Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

**дисциплины**

**«Искусственный интеллект и машинное обучение»**

**Вариант 9**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Кравчук Мирослав Витальевич  2 курс, группа ИТС-б-о-23-1,  11.03.02«Инфокоммуникационные технологии и системы связи», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверил:  Доцент департамента цифровых, робототехнических систем и электроники Воронкин Р.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2025 г.

**Тема:** Введение в pandas: изучение структуры Series и базовых операций.

**Цель:** познакомиться с основами работы с библиотекой pandas, в частности, со структурой данных Series.

**Ссылка на GitHub**: <https://github.com/miron2314/DLab-4.git>

**Порядок выполнения работы:**

1.Изучил теоретический материал работы.

2.Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный язык программирования.

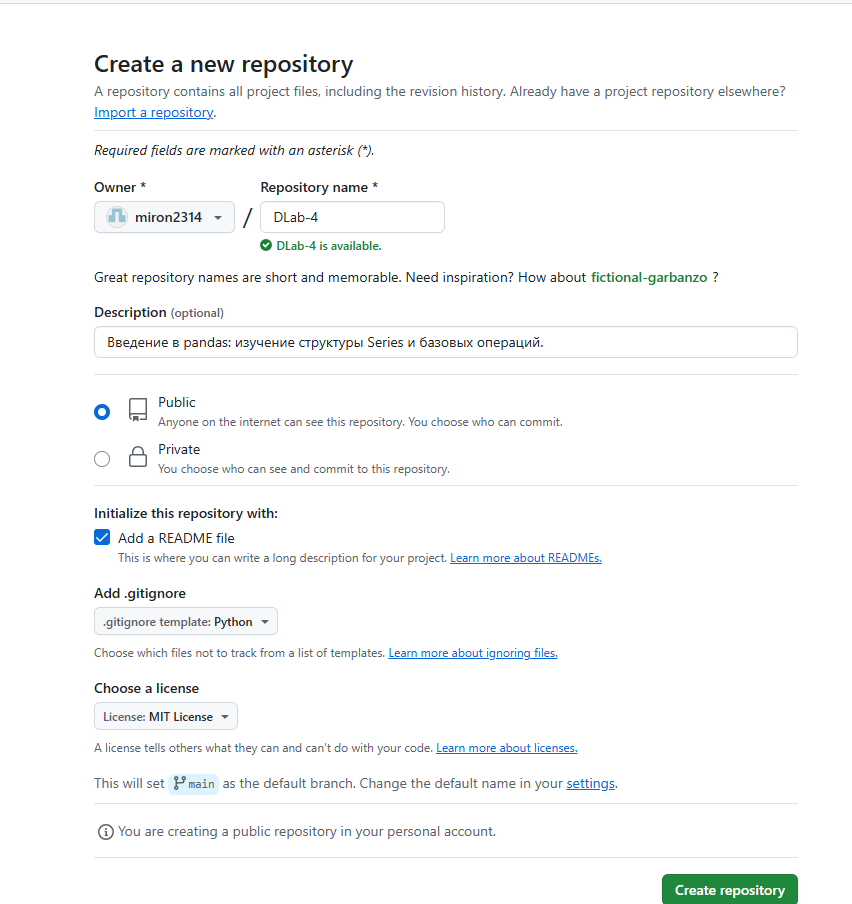


Рисунок 1. Репозиторий

3.Выполнил клонирование репозитория.

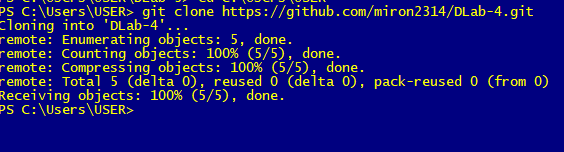


Рисунок 2. Клонирование

4.Проработал примеры работы.



