маркеры, линии Уметь: - Строить линейные графики, столбчатые диаграммы, гистограммы - Настраивать подписи, легенды, сетку и заголовки - Сохранять графики в файл в различных форматах (PNG, PDF и др.)	5
4.	-	базовый	Анализ и визуализация данных на Python.	Знать: - Понятие и цели одномерного анализа (распределение, мода, медиана, размах) - Назначение гистограммы и графика плотности - Типы распределений Уметь:	5

				- Строить графики функций и гистограммы по массиву данных - Интерпретировать распределения по графикам - Определять характер распределения	

Ключевые моменты:

Критерии оценивания: если задание решено верно, участник получает максимальный балл, иначе 0 баллов.

Описание хода практической части в случае очной дистанционной формы проведения этапа Конкурса: *категорически нельзя (ведет к отклонению работы):веб-поиском, методическими рекомендациями по направлениям Конкурса.* Во время проведения мероприятия участник может выйти из зоны проведения мероприятия не более чем на 5 минут, предупредив *проктора на камеру*. Мероприятие не продлевается на время отсутствия участника.

Трудности при подготовке – во избежание трудностей рекомендуется ознакомиться с полным комплектом методических материалов, самостоятельно решить демовариант, сверяясь с ответами и ходом решения, предложенном в методических материалах.

Для удобства повторения материала далее предложены 4 таблицы, в которых указаны ключевые термины/команды/методы, которые участнику Конкурса необходимо знать и уметь применять для успешного решения кейса.

Термин/команда/метод	Пояснение
библиотека	Набор готовых функций и структур для упрощения программирования (например, pandas, numpy, matplotlib).
массив (array)	Упорядоченная коллекция чисел или других объектов, часто используется в numpy.
таблица (DataFrame)	Основная структура данных в pandas, аналог таблицы с заголовками и индексами.
строка и столбец	Элементы таблицы — строки (записи) и столбцы (поля/признаки).
индекс	Уникальный идентификатор строки в DataFrame (например, номер ученика).

Термин/команда/метод	Пояснение
DataFrame	Двумерная структура таблицы в pandas.
Series	Одномерный столбец в таблице DataFrame.
df['Колонка']	Обращение к одному столбцу.
df.max(), df.min(), df.mean()	Методы для анализа данных по строкам или столбцам.
df.shape, df.size, len(df)	Размер таблицы (строки, столбцы), общее количество элементов.
df.describe()	Статистика по данным: среднее, минимум, максимум и др.
df.query('условие')	Отбор строк по логическому выражению.
df.groupby(...)	Группировка по значениям одного столбца.
df.to_csv(), pd.read_csv()	Сохранение/загрузка таблиц в/из CSV-файла.
index_col, sep, encoding, header	Параметры чтения CSV-файла.

Термин/команда/метод	Пояснение
NumPy	Библиотека для работы с числовыми массивами.
np.zeros(), np.ones(), np.full()	Создание массивов с одинаковыми значениями.
np.arange()	Генерация последовательностей с шагом.
np.linspace()	Генерация равномерно распределённых чисел.
np.random.rand(), np.random.randint()	Генерация случайных чисел.
dtype	Тип данных массива (int, float и т.д.).

Термин/команда/метод	Пояснение
matplotlib.pyplot (plt)	Библиотека для построения графиков.
plt.plot()	Построение линейного графика.
plt.bar(), plt.barh()	Столбчатые диаграммы (вертикальные и горизонтальные).
plt.scatter()	Диаграмма рассеяния (точки).
plt.title(), plt.xlabel(), plt.ylabel()	Заголовок и подписи к осям.
plt.grid(), plt.legend()	Сетка и легенда на графике.
plt.savefig()	Сохранение графика в файл.
plt.xlim(), plt.ylim()	Ограничения по осям X и Y.

Подробные решения кейсов, содержащих по 4 задачи из демоварианта.

Кейс

Задача 1

1.

Был создан датафрейм с оценками по предметам:

```
import pandas as pd
data = {'Математика': [4, 5, 3], 'Информатика': [5, 4, 4]}
df = pd.DataFrame(data)
```

Необходимо вычислить среднюю оценку по каждому предмету, в ответе укажите соответствующую команду.

