Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»

УТВЕРЖДАЮ
Директор Института магистратуры
_____Иванова Е.А.
«__03__» ____июня___ 2024___г.

Рабочая программа дисциплины
Питон для анализа данных

Направление 01.04.02 Прикладная математика и информатика
магистерская программа 01.04.02.04 "Искусственный интеллект: математические модели и прикладные решения"

Для набора 2024 года

Квалификация
Магистр

Составитель программы:

Богачев Т.В., к.ф.-м..н., доцент кафедры прикладной математики и технологий искусственного интеллекта.

ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цели освоения дисциплины: изучение языка программирования Python для дальнейшего использования в разработке научных приложений, ориентированных на обработку данных. Приобретение навыков работы с библиотеками Python обработки больших данных.

Задачи:

- Освоение базовых конструкций и синтаксиса языка программирования Python; изучение основных структур данных;
- Создание полноценной рабочей среды на компьютере учащегося, включающей в себя различный инструментарий, среду для разработки и коллекцию установленных модулей и программ, необходимых для удобной и эффективной работы;
- Приобретение навыков работы с библиотеками Python обработки больших данных.

МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОП ВО

2.1. Учебная дисциплина «Питон для анализа данных» (1 курс магистратуры, 1 семестр) относится к обязательной части блока дисциплин (модулей) и является обязательной дисциплиной.

2.2. Для изучения данной учебной дисциплины необходимы знания, умения и навыки, формируемые дисциплинами изучаемые в бакалавриате (курсы программирования, численные методы, базы данных).

2.3. Перечень последующих учебных дисциплин, для которых необходимы знания, умения и навыки, формируемые данной учебной дисциплиной: «Основы нейронных сетей», «Научно-исследовательский семинар», «Интеллектуальный анализ больших данных», «Прикладное машинное обучение». Знания и навыки, полученные в ходе изучения данной дисциплины, могут использоваться для решения профессиональных задач в научно-исследовательской, научно-производственной и проектной деятельности.

ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций в соответствии с ФГОС ВО и ОП ВО по данному направлению подготовки (специальности):

Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы:

Шифр и формулировка компетенций (результаты освоения ОП)	**Индикаторы компетенций**	**Элементы компетенций, формируемые дисциплиной**
Общепрофессиональные компетенции (ОПК)		
ОПК-4. Способен комбинировать и адаптировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности с учетом требований информационной безопасности	ОПК-4.1. Использует и комбинирует существующие информационно-коммуникационные технологии для решения поставленных задач в области профессиональной деятельности с учетом требований информационной безопасности	***Знания:*** Знает существующие информационно-коммуникационные технологии для решения поставленных задач в области профессиональной деятельности с учетом требований информационной безопасности ***Умения:*** Умеет использовать и комбинировать существующие информационно-коммуникационные технологии для решения поставленных задач в области профессиональной деятельности с учетом требований информационной безопасности ***Навыки:*** Имеет навыки применения существующих информационно-коммуникационных технологий для решения поставленных задач в области профессиональной деятельности с учетом требований информационной безопасно
	ОПК-4.2. Адаптирует существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности	***Знания:*** Знает существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности ***Умения:*** Умеет адаптировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности ***Навыки:*** Имеет навыки адаптирования существующих информационно-коммуникационных технологий для решения задач в области профессиональной деятельности
Профессиональные компетенции (ПК)		
ПК-2. Способен выбирать, разрабатывать и проводить экспериментальную проверку работоспособности программных компонентов систем искусственного интеллекта по обеспечению требуемых	ПК-2.1. Выбирает и разрабатывает программные компоненты систем искусственного интеллекта	ПК-2.1. З-1. Знает основные критерии эффективности и качества функционирования системы искусственного интеллекта: точность, релевантность, достоверность, целостность, быстрота решения задач, надежность, защищенность функционирования систем искусственного интеллекта ПК-2.1. З-2. Знает методы, языки и программные средства разработки программных компонентов систем искусственного интеллекта ПК-2.1. У-1. Умеет выбирать, адаптировать, разрабатывать и интегрировать программные компоненты систем искусственного интеллекта с учетом основных критериев эффективности и качества функционирования

критериев эффективности и качества функционирования	ПК-2.2. Проводит экспериментальную проверку работоспособности систем искусственного интеллекта	ПК-2.2. З-1. Знает методы постановки задач, проведения и анализа тестовых и экспериментальных испытаний работоспособности систем искусственного интеллекта ПК-2.2. У-1. Умеет ставить задачи и проводить тестовые и экспериментальные испытания работоспособности систем искусственного интеллекта анализировать результаты и вносить изменения

СОДЕРЖАНИЕ И СТРУКТУРА ДИСЦИПЛИНЫ

Трудоемкость дисциплины составляет 5 зачетных единиц, 180 часов,
из них 34 часов лекционных занятий, 34 часов лабораторных занятий, 112 часов на самостоятельную работу в течение семестра.

Форма отчетности: зачет

4.1 Содержание дисциплины, структурированное по темам, с указанием видов учебных занятий и отведенного на них количества академических часов

№ п/п	Раздел дисциплины/темы	Семестр	Виды учебной работы, включая самостоятельную работу обучающихся и трудоемкость (в часах)				Формы текущего контроля успеваемости Форма промежуточной аттестации *(по семестрам)*
			Контактная работа преподавателя с обучающимися			Самостоятельная работа	
			Лекции	Семинарские (практические занятия)	Лабораторные занятия		
1	**Раздел 1. Введение в язык программирования Python, простые встроенные типы (целый, вещественный, логический), управляющие конструкции языка (условные операторы, циклы, функции)**	1	3		11	45	Выполнение лабораторных работ, контрольная работа
2	**Раздел 2. Встроенные структуры данных (строки, списки, кортежи, множества, словари). Строки, кодировки, регулярные выражения. Текстовые и двоичные файлы**	1	3		11	45	Выполнение лабораторных работ, контрольная работа
3	**Раздел 3. Библиотеки языка Python для научных расчетов и визуализации данных**	1	4		10	44	Выполнение лабораторных работ
Итого часов			**10**		**32**	**134**	

4.2 План внеаудиторной самостоятельной работы обучающихся по дисциплине

Семестр	Название раздела, темы	Самостоятельная работа обучающихся			Оценочное средство	Учебно-методическое обеспечение самостоятельной работы
		Вид самостоятельной работы	Сроки выполнения	Затраты времени (час.)		
1	Раздел 1. Введение в язык программирования Python, простые встроенные типы (целый, вещественный, логический), управляющие конструкции языка (условные операторы, циклы, функции).	Изучение учебной литературы, выполнение лабораторных работ	6 недель	45	Проверка выполненных заданий	Материалы лекций, рекомендованная учебная литература
1	Раздел 2. Встроенные структуры данных (строки, списки, кортежи, множества, словари). Строки, кодировки, регулярные выражения. Текстовые и двоичные файлы.	Изучение учебной литературы, выполнение лабораторных работ	6 недель	45	Проверка выполненных заданий	
1	Раздел 3. Библиотеки языка Python для научных расчетов и визуализации данных	Изучение учебной литературы, выполнение лабораторных работ	6 недель	44	Проверка выполненных заданий	
Общая трудоемкость самостоятельной работы по дисциплине (час)				**134**		
Бюджет времени самостоятельной работы, предусмотренный учебным планом для данной дисциплины (час)				**134**		

4.3 Содержание учебного материала

Раздел 1. Введение в язык программирования Python, простые встроенные типы (целый, вещественный, логический), управляющие конструкции языка (условные операторы, циклы, функции). Ключевые слова, идентификаторы. Встроенные типы. Операции. Динамическая типизация. Логические выражения, равенство и идентичность объектов. Таблица приоритетов вычисления операторов. Структура программы. Целые и вещественные числа, логические значения. Условный оператор и условное выражение. Циклы for и while, примеры стандартных алгоритмов. Определение и вызов функций, инструкция return. Передача параметров: позиционный способ и передача по ключу, значения по умолчанию. Передача параметров (изменяемые и неизменяемые). Локальные и глобальные переменные.

