МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«ТЮМЕНСКИЙ ИНДУСТРИАЛЬНЫЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Председатель КСН
_____ Ю.В. Ваганов
«30» 08 2021г.

РАБОЧАЯ ПРОГРАММА

дисциплины/модуля: Математика и Python для анализа данных
направление подготовки: _21.03.01 Нефтегазовое дело_____
направленность (профиль):
_____Эксплуатация и обслуживание объектов добычи нефти
форма обучения: ____очная, очно-заочная_____

Рабочая программа разработана в соответствии с утвержденным учебным планом от 30.08.2021 г. и требованиями ОПОП 21.03.01 Нефтегазовое дело к результатам освоения дисциплины/модуля.

Рабочая программа рассмотрена
на заседании кафедры естественно-научных и гуманитарных дисциплин
Протокол № 1 от 30.08.2021г.

Заведующий кафедрой ЕНГД _____ Л.К. Иляшенко

СОГЛАСОВАНО:
Заведующий выпускающей кафедрой НД _____ Р.Д. Татлыев

30.08.2021г.

Рабочую программу разработал:

Л.К. Иляшенко, зав. кафедрой ЕНГД, к.п.н., доцент _____
(И.О. Фамилия, должность, ученая степень, ученое звание) *(подпись)*

1. Цели и задачи освоения дисциплины/модуля

Цель дисциплины/модуля заключается в овладении студентами основами работы с языком Python в анализе данных, расширении теоретической и практической подготовки в области математического анализа, линейной алгебры, методов оптимизации, теории вероятностей.

Задачи дисциплины/модуля:

- овладение особенностями языка Python для анализа данных, принципами чтения различных данных;
- изучение Python-библиотек, содержащих большое количество инструментов: от быстрых операций с многомерными массивами до визуализации и реализации различных математических методов, в том числе линейной алгебры как основного математического аппарата для работы с данными;
- изучение методов оптимизации как наилучшего инструмента для определения оптимальных параметров системы;
- знакомство с матричными разложениями, которые используются при построении регрессионных моделей, для уменьшения размерности данных, в рекомендательных системах и в анализе текстов;
- расширение знаний о базовых концепциях теории вероятностей и статистики, которые необходимы для понимания механизма работы практически всех методов анализа данных.

2. Место дисциплины/ модуля в структуре ОПОП ВО

Дисциплина относится к элективным дисциплинам (модулям) 1 (ДВ.1) «Digital & IT. Машинное обучение и анализ данных» части учебного плана.

Необходимыми условиями для освоения дисциплины/модуля являются:

– знание основных матричной алгебры, математического анализа, теории вероятностей и математической статистики;
– понимание основных принципов алгоритмизации и программирования;
– знание основ языка программирования Python;
– владение навыками использования компьютерных технологий и средств обработки информации.

Содержание дисциплины является логическим продолжением дисциплины «Математика» базовым для изучения следующих дисциплин модуля «Digital & IT. Машинное обучение и анализ данных»: «Машинное обучение и вопросы искусственного интеллекта», «Нейронные сети», «Прикладные задачи анализа данных».

3. Результаты обучения по дисциплине/модулю

Процесс изучения дисциплины/модуля направлен на формирование следующих компетенций:

Таблица 3.1

Код и наименование компетенции	Код и наименование индикатора достижения компетенции (ИДК)	Код и наименование результата обучения по дисциплине (модулю)
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1 Осуществляет выбор актуальных российских и зарубежных источников, а также поиск, сбор и обработку информации, необходимой для решения поставленной задачи	Знать: актуальные российские и зарубежные источники по дисциплине (УК-1.З1)
		Уметь: осуществлять выбор актуальных российских и зарубежных источников, а также поиск, сбор и обработку информации, необходимой для решения поставленной задачи (УК-1.У1)

		Владеть: навыками поиска, сбора и обработки информации, необходимой для решения поставленной задачи (УК-1.В1)
	УК-1.2 Систематизирует и критически анализирует информацию, полученную из разных источников, в соответствии с требованиями и условиями задачи	Знать: основные принципы, требования и правила систематизации и классификации информации, полученной из разных источников, а так же порядка ее анализа согласно выданного технического задания (УК-1.З2)
		Уметь: реализовывать основные требования и правила систематизации и анализа статистической информации, полученной из разных источников в соответствии с требованиями и условиями поставленной задачи (УК-1.У2)
		Владеть: принципами, требованиями, инструментами систематизации, классификации, анализа информации (УК-1.В2)
УК-2 Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2.1 Проводит анализ поставленной цели и формулирует совокупность взаимосвязанных задач, которые необходимо решить для ее достижения	Знать: цель и совокупность взаимосвязанных задач, которые необходимо решить для ее достижения (УК-2.З1)
		Уметь: проводить анализ поставленной цели и формировать совокупность взаимосвязанных задач, которые необходимо решить для ее достижения (УК-2.У1)
		Владеть: навыком постановки проанализированной цели и формирования совокупности взаимосвязанных задач, которые необходимо решить для ее достижения (УК-2.В1)
	УК-2.2 Выбирает оптимальный способ решения задач, исходя из имеющихся ресурсов и ограничений	Знать: оптимальный способ решения задач, исходя из имеющихся ресурсов и ограничений (УК-2.З2)
		Уметь: решать задачи, выбирая оптимальный способ вычисления, исходя из имеющихся ресурсов и ограничений (УК-2.У2)
		Владеть: навыком решения задач, выбирая оптимальный способ вычисления, исходя из имеющихся ресурсов и ограничений (УК-2.В2)
ПКС-1 Способность осуществлять и корректировать технологические процессы нефтегазового производства в соответствии с выбранной	ПКС-1.1 Осуществляет выбор и систематизацию информации о технологических процессах нефтегазового производства	Знать: способы сбора и анализа исходных данных о технологических процессах нефтегазового производства (ПКС-1.З1)
		Уметь: осуществлять выбор и систематизацию информации о технологических процессах нефтегазового производства (ПКС-1.У1)

сферой профессиональной деятельности		Владеть: навыками выбора и систематизации информации о технологических процессах нефтегазового производства(ПКС-1.В1)

4. Объем дисциплины/модуля

Общий объем дисциплины/модуля составляет 3 зачетные единицы, 108 часов.

Таблица 4.1.

Форма обучения	Курс/ семестр	Аудиторные занятия/контактная работа, час.			Самостоятельная работа, час.	Форма промежуточной аттестации
		Лекции	Практические занятия	Лабораторные занятия		
Очная	3/5	18	34	-	56	Зачет
Очно-заочная	3/5	12	24	-	72	Зачет

5. Структура и содержание дисциплины/модуля

5.1. Структура дисциплины/модуля

очная форма обучения (ОФО)

Таблица 5.1.1

№ п/п	Структура дисциплины/модуля		Аудиторные занятия, час.			СРС, час.	Всего, час.	Код ИДК	Оценочные средства
	Номер раздела	Наименование раздела	Л.	Пр.	Лаб.				
1	1	Введение. Знакомство с Python	4	24	-	12	40	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Выполнение и защита практической работы, коллоквиум
2	2	Библиотеки Python и линейная алгебра	6	4	-	16	26	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Выполнение и защита практической работы, коллоквиум
3	3	Оптимизация и матричные разложения	4	6	-	14	24	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Выполнение и защита практической работы, коллоквиум
4	4	Случайность	4	0	-	14	18	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Коллоквиум
5	Зачет		-	-	-	-	-	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Вопросы к зачету
Итого:			**18**	**34**	**-**	**56**	**108**		

очно-заочная форма обучения (ОЗФО)

Таблица 5.1.2

№ п/п	Структура дисциплины/модуля		Аудиторные занятия, час.			СРС, час.	Всего, час.	Код ИДК	Оценочные средства
	Номер раздела	Наименование раздела	Л.	Пр.	Лаб.				
1	1	Введение. Знакомство с Python	2	16	-	16	34	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Выполнение и защита практической работы, коллоквиум
2	2	Библиотеки Python и линейная алгебра	4	4	-	20	28	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Выполнение и защита практической работы, коллоквиум
3	3	Оптимизация и матричные разложения	4	4	-	18	26	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Выполнение и защита практической работы, коллоквиум
4	4	Случайность	2	0	-	18	20	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Коллоквиум
5	Зачет		-	-	-	-	-	УК-1.1 УК-1.2 УК-2.1 УК-2.2 ПКС-1.1	Вопросы к зачету
Итого:			**12**	**24**	**-**	**72**	**108**		

5.2. Содержание дисциплины/модуля.

5.2.1. Содержание разделов дисциплины/модуля (дидактические единицы).

Раздел 1. Введение. Знакомство с Python

Python — один из главных инструментов специалиста в науке о данных. Циклы, функции, генераторы, list comprehension. Функции и их свойства. Предел и производная. Геометрический смысл производной.

Раздел 2. Библиотеки Python и линейная алгебра

Pandas. Data Frame. NumPy, SciPy и Matplotlib. Решение оптимизационных задач в SciPy. Системы линейных уравнений. Матричные операции. Ранг и определитель

Раздел 3. Оптимизация и матричные разложения

Частные производные и градиент. Касательная плоскость и линейное приближение. Оптимизация негладких функций. Метод имитации отжига. Генетические алгоритмы и дифференциальная эволюция. Нелдер-Мид. Приближение матрицей меньшего ранга.

Раздел 4. Случайность

Случайность в теории вероятностей и статистике. Свойства вероятности. Условная вероятность. Оценка распределения по выборке. Важные характеристики распределений. Центральная предельная теорема. Доверительные интервалы.

Лекционные занятия

Таблица 5.2.1

№ п/п	Номер раздела дисциплины	Объем, час.		Тема лекции
		ОФО	ОЗФО	
1	1	4	2	Введение. Знакомство с Python
2	2	6	4	Библиотеки Python и линейная алгебра
3	2	4	4	Оптимизация и матричные разложения
4	3	4	2	Случайность
Итого:		**18**	**12**	

Практические занятия

Таблица 5.2.2

№ п/п	Номер раздела дисциплины	Объем, час.		Тема практического занятия
		ОФО	ОЗФО	
1	1	4	2	Введение в язык программирования Python
2	1	4	2	Структура ветвление в Python
3	1	4	2	Работа с циклами в Python
4	1	4	2	Работа со строками в Python
5	1	4	4	Работа со списками. Операции над списками в Python
6	1	4	4	Функции и процедуры в Python
7	2	4	4	Математические операции в Python
8	3	6	4	Работа с двумерными массивами
Итого:		**34**	**24**	

Лабораторные работы

Лабораторные работы учебным планом не предусмотрены.

Самостоятельная работа студента

Таблица 5.2.3

№ п/п	Номер раздела дисциплины	Объем, час.		Тема раздела	Вид СРС
		ОФО	ОЗФО		
1	1	12	16	Введение. Знакомство с Python	Изучение теоретического материала, подготовка к практическим занятиям, коллоквиуму
2	2	16	20	Библиотеки Python и линейная алгебра	Изучение теоретического материала, подготовка к практическим занятиям, коллоквиуму
3	3	14	18	Оптимизация и матричные разложения	Изучение теоретического материала, подготовка к практическим занятиям, коллоквиуму
4	4	14	18	Случайность	Изучение теоретического материала, подготовка к коллоквиуму
8	-	-	-	Зачет	Подготовка к зачету
Итого:		**56**	**72**		

5.2.3. Преподавание дисциплины/модуля ведется с применением следующих видов образовательных технологий: лекция-диалог (лекционные занятия); лекции-визуализации в PowerPoint в диалоговом режиме (в случае интерактивного метода обучения); работа в малых

группах, разбор практических ситуаций (лабораторные занятия), кейс-метод (разбор конкретных ситуаций).

6. Тематика курсовых работ/проектов

Учебным планом выполнение курсовых работ не предусмотрено.

7. Контрольные работы

Учебным планом выполнение контрольных работ не предусмотрено.

8. Оценка результатов освоения дисциплины/модуля

8.1. Критерии оценивания степени полноты и качества освоения компетенций в соответствии с планируемыми результатами обучения приведены в Приложении 1.

8.2. Рейтинговая система оценивания степени полноты и качества освоения компетенций обучающихся очной формы обучения представлена в таблице 8.1.

Таблица 8.1

№ п/п	Виды мероприятий в рамках текущего контроля	Количество баллов
1 текущая аттестация		
1	Выполнение практических работ по текущим темам дисциплины: Введение в язык программирования Python. Структура ветвление в Python. Работа с циклами в Python	0-20
2	Коллоквиум	0-10
	ИТОГО за первую текущую аттестацию	**0-30**
2 текущая аттестация		
3	Выполнение практических работ по текущим темам дисциплины: Работа со строками в Python. Работа со списками. Операции над списками в Python. Функции и процедуры в Python	0-20
4	Коллоквиум	0-10
	ИТОГО за вторую текущую аттестацию	**0-30**
3 текущая аттестация		
5	Выполнение практических работ по текущим темам дисциплины: Математические операции в Python. Работа с двумерными массивами	0-20
6	Коллоквиум	0-10
7	Тестирование	0-10
	ИТОГО за третью текущую аттестацию	**0-40**
	ВСЕГО	**0-100**

8.3. Рейтинговая система оценивания степени полноты и качества освоения компетенций обучающихся очно-заочной формы обучения представлена в таблице 8.2.

Таблица 8.2

№ п/п	Виды мероприятий в рамках текущего контроля	Количество баллов
1 текущая аттестация		
1	Выполнение практических работ по текущим темам дисциплины: Введение в язык программирования Python. Структура ветвление в Python. Работа с циклами в Python	0-20
2	Коллоквиум	0-10
	ИТОГО за первую текущую аттестацию	**0-30**
2 текущая аттестация		

3	Выполнение практических работ по текущим темам дисциплины: Работа со строками в Python. Работа со списками. Операции над списками в Python. Функции и процедуры в Python	0-20
4	Коллоквиум	0-10
	ИТОГО за вторую текущую аттестацию	**0-30**
3 текущая аттестация		
5	Выполнение практических работ по текущим темам дисциплины: Математические операции в Python. Работа с двумерными массивами	0-20
6	Коллоквиум	0-10
7	Тестирование	0-10
	ИТОГО за третью текущую аттестацию	**0-40**
	ВСЕГО	**0-100**

9. **Учебно-методическое и информационное обеспечение дисциплины/модуля**

9.1. Перечень рекомендуемой литературы представлен в Приложении 2.

9.2. Современные профессиональные базы данных и информационные справочные системы.