Рисунок 3. Проработка примеров

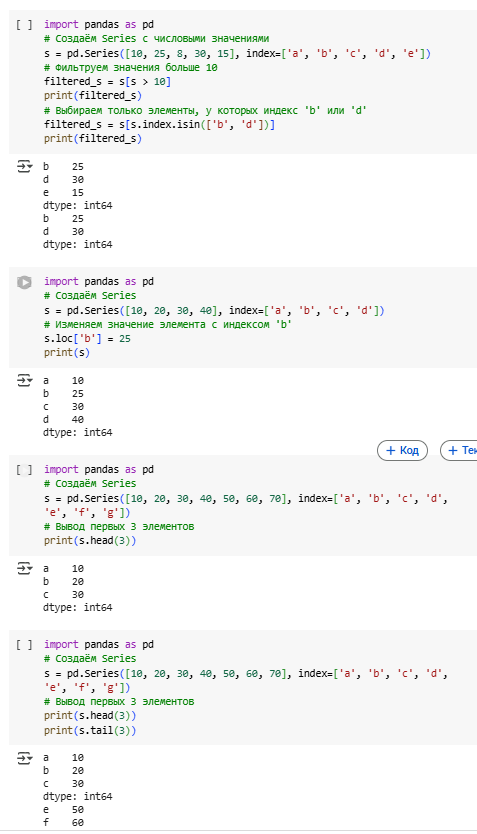


Рисунок 4. Проработка примеров

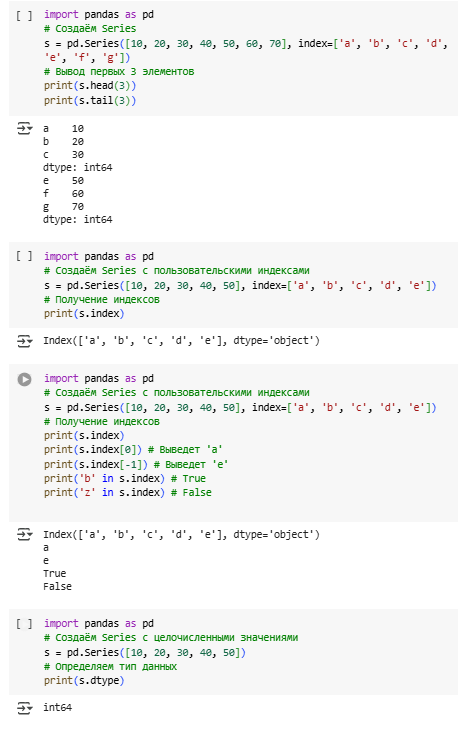
­

Рисунок 5. Проработка примеров

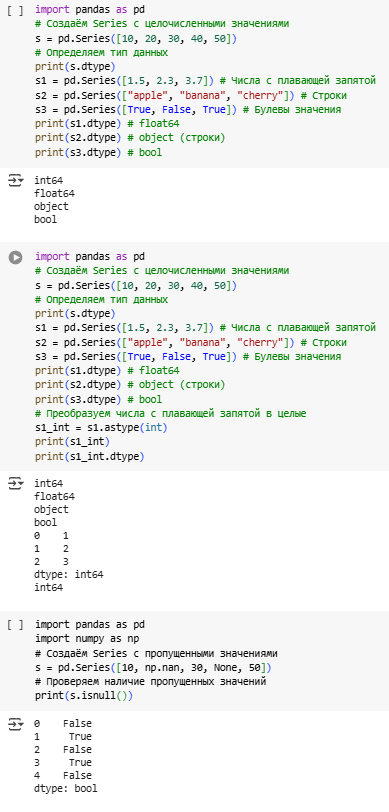


Рисунок 6. Проработка примеров

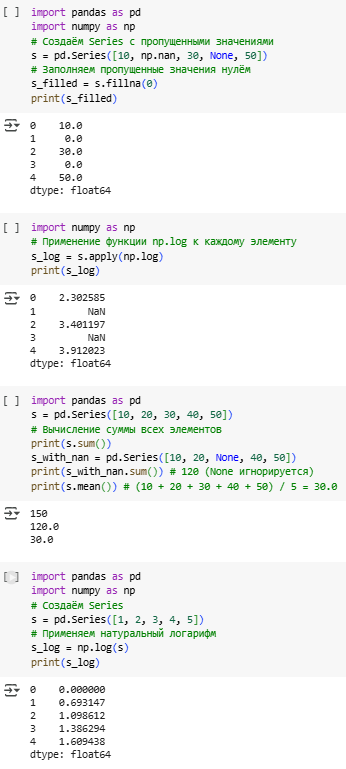


Рисунок 7. Проработка примеров

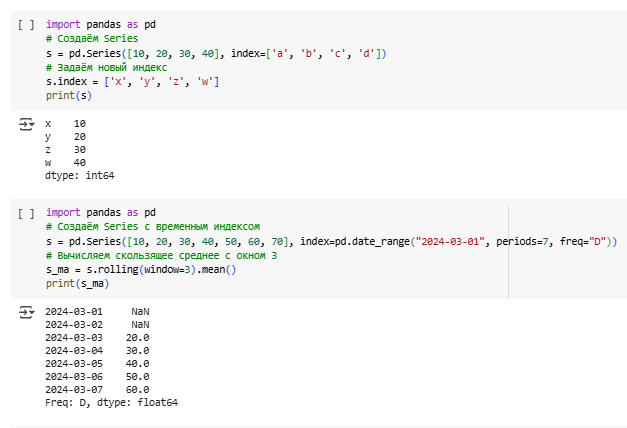


Рисунок 8. Проработка примеров

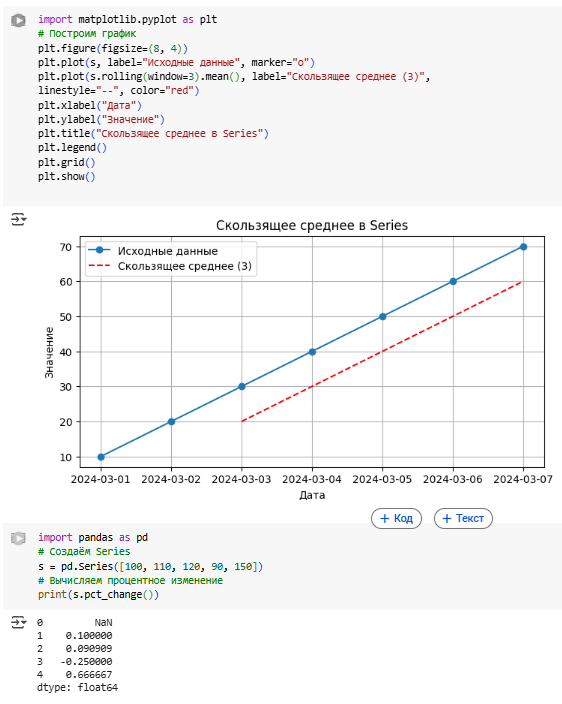


Рисунок 9. Проработка примеров



Рисунок 10. Проработка примеров

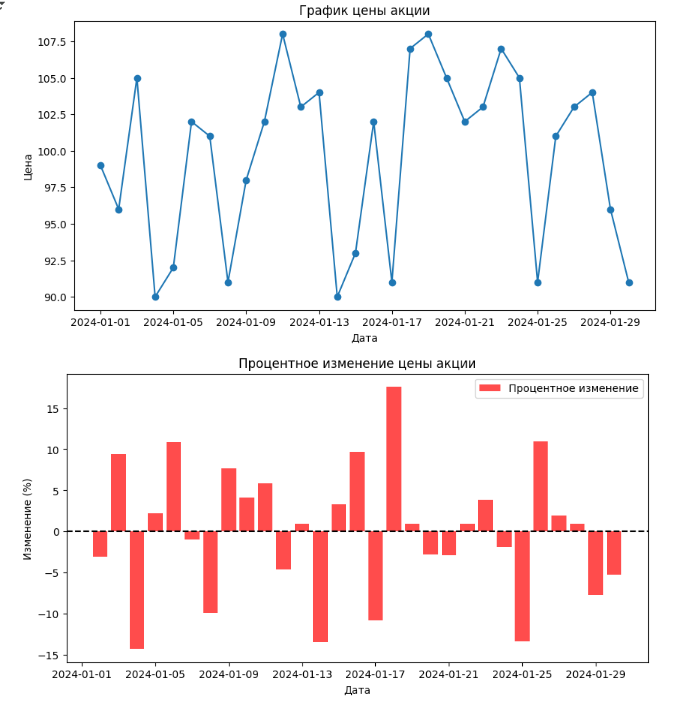


Рисунок 11. Проработка примеров

5.Выполнил практические задания.