Раздел 2. Встроенные структуры данных (строки, списки, кортежи, множества, словари). Строки, кодировки, регулярные выражения. Текстовые и двоичные файлы. Строки (str): литералы, экранированные последовательности. Срезы строк. Операции. Строковые методы. Форматирование строк. Кортежи (tuple). Списки (list). Срезы списков, изменение данных с помощью срезов. Операции, инструкция del, методы списков. Встроенные функ-ции. Списки и цикл for. Генераторы списков. Копирование списков, три вида копий. Использование последовательностей (кортежей, списков, строк) совместно с функциями. Использование звездочки в функциях. Алгоритмы. Линейный поиск. Поиск в упорядоченном списке, модуль bisect. Сортировка, метод sort. Сортировки обменом, выбором, включениями. Эффективные методы сортировки. Двумерные списки. Множества (set). Словари (dict). Операции и методы словарей. Генераторы словарей. Использование словарей совместно с функциями и **. Функции (продолжение). Имя функции как переменная. Примеры (функция в качестве параметра другой функции). lambda-выражение (анонимная функция). Итераторы. Создание функций-генераторов (с yield). Создание выражений-генераторов. Использование итераторов. Символы. Коды символов. Понятие кодировки. Однобайтовые кодировки. Двухбайтовые кодировки, Unicode. Регулярные выражения, модуль re в Python. Создание, открытие файлов. Классификация файлов по типу компонент и по способу доступа. Основные операции при работе с файлами. Способы открытия файлов, различия между ними. Текстовые файлы. Двоичные файлы. Последовательный и прямой доступ к файлу. Менеджер контекста with / as. Атрибуты объектов файлов. Модуль Pickle. Работа с файловой системой.

Раздел 3. Библиотеки языка Python для научных расчетов и визуализации данных

Язык Python для научной деятельности. Дистрибутив Anaconda, включающий библиотеки работы с данными. Среда Jupyter Notebook. Пакет для выполнения научных расчетов NumPy. Библиотека выполнения символьных вычислений SymPy. Библиотека для обработки и анализа данных Pandas. Библиотека визуализации данных Matplotlib.

ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

При проведении лекций и лабораторных занятий используются следующие образовательные технологии:

— мультимедийные лекции

— электронные формы контроля

Учебный процесс базируется на концепции компетентностного обучения, ориентированного на формирование конкретного перечня профессиональных компетенций, актуализацию получаемых теоретических знаний. Развертывание компетентностной модели обучения предполагает широкое применение инновационных способов организации учебного процесса, в т.ч. применение метода проектного обучения, технологий управляемого самостоятельного обучения в том числе балльно-рейтинговой системы, а также внедрение системы онлайн-поддержки внеаудиторной работы студентов.

Дисциплина может быть реализована частично или полностью с использованием ЭИОС Университета (ЭО и ДОТ). Аудиторные занятия и другие формы контактной работы обучающихся с преподавателем могут проводиться с использованием платформ Microsoft Teams, ZOOM, Skype, MOODLE и др., в том числе, в режиме онлайн-лекций и онлайн- семинаров

ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Полный комплект контрольно-оценочных материалов (Фонд оценочных средств) оформляется в виде приложения к рабочей программе дисциплины.

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

7.1. Основная литература.

1. {Печатный ресурс} Шелудько, Виктория Михайловна. Основы программирования на языке высокого уровня Python [Текст]: учебное пособие / В. М. Шелудько ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего образования "Южный федеральный университет", Инженерно-технологическая академия - Ростов-на-Дону: Издательство Южного федерального университета, 2017. - 146 с. Кол-во: 5 (2017)
2. [Электронный ресурс репозитория: https://hub.lib.sfedu.ru/repository/material/800757090/] Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули: Учебное пособие/ Виктория Михайловна Шелудько, 2017.
3. [Электронный ресурс biblioclub: http://biblioclub.ru/index.php?page=book&id=83142] Долгов, А.И. Алгоритмизация прикладных задач: учебное пособие / А.И. Долгов. - Москва: Флинта, 2011. - 136 с.

7.2. Дополнительная литература.

1. [Электронный ресурс biblioclub: http://biblioclub.ru/index.php?page=book&id=362986] Моделирование систем: Подходы и методы: учебное пособие / В.Н. Волкова, Г.В. Горелова, В.Н. Козлов и др.; Министерство образования и науки Российской Федерации, Санкт-Петербургский государственный политехнический университет. - Санкт-Петербург: Издательство Политехнического университета, 2013. - 568 с. (ресурс доступен до 05.02.2021)
2. {Печатный ресурс} Плас, Джейк Вандер. Python для сложных задач [Текст]: наука о данных и машинное обучение : [16+] / Дж. Вандер Плас ; [перевод с английского И. Пальти] - Санкт-Петербург [и др.]: Питер, 2018. - 572, [2] с. Кол-во: 1 (2018)
3. [Электронный ресурс biblioclub: http://biblioclub.ru/index.php?page=book&id=480500] Жуковский О. И. Информационные технологии и анализ данных: учебное пособие / О.И. Жуковский - Томск: Эль Контент, 2014. - 130 с. (ресурс доступен до 16.06.2022)
4. [Электронный ресурс biblioclub: http://biblioclub.ru/index.php?page=book&id=278426] Крутиков В. Н. Анализ данных / В.Н. Крутиков; В.В. Мешечкин - Кемерово: Кемеровский государственный университет, 2014. - 138 с. (ресурс доступен до 15.10.2022)

7.3. Список авторских методических разработок.

7.4. Периодические издания *(при необходимости)*

Периодические издания рекомендуются научным руководителем по выполнению ВКР.

7.5. Перечень ресурсов сети Интернет, необходимых для освоения дисциплины

Университетская библиотека online: http://biblioclub.ru/index.php?page=main_ub_red
MathWorld. http://mathworld.wolfram.com
Электронно-библиотечная система (ЭБС) ЮРАЙТ www.biblio-online.ru

7.6. Программное обеспечение информационно-коммуникационных технологий

Операционная система Microsoft Windows и пакет Microsoft Office

МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

8.1. Учебно-лабораторное оборудование

При проведении дисциплины учащиеся должны быть обеспечены:

1. Лекционной аудиторией с мультимедийным презентационным оборудованием для демонстрации презентаций и иллюстративного материала.

2. Аудиторией для лабораторных занятий с аппаратными и программными средствами в соответствии с реализуемой учебной тематикой.

8.2. Программные средства

Microsoft Windows, Microsoft Office, Windows CAL's Среда программирования Python.

Методические указания для обучающихся по освоению дисциплины

Методические указания приведены в учебных пособиях, перечисленных в разделе VII.

УЧЕБНАЯ КАРТА ДИСЦИПЛИНЫ

Питон для анализа данных

Трудоемкость: 5 зач.ед.

Форма промежуточной аттестации: зачет

Курс 1, семестр 1

Код и наименование направления подготовки (специальности): 01.04.02 «Прикладная математика и информатика»

Магистерская программа: «Искусственный интеллект: математические модели и прикладные решения»

№	Виды контрольных мероприятий	Текущий контроль	Рубежный контроль
	Раздел 1. Введение в язык программирования Python, простые встроенные типы (целый, вещественный, логический), управляющие конструкции языка (условные операторы, циклы, функции).	**23**	**10**
1.	Посещение лекций	3	
2.	Лабораторные работы	20	
3.	Контрольная работа		10
	Раздел 2. Встроенные структуры данных (строки, списки, кортежи, множества, словари). Строки, кодировки, регулярные выражения. Текстовые и двоичные файлы.	**23**	**10**
1.	Посещение лекций	3	
2.	Лабораторные работы	20	

3.	Контрольная работа		10
	Раздел 3. Библиотеки языка Python для научных расчетов и визуализации данных	**34**	
1.	Посещение лекций	4	
2.	Лабораторные работы	30	
	Всего	**80**	**20**
	ИТОГО	**100**	

Преподаватель_____

Приложение

к рабочей программе

(модулю)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»
Институт магистратуры
Кафедра фундаментальной и прикладной математики

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

ПО ДИСЦИПЛИНЕ (МОДУЛЮ) / ПРАКТИКЕ

ПИТОН ДЛЯ АНАЛИЗА ДАННЫХ

Направление подготовки / специальность

01.04.02 «Прикладная математика и информатика»

Ростов-на-Дону, 2024

ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ, ФОРМИРУЕМЫХ ДИСЦИПЛИНОЙ

«Питон для анализа данных»