2021/ 2022	Электронный каталог/Электронная библиотека Тюменского индустриального университета http://webirbis.tsogu.ru/	
	Договор №09-16/19 от 18.10.2019 взаимного оказания услуг двухстороннего доступа к ресурсам научно-технической библиотеки ФГАОУ ВО РГУ Нефти и газа (НИУ) им. И.М. Губкина и ФГБОУ ВО «ТИУ» http://elib.gubkin.ru/	С 18.10.2019 по 16.10.2021
	Договор №09-11/21 от 14.10.2021 взаимного оказания услуг двухстороннего доступа к ресурсам научно-технической библиотеки ФГАОУ ВО РГУ Нефти и газа (НИУ) им. И.М. Губкина и ФГБОУ ВО «ТИУ» http://elib.gubkin.ru/	С 14.10.2021 по 13.10.2022
	Договор № Б124/2019/09-20/2019 от 20.12.2019 на оказание услуг по предоставлению двустороннего доступа к ресурсам научно-технической библиотеки ФГБОУ ВО «УГНТУ» и ФГБОУ ВО «Тюменский индустриальный университет» http://bibl.rusoil.net	С 20.12.2019 по 18.12.2021
	Договор № 09-19/2019 от 12.12.2019 на оказание услуг двустороннего доступа к ресурсам научно-технической библиотеки ФГБОУ ВО «УГТУ» и ФГБОУ ВО «Тюменский индустриальный университет» http://lib.ugtu.net/books	С 12.12.2019 по 10.12.2021
	Договор №6631 – 20 от 29.12.2020 на оказание услуг по предоставлению доступа к ресурсам базы данных «Научная электронная библиотека eLIBRARY.RU» (эл.подписи)	с 01.01.2021 по 31.12.2021
	Гражданско-правовой договор №8232 от 18.08.2021 на оказание услуг по предоставлению доступа к электронным экземплярам произведений научного, учебного характера между ФГБОУ ВО «Тюменский индустриальный университет» и ООО «ЭБС ЛАНЬ» www.e.lanbook.ru	с 01.09.2021 по 31.08.2022
	Гражданско-правовой договор №7506 от 20.08.2021 на оказание услуг по предоставлению доступа к ЭБС между ФГБОУ ВО «Тюменский индустриальный университет» и ООО «Издательство ЛАНЬ» www.e.lanbook.com	с 01.09.2021 по 31.08.2022
	Гражданско-правовой договор №7508 от 23.08.2021 с ООО «Электронное издательство ЮРАЙТ» на оказание услуг по предоставлению доступа к образовательной платформе между ФГБОУ ВО «Тюменский индустриальный университет» и ООО «Электронное издательство ЮРАЙТ» www.urait.ru	с 01.09.2021 по 31.08.2022

	Гражданско-правовой договор № 7503 от 17.08.2021 на предоставление доступа к базе данных Консультант студента «Электронная библиотека технического ВУЗа» между ФГБОУ ВО «Тюменский индустриальный университет» и ООО «Политехресурс» http://www.studentlibrary.ru	с 01.09.2021 по 31.08.2022
	Гражданско-правовой договор №7507 от 26.08.2021 ООО «КноРус медиа» на оказание услуг по предоставлению доступа к электронно-библиотечной системе BOOK.ru https://www.book.ru	С 01.09.2021 по 31.08.2022
	Договор №7505 от 16.08.2021 на предоставление доступа к электронно-библиотечной системе «IPRbooks» между ФГБОУ ВО «Тюменский индустриальный университет» и ООО Компанией «Ай Пи Ар Медиа» http://www.iprbookshop.ru/	С 01.09.2021 по 31.08.2022
	Договор №101НЭБ/6258/09/17/2019 о подключении к Национальной электронной библиотеке и предоставлении доступа к объектам Национальной электронной библиотеки (через терминалы доступа)	С 29.10.2019 по 28.10.2024

9.3. Лицензионное и свободно распространяемое программное обеспечение, в т.ч. отечественного производства (Adobe Acrobat Reader), в т.ч. Microsoft Windows Microsoft Office Professional Plus (Договор №6714-20 от 31.08.2020 до 31.08.2021, Договор №7810 от 14.09.2021 до 13.09.2022), Microsoft Windows (Договор №6714-20 от 31.08.2020 до 31.08.2021, Договор №7810 от 14.09.2021 до 13.09.2022).

10. Материально-техническое обеспечение дисциплины/модуля

Помещения для проведения всех видов работы, предусмотренных учебным планом, укомплектованы необходимым оборудованием и техническими средствами обучения.

Таблица 10.1

Перечень оборудования, необходимого для освоения дисциплины/модуля	Перечень технических средств обучения, необходимых для освоения дисциплины/модуля (демонстрационное оборудование)
Стандартное оборудование (учебная мебель для обучающихся, рабочее место преподавателя, доска)	Комплект мультимедийного оборудования: проектор, экран, компьютер, акустическая система. Локальная и корпоративная сеть
Помещение для самостоятельной работы	Компьютеры с подключением к информационно-телекоммуникационной сети «Интернет», доступом в электронную информационно-образовательную среду ТИУ

11. Методические указания по организации СРС

11.1. Методические указания по подготовке к практическим занятиям.

На практических занятиях обучающиеся изучают методику и выполняют типовые задания. В процессе подготовки к практическим занятиям обучающиеся могут прибегать к консультациям преподавателя.

Практическая работа №1
Введение в язык программирования Python

Цель работы: познакомиться со средой разработки Python. Изучить основные типы данных, команды ввода и вывода данных.

Python – это объектно-ориентированный, интерпретируемый, переносимый язык сверхвысокого уровня. Программирование на Python позволяет получать быстро и качественно необходимые программные модули.

В комплекте вместе с интерпретатором Python идет IDLE (интегрированная среда разработки). По своей сути она подобна интерпретатору, запущенному в интерактивном режиме с расширенным набором возможностей (подсветка синтаксиса, просмотр объектов, отладка и т.п.).

Для запуска IDLE в Windows необходимо перейти в папку Python в меню “Пуск” и найти там ярлык с именем “IDLE (Python 3.X XX-bit)”.

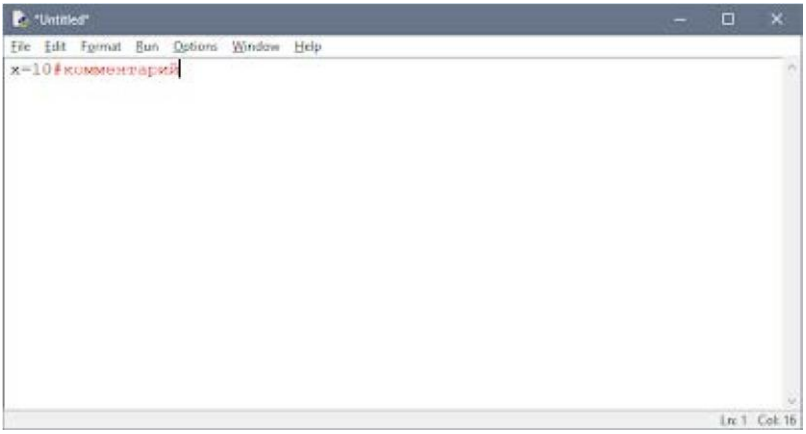
Для запуска редактора программы (кода) следует выполнить команду File->New File или сочетание клавиш Ctrl+N.

Любая Python-программа состоит из последовательности допустимых символов, записанных в определенном порядке и по определенным правилам.

Программа включает в себя:

- комментарии;
- команды;
- знаки пунктуации;
- идентификаторы;
- ключевые слова.

Комментарии в Python обозначаются предваряющим их символом # и продолжаются до конца строки (т.е. в Python все комментарии являются однострочными), при этом не допускается использование перед символом # кавычек:



Знаки пунктуации. В алфавит Python входит достаточное количество знаков пунктуации, которые используются для различных целей. Например, знаки "+" или " *" могут использоваться для сложения и умножения, а знак запятой "," - для разделения параметров функций.

Идентификаторы. Идентификаторы в Python это имена используемые для обозначения переменной, функции, класса, модуля или другого объекта.

Ключевые слова. Некоторые слова имеют в Python специальное назначение и представляют собой управляющие конструкции языка.

Ключевые слова в Python:

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif',

'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Типы данных

1. None (неопределенное значение переменной)
2. Логические переменные (Boolean Type)
3. Числа (Numeric Type)
   - int – целое число
   - float – число с плавающей точкой
   - complex – комплексное число

4. Списки (Sequence Type)
   - list – список
   - tuple – кортеж
   - range – диапазон
5. Строки (Text Sequence *Type*)
   - str

Ввод и вывод данных

Ввод данных осуществляется при помощи команды **input (**список ввода**):**

```
a = input()
print(a)
```

В скобках функции можно указать сообщение - комментарий к вводимым данным:

```
a = input ("Введите количество: ")
```

Команда input() по умолчанию воспринимает входные данные как строку символов. Поэтому, чтобы ввести целочисленное значение, следует указать тип данных int(): a = int (input())

Для ввода вещественных чисел применяется команда

```
a=float(input())
```

Вывод данных осуществляется при помощи команды **print** (список вывода):

```
a = 1
b = 2
print(a)
print(a + b)
print('сумма = ', a + b)
```

Существует возможность записи команд в одну строку, разделяя их через **;**.

Однако не следует часто использовать такой способ, это снижает удобочитаемость:

```
a = 1; b = 2; print(a)
print (a + b)
print ('сумма = ', a + b)
```

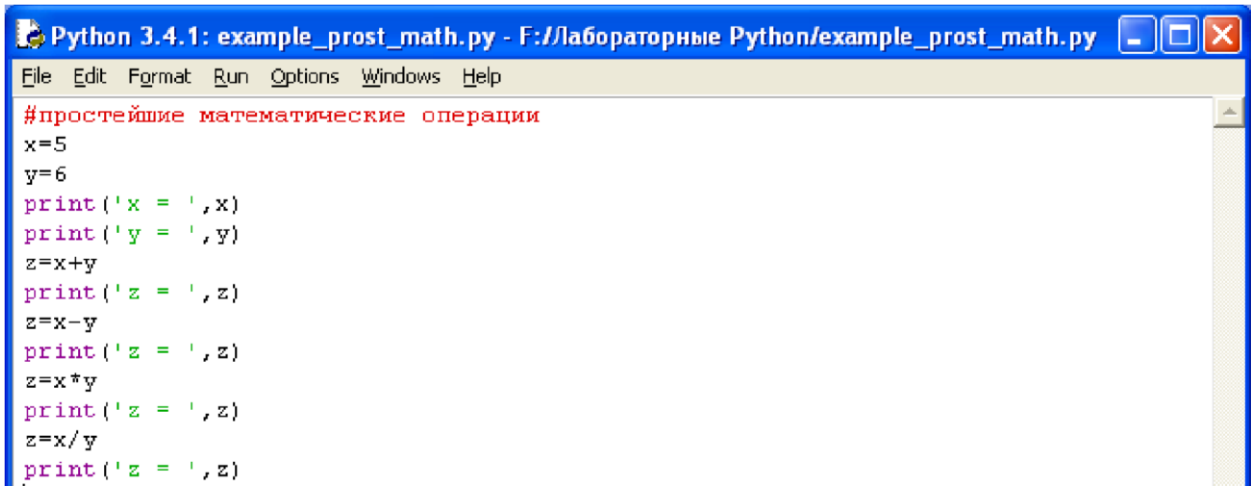
Для команды **print** может задаваться так называемый сепаратор — разделитель между элементами вывода:

```
x=2
y=5
print ( x, "+", y, "=", x+y, sep = " " )
```

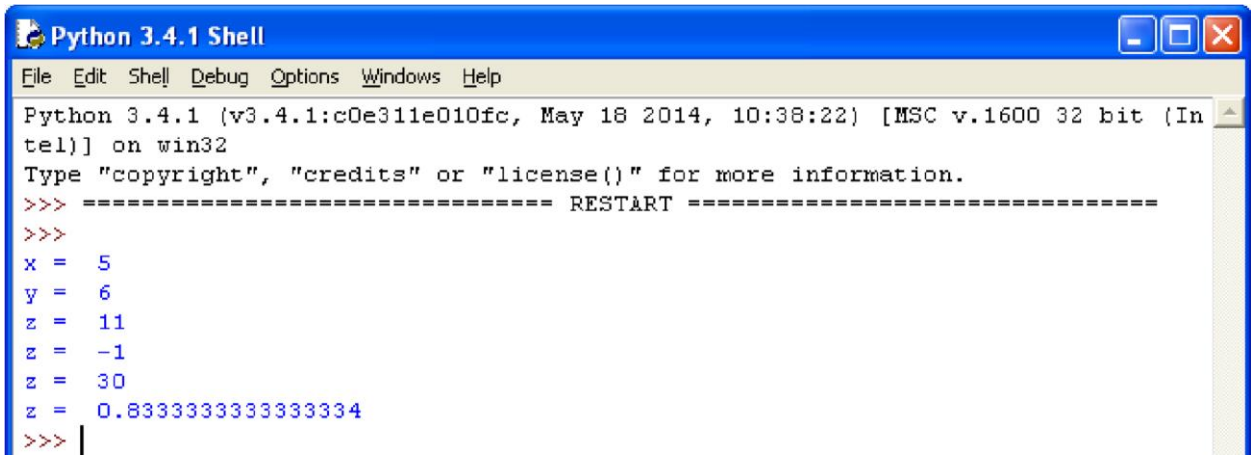
Результат отобразится с пробелами между элементами: 2 + 5 = 7

Простые арифметические операции над числами

x + y	Сложение
x - y	Вычитание
x * y	Умножение
x / y	Деление



Пример программы на Python



Результат выполнения программы с применением простых арифметических операций

Для форматированного вывода используется **format**:

Строковый метод format() возвращает отформатированную версию строки, заменяя идентификаторы в фигурных скобках {}. Идентификаторы могут быть позиционными, числовыми индексами, ключами словарей, именами переменных.

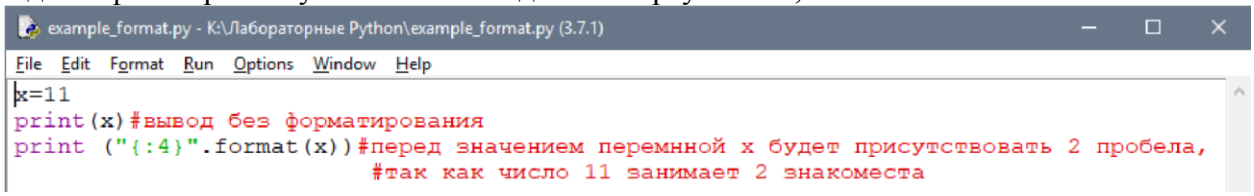
Синтаксис команды **format**:

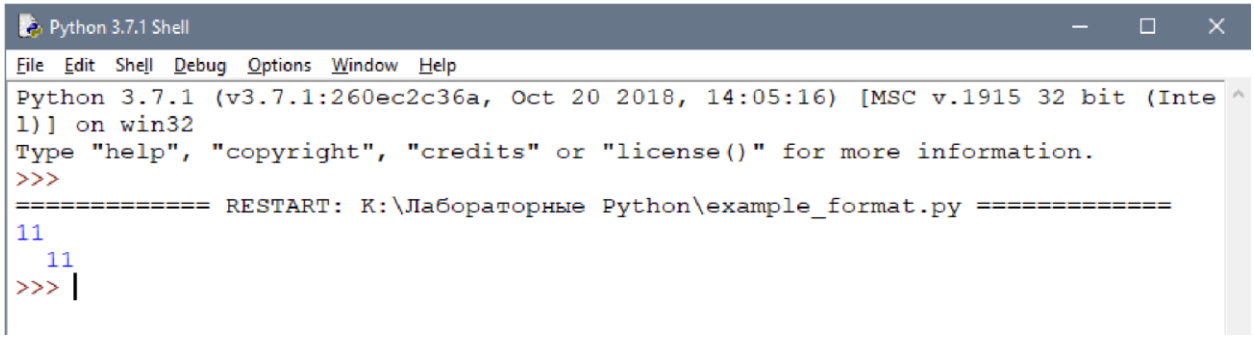
поле замены := "{" [имя поля] ["!" преобразование] [":" спецификация] "}" имя поля := arg_name ("." имя атрибута | "[" индекс "]")*

преобразование := "r" (внутреннее представление) | "s" (человеческое представление) спецификация := см. ниже

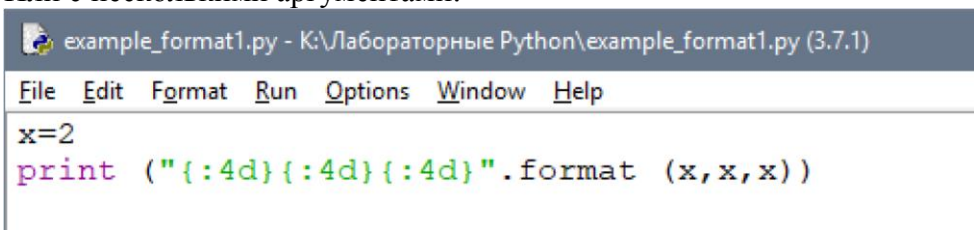
Аргументов в format() может быть больше, чем идентификаторов в строке. В таком случае оставшиеся игнорируются.

