МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет информатики и информационных технологий

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ
Технологии и методы программирования

Кафедра информационных систем и технологий программирования

Образовательная программа
10.03.01 Информационная безопасность

Направленность (профиль) программы:
Безопасность компьютерных систем

Уровень высшего образования
бакалавриат

Форма обучения
Очная

Статус дисциплины: входит в часть ОПОП, формируемую участниками образовательных отношений

Махачкала, 2022

Рабочая программа дисциплины “Технологии и методы программирования” составлена в 2021 году в соответствии с требованиями ФГОС ВО - бакалавриат по направлению подготовки 10.03.01 Информационная безопасность от 17 ноября 2020 г №1427 .

Разработчик (и): кафедра информационных систем и технологий программирования, доц. Баммаева Г.А.

Рабочая программа дисциплины одобрена:
на заседании кафедры ИСиТП от «1» марта 2022г., протокол № 8
Зав. кафедрой _____ Исмиханов З.Н.
(подпись)

на заседании Методической комиссии факультета ИиИТ
от «17» марта 2022г., протокол № 7.

Председатель _____ Бакмаев А.Ш.
(подпись)

Рабочая программа дисциплины согласована с учебно-методическим управлением «31» марта 2022г.

Начальник УМУ _____ Гасангаджиева А.Г.
(подпись)

Аннотация рабочей программы дисциплины

Дисциплина «Технологии и методы программирования» входит в часть формируемую участниками образовательных отношений по направлению подготовки 10.03.01 Информационная безопасность.

Дисциплина реализуется на факультете информатики и информационных технологий кафедрой информационных систем и технологий программирования.

Содержание дисциплины охватывает круг вопросов, связанных с формированием алгоритмического мышления у студентов, объектно-ориентированным программированием, созданием консольных и графических приложений.

Дисциплина нацелена на формирование следующих компетенций выпускника: общепрофессиональных - ОПК-3, профессиональных - ПК-1, ПК-6.

Преподавание дисциплины предусматривает проведение следующих видов учебных занятий: лекции, практические занятия, лабораторные занятия, самостоятельная работа.

Рабочая программа дисциплины предусматривает проведение следующих видов контроля успеваемости в форме контрольных работ, коллоквиума и промежуточный контроль в форме экзамена в пятом семестре.

Объем дисциплины 8 зачетных единиц, в том числе в академических часах по видам учебных занятий:

Очная форма обучения

Семестр	Учебные занятия								Форма промежуточной аттестации (зачет, дифференцированный зачет, экзамен)
	в том числе:								
	всего	Контактная работа обучающихся с преподавателем						СРС, в том числе экзамен	
		всего	из них						
			Лекции	Лабораторные занятия	Практические занятия	КСР	консультации		
4	108	66	34	16	16			42	
5	180	74	30	30	14			70+36	Экзамен
Итого	266	140	64	46	30			148	

1. Цели освоения дисциплины

Целями освоения дисциплины (модуля) «Технологии и методы программирования»являются овладение знаниями в области технологии программирования; подготовка к осознанному использованию, как языков программирования, так и методов программирования. Формирование у студентов научного, творческого подхода к освоению технологий, методов и средств производства программного обеспечения. Получение необходимых знаний, умений и навыков в области применения современной вычислительной техники для решения практических задач обработки данных, математического моделирования, информатики, получение высшего профессионального (на уровне бакалавра) образования, позволяющего выпускнику успешно работать в избранной сфере деятельности с применением современных компьютерных технологий.

2. Место дисциплины в структуре ОПОП бакалавриата

Дисциплина «Технологии и методы программирования» входит в часть, формируемую участниками образовательных отношений ОПОП бакалавриата по направлению подготовки 10.03.01 Информационная безопасность.

(в разделе дается характеристика места дисциплины в структуре ОПОП:

- приводится перечень дисциплин (или их разделов), необходимых для изучения даннойдисциплины;

- приводится перечень дисциплин (или их разделов), использующих результаты изученияданной дисциплины).

3. Компетенции обучающегося, формируемые в результате освоения дисциплины (перечень планируемых результатов обучения и процедура освоения).