Код компетенции	Формулировка компетенции
1	2
ПК	**ПРОФЕССИОНАЛЬНЫЕ КОМПЕТЕНЦИИ**
ПК-2	Способен выбирать, разрабатывать и проводить экспериментальную проверку работоспособности программных компонентов систем искусственного интеллекта по обеспечению требуемых критериев эффективности и качества функционирования
ОПК	**ОБЩЕПРОФЕССИОНАЛЬНЫЕ КОМПЕТЕНЦИИ**
ОПК-4	Способен комбинировать и адаптировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности с учетом требований информационной безопасности

ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ

«Питон для анализа данных»

№ п/п	*Контролируемые разделы дисциплины**	*Код контролируемой компетенции*	Наименование *оценочного средства***
1.	**Раздел 1. Введение в язык программирования Python, простые встроенные типы (целый, вещественный, логический), управляющие конструкции языка (условные операторы, циклы, функции)**	ОПК-4.1	Лабораторные работы, контрольная работа
2.	**Раздел 2. Встроенные структуры данных (строки, списки, кортежи, множества, словари). Строки, кодировки, регулярные выражения. Текстовые и двоичные файлы**	ОПК-4.2	Лабораторные работы, контрольная работа
3.	**Раздел 3. Библиотеки языка Python для научных расчетов и визуализации данных**	ПК-2	Лабораторные работы

** Наименование раздела указывается в соответствии с рабочей программой дисциплины.*

***Наименование оценочного средства указывается в соответствии с учебной картой дисциплины.*

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ КОНТРОЛЬНОЙ РАБОТЫ

по дисциплине «Питон для анализа данных»

Раздел 1. Контрольная 1. (10 баллов)

1. Написать программу вычисляющую выражение $y = \frac{\sin^3(x)}{2\sqrt{\pi}}$.

2. Какой результат вернут следующие выражения (здесь b1 = True, b2 = False):

(1) True and b1 or b2

(2) 5 and 3 and 4

(3) not 3 != 5 or not 2 >= True or False == 0

(4) 10 // 2 // 2 or not 1 and b1 * b1 == b1

Запишите эти выражения, и в каком порядке выполняются операции в них: сверху над каждой операцией поставьте номер – порядок ее выполнения.

3. Программа вводит натуральное число *N* и проверяет, верно ли, что все цифры в его десятичной записи нечетны. Результат работы программы – сообщение.

4. Программа вводит действительное число *X* >10, а затем вычисляет и выводит на экран элементы последовательности $a_k = (k^3 + 1)/(k^2 + 2)$, (где k = 1, 2, …) удовлетворяющие условию $X < a_k < 5 \cdot X$. Используйте цикл while.

Раздел 2. Контрольная 2. (10 баллов)

1. Решите *<u>одну</u>* из следующих задач:

   а. Дана строка S, состоящая из малых латинских букв и цифр. Требуется малые латинские буквы a, b, c заменить большими, например: 'f56iau0vbn2' $\rightarrow$ ' f56iAu0vBn2'.

   б. Дана строка S, являющаяся текстом (состоит из букв и пробелов). Требуется удалить из нее пятибуквенное слово, а если такого нет – выдать сообщение.

2. Дан список целых чисел. Определить количество элементов списка отличных от его последнего элемента.

3. Дан список целых чисел. Определить количество вхождений в него каждого числа, вывести его рядом с каждым элементом списка. Вспомогательный список не вводить.

4. Проверить, есть ли в двумерном списке A[n, m] с целочисленными элементами строка, числа в которой образуют возрастающую последовательность, и выдать сообщение. Если такая строка есть, она меняется с последней строкой.

Критерии оценки

Для каждого задания указано максимальное количество баллов M **(**учебная карта дисциплины).

Студенту выставляется

M баллов, если задание выполнено полностью и не содержит ошибок,

$0.75 * M$ баллов, если задание выполнено, но допущены неточности, например, вычислительного характера,

$0.5 * M$ баллов, если имеются существенные ошибки, но общая схема решения правильна

$0.25 * M$ баллов, если решение было начато, но задание выполнено лишь частично, существенные шаги не сделаны,

0 баллов, если задание не выполнено.

Для каждого задания указывается срок его выполнения. По 20% от максимального количества баллов снимается за каждую неделю просрочки.

После суммирования всех набранных баллов, округление производится до ближайшего целого числа (например, 45.5 округляется до 46, 45.25 округляется до 45).

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»
Кафедра фундаментальной и прикладной математики

КОМПЛЕКТ ЛАБОРАТОРНЫХ РАБОТЫ

по дисциплине «Питон для анализа данных»

Раздел 1. Занятие 3. Функции

Пример 1. Вычислите выражение (sign(x) + sign(y))*sign(x + y). При вычислении задачи определить и использовать функцию sign:

$$\text{sign}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$$

```
def sign(x):
    if x < 0: return -1
    elif x == 0: return 0
    else: return 1


# Ввод данных:
x = float(input('x = '))
y = float(input('y = '))
# Вычислить выражение:
res = (sign(x) + sign(y)) * sign(x + y)
# Результат:
print('res = %7.3f' % res)
```

Результат работы программы:

x = 5

y = 7

res = 2.000

Пример 2. Функция вычисляет min и max из 3-х чисел (возврат нескольких значений).

```
def minmax(a, b, c):
    _max = _min = a
    if b < _min: _min = b
    if c < _min: _min = c
    if b > _max: _max = b
    if c > _max: _max = c
    return _min, _max       # - возврат 2-х значений
```

```
print(minmax(5, 2, 7))       # Результат кортеж: (2, 7)
# Присваивание последовательностей:
m1, m2 = minmax(-1.2, 20.5, -7.7)
print('min =', m1)           # Результат: -7.7
print('max =', m2)           # Результат: 20.5
```

Пример 3. Найти все трехзначные натуральные числа, у которых сумма цифр равна 7 (сумма цифр – функция).

```
def sum_digits(n):
   return n // 100 + n // 10 % 10 + n % 10

# Основная программа:
for n in range(100, 1000):
  # В цикле вызываем функцию на каждой итерации:
   if sum_digits(n) == 7: print(n, end = ', ')
```

Результат работы программы:

106, 115, 124, 133, 142, 151, 160, 205, 214, 223, 232, 241, 250, 304, 313, 322, 331, 340, 403, 412, 421, 430, 502, 511, 520, 601, 610, 700,

Пример 4. Определяем функцию, вычисляющую расстояние между двумя точками на плоскости по формуле (евклидово расстояние):

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Берем по умолчанию значения точек равные (0, 0). Используем разные способы вызова: позиционный, по имени, с использованием значений по умолчанию.

```
# Функция со значениями по умолчанию:
def distance(x1 = 0, y1 = 0, x2 = 0, y2 = 0):
    return ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5

# Вызовы функции:
# Со значениями по умолчанию:
print(distance())            # Вернет 0

# Задаем 1-ю точку: (x1 = 3, y1 = 4),
# вторая точка по умолчанию (x2 = 0, y2 = 0):
print(distance(3, 4))  # Вернет 5

# Задаем по имени x1 и x2, y1 и y2 - по умолчанию.
#(x1 = 3, y1 = 0), (x2 = -4, y2 = 0):
print(distance(x1 = 3, x2 = - 4)) # Вернет 7

# Задаем (x1 = 1, y1 = 0), (x2 = 0, y2 = 1).
```

```
print(distance(1, y2 = 1)) # Вернет 1.4142135623730951
```

Самостоятельная работа в компьютерном классе

Функции

1. Даны два натуральных числа. Выяснить, в каком из них сумма цифр больше. Определить функцию для расчета суммы цифр натурального числа.
2. Даны два натуральных числа. Выяснить, в каком из них больше цифр. Определить функцию для расчета количества цифр натурального числа.
3. Найти значение выражения

$$c_n^k = \frac{n!}{k! \cdot (n-k)!},$$

при $0 \le k \le n$, и $n!$ означает факториал числа n. Определить функцию для расчета факториала числа: $n! = 1 \cdot 2 \ldots \cdot n$, для $n > 0$ и $0! = 1$.

Функции в этом блоке имеют 3 входных параметра-числа и возвращают 2 значения в return (см. пример 2).