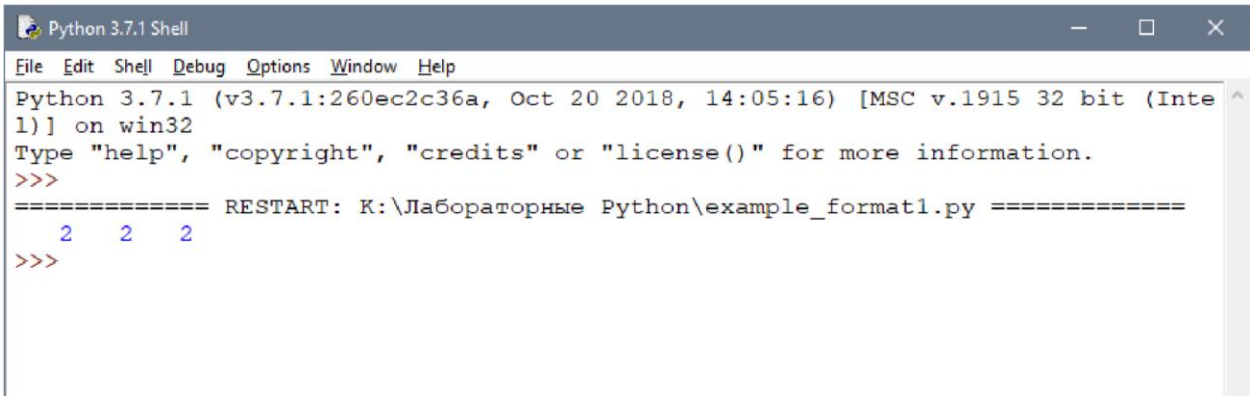
Идентификаторы могут быть либо индексами аргументов, либо ключами:





В результате выведется число 11, а перед ним два пробела, так как указано использовать для вывода четыре знакоместа.
Или с несколькими аргументами:





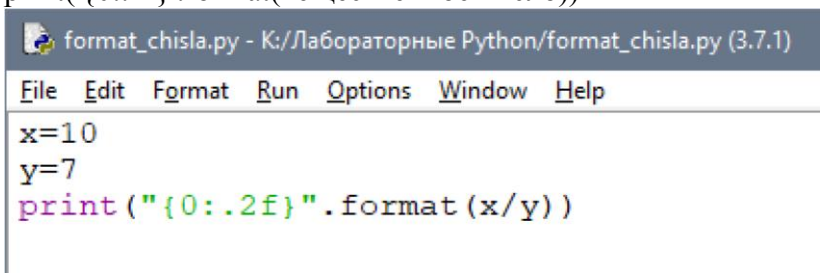
В итоге каждое из значений выводится из расчета 4 знакоместа.
Спецификация формата:

спецификация	:= [[fill]align][sign][#][0][width][,][.precision][type]
заполнитель	:= символ кроме '{' или '}'
выравнивание	:= "<" | ">" | "=" | "^"
знак	:= "+" | "-" | " "
ширина	:= integer
точность	:= integer
тип	:= "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"

Тип	**Значение**
'd', 'i', 'u'	Десятичное число

'o'	Число в восьмеричной системе счисления
'x'	Число в шестнадцатеричной системе счисления (буквы в нижнем регистре)
'X'	Число в шестнадцатеричной системе счисления (буквы в верхнем регистре)
'e'	Число с плавающей точкой с экспонентой (экспонента в нижнем регистре)
'E'	Число с плавающей точкой с экспонентой (экспонента в верхнем регистре)
'f', 'F'	Число с плавающей точкой (обычный формат)
'g'	Число с плавающей точкой. с экспонентой (экспонента в нижнем регистре), если она меньше, чем -4 или точности, иначе обычный формат
'G'	Число с плавающей точкой. с экспонентой (экспонента в верхнем регистре), если она меньше, чем -4 или точности, иначе обычный формат
'c'	Символ (строка из одного символа или число - код символа)
's'	Строка
'%'	Число умножается на 100, отображается число с плавающей точкой, а за ним знак %

Для форматирования вещественных чисел с плавающей точкой используется следующая команда:
print('{0:.2f}'.format(вещественное число))



В результате выведется число с двумя знаками после запятой.

Пример:
Напишите программу, которая запрашивала бы у пользователя:
- ФИО ("Ваши фамилия, имя, отчество?")
- возраст ("Сколько Вам лет?")
- место жительства ("Где вы живете?")

После этого выводила бы три строки:
"Ваше имя"
"Ваш возраст"

"Вы живете в"

Решение:

```
a=input('Введите ваши фамилию, имя, отчество ')
b=input('Сколько вам лет? ')
c=input('Где вы живёте? ')
print('Ваше имя ',a)
print('Ваш возраст ',b)
print('Вы живете в ',c)
```

```
Введите ваши фамилию, имя, отчество Иванов Иван Иванович
Сколько вам лет? 15
Где вы живёте? Уссурийск
Ваше имя  Иванов Иван Иванович
Ваш возраст  15
Вы живете в  Уссурийск
```

Задания для самостоятельной работы

1) Установите Python https://www.python.org/ftp/python/3.8.5/python-3.8.5.exe

2) Напишите программу, которая запрашивала бы у пользователя:

Имя, Фамилия, Возраст, Место жительства

- фамилия, имя ("Ваши фамилия, имя?")
- возраст ("Сколько Вам лет?")
- место жительства ("Где вы живете?")

После этого выводила бы три строки:

"Ваши фамилия, имя"

"Ваш возраст"

"Вы живете в"

Практическая работа №2
Структура ветвление в Python

Цель работы: познакомиться со структурой ветвление (if, if-else, if-elif-else). Научиться работать с числами и строками используя данную структуру.

Условный оператор ветвления if, if-else, if-elif-else

Оператор ветвления if позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Возможны следующие варианты использования.

1. Конструкция if

Синтаксис оператора if выглядит так:

if логическое выражение:

команда_1

команда_2

...

команда_n

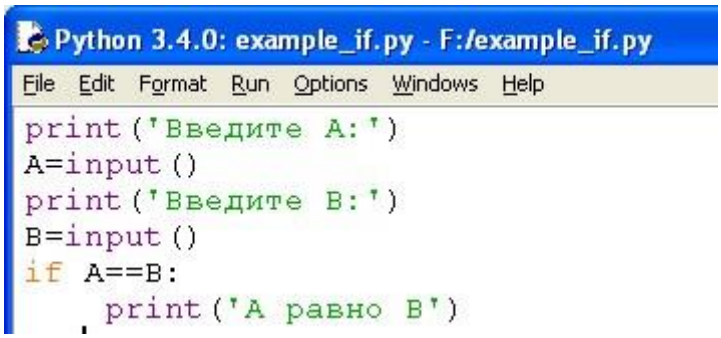
После оператора if записывается логическое выражение.

Логическое выражение — конструкция языка программирования, результатом вычисления которой является «истина» или «ложь».

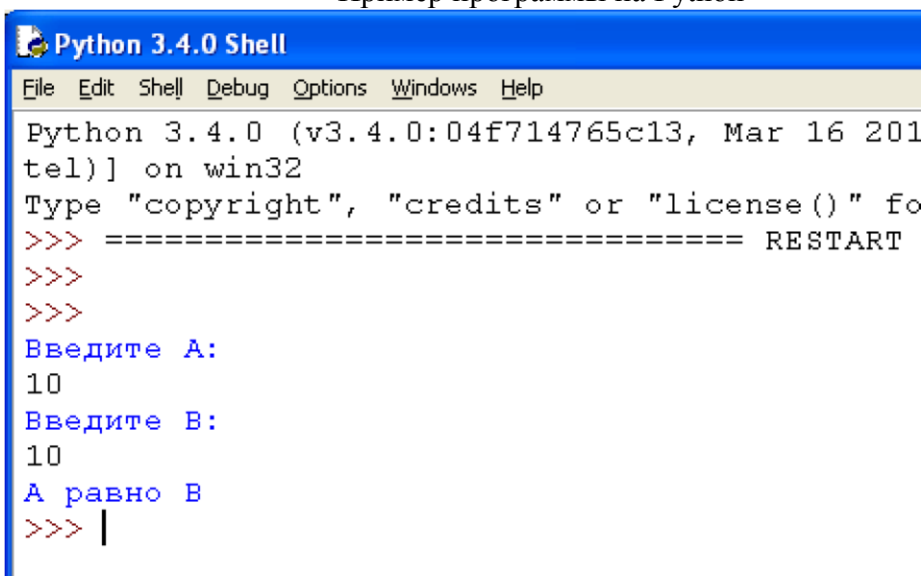
Если это выражение истинно, то выполняются инструкции, определяемые данным оператором. Выражение является истинным, если его результатом является число не равное нулю, непустой объект, либо логическое True. После выражения нужно поставить двоеточие ":".

ВАЖНО: блок кода, который необходимо выполнить, в случае истинности выражения, отделяется **четырьмя** пробелами слева!

Программа запрашивает у пользователя два числа, затем сравнивает их и если числа равны, то есть логическое выражение A==B истинно, то выводится соответствующее сообщение.



Пример программы на Python



Результат выполнения программы с использованием условного оператора if

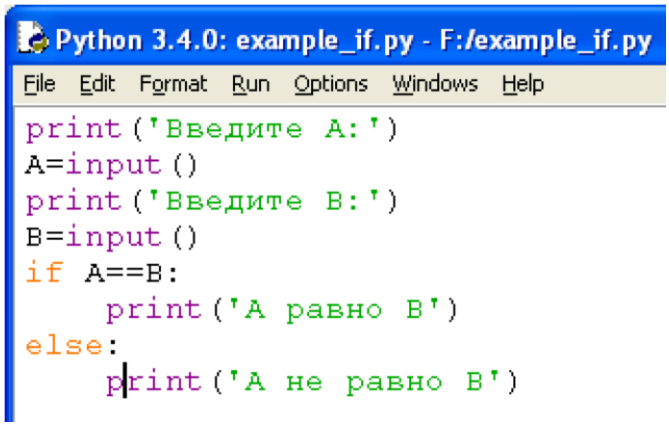
2. Конструкция if – else

Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т.е. при истинном условии нужно выполнить один набор инструкций, при ложном – другой. Для этого используется конструкция if – else.

Синтаксис оператора if – else выглядит так:

if логическое выражение:
команда_1
команда_2
...
команда_n
else:
команда_1
команда_2
...
команда_n

Программа запрашивает у пользователя два числа, затем сравнивает их и если числа равны, то есть логическое выражение A==B истинно, то выводится соответствующее сообщение. В противном случае выводится сообщение, что числа не равны.



Пример программы на Python

```
>>> ================================ RESTART
>>>
Введите A:
10
Введите B:
5
A не равно B
>>>
```

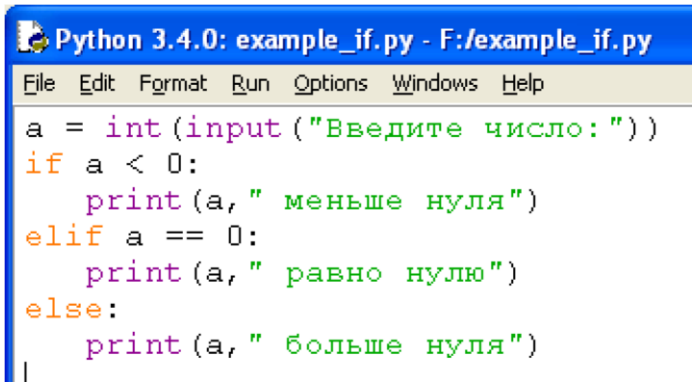
Результат выполнения программы с использованием условного оператора if-else

3. Конструкция if – elif – else

Для реализации выбора из нескольких альтернатив можно использовать конструкцию if – elif – else. Синтаксис оператора if – elif – else выглядит так:

if логическое выражение_1:
команда_1
команда_2
...
команда_n
elif логическое выражение_2:
команда_1
команда_2
...
команда_n
elif логическое выражение_3:
команда_1
команда_2
...
команда_n
else:
команда_1
команда_2
...
команда_n

Программа запрашивает число у пользователя и сравнивает его с нулём a<0. Если оно меньше нуля, то выводится сообщение об этом. Если первое логическое выражение не истинно, то программа переходит ко второму - a==0. Если оно истинно, то программа выведет сообщение, что число равно нулю, в противном случае, если оба вышеуказанных логических выражения оказались ложными, то программа выведет сообщение, что введённое число больше нуля.



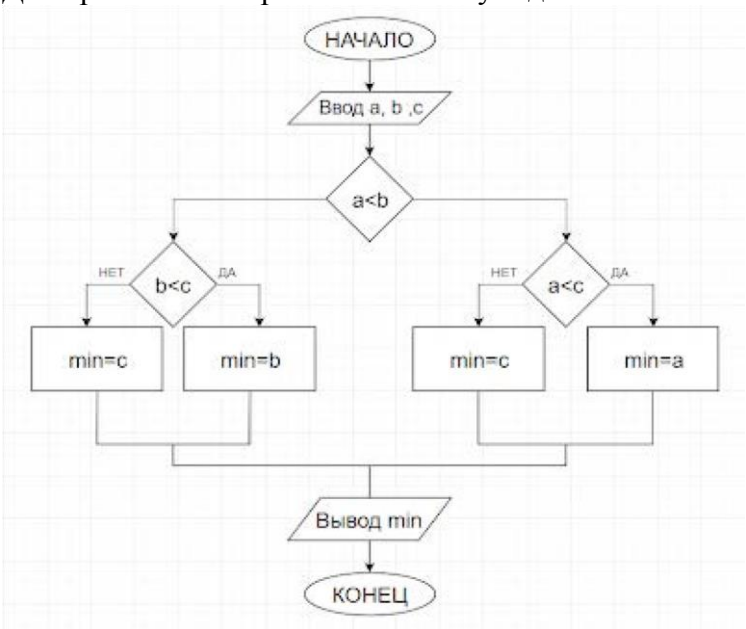
Пример программы на Python

```
Введите число:41
41  больше нуля
>>> ================================ RESTART =
>>>
Введите число:-5
-5  меньше нуля
>>> ================================ RESTART =
>>>
Введите число:0
0  равно нулю
>>>
```

Результат выполнения программы с использованием условного оператора if-elif-else

Пример

Задание:
Дано 3 числа. Найти минимальное среди них и вывести на экран.
Решение:
Для простоты построим блок-схему задачи.



Командами
a=input('')
b=input('')
c=input('')
введём три числа, присвоив значения переменным a, b, c.

Условной конструкцией if-else проверим на истинность логическое выражение a<b. Если оно истинно, то переходим на проверку логического выражения a<c. Если оно истинно, то переменной "y" присвоим значение переменной "a", т.е. "a" будет минимальным, а иначе "y" присвоится значение переменной "c".

Если в начале логическое выражение a<b оказалось ложным, то переходим на проверку другого логического выражения b<c.

Если оно истинно, то "y" присвоится значение переменной "b", иначе "c".

Командой print() выводим минимальное значение.

```
#нахождение минимального из 3-х чисел
a=input('Введите целое число \n')
b=input('Введите целое число \n')
c=input('Введите целое число \n')
if a<b:
    if a<c:
        y=a
    else:
        y=c
else:
    if b<c:
        y=b
    else:
        y=c
print('Минимальное:',y)
```

Пример программы

```
Введите целое число
2
Введите целое число
5
Введите целое число
1
Минимальное: 1
```

Результат выполнения программы

Задания для самостоятельной работы

Даны три целых числа. Выбрать из них те, которые принадлежат интервалу [1,3].

Практическая работа №3
Работа с циклами в Python

Цель работы: познакомиться с циклическими конструкциями

В Python существуют два типа цикличных выражений:

- Цикл while
- Цикл for

1. Цикл while в Python

Инструкция while в Python повторяет указанный блок кода до тех пор, пока указанное в цикле логическое выражение будет оставаться истинным.