**Задание 1.** Создание Series из списка

Создайте Series из списка чисел [5, 15, 25, 35, 45] с индексами ['a', 'b',

'c', 'd', 'e']. Выведите его на экран и определите его тип данных

**Листинг кода:**

#Задание №1

import pandas as pd

data = [5, 15, 25, 35, 45]

index\_labels = ['a', 'b', 'c', 'd', 'e']

series = pd.Series(data=data, index=index\_labels)

print("Серия:")

print(series)

print("\nТип данных:", series.dtype)

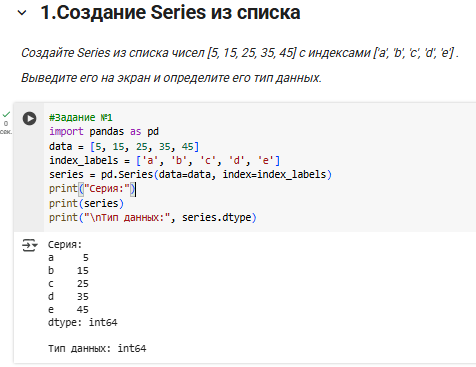
****

Рисунок 12. Выполнение задания 1

**Задание 2.** **Получение элемента Series**

Дан Series с индексами ['A', 'B', 'C', 'D', 'E'] и значениями [12, 24,36,48, 60]

Используйте. loc[] для получения элемента с индексом 'C' и .iloc[] для получения третьего элемента.

**Листинг кода:**

#Задание №2

import pandas as pd

import numpy as np

s = pd.Series([12, 24, 36, 48, 60], index=['A','B','C','D','E'])

print(s.loc['C'])

print(s.iloc[2])

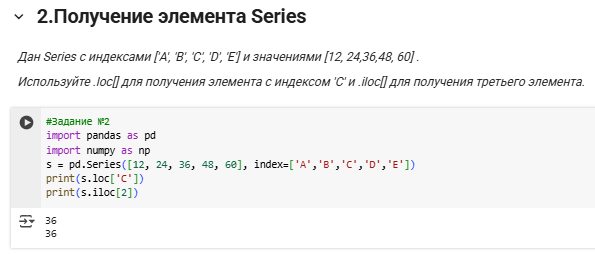


Рисунок 13. Практическое задание 2

**Задание 3.** **Фильтрация данных с помощью логической индексации**

Создайте Series из массива NumPy np.array([4, 9, 16, 25, 36, 49, 64]) .Выберите только те элементы, которые больше 20, и выведите результат.

**Листинг кода:**

#Задание №3

import numpy as np

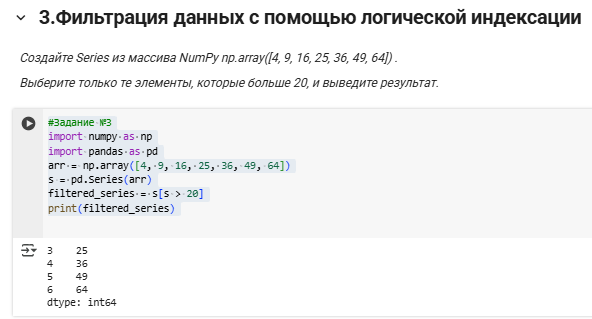
import pandas as pd

arr = np.array([4, 9, 16, 25, 36, 49, 64])

s = pd.Series(arr)

filtered\_series = s[s > 20]

print(filtered\_series)

  
Рисунок 14. Практическое задание 3

**Задание 4. Просмотр первых и последних элементов**

Создайте Series, содержащий 50 случайных целых чисел от 1 до 100 (используйте np.random.randint ).

Выведите первые 7 и последние 5 элементов с помощью. head() и .tail()

**Листинг кода:**

#Задание №4

import pandas as pd

import numpy as np

numbers = np.random.randint(1, 101, 50)

s = pd.Series(numbers)

print('Первые 7 элементов:\n', s.head(7).tolist())

print('Последние 5 элементов:\n',s.tail(5).tolist())

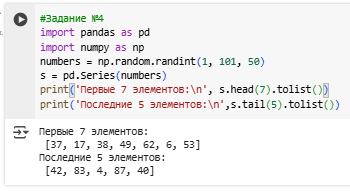


Рисунок 14. Практическое задание 4

**Задание 5.** **Определение типа данных**

Создайте Series из списка ['cat', 'dog', 'rabbit', 'parrot', 'fish'].

Определите тип данных с помощью. dtype, затем преобразуйте его в category с помощью. astype()

**Листинг кода:**

#Задание №5

import pandas as pd

animals = pd.Series(['cat', 'dog', 'rabbit', 'parrot', 'fish'])

current\_dtype = animals.dtype

print("Исходный тип данных:", current\_dtype)

animals\_category = animals.astype('category')

new\_dtype = animals\_category.dtype

print("Новый тип данных:", new\_dtype)

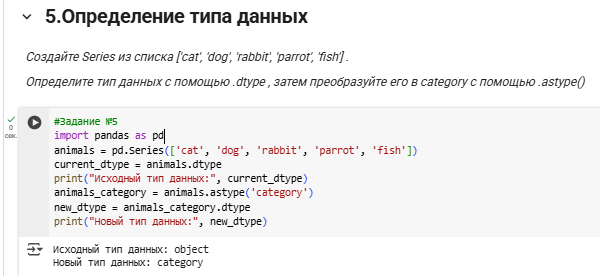


Рисунок 15. Практическое задание 5

**Задание 6.** **Проверка пропущенныъ значений**

Создайте Series с данными [1.2, np.nan, 3.4, np.nan, 5.6, 6.8].