Используйте значения по умолчанию в заголовке функции. Примените разные способы вызова: позиционный, по имени, с использованием значений по умолчанию (см. пример 4).

4. Находит сумму и произведение трех чисел.
5. Находит первый и второй максимум из трех чисел трех чисел ($\max_2 \le \max_1$). Например, для чисел 5, 7, 10, результат: $\max_2 = 7$ $\max_1 = 10$.
6. Среднее арифметическое и геометрическое трех чисел (Среднее арифметическое , Среднее геометрическое).

---

7. Найти все трехзначные простые числа. Определить функцию проверки числа на простоту.
8. Найти все четырехзначные числа палиндромы. Определить функцию, позволяющую распознавать числа-палиндромы.
9. Получить все шестизначные счастливые номера. Счастливым называют такое шестизначное число, в котором сумма его первых трех цифр равна сумме его последних трех цифр. Определить функцию для расчета суммы цифр трехзначного числа.

Каждому студенты из общего списка задач предлагается сделать часть задач (индивидуальная задача из каждого раздела).

Критерии оценки

Для каждого задания указано максимальное количество баллов M **(**учебная карта дисциплины).

Студенту выставляется

M баллов, если задание выполнено полностью и не содержит ошибок,

$0.75 * M$ баллов, если задание выполнено, но допущены неточности, например, вычислительного характера,

$0.5 * M$ баллов, если имеются существенные ошибки, но общая схема решения правильна

$0.25 * M$ баллов, если решение было начато, но задание выполнено лишь частично, существенные шаги не сделаны,

0 баллов, если задание не выполнено.

Для каждого задания указывается срок его выполнения. По 20% от максимального количества баллов снимается за каждую неделю просрочки.

После суммирования всех набранных баллов, округление производится до ближайшего целого числа (например, 45.5 округляется до 46, 45.25 округляется до 45).

КОМПЛЕКТ ЛАБОРАТОРНЫХ РАБОТЫ

по дисциплине «Питон для анализа данных»

Раздел 2. Занятие 4. Строки

Срезы. Примеры

```
# Определяем строку:
s = '0123456789'
print(len(s))                  # 10 – длина строки
# С одним параметром:
print(s[0])                    # Вернет 0
print(s[15])                   # ошибка! Выход за границы
s[1] = 'a'; print(s)           # ошибка! Нельзя менять

# С двумя параметрами:
print(s[0:1])                  # 0
print(s[15:])             # пустая строка (выход за границы – можно)
print(s[2:5])                  # 234
print(s[0:len(s) - 1])    # 012345678
print(s[:len(s)])         # 0123456789
print(s[-20:20])               # 0123456789
print(s[:])                    # 0123456789
print(s[-len(s): -1])          # 012345678
print(s[-len(s): ])            # 0123456789
print(s[-7: 7])                # 3456  ( == s[len(s)-7: 7])

# С тремя параметрами:
print(s[::2])                  # 02468
print(s[1::2])                 # 13579
print(s[::-1])                 # 9876543210 (переворот строки)
print(s[::-2])                 # 97531
print(s[::])  # 0123456789 или ошибка (зависит от версии!)
print(s[3:7:2])           # 35
print(s[-7:-4:2])         # 35
```

```
s = s[len(s) // 2:] + s[:len(s) // 2]
print(s)                          # 5678901234
```

Сравнение и коды символов, chr, ord, min, max

Пример. **Символы с кодами от 34 до 128:**

```
for i in range(34, 129):
    print('%d %s' % (i, chr(i)), end = ' | ')
```

34 " | 35 # | 36 $ | 37 % | 38 & | 39 ' | 40 (| 41) | 42 * | 43 + | 44 , | 45 - | 46 . | 47 / | 48 0 | 49 1 | 50 2 | 51 3 | 52 4 | 53 5 | 54 6 | 55 7 | 56 8 | 57 9 | 58 : | 59 ; | 60 < | 61 = | 62 > | 63 ? | 64 @ | 65 A | 66 B | 67 C | 68 D | 69 E | 70 F | 71 G | 72 H | 73 I | 74 J | 75 K | 76 L | 77 M | 78 N | 79 O | 80 P | 81 Q | 82 R | 83 S | 84 T | 85 U | 86 V | 87 W | 88 X | 89 Y | 90 Z | 91 [| 92 \ | 93] | 94 ^ | 95 _ | 96 ` | 97 a | 98 b | 99 c | 100 d | 101 e | 102 f | 103 g | 104 h | 105 i | 106 j | 107 k | 108 l | 109 m | 110 n | 111 o | 112 p | 113 q | 114 r | 115 s | 116 t | 117 u | 118 v | 119 w | 120 x | 121 y | 122 z | 123 { | 124 | | 125 } | 126 ~ | 127 ⍰ | 128 □ |

Пример. **Русские буквы:**

```
# Большие:
for i in range(ord('А'), ord('Я') + 1):
print('%d %s' % (i, chr(i)), end = ' | ')
i = 'Ё'
print('%d %s' % (ord(i), i))
```

1040 А | 1041 Б | 1042 В | 1043 Г | 1044 Д | 1045 Е | 1046 Ж | 1047 З | 1048 И | 1049 Й | 1050 К | 1051 Л | 1052 М | 1053 Н | 1054 О | 1055 П | 1056 Р | 1057 С | 1058 Т | 1059 У | 1060 Ф | 1061 Х | 1062 Ц | 1063 Ч | 1064 Ш | 1065 Щ | 1066 Ъ | 1067 Ы | 1068 Ь | 1069 Э | 1070 Ю | 1071 Я | 1025 Ё

```
# Маленькие:
for i in range(ord('а'), ord('я') + 1):
print('%d %s' % (i, chr(i)), end = ' | ')
i = 'ё'
print('%d %s' % (ord(i), i))
```

1072 а | 1073 б | 1074 в | 1075 г | 1076 д | 1077 е | 1078 ж | 1079 з | 1080 и | 1081 й | 1082 к | 1083 л | 1084 м | 1085 н | 1086 о | 1087 п | 1088 р | 1089 с | 1090 т | 1091 у | 1092 ф | 1093 х | 1094 ц | 1095 ч | 1096 ш | 1097 щ | 1098 ъ | 1099 ы | 1100 ь | 1101 э | 1102 ю | 1103 я | 1105 ё

Коды символов:

коды:	48–57	65–90	97–122	1025	1040–1071	1072–1105	1105

символы: **‘0’–‘9’** **‘A’–‘Z’** **‘a’–‘z’** **‘Ё’** **‘А’–‘Я’** **‘а’–‘я’** **‘ё’**

Строки можно сравнивать (< <= > >= == !=). Сравниваются коды символов.

```
'абв' == 'абв'              # True
'абв' > 'АБВ'               # True
'абв' > 'аб'          # True
max('z1+яЯёЁ')    # 'ё'  (возвращает символ с max-м кодом)
min('z1+яЯёЁ')    # '+'  (возвращает символ с min-м кодом)
```

Основные методы строк

s.find(t, start, end)	Возвращает позицию самого первого (крайнего слева) вхождения подстроки t в строку s (или в срез строки s[start:end]); если подстрока t не найдена, возвращается число –1
s.rfind(t, start, end)	Поиск самого последнего (крайнего справа) вхождения подстроки (или в срез строки s[start:end]); если подстрока t не найдена, возвращается число –1

Если надо просто узнать, входит ли одна строка в другую, то можно использовать t in s.