Синтаксис цикла while:

while логическое выражение:

команда 1

команда 2

...
команда n

После ключевого слова while указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций, который идет далее.
Все инструкции, которые относятся к циклу while, располагаются на последующих строках и должны иметь отступ от начала строки (4 пробела).

```
#! Программа по вычислению факториала
number = int(input("Введите число: "))
i = 1
factorial = 1
while i <= number:
    factorial *= i
    i += 1
print("Факториал числа", number, "равен", factorial)
```

Пример программы на Python

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Введите число: 5
Факториал числа 5 равен 120
>>>
```

Результат выполнения программы с использованием циклического оператора while

2. Цикл for в Python

Цикл for в Python обладает способностью перебирать элементы любого комплексного типа данных (например, строки или списка). Синтаксис цикла for: **for** int **in** range():
команда 1
команда 2
...
команда n

Переменной int присваивается значение первого элемента функции range(), после чего выполняются команды. Затем переменной int присваивается следующее по порядку значение и так далее до тех пор, пока не будут перебраны все элементы функции range().

Функция range() является универсальной функцией Python для создания списков (list) содержащих арифметическую прогрессию. Чаще всего она используется в циклах for.
range (старт, стоп, шаг) - так выглядит стандартный вызов функции range() в Python. По умолчанию старт равняется нулю, шаг единице.

Пример:

1. Найти сумму n элементов следующего ряда чисел: 1 -0.5 0.25 -0.125 ... n. Количество элементов (n) вводится с клавиатуры. Вывести на экран каждый член ряда и его сумму. Решить задачу используя циклическую конструкцию for.

Решение:

В данном случае ряд чисел состоит из элементов, где каждый следующий меньше предыдущего в два раза по модулю и имеет обратный знак. Значит, чтобы получить следующий элемент, надо предыдущий разделить на -2. Какой-либо переменной надо присвоить значение первого элемента ряда (в данном случае это 1). Далее в цикле добавлять ее значение к переменной, в которой накапливается сумма, после чего присваивать ей значение следующего элемента ряда, разделив текущее значение на -2. Цикл должен выполняться n раз.

```
n=int(input('Введите количество элементов последоваетльности: '))
x=1
s=0
print(x)
for i in range(n):
    s+=x
    x/=-2
    print(x)
print('Сумма ряда:',s)
```

Пример программы с циклом for

```
Введите количество элементов последоваетльности: 5
1
-0.5
0.25
-0.125
0.0625
-0.03125
Сумма ряда: 0.6875
```

Результат выполнения программы

2. Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.

Решение:

Для начала введём целое число командой int(input(текст сообщения)). Затем зададим переменной i значение 2. Переменная i выполняет роль счётчика. Если задать ей значение 1, то условие задачи не будет выполнено, а результатом всегда будет 1.

В цикле while в качестве логического выражения используется команда n%i сравниваемая с нулём. Таким образом, если остаток от деления введённого числа на текущее значение i не равно нулю, то счётчик увеличивается на 1, а если равно нулю цикл заканчивается и командой print() выводится сообщение и значение i.

```
n = int(input('Введите целое число не меньшее 2\n'))
i = 2
while n%i != 0:
    i+=1
print('наименьший натуральный делитель:',i)
```

Пример программы с циклом while

```
Введите целое число не меньшее 2
49
наименьший натуральный делитель: 7
```

Результат выполнения программы

Задание

1. Дано вещественное число – цена 1 кг конфет. Вывести стоимость 1, 2, … 10 кг конфет. Решить задачу используя циклическую конструкцию for.
2. Дана непустая последовательность целых чисел, оканчивающаяся нулем. Найти:

а) сумму всех чисел последовательности;

б) количество всех чисел последовательности

Решить задачу используя циклическую конструкцию while.

Практическая работа №4

Работа со строками в Python

Цель работы: познакомится с методами работы со строками.

Обучающийся должен:

владеть: навыками составления линейных алгоритмов на языке программирования Python с использованием строковых данных;

уметь: применять функции и методы строк при обработке строковых данных;

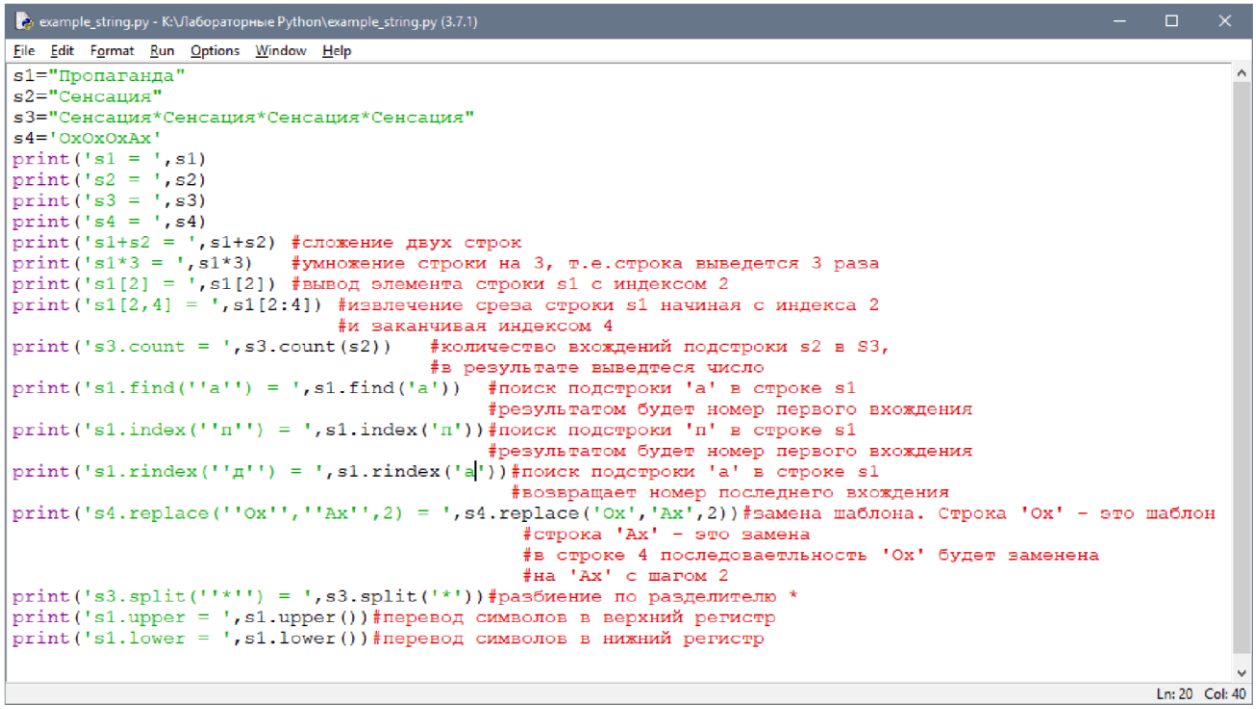
знать: операции и методы обработки строк.

Строка — базовый тип представляющий из себя неизменяемую последовательность символов; str от «string» — «строка».

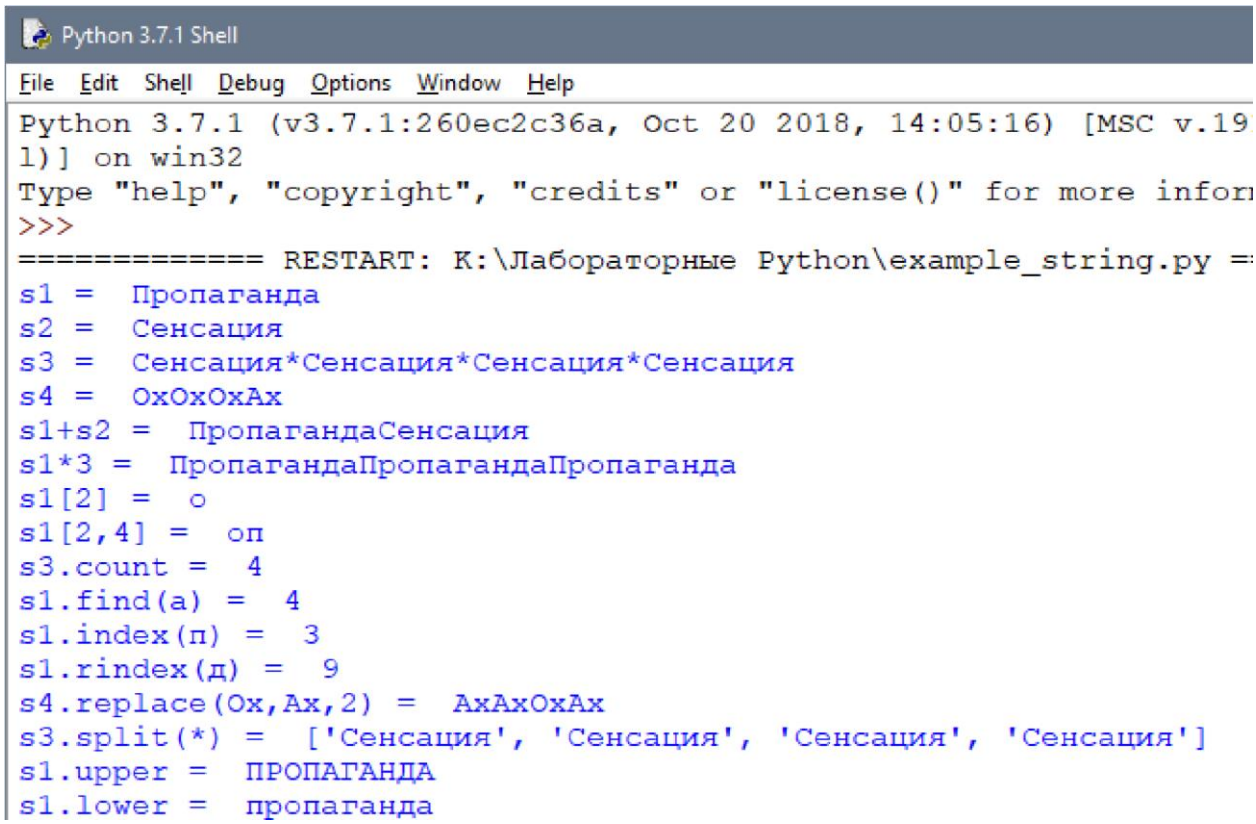
Функции и методы работы со строками

Функция или метод	Назначение
S1 + S2	Конкатенация (сложение строк)
S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза
len(S)	Длина строки
S.join(список)	Соединение строк из последовательности str через разделитель, заданный строкой
S1.count(S[, i, j])	количество вхождений подстроки s в строку s1. Результатом является число. Можно указать позицию начала поиска i и окончания поиска *j*
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
S.index(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
S.rindex(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
S.replace(шаблон, замена)	Замена шаблона
S.split(символ)	Разбиение строки по разделителю
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру

Ниже приведена программа, демонстрирующая использование функций и методов работы со строками.



Пример программы на Python



Результат выполнения программы с использованием функций и методов работы со строками

Пример:

Вариант 0

Проверить, будет ли строка читаться одинаково справа налево и слева направо (т. е. является ли она палиндромом).

Решение:

Сначала введём строку командой: s=input('Введите строку ').
Затем определим логическую переменную flag и присвоим ей значение 1: flag=1.

Для начала в введённой строке нужно удалить пробелы. Для этого воспользуемся циклической конструкцией for, которая выполнится столько раз, какую имеет длину строка. Длину строки определим функцией len(s).

В теле цикла будем проверять следующее условие: s[i]!=' '. Данное логическое выражение будет истинно в том случае, если i-ый элемент строки не будет равен пробелу, тогда выполнится команда следующая после двоеточия: string+=s[i].

К сроке string, которая была объявлена в начале программы, будет добавляться посимвольно строка s, но уже без пробелов.

Для проверки строки на "палиндром" воспользуемся циклической конструкцией for.

Длина половины строки находится делением нацело на 2. Если количество символов нечетно, то стоящий в середине не учитывается, т.к. его сравниваемая пара - он сам. Количество повторов цикла равно длине половины строки. Длину строки определим функцией len(s), где аргумент введённая нами строка s. Зная длину строки, можно вычислить количество повторов цикла. Для этого целочисленно разделим длину строки на 2: len(s)//2.

Для задания диапазона для цикла используем функцию range(), в которой аргументом будет являться половина длины строки: range(len(s//2)). for i in range(len(s//2)).

Если символ с индексом i не равен "симметричному" символу с конца строки (который находится путем индексации с конца) if s[i] != s[-1-i], то переменной flag присваивается значение 0 и происходит выход из цикла командой break.

Далее, при помощи условной конструкции if-else в зависимости от значения flag либо - 0, либо -1 выводится сообщение, что строка палиндром, либо нет.

```
s=input('Введите строку \n')
flag=1
string=''
for i in range(len(s)):
    if s[i]!=' ':
        string+=s[i]
print(string)
for i in range(len(s)//2):
    if string[i]!=string[-i-1]:
        flag=0
        break
if flag: print('Палиндром')
else: print('не палиндром')
```

Пример программы на Python

```
Введите строку
а роза упала на лапу азора
арозаупалаланалапуазора
Палиндром
```

Результат выполнения программы

Практическая работа №5
Работа со списками. Операции над списками в Python

Цель работы: Изучение одномерных массивов в Python.

Массивы (списки) в Python — это определенное количество элементов одного типа, которые имеют общее имя, и каждый элемент имеет свой индекс — порядковый номер. Часто для работы с массивами используются списки.

Список (list) – это структура данных для хранения объектов различных типов. Списки являются упорядоченными последовательностями, которые состоят из различных типов данных, заключающихся в квадратные скобки [ ] и отделяющиеся друг от друга с помощью запятой.

Создание списков на Python

Создать список можно несколькими способами:

1. Получение списка через присваивание конкретных значений.

Так выглядит в коде Python пустой список:

s = [] # Пустой список

Примеры создания списков со значениями:

```
l=[5,75,-4,7,-51]# список целых чисел
l=[1.13,5.34,12.63,4.6,34.0,12.8]# список из вещественных чисел
l=["Оля", "Владимир", "Михаил", "Дарья"]# список из строк
l=["Москва", "Иванов", 12, 124]    # смешанный список
l=[[0, 0, 0], [1, 0, 1], [1, 1, 0]]    # список, состоящий из списков
l=['s', 'p', ['isok'], 2] # список из значений и списка
```

Списки можно складывать (конкатенировать) с помощью знака «+»:

```
l=[1, 3]+[4,23]+[5]
print('l=[1, 3]+[4,23]+[5] =',l)
```

Результат:

```
>>>
l=[1, 3]+[4,23]+[5] = [1, 3, 4, 23, 5]
>>>
```

2. Создание списка при помощи функции Split()

Используя функцию split в Python можно получить из строки список. stroka ="Привет, страна" lst=stroka.split(",")

```
stroka ="Здравствуй, Дедушка Мороз"  #stroka - строка
lst=stroka.split(",") #lst - список
print('stroka = ',stroka)
print('lst=stroka.split(","):',lst)
```

Результат:

```
============== RESTART: C:/Users/maxim/Desktop/ex_list_
stroka =  Здравствуй, Дедушка Мороз
lst=stroka.split(","): ['Здравствуй', ' Дедушка Мороз']
```

3. Генераторы списков

В Python создать список можно также при помощи генераторов.

Первый способ:

Сложение одинаковых списков заменяется умножением: Список из 10 элементов, заполненный единицами l = [1]*10

Второй способ:

Пример 1.

l = [i for i in range(10)]

Пример 2.
c = [c * 3 for c in 'list'] print (c) # ['lll',
'iii', 'sss', 'ttt']

```
Создание списка из строки.
l = list (строка):
 ['с', 'т', 'р', 'о', 'к', 'а']

Создание списка при помощи функции Split().
stroka=" Hello, friend "
lst=stroka.split(","):
 ['Hello', ' friend']

Генераторы списков.
Первый способ.
l = [1]*10:
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Второй способ. Пример 1.
l = [i for i in range(10)]:
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Второй способ. Пример 2.
c=[c*3 for c in "list"]:
 ['lll', 'iii', 'sss', 'ttt']
```

Примеры использования генераторов списка

Пример 1.
Заполнить список квадратами чисел от 0 до 9, используя генератор списка.
Решение:
l = [i*i for i in range(10)]
Пример 2.
Заполнить список числами, где каждое последующее число больше на 2. l = [(i+1)+i for
i in range(10)] print(l)

```
Заполнить список квадратами чисел от 0 до 9, используя генератор списка.
l = [i*i for i in range(10)]:
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Заполнить список числами, где каждое последующее число больше на 2.
l = [(i+1)+i for i in range(10)]:
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

Модуль random предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности. random.randint(A, B) - случайное целое число N, A ≤ N ≤ B. random.random() - случайное число от 0 до 1.
Случайные числа в списке:
10 чисел, сгенерированных случайным образом в диапазоне (10,80) from random import
randint
l = [randint(10,80) for x in range(10)]
10 чисел, сгенерированных случайным образом в диапазоне (0,1) l = [random() for i in range(10)]

```
from random import *
l = [randint(10,80) for i in range(10)]
print('10 чисел, сгенерированных случайным образом в диапазоне (10,80).')
print('l = [randint(10,80) for x in range(10)]:')
print(l)
print()

l = [random() for i in range(10)]
print('10 чисел сгенерированных в диапазоне от 0 до 1.')
print('l = [random() for i in range(10):')
for i in range(len(l)):
    print ('{:.2f}'.format(l[i]), end = " " )
```

```
10 чисел, сгенерированных случайным образом в диапазоне (10,80).
l = [randint(10,80) for x in range(10)]:
[70, 33, 79, 61, 34, 27, 11, 55, 52, 31]

10 чисел сгенерированных в диапазоне от 0 до 1.
l = [random() for i in range(10):
0.66 0.97 0.87 0.57 0.54 0.83 0.57 0.65 0.04 0.07
```

4. Ввод списка (массива) в языке Python

Для ввода элементов списка используется цикл for и команда range (): for i in range(N): x[i] = int(input())

Более простой вариант ввода списка:

x = [int(input()) for i in range(N)]

```
print('Ввод списка. Пример 1:')
x=[]
for i in range(4):
    x.append(int(input()))
print(x)

x=[]
print('Ввод списка. Пример 2:')
x = [ int(input()) for i in range(4) ]
print(x)
```

```
Ввод списка. Пример 1:
45
4
85
2
[45, 4, 85, 2]
Ввод списка. Пример 2:
4
5
7
8
[4, 5, 7, 8]
```

Функция int здесь используется для того, чтобы строка, введенная пользователем, преобразовывалась в целые числа.