Код и наименование компетенции из ОПОП	**Код и наименование индикатора достижения компетенций** *(в соответствии с ОПОП*	**Планируемые результаты обучения**	**Процедура освоения**
ОПК-3. Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	ОПК-3.1. Знает принципы, методы и средства решения стандартных задач профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.	Знает принципы, методы и средства решения стандартных задач профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.	Письменный опрос, контрольная работа, коллоквиум, тестирование.
	ОПК-3.2. Умеет решать	Умеет решать стандартные задачи	

	стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.	Умеет решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.	
	ОПК-3.3. Владеет навыками подготовки обзоров, аннотаций, составления рефератов, научных докладов, публикаций, и библиографии по научно-исследовательской работе с учетом требований информационной безопасности.	Владеет навыками подготовки обзоров, аннотаций, составления рефератов, научных докладов, публикаций, и библиографии по научно-исследовательской работе с учетом требований информационной безопасности.	
ПК-1. способностью выполнять работы по установке, настройке и обслуживанию программных, программно-аппаратных (в том числе криптографических) и технических средств защиты информации	ПК- 1.1. Знает методики обследования организаций, выявления информационных потребностей пользователей.	Знать: основные способы и режимы обработки экономической информации; методику обследования организаций, выявления информационных потребностей пользователей; формирования требований к информационной системе; классы ИС и особенности корпоративных ИС; типы объектов проектирования и их структуры, состав компонент технологии проектирования, классы	Письменный опрос, контрольная работа, коллоквиум, тестирование.

		технологий проектирования, методы и инструментальные средства проектирования; особенности жизненного цикла проекта ИС; состав проектной и регламентной документации; состав стадий и этапов проектирования ИС для предметной области; виды моделей и методов моделирования ИС и информационных технологий и средства моделирования ИС.	
	ПК- 1.2. Умеет анализировать предметную область, выявлять информационные потребности пользователей, формировать требования к ИС.	Уметь: проводить анализ информационных потребностей пользователей и формировать требования к информационной системе; анализировать предметную область и выявлять состав подразделений, выполняемые функции и задачи; исследовать объекты проектирования как системы; проводить декомпозицию системы и выделять компоненты систем на различных уровнях изучения; классифицировать и выбирать типы моделей и методы моделирования ИС; выделять стадии цикла жизни проекта ИС и их содержание.	
	ПК- 1.3 Владеет навыками работы с технологиями и программным инструментарием	Владеть: навыками работы с технологиями и программным инструментарием формирования	

	формирования требований к информационной системе	требований к информационной системе; навыками осуществления декомпозиции сложных экономических и организационных систем на макро и микро уровне, на уровне процессов управления и функционирования системы, а также на уровне происходящих в системе процессов.	
ПК-6. Способность программировать приложения и создавать программные прототипы решения прикладных задач.	ПК- 6.1. Знает основные сведения о методах и способах построения эффективных алгоритмов для решения прикладных задач.	Владеть: навыками моделирования прикладных (бизнес) процессов и предметной области, использовать CASE-средства	Письменный опрос, контрольная работа, коллоквиум, тестирование.
	ПК- 6.2. Умеет создавать программные прототипы решения задач предметной области.	Уметь: разрабатывать программные приложения для предметной области; производить анализ сложности алгоритма и находить пути упрощения полученных алгоритмов	
	ПК- 6.3. Владеет практическими навыками разработки программных прототипов решения прикладных задач.	Владеть: практическими навыками использования языков программирования для создания программные прототипов решения прикладных задач; основные и наиболее популярные программные продукты, позволяющие проектировать и разрабатывать алгоритмы.	

4. Объем, структура и содержание дисциплины.

Объем дисциплины составляет 8 зачетных единиц, 266 академическихчасов.

Структура дисциплины.