Пример 1. Из строк вида “26.11.2020” или “1.1.21” будем вырезать число, месяц, год:

```
s = input('dd.mm.yyyy:')
date = s[:s.find('.')]
month = s[s.find('.') + 1 : s.rfind('.')]
year = s[s.rfind('.') + 1:]
print('дата: ', date)
print('месяц: ', month)
print('год: ', year)
```

Пример 2. Ищем вхождение строки с пересечениями. Например, строка ‘аа’ входит в строку ‘ааа аа’ 3 раза (без пересечений 2 раза найдет метод count).

```
s = input('s = ')
subs = input('subs = ')
count_substring = 0             # количество вхождений подстрки
pos = 0                         # текущая позиция
pos = s.find(subs, pos)  # ищем начиная с pos и до конца
while pos != -1:
    count_substring += 1
    pos = s.find(subs, pos + 1) # начиная с (pos + 1) и до конца
```

```
print('Количество вхождений: ', count_substring)
```

s.replace(t, u, n)	Заменяет в строке s, каждое (но не более n, если этот аргумент определен) вхождение подстроки t на подстроку u

Пример 1. Заменим все вхождения строки

```
s = ' Пример строки '
subs = ' '     # пробелы
newsubs = '*'       # будем менять на звездочки
s = s.replace(subs, newsubs) # обязательно нужно присваивать
print(s)
```

Примерстроки*

Пример 2. Заменим одно вхождения строки

```
verse = '''Наша Таня громко плачет:
Уронила в речку мячик.
- Тише, Танечка, не плачь:
Не утонет в речке мяч.'''
# Заменяем только первое вхождение:
verse_new = verse.replace('мяч', 'МЯЧ', 1)
print(verse_new)
```

Наша Таня громко плачет:

Уронила в речку МЯЧик.

- Тише, Танечка, не плачь:

Не утонет в речке мяч.

Пример 3. Удаление всех вхождений подстроки

```
s = '1101011101'
subs = '101' # удалим все 101
s = s.replace(subs, '') # замена на пустую строку
print(s)
```

101

Пример 4. Напишем свою замену одного вхождения с помощью поиска и срезов

```
s = input('s = ')
subs = input('subs = ')
newsubs = input('newsubs = ')
pos = s.find(subs)          # Поиск подстроки
if pos >= 0:                # Если нашли, то заменяем
    s = s[:pos] + newsubs + s[pos + len(subs):]
print(s)
```

s.count(t, start, end)	Возвращает число вхождений строки t в строку s (или в срез строки s[start:end])

Пример. **Ищет без пересечений, т.е. в строку ‘аааааа’ подстрока ‘аа’ входит 3 раза**

```
'aaaaaa'.count('aa')                    # вернет 3
# Сколько нулей в левой половине строки:
s = '1010110101100101010'
s.count('0', 0, len(s) // 2)          # вернет 4
```

s.join(seq)	Объединяет все элементы последовательности строк seq, вставляя между ними строку s
s.split(sep = None, maxsplit = –1)	Возвращает список слов в строке, используя sep в качестве строки-разделителя. Если задан maxsplit, то выполняется не более maxsplit расщеплений. Если задан sep, то последовательные разделители не группируются вместе и считаются разделителями пустых строк (например, '1,,2'.split(',') возвращает ['1', '', '2']). Аргумент sep может состоять из нескольких символов (например, '1<>2<>3'.split('<>') возвращает ['1', '2', '3']). Если sep не указан или отсутствует, применяется другой алгоритм разбиения: несколько последовательных пробелов рассматриваются как один разделитель, и результат не будет содержать пустых строк. Есть еще функция **rsplit**

Пример 1. **Разбиваем строку на слова (split)**

```
s = ' abc 123 AB АБВГ0 '
L = s.split(' ')    # Разбиваем на слова по пробелам
print(L)
```

```
# ['', '', 'abc', '', '', '123', 'AB', '', 'АБВГ0', '', '']
# Лучше так:
L = s.split() # Разбиваем на слова
print(L)      # ['abc', '123', 'AB', 'АБВГ0']
# Если не пустое слово и состоит лишь из цифр, выведем его:
for slovo in L:
    if slovo.isdigit(): print(slovo)
```

Результат:

123

Пример 2. Объединяем слова в одну строку (join)

```
res = '*'.join(['123', ' abc', ' ', 'AB']) # список или кортеж
print(res)          # Результат: 123* abc* *AB
```

Самостоятельная работа в компьютерном классе

Строки

Замечание 1. Одно и то же задание для строк часто можно сделать несколькими способами. Например, посимвольно проходя строку в цикле или используя срезы или встроенные методы строк. Рекомендуется делать задания разными способами, можно одно и то же задание сделать несколькими способами.

Замечание 2. Основной алгоритм можно оформлять в виде функции.

В этом разделе используйте срезы строк.

1. Дана строка. Определить, является ли строка палиндромом.
2. Дана строка. В строке с помощью срезов поменяйте первую и вторую половину. Если в строке нечетное количество символов, то 1-я половина большая (например: '123ab' → 'ab123').
3. Дана строка. В строке с помощью срезов поменяйте 2-ю букву на букву 'A' и удалите последнюю букву (примеры: '12345' → '1A34', 'abababab' → 'aAababa').
4. Дана строка. С помощью срезов переверните ее и удалите первый символ (пример: '12345' → '5432').
5. Дана строка. В строке с помощью срезов выделите только четные символы (пример: 'абвгд' → 'авд').

В этом разделе используйте методы строк.

6. В начале и конце строки удалить символы: пробелы, табуляции \t, переноса \n.
7. В начале и конце строки удалить символы: ', . : ; ! ?'.
8. В строке удалить все пробелы и все буквенные символы привести к нижнему регистру.

9. В строке заменить все символы табуляции \t и переноса \n на пробелы.
10. В строке посчитать количество латинских букв “Aa”.
11. В строке найти позицию первого вхождения подстроки “cat”.

12. Продублировать в строке все символы “/”. Например, из строки “/temp/1” сделать строку “//temp//1”.

---

13. Найти в строке позицию первого справа латинского символа. Если латинского символа в строке нет, выдать сообщение об этом.
14. Удалить из строки все символы-цифры (‘0’…‘9’).
15. Определить, чего больше в строке: последовательности символов ‘dog’ в ее первой половине или последовательности символов ‘cat’ во второй? (Например, в строке ‘dogdogdogcat’ 2 – dog и 1 – cat). В ответе указать больше, меньше или равно.
16. Даны две строки s1 и s2. Найти позицию начала первого появления в строке s1 строки s2 или перевернутой строки s2. Например, для строк s1 = '03400430' и s2 = ‘43’ вернет позицию 1. Если вхождения нет, выдать сообщение об этом.
17. Дана строка, состоящая из малых латинских символов. Посчитать сколько всего различных символов в ней есть. Например, в строке ‘abcabab’ всего 3 разных символа ‘abc’).

В этом разделе вводится строка, состоящая из слов, разделенных пробелами.

Для разбиения на слова удобно использовать метод split. Например, для строки s = ' abc 123 AB' следующий код L = s.split( ) вернет список L = ['abc', '123', 'AB'].

18. Подсчитать количество слов в тексте.
19. Напечатать слова из текста, в которых есть большие буквы.
20. Напечатать слова из текста, в которых есть цифры.
21. Напечатать слова из текста, в которых есть только латинские символы.

Каждому студенты из общего списка задач предлагается сделать часть задач (индивидуальная задача из каждого раздела).

Критерии оценки

Для каждого задания указано максимальное количество баллов M **(**учебная карта дисциплины).

Студенту выставляется

M баллов, если задание выполнено полностью и не содержит ошибок,

$0.75 * M$ баллов, если задание выполнено, но допущены неточности, например, вычислительного характера,

$0.5 * M$ баллов, если имеются существенные ошибки, но общая схема решения правильна

$0.25 * M$ баллов, если решение было начато, но задание выполнено лишь частично, существенные шаги не сделаны,

0 баллов, если задание не выполнено.

Для каждого задания указывается срок его выполнения. По 20% от максимального количества баллов снимается за каждую неделю просрочки.

После суммирования всех набранных баллов, округление производится до ближайшего целого числа (например, 45.5 округляется до 46, 45.25 округляется до 45).

КОМПЛЕКТ ЛАБОРАТОРНЫХ РАБОТЫ

по дисциплине «Питон для анализа данных»

Раздел 3. Занятие 9. Библиотека Numeric Python (NumPy)

Для быстрой работы с большими данными используются массивы из пакета numpy (см. www.numpy.org).

Используются массивы (array) элементов одинакового типа:

Размер задается при создании, нельзя поменять. Можно менять значения. Можно создавать одномерные (вектора), двумерные (матрицы) и большей размерности массивы. Все элементы массива одного типа (например, все целые или вещественные). Быстрый доступ к элементам. Пакет содержит кроме самих массивов еще и методы для работы с ними. Быстрая обработка данных в массиве, особенно, если использовать методы пакетов numpy, scipy, pandas, и не использовать циклы и методы базового пакета Python.