Напишите код, который проверяет, есть ли в Series пропущенные значения ( NaN ), и выведите индексы таких элементов.

**Листинг кода:**

#Задание №6

import pandas as pd

import numpy as np

data = pd.Series([1.2, np.nan, 3.4, np.nan, 5.6, 6.8])

Nan = data.isnull()

print('Вывод индексов пропущенных значений', data.index[Nan].tolist())

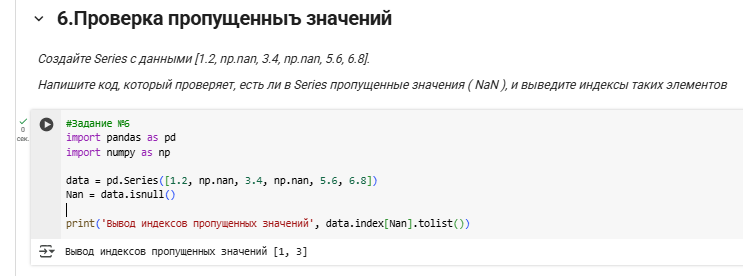
****

Рисунок 16. Практическое задание 6

**Задание 7. Заполнение пропущенных значений**

Используйте Series из предыдущего задания и замените все NaN на среднее значение всех непустых элементов. Выведите результат.

**Листинг кода:**

#Задание №7

import numpy as np

import pandas as pd

data = pd.Series([1.2, np.nan, 3.4, np.nan, 5.6, 6.8])

mean = data.mean()

filled\_data = data.fillna(mean)

print(filled\_data)

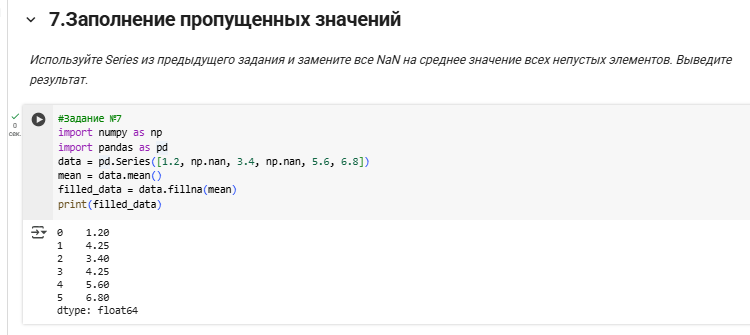


Рисунок 17. Практическое задание 7

**Задание 8** **Арифметические операции с Series**

Создайте два Series

s1 = pd.Series([10, 20, 30, 40], index=['a', 'b', 'c', 'd'])

s2 = pd.Series([5, 15, 25, 35], index=['b', 'c', 'd', 'e'])

Выполните сложение s1 + s2 . Объясните, почему в результате появляются NaN , и замените их на 0.

**Листинг кода:**

#Задание №8

import pandas as pd

import numpy as np

s1 = pd.Series([10, 20, 30, 40], index = ['a', 'b', 'c', 'd'])

s2 = pd.Series([5, 15, 25, 35], index = ['b', 'c', 'd', 'e'])

summ = s1 + s2

print('Полученная сумма:',summ.tolist())

print('Nan появляется из-за несовпадения индексов некоторых элементов.')

zamena = summ.fillna(0)

print('Данные после замены "nan" на 0:',zamena.tolist())

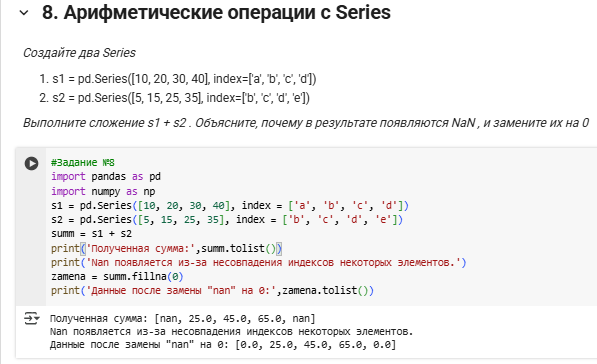


Рисунок 18. Практическое задание 8

**Задание 9.** **Применение функции к Series**

Создайте Series из чисел [2, 4, 6, 8, 10]. Напишите код, который применяет к каждому элементу функцию вычисления квадратного корня с помощью. apply(np.sqrt)

**Листинг кода:**

#Задание №9

import pandas as pd

import numpy as np

s = pd.Series([2, 4, 6, 8, 10])

koren = s.apply(np.sqrt)

print('Вычисленный квадрат каждого из чисел:\n',koren.tolist())

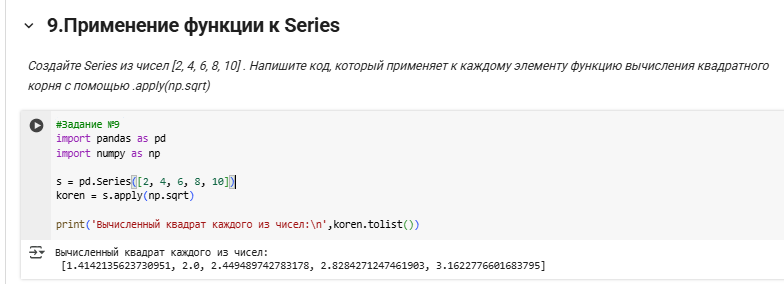


Рисунок 19. Практическое задание 9

**Задание 10. Основные статические методы**

Создайте Series из 20 случайных чисел от 50 до 150 (используйте np.random.randint ). Найдите сумму, среднее, минимальное и максимальное значение. Выведите также стандартное отклонение.