Структура дисциплины в очной форме

№	Разделы и темы дисциплины	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость(в часах)				Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)
				Лекции	Практические	Лабораторные	Самостоятельная работа	
	Семестр 4. Программирование на Python							
	Модуль 1. Основы программирования.							
1	Алгоритмизация задачи. Понятие алгоритма. Свойства алгоритма.	1	1	2	2	2	10	Опрос, тестирование, контрольная работа
2	Введение в Python. Ввод и вывод в Python. Переменные в Python. Оператор присваивания. Логический тип данных	1	2-6	6	2	2	10	Опрос, тестирование, контрольная работа
	Итого по модулю 1:	**36**	**1-6**	**8**	**4**	**4**	**20**	
	Модуль 2. Строки и списки							
3	Строки и операции над ними. Строки: индексы и срезы.	4	7-8	4	2	2	6	Опрос, тестирование, контрольная работа
4	Методы строк. Комментарии в коде. F-строки в Python.	4	9	4	2	2	6	Опрос, тестирование, контрольная работа
5	Списки и операции над ними. Списки: индексы и срезы.	4	10	2	2	2	2	Опрос, тестирование, контрольная работа
	Итого по модулю 2:	**36**	**7-9**	**10**	**6**	**6**	**14**	
	Модуль 3. Условия и циклы в Python							
6	Условный оператор. Вложенный оператор if. Множественный оператор elif.	4	11	2	2	2		Опрос, тестирование, контрольная работа
7	Цикл for. Функция range() с одним	4	12-14	4	2	2	2	Опрос, тестирование,

	параметром. Перегрузка range() с двумя параметрами. Перегрузка range() с 3 параметрами.							контрольная работа
8	Цикл while. Цикл for VS цикл while. Считывание данных до стоп значения	4	15-16	4			2	Опрос, тестирование, контрольная работа
9	Оператор прерывания цикла break. Бесконечные циклы. Оператор continue.	4	17	2			2	Опрос, тестирование, контрольная работа
10	Вложенные циклы. Операторы break и continue во вложенных циклах.	4	18	4	2	2	2	Опрос, тестирование, контрольная работа
	Итого по модулю 3:	**36**	**10-14**	**16**	**6**	**6**	**8**	
	Итого (4 сем)	**108**		**34**	**16**	**16**	**42**	
	Семестр 5.							
	Модуль 4. None, словари, множества и кортежи.							
1	Кортежи (tuple). Операции и методы кортежей. Вид данных кортеж (tuple)	4	1	2		2	8	Опрос, тестирование, контрольная работа
2	Словарь. Знакомство с типом данных dict. Операции со словарями. Методы словаря.	4	2-3	2		2	8	Опрос, тестирование, контрольная работа
3	Операции с множествами. Методы множеств. Неизменяемое множество frozenset	4	4-5	2		2	8	Опрос, тестирование, контрольная работа
	Итого по модулю 4:	**36**	**1-5**	**6**		**6**	**24**	
	Модуль 5. Функции в Python.							
4	Определение и вызов функции. Инструкция def. Возвращаемое значение функции. Оператор return	4	6-7	2		2	8	Опрос, тестирование, контрольная работа
5	Область видимости: локальная, глобальная и встроенная. Передача аргументов. Сопоставление	4	8	2		2	8	Опрос, тестирование, контрольная работа

	аргументов по имени и позиции							
6	Рекурсия в Python. Рекурсивная функция. Вложенные функции в Python	4	9	2		2	8	Опрос, тестирование, контрольная работа
	Итого по модулю 5:	**36**	**6-9**	**6**		**6**	**24**	
	Модуль 6. Работа с модулями.							
7	Установка модулей в Python.	4	10	2		2	6	Опрос, тестирование, контрольная работа
8	Импорт стандартных модулей	4	11	2	2	2	6	Опрос, тестирование, контрольная работа
9	Импорт собственных модулей в Python. Пакеты в Python. Файл __init__, переменная __all__	4	12	2	2	2	8	Опрос, тестирование, контрольная работа
	Итого по модулю 6:	**36**	**10-12**	**6**	**4**	**6**	**20**	
	Модуль 7. Работа с файлами							
10	Чтение и запись данных. Функция open. Контекстный менеджер	4	13-14	2	2	2	10	Опрос, тестирование, контрольная работа
11	Выражения-генераторы. Функция генератор. Создание генератора при помощи yield.	4	15	2	2	2	12	Опрос, тестирование, контрольная работа
	Итого по модулю 7:	**36**	**13-16**	**4**	**4**	**4**	**24**	
	Модуль 8. Подготовка к экзамену.							
	Подготовка и сдача экзамена.	4	**36**					
	ИТОГО 5 сем.:	**180**		**30**	**14**	**30**	**70 +36**	**Экзамен**
	ИТОГО:	**266**		**64**	**30**	**46**	**148**	

Содержание дисциплины, структурированное по темам (разделам).