Функции создания массивов

https://numpy.org/doc/stable/reference/routines.array-creation.html и
https://numpy.org/doc/stable/user/basics.creation.html#arrays-creation

Функция	Описание
array(object, dtype=None, …)	Создание массива из объекта object
arange([start,]stop, [step,] dtype=None)	Создание массива из диапазона
linspace(start, stop, num=50, …)	Создание массива num чисел, равномерно расположенных на отрезке [start, stop]
zeros(shape, dtype = float, order = 'C')	Возвращает новый массив заданной формы и типа (по умолчанию 'float64'), заполненный нулями
empty(shape, …)	новый массив (не заполнен)
ones(shape, …)	новый массив из 1
full(shape, fill_value, dtype = None, order = 'C')	новый массив, заполненный заданным значением
empty_like, full_like, ones_like, zeros_like	новый массив (размер и тип по образцу другого массива)

Пример. Создание массивов

```
# Подключаем пакет numpy:
import numpy as np
# Создание массива из списка:
A = np.array([1, 4, 2, 5, 3])
# Создание двумерного массива из списка:
np.array([[i]*3 for i in [2, 4, 6]])
# Результат: array([[2, 2, 2],[4, 4, 4],[6, 6, 6]])

# Создание массивов заполненных нулями:
np.zeros(5)  # числа по умолчанию вещественные
#   array([ 0.,  0.,  0.,  0.,  0.])
np.zeros((5,), dtype = int)
#   array([0, 0, 0, 0, 0])
np.empty((2, 1)) # без заполнения
#   array([[ 8.46612366e-83],[-1.84402114e-26]])
np.ones((2, 2)) # заполненного единицами
#   array([[ 1.,  1.],[ 1.,  1.]])

np.arange(0, 10, 2) # диапазон
#   array([0, 2, 4, 6, 8])
```

```
np.linspace(0, 1, 5) # 5 чисел на отрезке [0, 1]
array([0., 0.25, 0.5, 0.75, 1.])
# случайные числа:
np.random.random((2, 3)) # вещественные
# array([[0.61646814, 0.57419779, 0.1674393 ],
#        [0.92308608, 0.69372043, 0.24111806]])
np.random.randint(0, 10, (3, 3)) # целые
#  array([[9, 7, 4], [1, 7, 2], [1, 6, 6]])
```

Стандартные типы данных библиотеки NumPy

https://numpy.org/doc/stable/reference/arrays.scalars.html

Тип данных	Описание
bool_	Булев тип (True или False), 1 байт в памяти
int_	Целочисленное значение по умолчанию (аналогичен типу long языка C; обычно int64 или int32)
intc	Идентичен типу int языка C (обычно int32 или int64)
intp	Целочисленное значение, используемое для индексов (аналогично типу ssize_t языка C; обычно int32 или int64)
int8	Байтовый тип (от –128 до 127)
int16	Целое число (от –32 768 до 32 767)
int32	Целое число (от –2 147 483 648 до 2 147 483 647)
int64	Целое число (от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807)
uint8	Беззнаковое целое число (от 0 до 255)
uint16	Беззнаковое целое число (от 0 до 65 535)
uint32	Беззнаковое целое число (от 0 до 4 294 967 295)
uint64	Беззнаковое целое число (от 0 до 18 446 744 073 709 551 615)
float_	Сокращение для названия типа float64
float16	Число с плавающей точкой с половинной точностью: 1 бит знак, 5 бит порядок, 10 бит мантисса
float32	Число с плавающей точкой с одинарной точностью: 1 бит знак, 8 бит порядок, 23 бита мантисса
float64	Число с плавающей точкой с удвоенной точностью: 1 бит знак, 11 бит порядок, 52 бита мантисса
complex_	Сокращение для названия типа complex128
complex64	Комплексное число, представленное двумя 32-битными числами
complex128	Комплексное число, представленное двумя 64-битными числами

```
# Создаем массив целых чисел:
b = np.zeros(10, dtype = 'int32')
b.dtype   #                  dtype('int32')
c = np.zeros(10)
c.dtype  # По умолчанию тип float64
#           dtype('float64')
```

Атрибуты массивов библиотеки NumPy

```
# двумерный массив:
x2 = np.random.randint(10, size = (3, 4))
x2.ndim             #  2 – размерность
x2.shape            #  (3, 4) – размер каждого измерения
x2.size             #  12 – количество элементов
x2.itemsize         #  4 – байт на один элемент
x2.nbytes           #  48 – байт на весь массив = 12 * 4
```

Индексация массива: срезы

Можно использовать срезы. Одномерные массивы:

```
x1 = np.arange(10)
#   array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
x1[0]               # 0
x1[1:5]         # array([1, 2, 3, 4])
x1[-1]              # 9
x1[::-1]        # array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

Многомерные массивы:

```
x2 = np.arange(12)
x2 = x2.reshape((3, 4))
# array([[0, 1, 2, 3],[4, 5, 6, 7],[8, 9, 10, 11]])
x2[1, 3]        #  7  индексы задаются через x2[i, j]
#  В списках не так: L[i][j]
x2[:2, ::2]         # Строки 0, 1 и столбцы 0, 2:
         array([    [0, 2],
                        [4, 6]])
x2[:, 0]            # первый столбец массива x2
    array([0, 4, 8])
x2[0, :]            # первая строка массива x2
    array([0, 1, 2, 3])
```

Срезы массивов возвращают представления (views), а не копии (copies) данных массива. Этим срезы массивов библиотеки NumPy отличаются от срезов списков языка Python.

```
x3 = x2[:2, ::2]          # извлекаем подмассив из x2
#   array([[0, 2], [4, 6]])
x3[:, :] = 55  # заменить все элементы в x3 на 55
x2  # массив x2 тоже поменялся:
#  array([[55,  1, 55,  3],
          [55,  5, 55,  7],
          [ 8,  9, 10, 11]])
```

Срез x3 продолжает ссылаться на x2 (не создается новый массив). Представление – это работа с теми же данными, но новые индексы.

Создание копий массивов

```
# Если надо создать копию, вызываем метод copy:
x2_copy = x2[:2, ::2].copy()
```

Изменение формы массивов

```
x = np.arange(1, 10).reshape((3, 3))
#   array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# Преобразование в столбец:
y = x.reshape((9, 1))
#  y = array([[1],[2],[3],[4],[5],[6],[7],[8],[9]])
```

Добавление новой размерности

```
x = np.arange(1, 10)
#  array([1, 2, 3, 4, 5, 6, 7, 8, 9])
# Преобразование из строки в столбец:
# Из массива делаем матрицу размера (n, 1):
z = x[:, np.newaxis]
# и старые данные записываются в разные строки:
#  z = array([[1],[2],[3],[4],[5],[6],[7],[8],[9]])
```

Это тоже представления, т.е. массивы продолжают ссылаться на одни и те же данные! Изменяя данные в y или z, меняем и x.

Слияние и разбиение массивов

Объединение: concatenate((a1, a2, ...), axis=0, out=None, dtype=None)

```
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6]])
# Вдоль первой оси (axis = 0):
np.concatenate((a, b), axis = 0)
#       array([[1, 2], [3, 4], [5, 6]])
# Вдоль второй оси (axis = 1):
# b.T - транспонирование (переворот):
np.concatenate((a, b.T), axis = 1)
#       array([[1, 2, 5], [3, 4, 6]])
# Если axis = None, массивы преобраз. в одномерные:
np.concatenate((a, b), axis = None)
#       array([1, 2, 3, 4, 5, 6])

a = np.array([1, 2, 3])
b = np.array([2, 3, 4])
# np.vstack (вертикальное объединение):
np.vstack((a, b))
#       array([[1, 2, 3], [2, 3, 4]])
# np.hstack (горизонтальное объединение):
np.hstack((a, b))
#       array([1, 2, 3, 2, 3, 4])
```

Разбиение массивов выполняется с помощью функций: np.split,

np.hsplit, np.vsplit.