5. Вывод списка (массива) в языке Python

Вывод целого списка (массива):

print (L)

Поэлементный вывод списка (массива): for i in range(N):

```
  print ( L[i], end = " " )
```

```
Вывод целого списка (массива)
[1, 56, 6, 3, 6, 7, 3, 37, 7, 37, 37]

Поэлементный вывод списка (массива)
1 56 6 3 6 7 3 37 7 37 37
```

2. Методы списков

Метод	Что делает
list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет перед i-ым элементом значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
list.count(x)	Возвращает количество элементов со значением x
list.reverse()	Разворачивает список
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

Ниже приведена программа, демонстрирующая методы работы списков.

```
a=[0,2,2,2,4] #список a
b=[5,6,7,2,9] #список b
print('Исходный список a:',a)
print('Исходный список b:',b)
x=99
y=5

a.append(x)
print('a.append(x):',a)

a.extend(b)
print('a.extend(b):',a)

a.insert(3,x)
print('a.insert(3,x):',a)

a.remove(x)
print('a.remove(x):',a)

print('a.pop(5):',a.pop(5))
print(a)

print('a.index(y,0,len(a)):',a.index(y,0,len(a)))

print('a.count(2):',a.count(2))

a.reverse()
print('a.reverse():',a)

z=a.copy()
print('z=a.copy():',z)

z.clear()
print('z.clear():')
print('z =',z)
```

Пример программы на Python

```
Исходный список a: [0, 2, 2, 2, 4]
Исходный список b: [5, 6, 7, 2, 9]
a.append(x): [0, 2, 2, 2, 4, 99]
a.extend(b): [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.insert(3,x): [0, 2, 2, 99, 2, 4, 99, 5, 6, 7, 2, 9]
a.remove(x): [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.pop(5): 99
[0, 2, 2, 2, 4, 5, 6, 7, 2, 9]
a.index(y,0,len(a)): 5
a.count(2): 4
a.reverse(): [9, 2, 7, 6, 5, 4, 2, 2, 2, 0]
z=a.copy(): [9, 2, 7, 6, 5, 4, 2, 2, 2, 0]
z.clear():
z = []
```

Результат выполнения программы

Вариант 0

1. Из массива X длиной n, среди элементов которого есть положительные, отрицательные и равные нулю, сформировать новый массив Y, взяв в него только те элементы из X, которые больше по модулю заданного числа M.

Вывести на экран число M, данный и полученные массивы.

Решение:

```
n=int(input('Введите длину массива\n'))
m=int(input('Введите число M\n'))
x=[]
y=[]
for i in range(n):
    print('Введите ',i,'элемент:')
    x.append(int(input()))
for i in range(n):
    if abs(x[i])>m:
        y.append(x[i])
print('Введённое число M:',m)
print('Массив X:',x)
print('Массив Y:',y)
```

```
Введите длину массива
5
Введите число M
20
Введите  0 элемент:
21
Введите  1 элемент:
22
Введите  2 элемент:
5
Введите  3 элемент:
6
Введите  4 элемент:
8
Введённое число M: 20
Массив X: [21, 22, 5, 6, 8]
Массив Y: [21, 22]
```

2. В массиве целых чисел все отрицательные элементы заменить на положительные. Вывести исходный массив и полученный.

Решение:

```
n=int(input('Введите длину массива:'))
a=[]
for i in range(n):
    print('Введите',i,'элемент:')
    a.append(int(input()))
print('Исходный массив:',a)
for i in range(n):
    if a[i]<0:
        a[i]=-a[i]
print('Полученный массив:',a)
```

```
Введите длину массива:5
Введите 0 элемент:
-5
Введите 1 элемент:
-4
Введите 2 элемент:
-6
Введите 3 элемент:
5
Введите 4 элемент:
-7
Исходный массив: [-5, -4, -6, 5, -7]
Полученный массив: [5, 4, 6, 5, 7]
```

Практическая работа №6
Функции и процедуры в Python

Цель работы: изучение процедур и функций в Python.

Обучающийся должен:

знать - синтаксис процедур и функций, процедура с параметром, локальные и глобальные переменные.

уметь - применять синтаксис процедур и функций при составлении программы;

владеть - основными навыками работы с функциями и процедурами.

Подпрограмма - это именованный фрагмент программы, к которому можно обратиться из другого места программы Подпрограммы делятся на две категории: процедуры и функции.

1. Процедуры.

Рассмотрим синтаксис процедуры:

def имя процедуры (Список параметров):

Система команд.

Для определения процедуры используется ключевое слово def, затем указывается имя процедуры и в скобках её формальные параметры, если они присутствуют. После ставится двоеточие и со следующей строки с отступом в 4 пробела указываются команды. Процедура — вспомогательный алгоритм, выполняющий некоторые действия. Процедура должна быть определена к моменту её вызова. Определение процедуры начинается со служебного слова def.

Вызов процедуры осуществляется по ее имени, за которым следуют круглые скобки, например, Err(). В одной программе может быть сколько угодно много вызовов одной и той же процедуры. Использование процедур сокращает код и повышает удобочитаемость.

Процедура с параметрами.

Как используются в Python параметры процедуры, рассмотрим на примере.

Пример:

Написать процедуру, которая печатает раз указанный символ (введенный с клавиатуры), каждый с новой строки.

```
def printChar(s):
print (s)
sim = input('введите символ') printChar(sim) # первый вызов, вывод
введенного символа
```

printChar('*') # второй вызов, вывод *

```
def printChar(s):
    print (s)
sim = input('введите символ: ')
printChar(sim) # первый вызов, вывод введенного символа
printChar('*') # второй вызов, вывод *
```

```
>>>
введите символ: 41
41
*
```

Глобальная переменная — если ей присвоено значение в основной программе (вне процедуры).

Локальная переменная (внутренняя) известна только на уровне процедуры, обратиться к ней из основной программы и из других процедур нельзя.

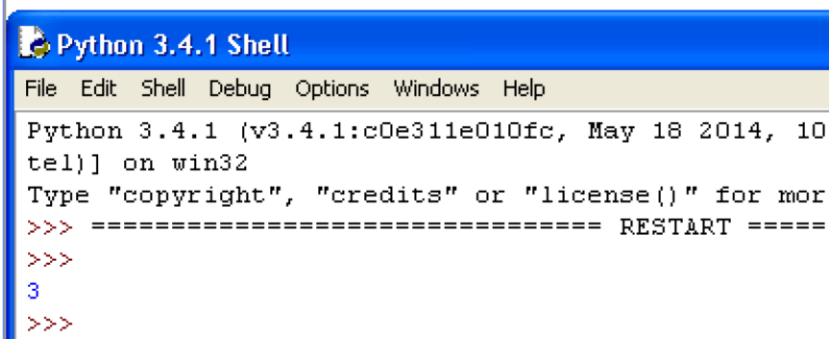
Параметры процедуры — локальные переменные.

Примеры использования локальных и глобальных переменных

Пример 1.

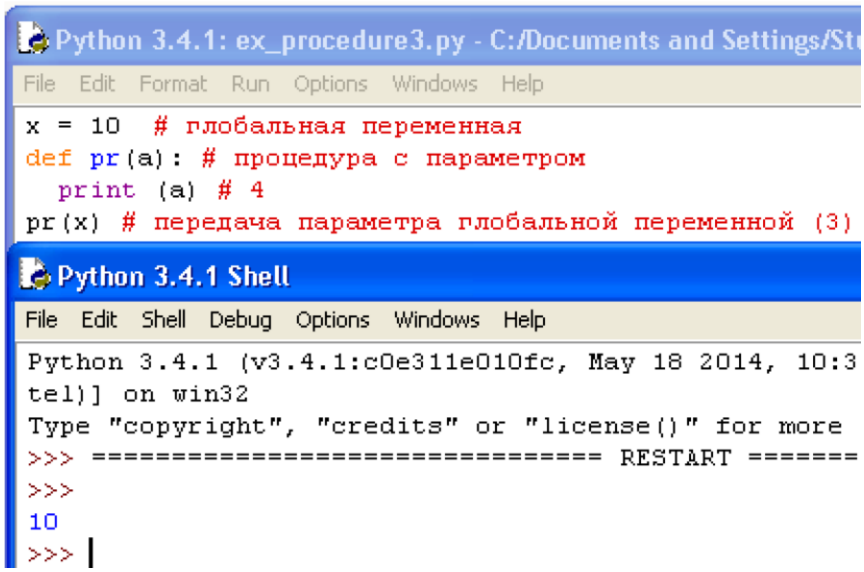
x = 3 # глобальная переменная def pr(): # процедура без параметров
print (x) # вывод значения глобальной переменной pr()

```
x = 3 # глобальная переменная
def pr(): # процедура без параметров
  print (x) # вывод значения глобальной переменной
pr()
```

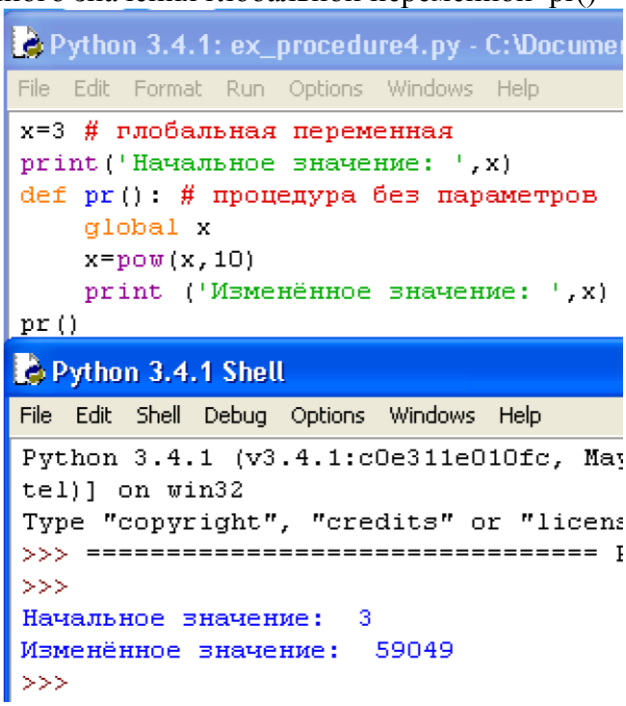


Пример 2.

x = 3 # глобальная переменная def pr(a): # процедура с параметром
print (a) # 4
pr(x) # передача параметра глобальной переменной (3)



Существует возможность изменить значение глобальной переменной (не создавая локальную). В процедуре с помощью слова global:
x = 3 # глобальная переменная
def pr(): # процедура без параметров global x
x = pow(x,10)
print (x) # вывод измененного значения глобальной переменной pr()



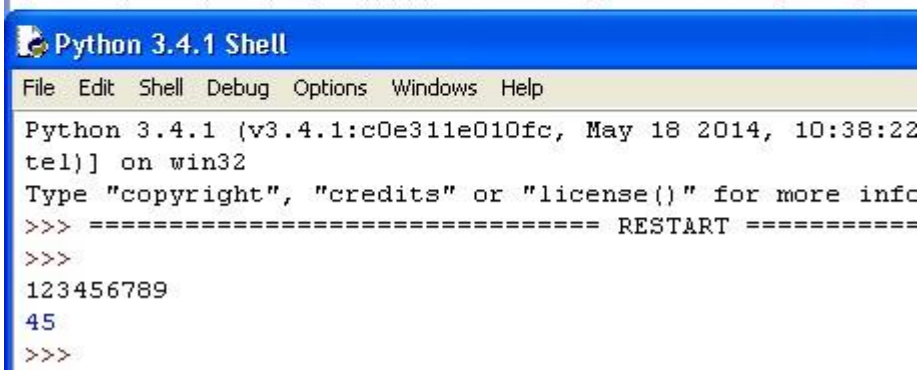
Функция - подпрограмма, к которому можно обратиться из другого места программы. Для создания функции используется ключевое слово def, после которого указывается имя и список аргументов в круглых скобках. Тело функции выделяется также как тело условия (или цикла): четырьмя пробелами.

Рассмотрим синтаксис функции:
def имя функции (Список параметров):
Система команд return выражение

Часть функций языка Python являются встроенными функциями, которые обеспечены синтаксисом самого языка. Например, int, input, randint. Рассмотрим пример создания пользовательских функций.

Пример 1.
Вычислить сумму цифр числа. def sumD(n): # определение функции с параметром sumD = 0
while n!= 0:
sumD += n % 10
n = n // 10
return sumD # возврат значения функции
основная программа
print (sumD(int(input())) # вызов функции с параметром

```
def sumD(n): # определение функции с параметром
    summa = 0
    while n!= 0:
        summa += n % 10
        n=n//10
    return summa # возврат значения функции
# основная программа
print (sumD(int(input()))) # вызов функции с параметром
```



Вариант 0

1. Определить, являются ли три треугольника равновеликими. Длины сторон вводить с клавиатуры. Для подсчёта площади треугольника использовать формулу Герона. Вычисление площади оформить в виде функции с тремя параметрами.
Формула Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

$$\text{где } p = \frac{a+b+c}{2}$$

Решение:

```
import math
def s(x,y,z):
    p=(x+y+z)/2
    s=math.sqrt(p*(p-x)*(p-y)*(p-z))
    return s
A=[]
for i in range(3):
    print('Введите стороны ',i,'-го треугольника:')
    a=int(input('a:'))
    b=int(input('b:'))
    c=int(input('c:'))
    A.append(s(a,b,c))
for i in range(3):
    print('Площадь ',i,'-го треугольника {:.2f}'.format(A[i]))
if A[0]==A[1]:
    if A[0]==A[2]:
        print('Треугольники равновеликие')
else: print('Треугольники не равновеликие')
```

```
Введите стороны  0 -го треугольника:
a:3
b:4
c:5
Введите стороны  1 -го треугольника:
a:6
b:7
c:8
Введите стороны  2 -го треугольника:
a:9
b:10
c:11
Площадь  0 -го треугольника 6.00
Площадь  1 -го треугольника 20.33
Площадь  2 -го треугольника 42.43
Треугольники не равновеликие
```

2. Ввести одномерный массив A длиной m. Поменять в нём местами первый и последний элементы. Длину массива и его элементы ввести с клавиатуры. В программе описать процедуру для замены элементов массива. Вывести исходные и полученные массивы.