Семестр 1. Программирование на Python. Модуль 1. Основы программирования.

Тема 1. Алгоритмизация задачи. Понятие алгоритма. Свойства алгоритма

Тема 2. Введение в Python.Ввод и вывод в Python. Переменные в Python. Оператор присваивания. Логический тип данных

Модуль 2. Строки и списки

Тема 3. Строки и операции над ними. Строки: индексы и срезы.

Тема 4. Методы строк. Комментарии в коде.F-строки в Python.
Тема 5. Списки и операции над ними. Списки: индексы и срезы.

Модуль 3. Условия и циклы в Python
Тема 6. Условный оператор. Вложенный оператор if. Множественный оператор elif.
Тема 7. Цикл for. Функция range() с одним параметром. Перегрузка range() с двумя параметрами. Перегрузка range() с 3 параметрами.
Тема 8. Цикл while. Цикл for VS цикл while. Считывание данных до стоп значения
Тема 9. Оператор прерывания цикла break. Бесконечные циклы. Оператор continue**.**
Тема 10. Вложенные циклы. Операторы break и continue во вложенных циклах.

Семестр 5.
Модуль 4. None, словари, множества и кортежи.
Тема 1. Кортежи (tuple). Операции и методы кортежей. Вид данных кортеж (tuple)
Тема 2. Словарь. Знакомство с типом данных dict. Операции со словарями. Методы словаря.
Тема 3. Операции с множествами. Методы множеств. Неизменяемое множество frozenset

Модуль 5. Функции в Python.
Тема 4. Определение и вызов функции. Инструкция def. Возвращаемое значение функции.
Тема 5. Область видимости: локальная, глобальная и встроенная. Передача аргументов. Сопоставление аргументов по имени и позиции
Тема 6. Рекурсия в Python. Рекурсивная функция. Вложенные функции в Python

Модуль 6. Функции в Python.
Тема 7. Установка модулей в Python.
Тема 8. Импорт стандартных модулей
Тема 9. Импорт собственных модулей в Python. Пакеты в Python. Файл __init__, переменная __all__

Модуль 7. Работа с файлами.
Тема 10. Чтение и запись данных. Функция open. Контекстный менеджер
Тема 11. Выражения-генераторы. Функция генератор. Создание генератора при помощи yield.

Темы лабораторных занятий

Лабораторные и практические занятия проводятся в специально оборудованных помещениях - компьютерных классах, где установлено необходимое программное обеспечение.

Ниже приведены темы лабораторных и практических занятий занятий:

Лабораторная работа №1.
Тема: Ввод и вывод данных в Python

Python 3 - это современный язык, на котором просто и приятно писать программы.

Для печати значений в Питоне есть функция print(). Внутри круглых скобок через запятую мы пишем то, что хотим вывести. Вот программа, которая делает несколько вычислений:

```
print(5 + 10)
print(3 * 7, (17 - 2) * 8)
print(2 ** 16)  # две звёздочки означают возведение в степень
print(37 / 3)  # один слэш - это деление с ответом-дробью
print(37 // 3)  # два слэша считают частное от деления нацело
          # это как операция div в других языках
print(37 % 3)  # процент считает остаток от деления нацело
          # это как операция mod в других языках
```

Для ввода данных в программу мы используем функцию `input()`. Она считывает одну строку. Вот программа, которая считывает имя пользователя и приветствует его:

```
print('Как вас зовут?')
name = input()  # считываем строку и кладём её в переменную name
print('Здравствуйте, ' + name + '!')
```

Мы будем писать программы, которые считывают данные, перерабатывают их и выводят какой-то результат. При запуске на компьютере такие программы считывают данные, которые пользователь вводит с клавиатуры, а результат выводят на экран.