```
a = np.array([10, 20, 30, 40, 50])
# Разбить по 2-му элементу:
np.split(a, [2])
#      [array([10, 20]), array([30, 40, 50])]
```

Выполнение вычислений над массивами библиотеки
NumPy: универсальные функции

https://numpy.org/doc/stable/reference/ufuncs.html#ufuncs

С массивами пакета numpy циклы работают медленно. Лучше использовать векторизованные операции и функции из numpy.

Операции: +, –, *, /, //, %, **.

Операция	Функция	Описание
+	np.add	Сложение
–	np.subtract	Вычитание
–	np.negative	Унарная операция изменения знака
*	np.multiply	Умножение
/	np.divide	Деление
//	np.floor_divide	Деление с остатком
**	np.power	Возведение в степень
%	np.mod	Остаток

Примеры.

```
# Определяем два массива:
x = np.array([1, 2, 3])
y = np.array([2, 4, 8])
# Операции (поэлементно с одинаковыми массивами):
# Сумма через операцию или функцию:
x + y                     # array([ 3, 6, 11])
np.add(x, y)        # array([ 3, 6, 11])
x - y                           # array([-1, -2, -5])
-x                              # array([-1, -2, -3])
x * y                           # array([ 2, 8, 24])
x / y                           # array([0.5 , 0.5 , 0.375])
x // y                          # array([0, 0, 0], dtype=int32)
x % y                           # array([1, 2, 3], dtype=int32)
# Операции (поэлементно массив и число):
x + 5                           # array([6, 7, 8])
x ** 2                          # array([1, 4, 9], dtype=int32)
```

Функции: absolute (или abs), exp, log, sqrt, тригонометрические (sin, cos, tan, arcsin, arcos, arctan) и др.

```
# Определяем массив:
x = np.array([1, 2, 4])
# Вычисляем функции:
np.sqrt(x)          # array([1., 1.41421356, 2.])
np.sin(x)           # array([ 0.84147098, 0.90929743,
                        -0.7568025 ])
```

Линейная алгебра

Операции линейной алгебры – умножение и разложение матриц, вычисление определителей и другие – важная часть любой библиотеки для работы с массивами. В отличие от некоторых пакетов, например MATLAB, в NumPy применение оператора * к двум двумерным массивам вычисляет поэлементное, а не матричное произведение. А для перемножения матриц имеется функция dot:

```
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, -6], [-1, 7]])
# Умножение матриц:
z = np.dot(x, y)
```

```
# или   z = x.dot(y)
#       z = [[ 3  8], [11 10]]
```

$$z = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 5 & -6 \\ -1 & 7 \end{pmatrix} = \begin{pmatrix} 3 & 8 \\ 11 & 10 \end{pmatrix}.$$

В модуле numpy.linalg имеется стандартный набор алгоритмов, в частности, разложение матриц, нахождение обратной матрицы и вычисление определителя.

```
A = np.array([[1, 2], [3, 0]])
# Обратная матрица к A:
B = np.linalg.inv(A)
# [[ 0.  0.33333333] [ 0.5 -0.16666667]]
# Произведение матриц A * A^-1 = E:
np.dot(A, B)    #  [[1. 0.] [0. 1.]]
```

Некоторые функции из модуля numpy.linalg

numpy.org/doc/stable/reference/routines.linalg.html?highlight=linalg#module-numpy.linalg

Функция	Описание
trace	Вычисляет след матрицы – сумму диагональных элементов
det	Вычисляет определитель матрицы
eig	Вычисляет собственные значения и собственные векторы квадратной матрицы
eigvals	Вычисляет собственные значения квадратной матрицы
inv	Вычисляет обратную матрицу
norm	Вычисляет норму матрицы или вектора
qr	Вычисляет QR-разложение
svd	Вычисляет сингулярное разложение (SVD)
solve	Решает линейную систему Ax = b, где A – квадратная матрица
lstsq	Вычисляет решение уравнения y = Xb по методу наименьших квадратов

Сводные показатели (reduce, accumulate)

Операция reduce применяет операцию к элементам массива до тех пор, пока не останется только один результат:

```
x = np.array([1, 2, 4])
np.add.reduce(x)          # 7  = 1 + 2 + 4
```

Функция accumulate сохраняет все промежуточные результаты:

```
np.multiply.accumulate(x)
#         array([1, 2, 8], dtype = int32)
```

Внешнее произведение двух векторов

```
x = np.array([1, 2, 4])
y = np.array([0, 3, 7])
# Произведение outher (каждый с каждым):
np.multiply.outer(x, y)
array([[ 0,  3,  7],
       [ 0,  6, 14],
       [ 0, 12, 28]])
# Сумма внешняя outher (каждый с каждым):
np.add.outer(x, y)
array([[ 1,  4,  8],
       [ 2,  5,  9],
       [ 4,  7, 11]])
```

Агрегирование: минимум, максимум и др.

https://numpy.org/doc/stable/reference/routines.statistics.html#averages-and-variances

Функция	NaN-безопасная версия	Описание
np.sum	np.nansum	Вычисляет сумму элементов
np.prod	np.nanprod	Вычисляет произведение элементов
np.mean	np.nanmean	Вычисляет среднее значение элементов
np.std	np.nanstd	Вычисляет стандартное отклонение
np.var	np.nanvar	Вычисляет дисперсию
np.min	np.nanmin	Вычисляет минимальное значение
np.max	np.nanmax	Вычисляет максимальное значение
np.argmin	np.nanargmin	Индекс минимального значения
np.argmax	np.nanargmax	Индекс максимального значения
np.median	np.nanmedian	Вычисляет медиану элементов
np.percentile	np.nanpercentile	Вычисляет квантили элементов
np.any		Существуют ли значения true
np.all		Все ли элементы имеют значение true

Причем можно вычислять для всего многомерного массива и по отдельным осям.

```
a = np.array([[1, 2], [3, 4]])
np.min(a)                                   # 1 - во всем массиве
np.min(a, axis = 0)       # array([1, 2])
np.min(a, axis = 1)       # array([1, 3])

np.mean(a)                # 2.5
np.std(a)                 # 1.118033988749895
np.var(a)                 # 1.25

b = np.array([[1, 2], [3, np.nan]])
np.nanmin(b)         # 1.0
```

Транслирование

Транслирование представляет собой набор правил по применению бинарных универсальных функций (сложение, вычитание, умножение и т. д.) к массивам различного размера.

```
a = np.array([0, 1, 2])
b = np.array([5, 5, 5])
# массивы одинакового размера - поэлементно:
a + b    # [5 6 7]
# массивы и скаляр - к каждому элементу:
a + 5    # [5 6 7]
# массивы разной размерности:
M = np.ones((3, 3))
#  [[1. 1. 1.] [1. 1. 1.] [1. 1. 1.]]
print(a + M)          # (1, 3) и (3, 3) → (3, 3):
#  [[1. 2. 3.] [1. 2. 3.] [1. 2. 3.]]
# массивы разной размерности:

c = np.array([0, 10, 20]).reshape(3,1)
```

```
#  c = [[ 0] [10] [20]] # вектор-стобец
print(a + c)  # (1, 3) и (3, 3):
# [[ 0  1  2]
#  [10 11 12]
#  [20 21 22]]
```

Сравнения, маски и булева логика

https://numpy.org/doc/stable/reference/routines.logic.html

Оператор	Эквивалентная универсальная функция
==	np.equal
!=	np.not_equal
<	np.less
<=	np.less_equal
>	np.greater
>=	np.greater_equal

Пример.

```
np.random.seed(7)
x = np.random.randint(-100, 101, 5)
# x = [ 75  96 -75 -33  51]
x > 0    #     [ True  True False False  True]
x < 0    # [False False  True  True False]
abs(x) == 75  # [ True False  True False False]
abs(x) != 75  # [False  True False  True  True]
```

Работа с булевыми массивами

Созданные булевы массивы можно использовать.

Пример.

```
# количество ненулевых элементов (= True):
np.count_nonzero(x > 0)   #  3
# Сумма элементов (True = 1, False = 0):
np.sum(x > 0)                              #  3
# Есть ли элементы > 0:
np.any(x > 0)                              #  True
# Все ли элементы > 0:
np.all(x > 0)                              #  False
```

Можно объединять несколько условий с помощью битовых и булевых операций.

Оператор	Эквивалентная универсальная функция
Битовые	
&	np.bitwise_and
|	np.bitwise_or
^	np.bitwise_xor
~	np.bitwise_not

Логические	
	logical_and, logical_or, logical_not, logical_xor

Для логических значений (True, False) они работают одинаково.

Пример.