Ответ: df.mean()

In [1]:
```
import pandas as pd

data = {'Математика': [4, 5, 3], 'Информатика': [5, 4, 4]}
df = pd.DataFrame(data)

# Считаем среднюю оценку по каждому предмету
print(df.mean())

```

```
Математика     4.000000
Информатика    4.333333
dtype: float64
```

Комментарий:

df.mean() считает среднее значение по столбцам (по предметам).

Задача 2

2.

Необходимо создать массив из 5 одинаковых значений 7. В ответе напишите соответствующую команду. Способ решения должен быть корректным с помощью NumPy.

Ответ: np.full(5, 7), либо любой другой корректный способ решения.

In [2]:
```
import numpy as np

# Создаём массив из 5 семёрок
array = np.full(5, 7)

print(array)

```

```
[7 7 7 7 7]
```

Комментарий: np.full создаёт массив заданной длины и значений.

Альтернатива — np.repeat или [7]*5.

Задача 3

Необходимо построить график зависимости температуры от времени. Массивы time, temperature заданы. Напишите команду, учитывая, что она должна быть в формате plt.___(___, ___).

Ответ: plt.plot(time, temperature)

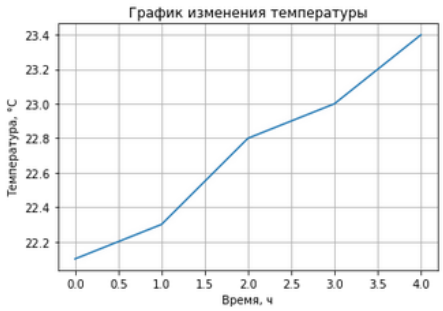
Приведем пример с данными:

```python
import matplotlib.pyplot as plt

time = [0, 1, 2, 3, 4]
temperature = [22.1, 22.3, 22.8, 23.0, 23.4]

plt.plot(time, temperature)
plt.xlabel('Время, ч')
plt.ylabel('Температура, °C')
plt.title('График изменения температуры')
plt.grid(True)
plt.show()

```



Комментарий: plt.plot — базовая функция для построения графика. Дополнительные команды делают график нагляднее.

Задача 4

Необходимо загрузить CSV-файл inf.csv в DataFrame для анализа. Напишите команду, учитывая, что она должна быть в формате pd.___(___)

```python
import pandas as pd

# Загрузка CSV-файла
df = pd.read_csv('inf.csv')

print(df.head())

```

```
    Имя  Возраст           Город
0   Аня       25          Москва
1  Борис      30  Санкт-Петербург
2   Вика      28          Казань
3  Игорь      32      Новосибирск
4  Мария      27     Екатеринбург
```

Комментарий: read_csv — основной способ чтения CSV в pandas. Можно указать путь, кодировку, разделитель и др.

Кейс

Задача 1

Необходимо выбрать из DataFrame только те строки, где в столбце "Баллы" значение больше 80, используя метод query:

```
import pandas as pd
df = pd.DataFrame({'Имя': ['Аня', 'Борис', 'Вика'], 'Баллы': [75, 90, 85]})
```

В ответе укажите соответствующую команду.

Ответ: df.query('Баллы > 80')

```
import pandas as pd

df = pd.DataFrame({
    'Имя': ['Аня', 'Борис', 'Вика'],
    'Баллы': [75, 90, 85]
})

# Отбираем строки, где Баллы > 80
result = df.query('Баллы > 80')

print(result)

```

```
    Имя  Баллы
1  Борис     90
2   Вика     85
```

Комментарий: метод query позволяет фильтровать строки по условиям, используя названия столбцов как переменные.

Задача 2

Создайте массив из 100 случайных чисел от 0 до 10 с равномерным распределением и округлите до 2 знаков, учитывая, что ответ должен быть записан в виде np.___(np.random.___(), ___).