**Листинг кода:**

#Задание №10

import pandas as pd

import numpy as np

numbers = np.random.randint(50, 151, 20)

s = pd.Series(numbers)

print('Полученные числа:', s.tolist())

print('Полученная сумма:', np.sum(s))

print('Среднее значение:', np.mean(s))

print('Минимальное значение:', np.min(s))

print('Максимальное значение:', np.max(s))

print('Стандартное отклонение:', np.std(s))

****

Рисунок 20. Практическое задание 10

**Задание 11. Работа с временными рядами**

Создайте Series , где индексами будут даты с 1 по 10 марта 2024 года ( pd.date\_range(start='2024-03-01', periods=10, freq='D') ), а значениями –случайные числа от 10 до 100. Выберите данные за 5–8 марта.

**Листинг кода:**

#Задание №11

import pandas as pd

import numpy as np

znach = np.random.randint(10, 101, 10)

s = pd.Series(znach, index = pd.date\_range(start='2024-03-01', periods=10, freq='D'))

print(s.loc['2024-03-05': '2024-03-08'])

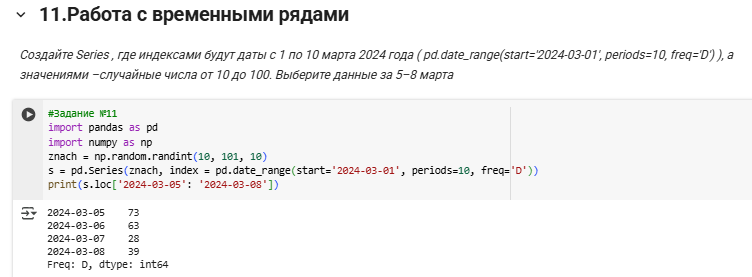


Рисунок 21. Практическое задание 11

**Задание 12. Проверка уникальности индексов**

Создайте Series с индексами ['A', 'B', 'A', 'C', 'D', 'B'] и значениями [10, 20, 30, 40, 50, 60]. Проверьте, являются ли индексы уникальными. Если нет, сгруппируйте повторяющиеся индексы и сложите их значения.

**Листинг кода:**

#Задание №12

import pandas as pd

s = pd.Series([10, 20, 30, 40, 50, 60], index=['A', 'B', 'A', 'C', 'D', 'B'])

if not s.index.is\_unique:

    print("Индекс не уникален.")

    grouped\_s = s.groupby(level=0).sum()

else:

    print("Индекс уникален.")

print(grouped\_s if 'grouped\_s' in locals() else s)

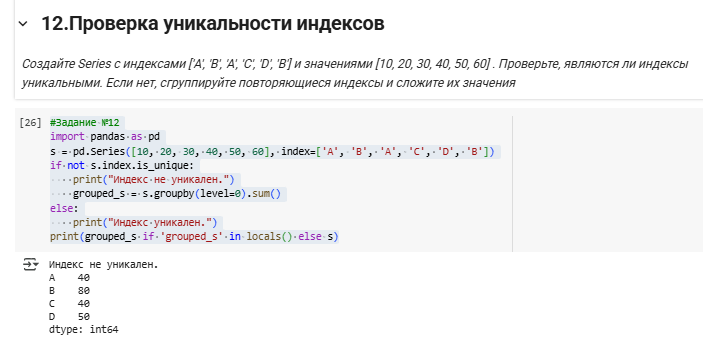


Рисунок 22. Практическое задание 12

**Задание 13. Преобразование строковых дат в DatetimeIndex**

Создайте Series, где индексами будут строки ['2024-03-10', '2024-03-11', '2024-03-12'], а значениями [100, 200, 300].

Преобразуйте индексы в DatetimeIndex и выведите тип данных индекса

**Листинг кода:**

#Задание №13

import pandas as pd

s = pd.Series([100, 200, 300], index=['2024-03-10', '2024-03-11', '2024-03-12'])

s.index = pd.to\_datetime(s.index)

print("Новый индекс:", s.index)

print("Тип данных индекса:", s.index.dtype)

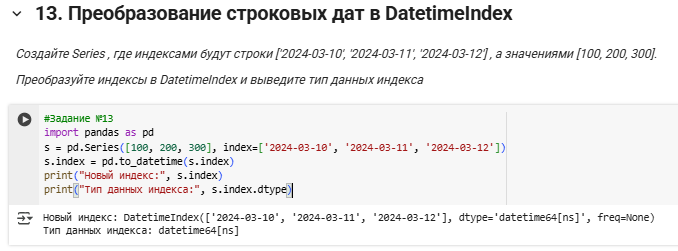


Рисунок 23. Практическое задание 13

**Задание 14. Чтение данных из CSV-файла**

Создайте CSV-файл data.csv со следующими данными:

Дата,Цена

2024-03-01,100

2024-03-02,110

2024-03-03,105

2024-03-04,120

2024-03-05,115

Прочитайте файл и создайте Series, используя "Дата" в качестве индекса

**Листинг кода:**

#Задание №14

import pandas as pd

import numpy as np

s = pd.Series([100, 200, 300], index = ['2024-03-10', '2024-03-11', '2024-03-12'])

s.index = pd.to\_datetime(s.index)

print(s)

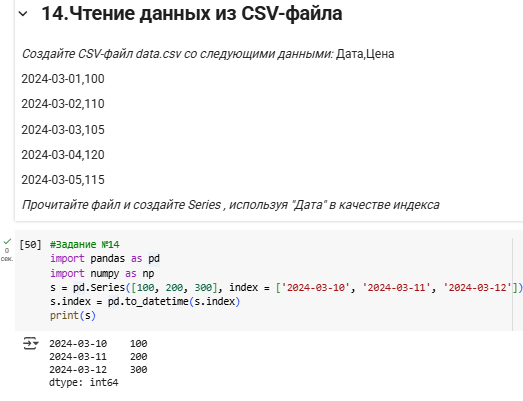


Рисунок 24. Практическое задание 14

**Задание 15. Построение графика на основе Series**

Создайте Series, где индексами будут даты с 1 по 30 марта 2024 года, а значениями – случайные числа от 50 до 150. Постройте график значений с помощью matplotlib . Добавьте заголовок, подписи осей и сетку.