```
np.random.seed(7)
x = np.random.randint(-100, 101, 5)
# x = [ 75 96 -75 -33 51]
y = np.random.randint(-100, 101, 5)
# [ 3 -8 85 42 -77]
# Все 3 оператора вернут одинаковый результат:
np.logical_and(x > 0, y > 0)
np.bitwise_and(x > 0, y > 0)
(x > 0) & (y > 0)
# array([ True, False, False, False, False])
```

Также булевы массивы можно использовать как индексы:

```
np.random.seed(0)
x = np.random.randint(-10, 11, size = (3, 3))
# x = [[  2   5 -10]
       [ -7  -7  -3]
       [ -1   9   8]]
x > 0
# [[ True True  False]
   [False False False]
   [False True  True]]
# Выбрать только > 0 числа из массива x:
x[x > 0]
# Возвращается одномерный массив:
#  [2 5 9 8]
```

Операции and, or, not не работают с массивами numpy. Для чисел битовые (bitwise) и логические (logical) операции отличаются

Функция where, выбор по условию

https://numpy.org/doc/stable/reference/routines.sort.html#searching

where(condition[, x, y]) – возвращает элемент массива x или y в зависимости от условия.

```
a = np.arange(10) - 5
# a = [-5 -4 -3 -2 -1 0 1 2 3 4]
# В зависимости от условия выбираем -a или a:
b = np.where(a < 0, -a, a)
# b = [5 4 3 2 1 0 1 2 3 4]
```

nonzero(a) – возвращает индексы ненулевых элементов.

```
x = np.array([[3, 0, 0], [0, 4, 0], [5, 6, 0]])
np.nonzero(x)
# (array([0, 1, 2, 2], dtype = int64),
   array([0, 1, 0, 1], dtype = int64))
```

«Прихотливая» индексация (fancy indexing)

Похожа на уже рассмотренную простую индексацию, но вместо скалярных значений передаются массивы индексов.

```
np.random.seed(0)
x = np.random.randint(0, 50, size = 10)
# x = [44 47 0 3 3 39 9 19 21 36]
ind = [1, 5, 3]
x[ind]
#  [47 39 3]
```

Форма результата отражает форму массивов индексов, а не форму индексируемого массива:

```
ind = np.array([[1, 5], [3, 9]])
x[ind]
# [[47 39] [ 3 36]]
```

Работает и в случае многомерных массивов.

```
X = np.arange(12).reshape((3, 4))
# X = [[ 0 1 2 3]
       [ 4 5 6 7]
       [ 8 9 10 11]]
# Определяем индексы:
row = np.array([0, 1, 2])
col = np.array([2, 1, 3])
X[row, col]          # Выбирает X[0, 2] X[1, 1] X[2, 3]:
#  [ 2 5 11]
```

Можно смешивать разные индексы между собой.

Сортировка массивов

https://numpy.org/doc/stable/reference/routines.sort.html

В пакете есть свои более быстрые методы сортировки массивов.

1) Функция возвращает отсортированную копию массива:

numpy.sort(a, axis = –1, kind = None, order = None)

2) Метод массива сортирует массив на месте:

ndarray.sort(axis = –1, kind = None, order = None)

Параметры:

axis – выбор оси (по умолчанию последняя);

kind – метод (‘quicksort’, ‘mergesort’, ‘heapsort’, ‘stable’);

order – можно по нескольким полям.

```
x = np.array([2, 1, 4, 3, 5])
y = np.sort(x)  # создаем новый массив
#  y =  [1 2 3 4 5]
# или
x.sort( )           # сортируем x
#  x =  [1 2 3 4 5]
```

Пример. По разным осям

```
X = np.random.randint(0, 10, (3, 3))
# X = [[5 0 3]
       [3 7 9]
       [3 5 2]]
```

```
print(np.sort(X))
# или print(np.sort(X, axis = 1))
#     [[0 3 5]
       [3 7 9]
       [2 3 5]]
print(np.sort(X, axis = 0))
#     [[3 0 2]
       [3 5 3]
       [5 7 9]]
# Сортирует все элементы в списке:
print(np.sort(X, axis = None))
#   [0 2 3 3 3 5 5 7 9]
```

Сохранение массива в файл

https://numpy.org/doc/stable/reference/routines.io.html

Сохранить в файл массив можно с помощью метода save. Создает файл со специальной структурой, в которой хранится размерность массива, тип и сами данные.

```
import numpy as np
arr = np.random.randint(-5, 6, size=(2, 4))
# Сохраняем в файл:
np.save('some_array.npy', arr)
```

Загрузить из файла такой массив обратно в программу можно с помощью метода load:

```
# Загружаем из файла и создаем массив a:
a = np.load('some_array.npy')
print(a)
#     [[-3 -2 2 2]
       [ 4 -4 1 3]]
```

Задания

Создание одномерных массивов (векторов) и двумерных массивов (матриц)

1. Создайте 2 вектора с помощью функции array.
2. Создайте 2 вектора с помощью функции arrange.
3. Создайте 2 вектора с помощью функции linspace.
4. Создайте 2 вектора с помощью функции random.
5. Создайте 2 вектора с помощью функции randint.

Атрибуты массивов, операции с векторами

6. Выдайте на экран информацию о созданном выше массиве: тип (dtype), размерность (ndim, shape), количество элементов (size), itemsize, nbytes.
7. С помощью срезов выделите несколько элементов в массиве и поменяйте их значения.
8. Найти результаты арифметических операций для 2 массивов (или числа и массива). Используйте операции или функции numpy (т.е., например, “+” или “np.add”). Вычислите еще любые две функции от массивов.

Расстояние, норма и скалярное произведение
(определения см. в лекции по алгебре)

9. Вычислите расстояние ρ_1 между двумя векторами.
10. Вычислите расстояние ρ_2 между двумя векторами.
11. Вычислите расстояние ρ_∞ между двумя векторами.
12. Вычислите норму $\|a\|_1$ вектора.
13. Вычислите норму $\|a\|_2$ вектора.

14. Вычислите норму $\|a\|_\infty$ вектора.
15. Вычислите скалярное произведение двух векторов.

Определение матрицы и операции с матрицами

16. Создайте 2 матрицы одинакового размера. Найти результаты арифметических операций для 2 матриц. Кроме поэлементного умножения “*”, вычислите еще произведение матриц, определяемое в алгебре, с помощью функции “np.dot”.
17. Найдите определитель квадратной матрицы.
18. Найдите собственные значения квадратной матрицы.
19. Найдите собственные значения и собственные вектора квадратной матрицы.
20. Найдите обратную матрицу к квадратной матрице.
21. Найдите норму квадратной матрицы.

Агрегирование

Определите произвольный массив. Найдите для него следующие величины (можно по одной из осей или для всего массива):

22. Сумму, среднее и минимальное значения.
23. Произведение, стандартное отклонение и индекс минимального значения.
24. Среднее значение, дисперсию и медиану.

Сравнения, маски и булева логика

Определите произвольный массив.

25. Найдите количество элементов из промежутка [–5, 5].
26. Определите, есть ли среди его элементов числа большие 100.
27. Определите, все ли они положительные.
28. Выбрать из массива все элементы из промежутка [0, 10].
    Заменить в массиве все отрицательные элементы на 0.

Каждому студенты из общего списка задач предлагается сделать часть задач (индивидуальная задача из каждого раздела).

Критерии оценки

Для каждого задания указано максимальное количество баллов M **(**учебная карта дисциплины).

Студенту выставляется

M баллов, если задание выполнено полностью и не содержит ошибок,

$0.75 * M$ баллов, если задание выполнено, но допущены неточности, например, вычислительного характера,

$0.5 * M$ баллов, если имеются существенные ошибки, но общая схема решения правильна

$0.25 * M$ баллов, если решение было начато, но задание выполнено лишь частично, существенные шаги не сделаны,

0 баллов, если задание не выполнено.

Для каждого задания указывается срок его выполнения. По 20% от максимального количества баллов снимается за каждую неделю просрочки.

После суммирования всех набранных баллов, округление производится до ближайшего целого числа (например, 45.5 округляется до 46, 45.25 округляется до 45).