Ответ: np.round(np.random.uniform(0, 10, 100), 2)

```
import numpy as np

# Генерируем 100 чисел от 0 до 10 и округляем
array = np.round(np.random.uniform(0, 10, 100), 2)

print(array)

```

```
[7.22 8.9  4.11 3.83 7.51 7.25 8.61 7.34 5.69 4.28 7.17 2.39 7.18 5.47
 6.94 1.22 3.33 3.31 7.63 8.24 8.   1.1  3.43 8.17 2.95 9.1  9.63 9.64
 2.54 3.33 0.19 8.72 8.97 0.75 9.69 4.3  5.23 6.48 8.94 1.2  1.85 5.75
 5.62 0.31 9.6  4.35 2.64 7.73 6.71 3.89 0.54 6.56 2.95 3.43 8.29 0.19
 5.89 5.08 4.68 7.53 6.64 8.25 7.95 5.36 6.15 2.81 1.83 9.92 1.46 6.74
 7.58 0.76 2.38 7.98 4.77 0.99 3.8  1.36 4.05 3.52 8.67 3.96 0.06 9.63
 9.13 0.42 7.39 5.34 8.26 4.83 6.83 8.66 1.56 5.31 5.56 3.39 8.07 4.55
 6.93 0.66]
```

Комментарий: uniform — равномерное распределение. round округляет до заданного количества знаков после запятой.

Задача 3

Постройте гистограмму из массива data с 20 равными интервалами и отобразите сетку.

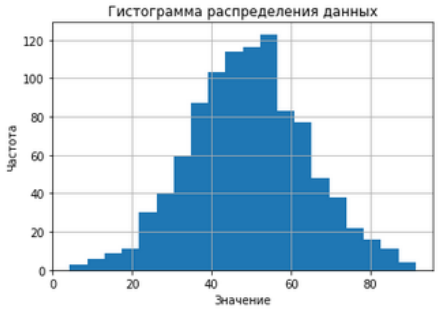
Ответ должен быть записан в виде plt.___(__, ___); plt.___().

Ответ: plt.hist(data, bins=20); plt.grid()

```python
import matplotlib.pyplot as plt
import numpy as np

# Генерация случайных данных (нормальное распределение)
np.random.seed(0)
data = np.random.normal(loc=50, scale=15, size=1000)

# Построение гистограммы с 20 интервалами
plt.hist(data, bins=20)
plt.grid()
plt.title("Гистограмма распределения данных")
plt.xlabel("Значение")
plt.ylabel("Частота")
plt.show()
```



Комментарий:
bins=20 разбивает данные на 20 групп. Сетка улучшает восприятие данных на графике.

Задача 4

Постройте график плотности распределения (kernel density estimation) для столбца 'score' датафрейма df. В ответ впишите верную команду.

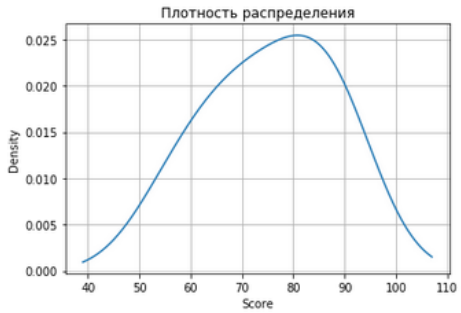
Ответ: df['score'].plot.kde()

```python
import pandas as pd

# Примерные данные
df = pd.DataFrame({'score': [56, 70, 88, 90, 77, 65, 82]})

# График плотности распределения
df['score'].plot.kde()
plt.xlabel('Score')
plt.title('Плотность распределения')
plt.grid(True)
plt.show()

```



Комментарий: plot.kde() строит сглаженную кривую распределения, отражающую вероятностную плотность.