**Листинг кода:**

#Задание №15

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

dates = pd.date\_range(start="2024-03-01", end="2024-03-30")

values = np.random.randint(50, 151, len(dates))

ts = pd.Series(values, index=dates)

plt.figure(figsize=(10, 6))

plt.plot(ts.index, ts.values, marker='o', linestyle='-', color='blue')

plt.title("График временных рядов за март 2024")

plt.xlabel("Дата")

plt.ylabel("Значения")

plt.grid(True)

plt.show()

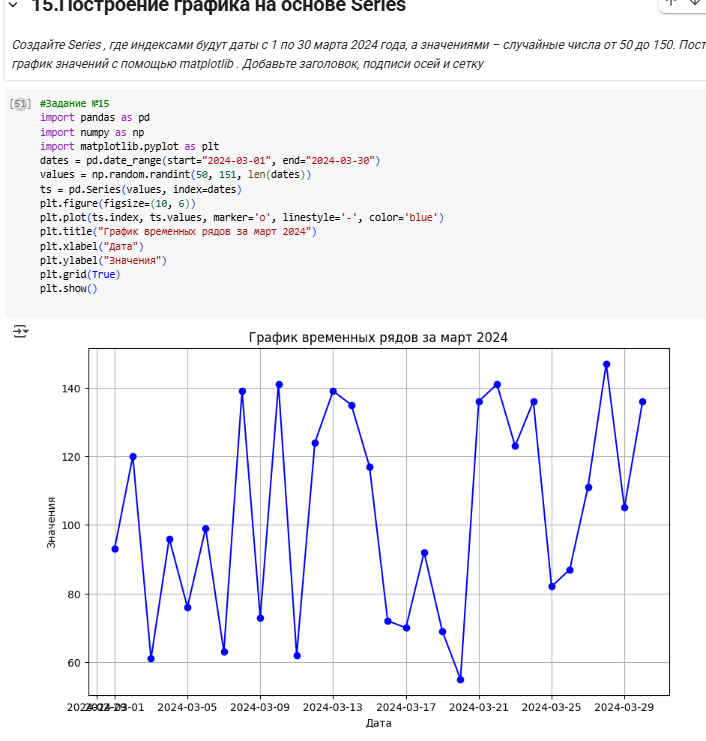
**

Рисунок 25. Практическое задание 15

6. Выполнил индивидуальное задание.

**Вариант 9**

**Индивидуальное задание. Погодные аномалии: резкие изменения температуры**

Создайте Series, где индексами будут даты с 1 по 15 мая 2024 года, а значениями – случайные температуры от -10 до +25 градусов.

Преобразуйте индекс в DatetimeIndex , найдите разницу температур между днями (diff()), выделите дни с изменением больше 5 градусов и постройте график с отображением таких скачков.

**Листинг кода:**

#Индивидуальное задание

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

dates = pd.date\_range(start='2024-05-01', end='2024-05-15')

temperatures = np.random.randint(-10, 26, size=len(dates))

temperature\_series = pd.Series(data=temperatures, index=dates)

temperature\_diff = temperature\_series.diff()

significant\_changes = temperature\_diff[temperature\_diff.abs() > 5]

plt.figure(figsize=(10, 5))

plt.plot(temperature\_series.index, temperature\_series, marker='o', label='Температура')

plt.scatter(significant\_changes.index, temperature\_series[significant\_changes.index], color='red', label='Скачки > 5°C')

plt.title('Температуры с изменениями более 5 градусов')

plt.xlabel('Дата')

plt.ylabel('Температура (°C)')

plt.axhline(0, color='black', linewidth=0.5, linestyle='--')

plt.grid()

plt.legend()

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

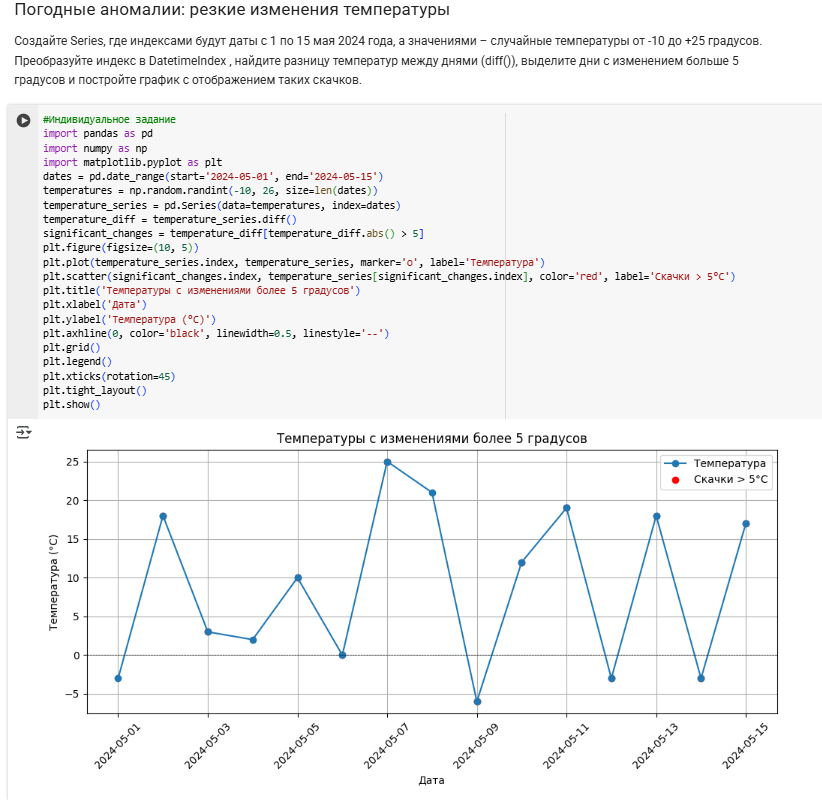


Рисунок 26. Индивидуальное задание

7.Зафиксированы изменения на репозитории и отправлены на сервер GitHub.