Решение:

```
def zam(X):
    tmp=X[0]
    X[0]=X[len(X)-1]
    X[len(X)-1]=tmp
A=[]
m=int(input('Введите длину массива:'))
for i in range(m):
    print('Введите ',i,'элемент массива')
    A.append(int(input()))
print(A)
zam(A)
print(A)
```

```
Введите длину массива:5
Введите  0 элемент массива
0
Введите  1 элемент массива
1
Введите  2 элемент массива
2
Введите  3 элемент массива
3
Введите  4 элемент массива
4
[0, 1, 2, 3, 4]
[4, 1, 2, 3, 0]
```

Практическая работа №7
Математические операции в Python

Цель работы: познакомиться с основными математическими операциями в Python.

Язык Python, благодаря наличию огромного количества библиотек для решения разного рода вычислительных задач, сегодня является конкурентом таким пакетам как Matlab и Octave. Запущенный в интерактивном режиме, он, фактически, превращается в мощный калькулятор. В этом уроке речь пойдет об арифметических операциях, доступных в данном языке Арифметические операции изучим применительно к числам.

Если в качестве операндов некоторого арифметического выражения используются только целые числа, то результат тоже будет целое число. Исключением является операция деления, результатом которой является вещественное число. При совместном использовании целочисленных и вещественных переменных, результат будет вещественным.

В этом уроке речь пойдет об арифметических операциях, доступных в данном языке.

Если в качестве операндов некоторого арифметического выражения используются только целые числа, то результат тоже будет целое число. Исключением является операция деления, результатом которой является вещественное число. При совместном использовании целочисленных и вещественных переменных, результат будет вещественным.

Целые числа (int)

Числа в Python 3 поддерживают набор самых обычных математических операций:

x + y	Сложение
x - y	Вычитание
x * y	Умножение
x / y	Деление
x // y	Получение целой части от деления
x % y	Остаток от деления
-x	Смена знака числа
abs(x)	Модуль числа
divmod(x, y)	Пара (x // y, x % y)

x ** y	Возведение в степень
pow(x, y[, z])	x : Число, которое требуется возвести в степень. y : Число, являющееся степенью, в которую нужно возвести первый аргумент. Если число отрицательное или одно из чисел "x" или "y" не целые, то аргумент "z" не принимается. z : Число, на которое требуется произвести деление по модулю. Если число указано, ожидается, что "x" и "y" положительны и имеют тип int.

Пример применения вышеописанных операций над целыми числами

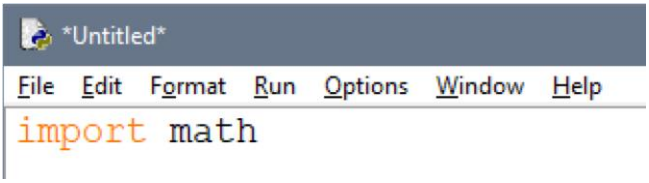
```
x = 5
y = 2
z = 3
x+y = 7
x-y = 3
x*y = 10
x/y = 2.5
x//y = 2
x%y = 1
-x= -5
abs(-x) = 5
divmod(x,y) = (2, 1)
x**y = 25
pow(x,y,z) = 1
```

Вещественные числа (float). Вещественные числа поддерживают те же операции, что и целые. Однако (из-за представления чисел в компьютере) вещественные числа неточны, и это может привести к ошибкам.

Пример применения вышеописанных операций над вещественными числами

```
x = 5.5
y = 2.3
x+y = 7.8
x-y = 3.2
x*y = 12.649999999999999
x/y = 2.3913043478260874
x//y = 2.0
x%y = 0.9000000000000004
-x= -5.5
abs(-x) = 5.5
divmod(x,y) = (2.0, 0.9000000000000004)
x**y = 50.44686540422945
```

Библиотека (модуль) math. В стандартную поставку Python входит библиотека math, в которой содержится большое количество часто используемых математических функций.
Для работы с данным модулем его предварительно нужно импортировать.



Рассмотрим наиболее часто используемые функции модуля math

math.ceil(x)	Возвращает ближайшее целое число большее, чем x
math.fabs(x)	Возвращает абсолютное значение числа x
math.factorial(x)	Вычисляет факториал x
math.floor(x)	Возвращает ближайшее целое число меньшее, чем x
math.exp(x)	Вычисляет e**x
math.log2(x)	Логарифм по основанию 2
math.log10(x)	Логарифм по основанию 10
math.log(x[, base])	По умолчанию вычисляет логарифм по основанию e, дополнительно можно указать основание логарифма
math.pow(x, y)	Вычисляет значение x в степени y
math.sqrt(x)	Корень квадратный от x

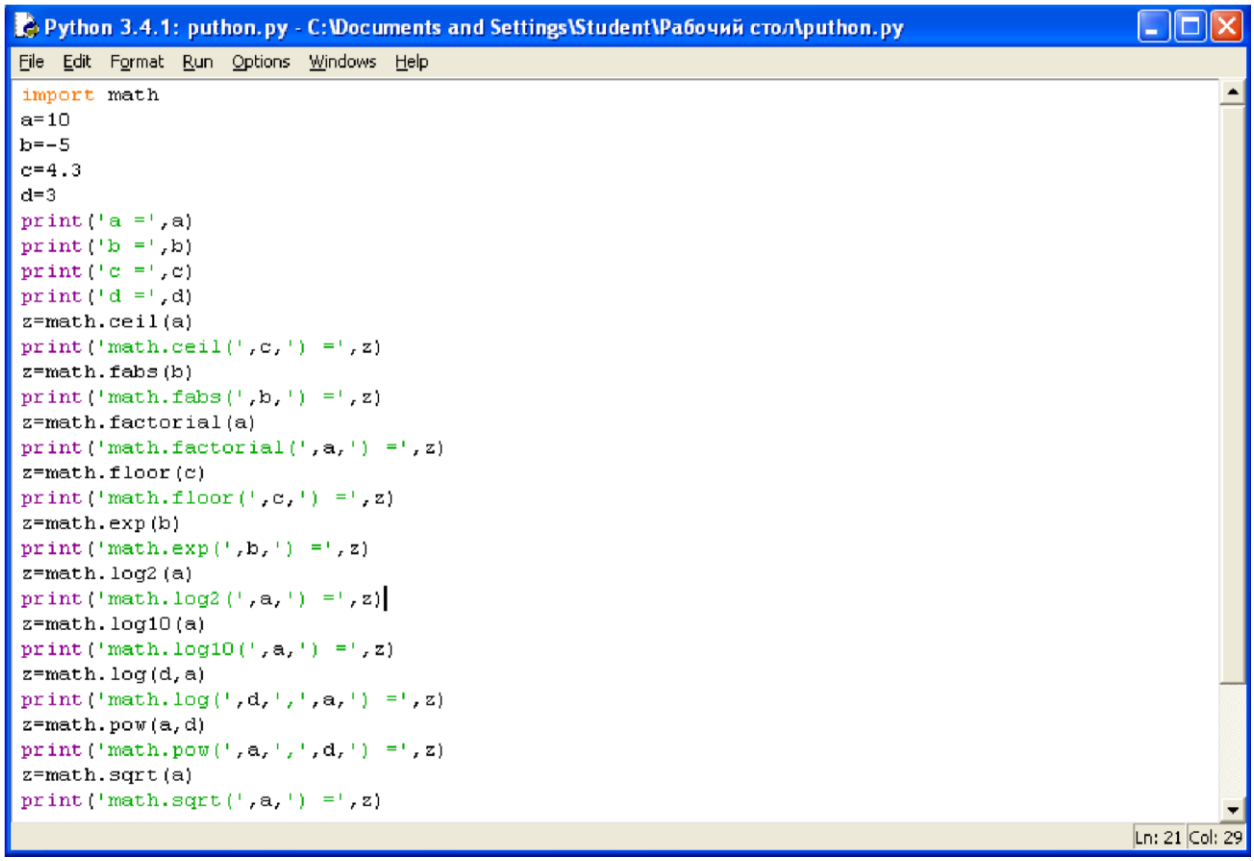
Пример применения вышеописанных функций над числами

В программе определены 4 переменные - a, b, c, d, каждая из которых является либо целым числом, либо вещественным, либо отрицательным.

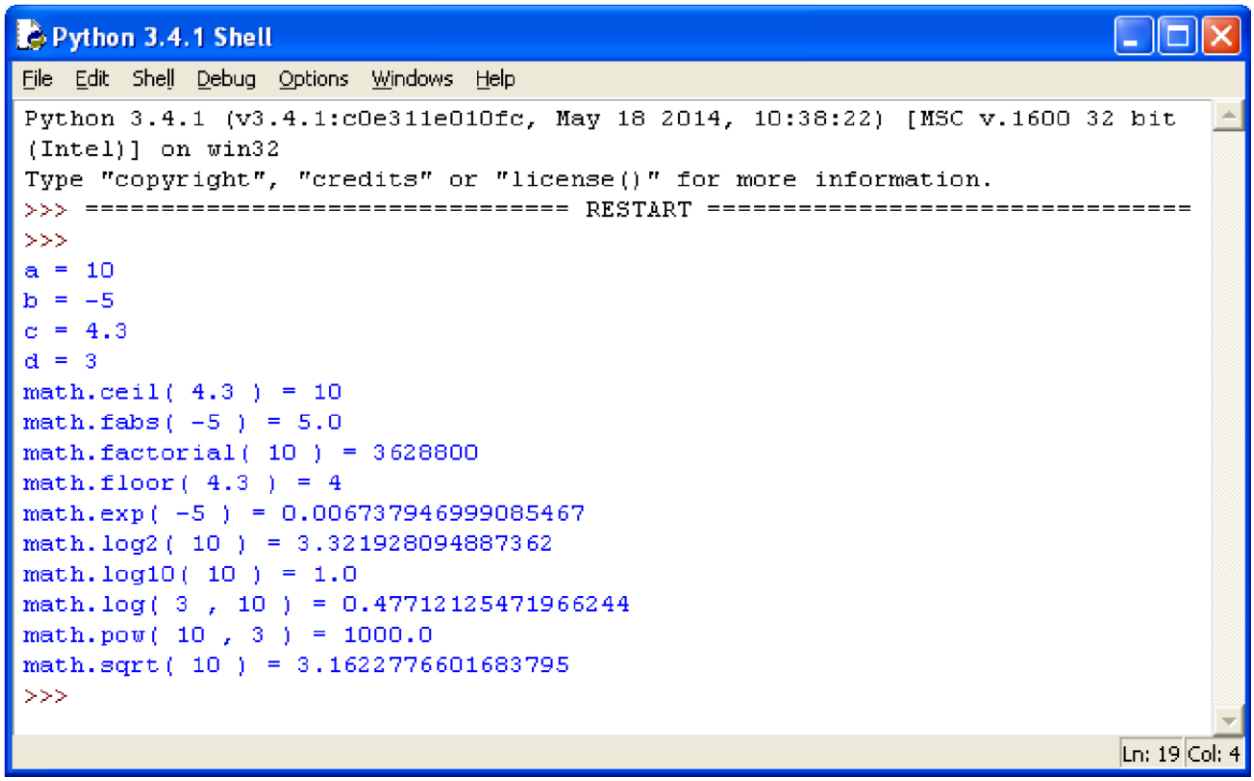
Командой print() выводится значение каждой переменной на экран при выполнении программы.

В переменную z помещается результат выполнения функции модуля math.

Затем командой print() выводится сообщение в виде используемой функции и её аргумента и результат её выполнения.



Пример программы на Python



Результат выполнения программы с применением функций модуля math

Тригонометрические функции модуля math

math.cos(x)	Возвращает cos числа X

math.sin(x)	Возвращает sin числа X
math.tan(x)	Возвращает tan числа X
math.acos(x)	Возвращает acos числа X
math.asin(x)	Возвращает asin числа X
math.atan(x)	Возвращает atan числа X

Пример применения вышеописанных функций над числами

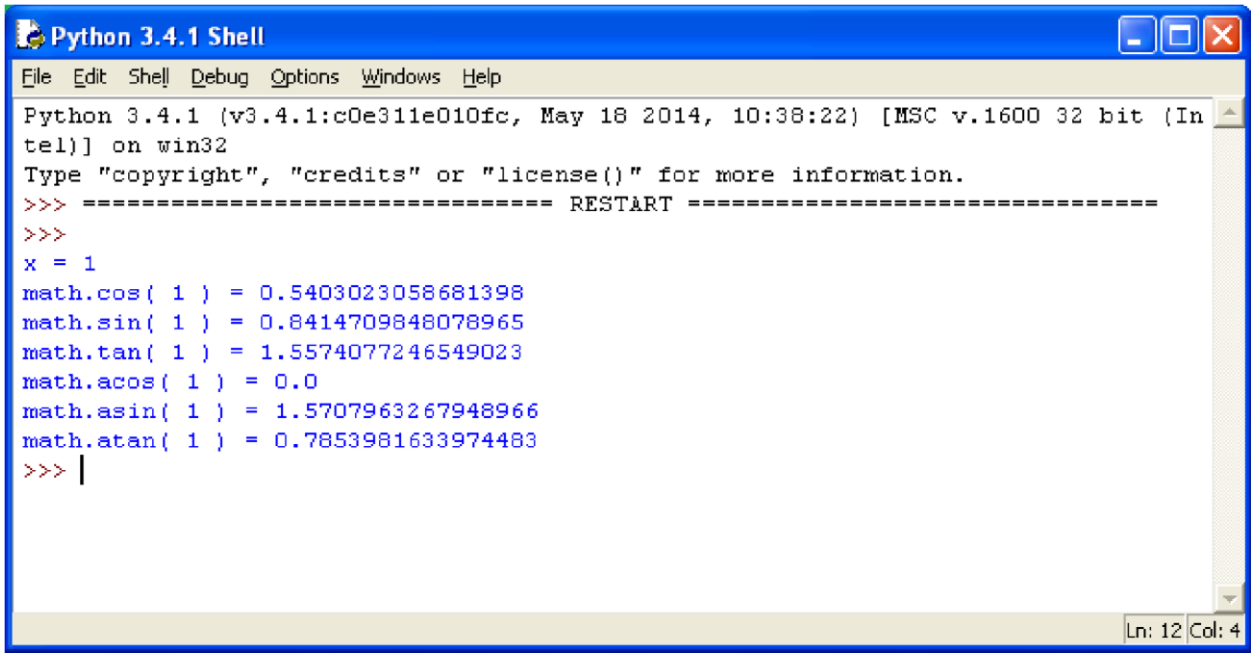
В программе определена переменная x, содержащая целое число.

Значение переменной выводится командой print() на экран.

В переменную z помещается результат выполнения тригонометрической функции модуля math.

Затем командой print() выводится сообщение в виде используемой функции и её аргумента и результат её выполнения.

Пример программы с использованием тригонометрических функций модуля math



Результат выполнения программы с
применением тригонометрических функций модуля math

Константы:

• **math.pi** - число Pi.

• **math.e** - число e (экспонента).

Пример:

Напишите программу, которая бы вычисляла заданное арифметическое выражение при заданных переменных. Ввод переменных осуществляется с клавиатуры. Вывести результат с 2-мя знаками после запятой.

Задание:

$$Z = \frac{9\pi t + 10\cos(x)}{\sqrt{t} - |\sin(t)|} * e^{x}$$

x=10; t=1

Решение:

Сначала импортируем модуль math. Для этого воспользуемся командой import math. Затем следует ввести значения двух переменных целого типа x и t.

Для ввода данных используется команда input, но так как в условии даны целые числа, то нужно сначала определить тип переменных: x=int(), t=int(). Определив тип переменных, следует их ввести, для этого в скобках команды int() нужно написать команду input().

Для переменной x это выглядит так: x=int(input("сообщение при вводе значения")).

Для переменной t аналогично: t=int(input("сообщение при вводе значения")). Следующий шаг - это составление арифметического выражения, результат которого поместим в переменную z.

Сначала составим числитель. Выглядеть он будет так:
9*math.pi*t+10*math.cos(x).

Затем нужно составить знаменатель, при этом обратим внимание на то, что числитель делится на знаменатель, поэтому и числитель и знаменатель нужно поместить в скобки (), а между ними написать знак деления /. Выглядеть это будет так: (9*math.pi*t+10*math.cos(x))/(math.sqrt(t)math.fabs(math.sin(t))).