Подборка методов, с которыми следует ознакомиться:

df.max()

Возвращает максимальные значения по каждому столбцу. Используется для оценки наибольшего значения в наборе данных.

df.min()

Возвращает минимальные значения по каждому столбцу. Полезно для поиска наименьших показателей.

df.mean(axis=1)

Вычисляет среднее значение по строкам (axis=1). Удобно при подсчёте средних оценок или других метрик по каждому объекту.

df.describe()

Предоставляет статистическое описание данных: количество, среднее, стандартное отклонение, минимум, максимум и квартильные значения.

df.query('x > 5')

Позволяет отбирать строки из DataFrame по логическим условиям в строковом формате. Используется для фильтрации данных.

np.zeros(10)

Позволяет создать массив из 10 нулей. Применяется для инициализации пустых значений.

np.ones(6)

Позволяет создать массив из 6 единиц. Удобно для тестов, масок и шаблонов.

np.arange(10)

Позволяет создать массив от 0 до 9 (без включения 10). Используется для генерации последовательностей.

np.arange(5, 16)

Позволяет сгенерировать массив из чисел от 5 до 15 включительно. Подходит для интервалов и отрезков.

np.linspace(0, 1, 7)

Позволяет создать массив из 7 чисел, равномерно распределённых от 0 до 1. Полезно в графиках.

np.arange(5)

Позволяет сгенерировать массив из целых чисел от 0 до 4. Быстрая генерация небольших массивов.

np.random.rand(5)

Позволяет создать массив из 5 случайных чисел от 0 до 1 (равномерное распределение).

np.random.randint(1, 11, size=4)

Позволяет сгенерировать массив из 4 случайных целых чисел от 1 до 10 включительно.

np.full(3, 3.14)

Позволяет создать массив из 3 одинаковых значений 3.14. Полезно для шаблонов и заполнений.

plt.plot(x, y)

Строит линейный график по точкам x и y. Подходит для отображения изменений во времени или трендов.

plt.bar(labels, values)

Строит столбчатую диаграмму. Подходит для категориальных данных, например, оценок, продаж, частот.

plt.title('...')

Позволяет добавить заголовок к графику. Полезно для пояснения отображаемых данных.

plt.xlabel('...') и plt.ylabel('...')

Добавляют подписи к осям X и Y. Делают график понятнее.

plt.scatter(x, y)

Строит диаграмму рассеяния — точки на плоскости. Удобно для анализа распределений и кластеров.

plt.legend()
Позволяет добавить легенду к графику, если используются метки (label='...') при построении графиков.
plt.grid()
Позволяет добавить сетку на график. Улучшает визуальное восприятие данных.
plt.savefig('filename.png')
Сохраняет текущий график в файл. Удобно для отчётов и публикаций.
plt.xlim(a, b) и plt.ylim(c, d)
Устанавливают границы отображаемых значений по осям X и Y.
pd.read_csv('data.csv')
Позволяет загрузить CSV-файл в DataFrame с использованием стандартных настроек. Один из самых часто используемых методов.
pd.read_csv('data.csv', header=None)
Позволяет загрузить CSV без использования первой строки в качестве заголовков. Полезно при отсутствии заголовков в файле.
pd.read_csv('data.csv', sep=';')
Позволяет прочесть CSV, где в качестве разделителя используется точка с запятой.
pd.read_csv('data.csv', encoding='utf-16')
Позволяет загрузить CSV с заданной кодировкой. Используется, если файл не в UTF-8 и возникают ошибки при чтении.
pd.read_csv('data.csv', index_col='id')
Использует столбец 'id' в качестве индекса DataFrame. Удобно для дальнейшей адресации по строкам.
pd.read_csv('data.csv', nrows=10)
Позволяет загрузить только первые 10 строк. Полезно при предварительном просмотре больших файлов.
pd.read_csv('data.csv', skiprows=1)
Пропускает первую строку при загрузке. Используется, если первая строка содержит неактуальные данные или комментарии.
pd.read_csv('data.csv', usecols=['Name', 'Score'])
Позволяет загрузить только указанные столбцы. Уменьшает объём данных и ускоряет обработку.
pd.read_csv('data.csv', na_values='?')
Заменяет указанные символы (например, '?') на NaN при загрузке. Полезно при нестандартных обозначениях пропусков.
df.to_csv('output.csv')
Сохраняет DataFrame в CSV-файл. По умолчанию сохраняет и заголовки, и индексы.
df.to_csv('output.csv', index=False)
Сохраняет DataFrame в CSV без индексов. Используется при подготовке к импорту в другие системы.
df.to_csv('output.csv', sep=';')
Сохраняет CSV-файл с точкой с запятой в качестве разделителя. Удобно для Excel и других европейских форматов.