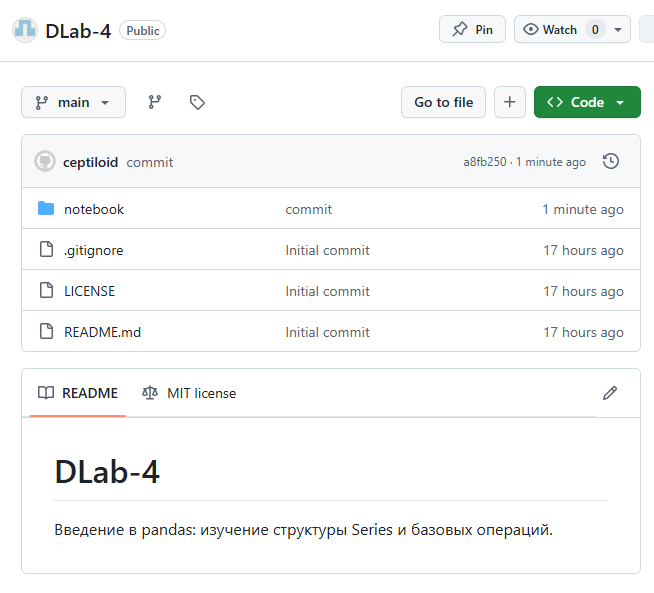


Рисунок 27. Отправка на сервер GitHub

**Ответы на контрольные вопросы:**

**1.Что такое pandas.Series и чем она отличается от списка в Python Pandas.Series** — это одномерный массив индексированных данных из

библиотеки Pandas для работы с данными на Python. Series можно рассматривать как столбец в таблице, который может хранить данные различных типов.

**Некоторые отличия Pandas.Series от списка в Python:**

Однородность. В Series все элементы должны быть одного типа данных, в то время как список может содержать элементы разных типов.

**Эффективность использования памяти.** Series более эффективны, чем списки, так как внутри используют массивы NumPy, которые более компактны и быстрее для числовых вычислений

**2.Какие типы данных можно использовать для создания Series?**

Для создания Series в библиотеке pandas можно использовать различные типы данных, в том числе:

Массивы из numpy: ndarray.

Скалярные величины.

Некоторые основные типы данных, используемые в pandas:

object — текстовые или смешанные числовые и нечисловые значения;

int64 — целые числа;

float64 — числа с плавающей точкой;

bool — булевое значение: True/False;

**3.Как задать индексы при создании Series**

Чтобы задать индексы при создании Series в Pandas, необходимо при вызове конструктора включить параметр index и присвоить ему массив строк с метками.

Общий вид синтаксиса: pd.Series(data, index=index)

**­4. Каким образом можно обратиться к элементу Series по его индексу?**

В библиотеке Pandas для обращения к элементу Series по индексу используют квадратные скобки языка Python. Индекс должен быть целым числом.

**5.В чём разница между. iloc [] и. loc [] при индексации Series?**

loc позволяет индексировать по метке, а. iloc по целому числу позиций, по которым необходимо сделать выборку.

**6.Как использовать логическую индексацию в Series**

Логическая индексация в Series позволяет отбирать элементы структуры на основе логического выражения. Для этого в квадратных скобках записывается логическое выражение, согласно которому будет произведён отбор.

**7.Какие методы можно использовать для просмотра первых и последних элементов Series?**

head(n): Этот метод возвращает первые n элементов Series. Если n не указан, по умолчанию возвращаются первые 5 элементов.

tail(n): Этот метод возвращает последние n элементов Series. Если n не указан, по умолчанию возвращаются последние 5 элементов.

**8.Как проверить тип данных элементов Series?**

В библиотеке pandas для проверки типа данных элементов объекта Series можно использовать атрибут. dtype.

**9.Каким способом можно изменить тип данных Series?**

В библиотеке pandas для изменения типа данных объекта Series можно использовать метод. astype(). Этот метод позволяет преобразовать элементы Series в другой тип данных.

**10.Как проверить наличие пропущенных значений в Series?**

В библиотеке pandas для проверки наличия пропущенных значений в объекте Series можно использовать метод. isnull() или .isna(), а также метод

.any() для определения, есть ли хотя бы одно пропущенное значение.

**11.Методы для заполнения пропущенных значений в Series**

.fillna(): Заполняет пропущенные значения указанным значением, результатом вычисления или методом

**12.Разница между. fillna() и .dropna()**

.fillna(): Заменяет пропущенные значения на указанное значение.

.dropna(): Удаляет строки или столбцы, содержащие пропущенные значения.

**13.Математические операции с Series**

Сложение (+), вычитание (-), умножение (), деление (/), возведение в степень (\\\*), взятие остатка (%), floor division (//). Эти операции выполняются поэлементно.

**14.Преимущество векторизированных операций перед циклами Python**

Скорость: Векторизированные операции (использующие NumPy и pandas) выполняются гораздо быстрее, чем циклы Python, так как они реализованы на C/C++ и используют оптимизированные алгоритмы.

Удобство: Код становится более лаконичным и читаемым, так как не нужно писать циклы для обработки каждого элемента.

**15.Применение пользовательской функции к каждому элементу Series**

Использовать метод .apply(), передав в него имя пользовательской функции.

**16.Агрегирующие функции в Series**

.sum(), .mean(), .median(), .min(), .max(), .std(), .var(), .count(), .size(),

.nunique()

**17.Как узнать минимальное, максимальное, среднее и стандартное отклонение Series**

.min(): Минимальное значение.

.max(): Максимальное значение.

.mean(): Среднее значение.

.std(): Стандартное отклонение.

**18.Сортировка Series**

.sort\_values(): Сортировка по значениям.

.sort\_index(): Сортировка по индексам

**19.Проверка уникальности индексов Series**

series.index.is\_unique: возвращает True, если все индексы уникальны, и False в противном случае.

**20.Как сбросить индексы Series и сделать их числовыми?**

Для сброса индексов Series и присвоения числовых: series.reset\_index(drop=True, inplace=True)

drop=True: удаляет старые индексы.

inplace=True: изменяет Series "на месте".