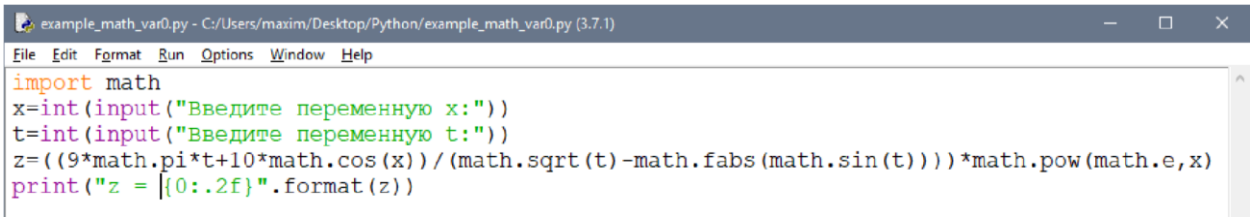
Последним шагом является умножение дроби на экспоненту в степени x. Так как умножается вся дробь, то следует составленное выражение поместить в скобки (), а уже потом написать функцию math.pow(math.e,x).

В результате выражение будет иметь вид:
z=((9*math.pi*t+10*math.cos(x))/(math.sqrt(t)math.fabs(math.sin(t))))*math.pow(math.e,x).

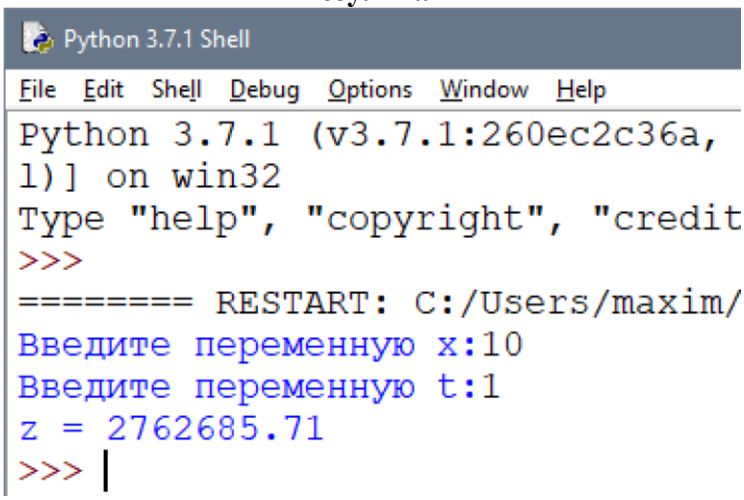
При составлении данного выражения следует обратить внимание на количество открывающихся и закрывающихся скобок.

Командой print() выведем значение переменной, отформатировав его командой format. Сам формат записывается в апострофах в фигурных скобках {}.

В задаче требуется вывести число с двумя знаками после запятой, значит вид формата будет выглядеть следующим образом: {0:.2f}, где 2 - это количество знаков после запятой, а f указывает на то, что форматируется вещественное число. При этом перед 2 нужно поставить точку, указав тем самым на то, что форматируем именно дробную часть числа.



Результат



Задания для самостоятельной работы

Воспроизвести задание из примера. Сделать скриншоты кода и результата.

Практическая работа №8
Работа с двумерными массивами

Цель работы: изучение двумерных массивов в Python.

Обучающийся должен:

знать - способ описания двумерного массива, способы ввода элементов двумерного массива;

уметь - вводить массивы, получать списки через присваивание конкретных значений, применять функции;

владеть - основными навыками создания программ обработки двумерных массивов.

Матрицами называются массивы элементов, представленные в виде прямоугольных таблиц, для которых определены правила математических действий. Элементами матрицы могут являться числа, алгебраические символы или математические функции.

Для работы с матрицами в Python также используются списки. Каждый элемент списка-матрицы содержит вложенный список.

Таким образом, получается структура из вложенных списков, количество которых определяет количество столбцов матрицы, а число элементов внутри каждого вложенного списка указывает на количество строк в исходной матрице.

1. Создание списка

Пусть даны два числа: количество строк n и количество столбцов m.
Необходимо создать список размером n×m, заполненный нулями. Очевидное решение оказывается неверным:
A = [[0] *m]*n

В этом легко убедиться, если присвоить элементу A[0][0] значение 1, а потом вывести значение другого элемента A[1][0] — оно тоже будет равно 1! Дело в том, что [0] * m возвращает ссылку на список из m нулей. Но последующее повторение этого элемента создает список из n элементов, которые являются ссылкой на один и тот же список (точно так же, как выполнение операции B = A для списков не создает новый список), поэтому все строки результирующего списка на самом деле являются одной и той же строкой.

Таким образом, двумерный список нельзя создавать при помощи операции повторения одной строки.

Первый способ:

Сначала создадим список из n элементов (для начала просто из n нулей). Затем сделаем каждый элемент списка ссылкой на другой одномерный список из m элементов:
A = [0] * n for i in
range(n):
A[i] = [0] * m

```
n=3
m=3
A=[0]*n
for i in range(n):
    A[i]=[0]*m
print('A:',A)
```

```
A: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>>
```

Второй способ:

Создать пустой список, потом n раз добавить в него новый элемент, являющийся списком-строкой:
A = [] for i in range(n):
A.append([0] * m)

```
n=3
m=4
A = []
for i in range(n):
    A.append([0]*m)
print(A)
```

```
A: [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
>>>
```

2. Ввод вложенного списка (двумерного массива)

Пример: n=5

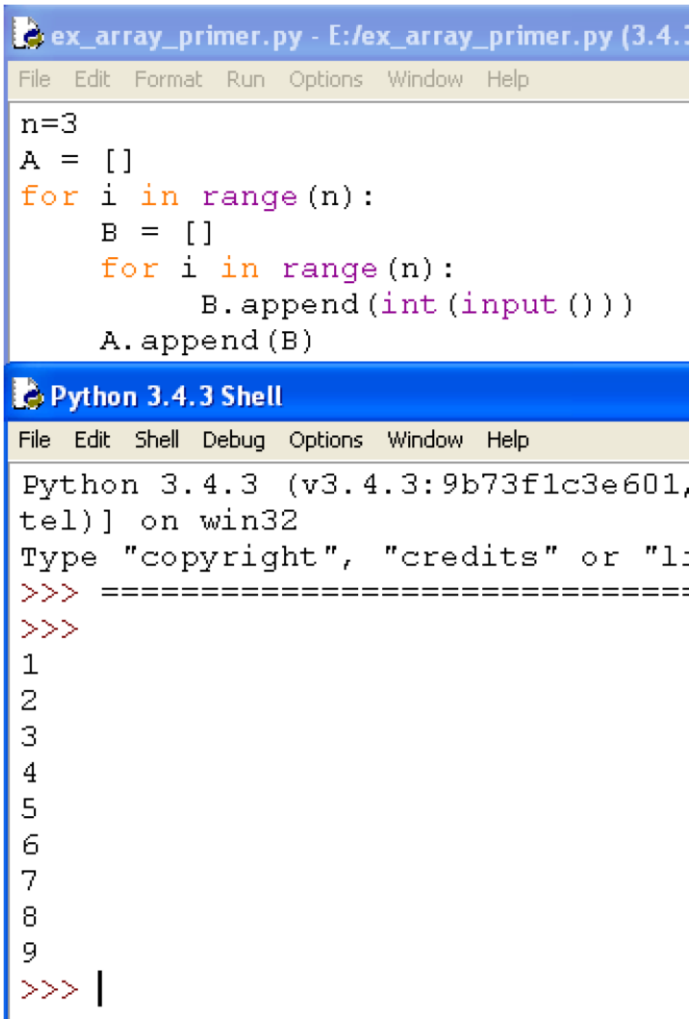
A = [] for i in range(n):

b = input()

for i in range(len(row)):

row[i] = int(row[i])

A.append(row)



3. Вывод вложенного списка (двумерного массива)

Для обработки и вывода списка как правило используется два вложенных цикла. Первый цикл по номеру строки, второй цикл по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

for i in range(n):

for j in range(n]):

print(A[i][j], end = ' ')

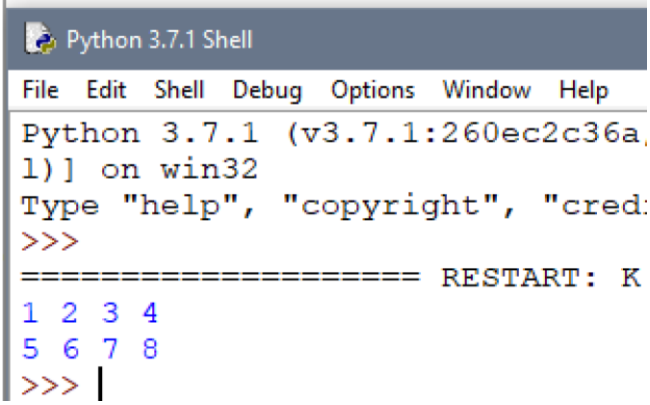
print()

```
n=3
A = []
#ввод массива
for i in range(n):
    B = []
    for i in range(n):
        B.append(int(input()))
    A.append(B)
#вывод массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end=' ')
    print()
```

```
1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9
```

То же самое, но циклы не по индексу, а по значениям списка:
for row in A:
    for elem in row:
        print(elem, end = ' ')   print()

```
A=[[1,2,3,4],[5,6,7,8]]
for row in A:
    for elem in row:
        print(elem, end = ' ')
    print()
```



Для вывода одной строки можно воспользоваться методом join. Использовав этот метод в цикле for можно for row in A: print(' '.join(list(map(str, row))))

4. Обработка и вывод вложенных списков

Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются матрицами или двумерными массивами. В языке программирования Питон таблицу можно представить в виде списка строк, каждый элемент которого является в свою очередь списком, например, чисел. Например, создать числовую таблицу из двух строк и трех столбцов можно так:

A = [[1, 2, 3], [4, 5, 6]]

Здесь первая строка списка A[0] является списком из чисел [1, 2, 3].

То есть

A[0][0]= 1,
A[0][1]= 2,
A[0][2]= 3,
A[1][0]=4,
A[1][1]=5,
A[1][2]=6.

Используем два вложенных цикла для подсчета суммы всех чисел в списке:

S = 0 for i in range(len(A)): for j in range(len(A[i])): S += A[i][j]

Или то же самое с циклом не по индексу, а по значениям строк:

S = 0 for row in A:
for elem in row:
S += elem

```
A=[[1, 2, 3,4],[ 5, 6,7,8]]
#вывод при помощи цикла for и метода join
print('Массив A:')
for i in A:
    print(' '.join(list(map(str, i))))
#Пример 1. Подсчёт суммы всех элементов
S = 0
for i in range(len(A)):
    for j in range(len(A[i])):
        S += A[i][j]
print('Пример 1. Сумма элементов:',S)
#Пример 2. Подсчёт суммы всех элементов
S = 0
for row in A:
    for elem in row:
        S += elem
print('Пример 2. Сумма элементов:',S)
```

```
Массив A:
1 2 3 4
5 6 7 8
Пример 1. Сумма элементов: 36
Пример 2. Сумма элементов: 36
```

5. Пример сложной обработки массива

Пусть дана квадратная матрица из n строк и n столбцов. Необходимо элементам, находящимся на главной диагонали, проходящей из левого верхнего угла в правый нижний (то есть тем элементам A[i][j], для которых i==j) присвоить значение 0, элементам, находящимся выше главной диагонали – значение 1, элементам, находящимся ниже главной диагонали – значение 2. То есть получить такой массив (пример для n==3):

0 1 1

2 0 1
2 2 0

Рассмотрим несколько способов решения этой задачи.

Первый способ:

Элементы, которые лежат выше главной диагонали – это элементы A[i][j], для которых i<j, а для элементов ниже главной диагонали i>j. Таким образом, мы можем сравнивать значения i и j и по ним определять значение A[i][j]. Получаем следующий алгоритм:

```
for i in range(n):      for j in
range(n):               if i < j:
A[i][j] = 0             elif i > j:
A[i][j] = 2         else:
A[i][j] = 1
```

Ниже приведён пример программы, в котором квадратная матрица 3x3 заполняется элементами со значением 9, а затем элементам, находящимся на главной диагонали, проходящей из левого верхнего угла в правый нижний (то есть тем элементам A[i][j], для которых i==j) присваивается значение 0, элементам, находящимся выше главной диагонали – значение 1, элементам, находящимся ниже главной диагонали – значение 2.

```
n=3
A=[]
#заполняем массив 9-ми
for i in range(n):
    A.append([9]*n)
#вывод исходного массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
#заменяем элементы главной диагонали, выше и ниже неё
for i in range(n):
    for j in range(n):
        if i < j:
            A[i][j] = 1
        elif i > j:
            A[i][j] = 2
        else:
            A[i][j] = 0
#вывод изменённого массива
print()
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```

```
9 9 9
9 9 9
9 9 9

0 1 1
2 0 1
2 2 0
>>>
```

Второй способ:

Данный алгоритм плох, поскольку выполняет одну или две инструкции if для обработки каждого элемента. Если мы усложним алгоритм, то мы сможем обойтись вообще без условных инструкций.

Сначала заполним главную диагональ, для чего нам понадобится один цикл:

for i in range(n): A[i][i] = 1

Затем заполним значением 0 все элементы выше главной диагонали, для чего нам понадобится в каждой из строк с номером i присвоить значение элементам A[i][j] для j=i+1, ..., n-1. Здесь нам понадобятся вложенные циклы:

for i in range(n):
for j in range(i + 1, n):
A[i][j] = 0

Аналогично присваиваем значение 2 элементам
A[i][j] для j=0, ..., i-1:
for i in range(n):
for j in range(0, i):
A[i][j] = 2

Можно также внешние циклы объединить в один и получить еще одно, более компактное решение:

for i in range(n):
for j in range(0, i):
A[i][j] = 2 A[i][i] = 1 for j in range(i + 1, n):
A[i][j] = 0

```
n=3
A=[]
#заполняем массив 9-ми
for i in range(n):
    A.append([9]*n)
#вывод исходного массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
#заменяем элементы главной диагонали, выше и ниже неё
for i in range(n):
    for j in range(0, i):
        A[i][j] = 2
    A[i][i] = 0
    for j in range(i + 1, n):
        A[i][j] = 1
#вывод изменённого массива
print()
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```

```
9 9 9
9 9 9
9 9 9

0 1 1
2 0 1
2 2 0
>>>
```

Третий способ:

А вот такое решение использует операцию повторения списков для построения очередной строки списка. i-я строка списка состоит из i чисел 2, затем идет одно число 1, затем идет n-i-1 число 0:

for i in range(n):

A[i] = [2] * i + [1] + [0] * (n - i - 1)

```
n=3
A=[]
#заполняем массив 9-ми
for i in range(n):
    A.append([9]*n)
#вывод исходного массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
#заменяем элементы главной диагонали, выше и ниже неё
for i in range(n):
    A[i] = [2] * i + [0] + [1] * (n - i - 1)
#вывод изменённого массива
print()
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```

```
9 9 9
9 9 9
9 9 9

0 1 1
2 0 1
2 2 0
>>>
```

Вариант 0

1. Дан двумерный массив размером 3x3. Определить максимальное значение среди элементов третьего столбца массива; максимальное значение среди элементов второй строки массива. Вывести полученные значения.

Решение:

```
n=3
a=[]
for i in range(n):
    b = []
    for j in range(n):
        print('Введите [',i,',',j,'] элемент')
        b.append(int(input()))
    a.append(b)
#вывод массива
for i in range(n):
    for j in range(n):
        print(a[i][j], end=' ')
    print()

#максимальное значение среди элементов третьего столбца
maximum=a[0][2]
for i in range(n):
    for j in range(n):
        if maximum<a[i][2]:
            maximum=a[i][2]
print('Максимальный в 3 столбце:',maximum)

#максимальное значение среди элементов второй строки
maximum=a[1][0]
for i in range(n):
    for j in range(n):
        if maximum<a[1][j]:
            maximum=a[1][j]
print('Максимальный во второй строке:',maximum)
```