**21.Как можно задать новый индекс в Series?**

series.index = new\_index: Просто присвоить новый список/массив/Ind ex-объект свойству index. Длина new\_index должна совпадать с длиной series.

series.reindex(new\_index): Создаёт новый Series с указанным new\_index. Если в new\_index есть значения, отсутствующие в старом индексе, им присваивается NaN.

series.set\_axis(new\_index, axis=0): (менее распространенный) Более общий метод для изменения индексов (и столбцов в DataFrame). axis=0 указывает на изменение индекса. Возвращает новый Series.

Выбор зависит от того, нужно ли вам заменить существующий индекс (способ 1) или создать новый Series с другим набором индексов (способы 2 и 3).

**22.Как работать с временными рядами в Series ?**

Создание: pd.Series(data, index=pd.to\_datetime(dates)) - Индекс должен быть DatetimeIndex.

Доступ: series['YYYY-MM-DD'], series['YYYY-MM-DD':'YYYY-MM-

DD'], series.index.dt.year/month/day

Resample: series.resample('D/W/M/A').mean() - Изменение частоты, агрегация.

Shift: series.shift(periods=1) - Сдвиг данных.

Rolling: series.rolling(window=N).mean() - Скользящее среднее.

Пропуски: series.fillna(), series.dropna(), series.interpolate()

**23.Как преобразовать строковые даты в формат DatetimeIndex ?**

pd.to\_datetime(серия\_строк): Самый простой способ. Преобразует Series или список строк в DatetimeIndex.

pd.DatetimeIndex(серия\_строк): создаёт DatetimeIndex напрямую из Series или списка строк.

Оба способа автоматически распознают большинство распространенны х форматов дат. Если формат нестандартный, используйте параметр format=, чтобы указать формат строки даты.

**24.Каким образом можно выбрать данные за определённый временной диапазон?**

Если индекс Series/DataFrame - DatetimeIndex:

Слайсинг строками: df['YYYY-MM-DD':'YYYY-MM-DD'] (включает обе границы диапазона)

loc со строками: df.loc['YYYY-MM-DD':'YYYY-MM-DD'] (то же, но более явный)

Если столбец с датами (не индекс):

Логическая индексация: start\_date = 'YYYY-MM-DD' end\_date = 'YYYY-MM-DD'

mask = (df['date\_column'] >= start\_date) & (df['date\_column'] <= end\_date)

df.loc[mask]

**25.Как загрузить данные из CSV-файла в Series ?**

import pandas as pd

# 1. Загрузка CSV в DataFrame

df = pd.read\_csv('имя\_файла.csv', index\_col='имя\_столбца\_с\_индексом') # 2. Преобразование столбца DataFrame в Series

series = df['имя\_столбца'].squeeze() #squeeze() преобразует DataFrame с одним столбцом в Series

# Альтернатива (если индекс не нужен из CSV)

# series = pd.read\_csv('имя\_файла.csv', usecols=['имя\_столбца']).squeeze() #Если столбец с датами и должен быть индексом

#series = pd.read\_csv('имя\_файла.csv', index\_col='имя\_столбца\_с\_индекс ом', parse\_dates=['имя\_столбца\_с\_индексом'])['имя\_столбца'].squeeze() #если нужно, parse\_dates

**26.Как установить один из столбцов CSV-файла в качестве индекса Series?**

import pandas as pd

series = pd.read\_csv('имя\_файла.csv', index\_col='имя\_столбца\_с\_индексо м')['имя\_столбца'].squeeze()

**27.Для чего используется метод .rolling().mean() в Series ?**

.rolling().mean() используется для вычисления скользящего среднего (или moving average) в Series. Он берёт окно из N последовательных значений и вычисляет их среднее, затем сдвигает окно на одно значение и повторяет процесс. Это сглаживает колебания и показывает тренд.

**28.Как работает метод .pct\_change() ?** **Какие задачи он решает?**

.pct\_change() вычисляет процентное изменение между текущим и предыдущим элементом в Series/DataFrame.

Задачи:

Анализ роста: Определение процентного роста или падения во времени (например, изменение цены акции, рост продаж).

Сравнение изменений: Сравнение скорости изменений между разными периодами или разными временными рядами.

Нормализация данных: Приведение данных к процентным изменениям, чтобы убрать влияние абсолютных значений.

Кратко: вычисляет процентное изменение между последовательными значениями, что полезно для анализа роста и сравнения изменений.

**29.В каких ситуациях полезно использовать. rolling() и .pct\_change() ?**

\*. rolling():

1)Сглаживание временных рядов от шума и случайных колебаний.

2)Выявление трендов и долгосрочных изменений.

3)Фильтрация данных для упрощения анализа.

4).pct\_change():

5)Измерение темпов роста/падения (экономика, финансы).

6)Сравнение волатильности разных активов.

7)Визуализация изменений в данных относительно предыдущего

периода.

**30.Почему NaN могут появляться в Series, и как с ними работать?**

Почему появляются NaN в Series:

Отсутствие данных: Явное отсутствие значения в данных (например, в CSV-файле).

Неопределенные вычисления: Операции, которые не могут быть выполнены (например, деление на ноль).

Объединение/переиндексация: Объединение Series/DataFrames с разными индексами, где некоторые индексы отсутствуют в другом Series.

Сдвиг данных (shift): Сдвиг временного ряда приводит к появлению NaN в начале или конце.

Как работать с NaN:

Обнаружение: series.isna() или series.isnull() - возвращают Series с True/False.

Удаление: series.dropna() - удаляет строки с NaN.

Заполнение: series.fillna(value) - заменяет NaN указанным значением (value может быть числом, средним, предыдущим значением и т.д.).

Интерполяция: series.interpolate() - заполняет NaN на основе соседних значений (линейно, полиномиально и т.д.).

Выбор метода обработки зависит от контекста и задачи анализа.

**Вывод**: в ходе выполнения работы познакомился с основами работы с библиотекой pandas, в частности, со структурой данных Series