```
Введите [ 0 , 0 ] элемент
1
Введите [ 0 , 1 ] элемент
2
Введите [ 0 , 2 ] элемент
3
Введите [ 1 , 0 ] элемент
4
Введите [ 1 , 1 ] элемент
5
Введите [ 1 , 2 ] элемент
6
Введите [ 2 , 0 ] элемент
7
Введите [ 2 , 1 ] элемент
8
Введите [ 2 , 2 ] элемент
9
1 2 3
4 5 6
7 8 9
Максимальный в 3 столбце: 9
Максимальный во второй строке: 6
```

2. Дан двумерный массив размером mxn. Сформировать новый массив заменив положительные элементы единицами, а отрицательные нулями. Вывести оба массива.

Решение:

```
m=int(input('Введите количество строк'))
n=int(input('Введите количство столбцов'))
a=[]
for i in range(m):
    b = []
    for j in range(n):
        print('Введите [',i,',',j,'] элемент')
        b.append(int(input()))
    a.append(b)
#вывод массива
print('Исходный массив:')
for i in range(m):
    for j in range(n):
        print(a[i][j], end=' ')
    print()

for i in range(m):
    for j in range(n):
        if a[i][j]<0: a[i][j]=0
        elif a[i][j]>0: a[i][j]=1

#вывод массива
print('Изменённый массив:')
for i in range(m):
    for j in range(n):
        print(a[i][j], end=' ')
    print()
```

```
Введите количество строк:3
Введите количество столбцов:4
Введите [ 0 , 0 ] элемент
-1
Введите [ 0 , 1 ] элемент
5
Введите [ 0 , 2 ] элемент
4
Введите [ 0 , 3 ] элемент
-5
Введите [ 1 , 0 ] элемент
-2
Введите [ 1 , 1 ] элемент
-1
Введите [ 1 , 2 ] элемент
0
Введите [ 1 , 3 ] элемент
4
Введите [ 2 , 0 ] элемент
-5
Введите [ 2 , 1 ] элемент
4
Введите [ 2 , 2 ] элемент
5
Введите [ 2 , 3 ] элемент
-5
Исходный массив:
-1 5 4 -5
-2 -1 0 4
-5 4 5 -5
Полученный массив:
0 1 1 0
0 0 0 1
0 1 1 0
```

11.2. Методические указания по организации самостоятельной работы.

Самостоятельная работа обучающихся заключается в получении заданий (тем) у преподавателя для индивидуального освоения дисциплины. Преподаватель на занятии дает рекомендации необходимые для освоения материала.

Самостоятельная работа способствует закреплению навыков работы с учебной и научной литературой, осмыслению и закреплению теоретического материала. Самостоятельная работа выполняется во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль в контроле за работой студентов).

В процессе изучения дисциплины «Математика и Python для анализа данных» обучающимися *основными видами самостоятельной работы* являются:

- подготовка к аудиторным занятиям (лекциям, и практическим занятиям) и выполнение соответствующих заданий;
- самостоятельная работа над отдельными темами учебной дисциплины в соответствии с учебно-тематическим планом;
- подготовка к коллоквиуму;
- подготовка к зачету.

Вопросы для самостоятельного изучения

Раздел 1. Введение. Знакомство с Python

1. Предел и производная. Геометрический смысл производной.

Раздел 2. Библиотеки Python и линейная алгебра

1. Решение оптимизационных задач в SciPy.
2. Системы линейных уравнений.
3. Матричные операции. Ранг и определитель

Раздел 3. Оптимизация и матричные разложения

1. Касательная плоскость и линейное приближение.
2. Оптимизация негладких функций.
3. Метод имитации отжига.
4. Генетические алгоритмы и дифференциальная эволюция. Нелдер-Мид.
5. Приближение матрицей меньшего ранга.

Раздел 4. Случайность

1. Оценка распределения по выборке.
2. Важные характеристики распределений.
3. Центральная предельная теорема. Доверительные интервалы.

Приложение 1

Планируемые результаты обучения для формирования компетенции и критерии их оценивания

Дисциплина/модуль Математика и Python для анализа данных
Код, направление подготовки 21.03.01 Нефтегазовое дело
Направленность (профиль) Эксплуатация и обслуживание объектов добычи нефти

Код компетенции	Код, наименование ИДК	Код и наименование результата обучения по дисциплине	Критерии оценивания результатов обучения			
			Менее 61	61-75	76-90	91-100
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1 Осуществляет выбор актуальных российских и зарубежных источников, а также поиск, сбор и обработку информации, необходимой для решения поставленной задачи	Знать: актуальные российские и зарубежные источники по дисциплине (УК-1.З1)	Не знает актуальные российские и зарубежные источники по дисциплине	Удовлетворительно знает актуальные российские и зарубежные источники по дисциплине	Хорошо знает Актуальные российские и зарубежные источники по дисциплине	Отлично (комплексно) знает актуальные российские и зарубежные источники по дисциплине
		Уметь: осуществлять выбор актуальных российских и зарубежных источников, а также поиск, сбор и обработку информации, необходимой для решения поставленной задачи (УК-1.У1)	Не умеет осуществлять выбор актуальных российских и зарубежных источников, а также поиск, сбор и обработку информации, необходимой для решения поставленной задачи	Умеет осуществлять выбор актуальных российских и зарубежных источников, а также поиск, сбор и обработку информации, необходимой для решения поставленной задачи при помощи преподавателя	Частично умеет осуществлять выбор актуальных российских и зарубежных источников, а также поиск, сбор и обработку информации, необходимой для решения поставленной задачи	Уметь самостоятельно осуществлять выбор актуальных российских и зарубежных источников, а также поиск, сбор и обработку информации, необходимой для решения поставленной задачи
		Владеть: навыками поиска, сбора и обработки информации, необходимой для решения	Не владеет навыками поиска, сбора и обработки информации, необходимой для решения поставленной задачи	Владеет навыками поиска, сбора и обработки информации, необходимой для решения поставленной задачи, допуская ряд ошибок	Владеет навыками поиска, сбора и обработки информации, необходимой для решения поставленной задачи,	В совершенстве владеет навыками поиска, сбора и обработки информации, необходимой для

		поставленной задачи (УК-1.В1)			допуская незначительные неточности в расчетах	решения поставленной задачи
	УК-1.2 Систематизирует и критически анализирует информацию, полученную из разных источников, в соответствии с требованиями и условиями задачи	Знать: основные принципы, требования и правила систематизации и классификации информации, полученной из разных источников, а так же порядка ее анализа согласно выданного технического задания (УК-1.З2)	Не знает основные принципы, требования и правила систематизации и классификации информации, полученной из разных источников, а так же порядка ее анализа согласно выданного технического задания	Удовлетворительно знает основные принципы, требования и правила систематизации и классификации информации, полученной из разных источников, а так же порядка ее анализа согласно выданного технического задания	Допускает не точности в формулировках основных принципов, требований и правил систематизации и классификации информации, полученной из разных источников, а так же порядка ее анализа согласно выданного технического задания	Знает в совершенстве основные принципы, требования и правила систематизации и классификации информации, полученной из разных источников, а так же порядка ее анализа согласно выданного технического задания
		Уметь: реализовывать основные требования и правила систематизации и анализа статистической информации, полученной из разных источников в соответствии с требованиями и условиями поставленной задачи (УК-1.У2)	Не умеет реализовывать основные требования и правила систематизации и анализа статистической информации, полученной из разных источников в соответствии с требованиями и условиями поставленной задачи	Умеет реализовывать основные требования и правила систематизации и анализа статистической информации, полученной из разных источников в соответствии с требованиями и условиями поставленной задачи при помощи преподавателя	Частично умеет реализовывать основные требования и правила систематизации и анализа статистической информации, полученной из разных источников в соответствии с требованиями и условиями поставленной задачи	Уметь самостоятельно реализовывать основные требования и правила систематизации и анализа статистической информации, полученной из разных источников в соответствии с требованиями и условиями поставленной задачи
		Владеть: принципами, требованиями, инструментами систематизации, классификации,	Не владеет принципами, требованиями, инструментами систематизации,	Владеет принципами, требованиями, инструментами систематизации, классификации, анализа	Владеет принципами, требованиями, инструментами систематизации, классификации, анализа информации,	В совершенстве владеет принципами, требованиями, инструментами систематизации,

		анализа информации (УК-1.В2)	классификации, анализа информации	информации, допуская грубые ошибки	допуская незначительные ошибки	классификации, анализа информации
УК-2 Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2.1 Проводит анализ поставленной цели и формулирует совокупность взаимосвязанных задач, которые необходимо решить для ее достижения	Знать: цель и совокупность взаимосвязанных задач, которые необходимо решить для ее достижения (УК-2.З1)	Не знает цель и задачи, которые необходимо решить	Удовлетворительно знает цель и задачи, которые необходимо решить	Хорошо знает цель и задачи, которые необходимо решить	Отлично знает и самостоятельно ставит цель и задачи, которые необходимо решить
		Уметь: проводить анализ поставленной цели и формировать совокупность взаимосвязанных задач, которые необходимо решить для ее достижения (УК-2.У1)	Не умеет проводить анализ поставленной цели и формировать совокупность взаимосвязанных задач, которые необходимо решить для ее достижения	Удовлетворительно умеет проводить анализ поставленной цели и формировать совокупность взаимосвязанных задач, которые необходимо решить для ее достижения	Хорошо умеет проводить анализ поставленной цели и формировать совокупность взаимосвязанных задач, которые необходимо решить для ее достижения	Отлично умеет проводить анализ поставленной цели и формировать совокупность взаимосвязанных задач, которые необходимо решить для ее достижения
		Владеть: навыком постановки проанализированной цели и формирования совокупности взаимосвязанных задач, которые необходимо решить для ее достижения (УК-2.В1)	Не владеет навыком постановки проанализированной цели и формирования совокупности взаимосвязанных задач, которые необходимо решить для ее достижения	Удовлетворительно владеет навыком постановки проанализированной цели и формирования совокупности взаимосвязанных задач, которые необходимо решить для ее достижения	Хорошо владеет навыком постановки проанализированной цели и формирования совокупности взаимосвязанных задач, которые необходимо решить для ее достижения	Отлично владеет навыком постановки проанализированной цели и формирования совокупности взаимосвязанных задач, которые необходимо решить для ее достижения
	УК-2.2 Выбирает оптимальный способ решения задач, исходя из имеющихся	Знать: оптимальный способ решения задач, исходя из имеющихся ресурсов и ограничений (УК-2.З2)	Не знает оптимальный способ решения задач, исходя из имеющихся ресурсов и ограничений	Удовлетворительно знает оптимальный способ решения задач, исходя из имеющихся ресурсов и ограничений	Хорошо знает оптимальный способ решения задач, исходя из имеющихся ресурсов и ограничений	Отлично знает оптимальный способ решения задач, исходя из имеющихся ресурсов и ограничений

	ресурсов и ограничений	Уметь: решать задачи, выбирая оптимальный способ вычисления, исходя из имеющихся ресурсов и ограничений (УК-2.У2)	Не умеет решать задачи и выбирать оптимальный способ вычисления	Удовлетворительно умеет решать задачи и выбирать оптимальный способ вычисления	Решает задачи и умеет выбирать оптимальный способ вычисления, исходя из имеющихся ресурсов и ограничений. Но допускает определенные погрешности и ошибки	Самостоятельно, без посторонней помощи умеет решать задачи и выбирать оптимальный способ вычисления, исходя из имеющихся ресурсов и ограничений
		Владеть: навыком решения задач, выбирая оптимальный способ вычисления, исходя из имеющихся ресурсов и ограничений (УК-2.В2)	Не владеет навыком решения задачи и выбора оптимальный способ вычисления	Удовлетворительно владеет навыком решения задачи и выбора оптимальный способ вычисления	Хорошо владеет навыком решения задачи и выбора оптимальный способ вычисления	Отлично владеет навыком решения задачи и выбора оптимальный способ вычисления
ПКС-1 Способность осуществлять и корректировать технологические процессы нефтегазового производства в соответствии с выбранной сферой профессиональной деятельности	ПКС-1.1 Осуществляет выбор и систематизацию информации о технологических процессах нефтегазового производства	Знать: способы сбора и анализа исходных данных о технологических процессах нефтегазового производства (ПКС-1.З1)	Не знает способы сбора и анализа исходных данных о технологических процессах нефтегазового производства	Удовлетворительно знает способы сбора и анализа исходных данных о технологических процессах нефтегазового производства	Хорошо знает способы сбора и анализа исходных данных о технологических процессах нефтегазового производства	Отлично знает способы сбора и анализа исходных данных о технологических процессах нефтегазового производства
		Уметь: осуществлять выбор и систематизацию информации о технологических процессах нефтегазового производства (ПКС-1.У1)	Не умеет осуществлять выбор и систематизацию информации о технологических процессах нефтегазового производства	Удовлетворительно умеет осуществлять выбор и систематизацию информации о технологических процессах нефтегазового производства	Хорошо умеет осуществлять выбор и систематизацию информации о технологических процессах нефтегазового производства	Отлично умеет осуществлять выбор и систематизацию информации о технологических процессах нефтегазового производства

		Владеть: навыками выбора и систематизации информации о технологических процессах нефтегазового производства (ПКС-1.В1)	Не владеет навыками выбора и систематизации информации о технологических процессах нефтегазового производства	Удовлетворительно владеет навыками выбора и систематизации информации о технологических процессах нефтегазового производства	Хорошо владеет навыками выбора и систематизации информации о технологических процессах нефтегазового производства	Отлично владеет навыками выбора и систематизации информации о технологических процессах нефтегазового производства

Приложение 2

КАРТА

обеспеченности дисциплины учебной и учебно-методической литературой

Дисциплина/модуль Математика и Python для анализа данных

Код, направление подготовки 21.03.01 Нефтегазовое дело

Направленность (профиль) Эксплуатация и обслуживание объектов добычи нефти

№ п/п	Название учебного, учебно-методического издания, автор, издательство, вид издания, год издания	Количество экземпляров в БИК	Контингент обучающихся, использующих литературу	Обеспеченность обучающихся литературой, %	Наличие электронного варианта в ЭБС (+/-)
1	Федоров, Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для вузов / Д. Ю. Федоров. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2022. — 210 с. — (Высшее образование). — ISBN 978-5-534-14638-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/492920	Электр. вариант	150	100	+
2	Маккинни, У. Python и анализ данных / У. Маккинни ; перевод с английского А. А. Слинкина. — 2-ое изд., испр. и доп. — Москва : ДМК Пресс, 2020. — 540 с. — ISBN 978-5-97060-590-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131721	Электр. вариант	150	100	+
3	Бонцанини, М. Анализ социальных медиа на Python. Извлекайте и анализируйте данные из всех уголков социальной паутины на Python / М. Бонцанини ; перевод с английского А. В. Логунова. — Москва : ДМК Пресс, 2018. — 288 с. — ISBN 978-5-97060-574-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/108129	Электр. вариант	150	100	+

4	Программные системы статистического анализа. Обнаружение закономерностей в данных с использованием системы R и языка Python : учебное пособие / В. М. Волкова, М. А. Семёнова, Е. С. Четвертакова, С. С. Вожов. — Новосибирск : НГТУ, 2017. — 74 с. — ISBN 978-5-7782-3183-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/118287	Электр. вариант	150	100	+

Заведующий кафедрой ЕНГД _____ Л.К. Иляшенко

Дополнения и изменения
к рабочей программе дисциплины (модуля)

_____

на 20_ - 20_ учебный год

В рабочую программу вносятся следующие дополнения (изменения):

_____

_____

_____

_____

_____

Дополнения и изменения внес:

_____ _____

(должность, ученое звание, степень) *(подпись)* *(И.О. Фамилия)*

Дополнения (изменения) в рабочую программу рассмотрены и одобрены на заседании кафедры _____

(наименование кафедры)

Протокол от «____» _____20__ г. № _____

Заведующий кафедрой_____ И.О. Фамилия

СОГЛАСОВАНО:

Заведующий выпускающей кафедрой/
Руководить образовательной программы _____ И.О. Фамилия

«_____» _____ 20___ г.