Попробуем написать программу, которая считывает два числа и выводит их сумму. Для этого считаем два числа и сохраним их в переменные `a` и `b`, пользуясь оператором присваивания `=`. Слева от оператора присваивания в программах на Питоне ставится имя переменной - например, строка из латинских букв. Справа от оператора присваивания ставится любое выражение. Имя станет указывать на результат вычисления выражения. Проиграйте эту программу и посмотрите на результаты её работы:

```
a = input()
b = input()
s = a + b
print(s)
```

Мы видим, что программа выводит `57`, хотя в реальной жизни `5 + 7` будет `12`. Это произошло потому, что Питон в третьей строчке «сложил» две строки, а не два числа. В Питоне две строки складываются так: к первой строке приписывается вторая.

Обратите внимание, что в визуализаторе содержимое переменных `a` и `b` заключено в кавычки. Это означает, что в `a` и `b` лежат строки, а не числа.

В Питоне все данные называются объектами. Число 2 представляется объектом «число 2», строка `'hello'` – это объект «строка `'hello'`».

Каждый объект относится к какому-то типу. Строки хранятся в объектах типа `str`, целые числа хранятся в объектах типа `int`, дробные числа (вещественные числа) - в объектах типа `float`. Тип объекта определяет, какие действия можно делать с объектами этого типа. Например, если в переменных `first` и `second` лежат объекты типа `int`, то их можно перемножить, а если в них лежат объекты типа `str`, то их перемножить нельзя:

```
first = 5
second = 7
print(first * second)
first = '5'
second = '7'
print(first * second)
```

Чтобы преобразовать строку из цифр в целое число, воспользуемся функцией int(). Например, int('23') вернет число 23.

Вот пример правильной программы, которая считывает два числа и выводит их сумму:

```
a = int(input())
b = int(input())
s = a + b
print(s)
```

Лабораторная работа №2
Условие в Python

Все ранее рассматриваемые программы имели линейную структуру: все инструкции выполнялись последовательно одна за одной, каждая записанная инструкция обязательно выполняется.

Допустим мы хотим по данному числу x определить его абсолютную величину (модуль). Программа должна напечатать значение переменной x, если x>0 или же величину -x в противном случае. Линейная структура программы нарушается: в зависимости от справедливости условия x>0 должна быть выведена одна или другая величина. Соответствующий фрагмент программы на Питоне имеет вид:

```
x = int(input())
if x > 0:
    print(x)
else:
    print(-x)
```

В этой программе используется условная инструкция if (если). После слова if указывается проверяемое условие (x > 0), завершающееся двоеточием. После этого идет блок (последовательность) инструкций, который будет выполнен, если условие истинно, в нашем примере это вывод на экран величины x. Затем идет слово else (иначе), также завершающееся двоеточием, и блок инструкций, который будет выполнен, если проверяемое условие неверно, в данном случае будет выведено значение -x.

Итак, условная инструкция в Питоне имеет следующий синтаксис:

if *Условие*:
    Блок инструкций 1
else:
    Блок инструкций 2

Блок инструкций 1 будет выполнен, если *Условие* истинно. Если *Условие* ложно, будет выполнен *Блок инструкций 2*.

В условной инструкции может отсутствовать слово else и последующий блок. Такая инструкция называется неполным ветвлением. Например, если дано число x и мы хотим заменить его на абсолютную величину x, то это можно сделать следующим образом:

```
x = int(input())
if x < 0:
    x = -x
print(x)
```

В этом примере переменной x будет присвоено значение -x, но только в том случае, когда x<0. А вот инструкция print(x) будет выполнена всегда, независимо от проверяемого условия.

Для выделения блока инструкций, относящихся к инструкции if или else в языке Питон используются отступы. Все инструкции, которые относятся к одному блоку, должны иметь равную величину отступа, то есть одинаковое число пробелов в начале строки. Рекомендуется использовать отступ в 4 пробела и не рекомедуется использовать в качестве отступа символ табуляции.

Это одно из существенных отличий синтаксиса Питона от синтаксиса большинства языков, в которых блоки выделяются специальными словами, например, нц... кц в Кумире, begin... end в Паскале или фигурными скобками в Си.

2. Вложенные условные инструкции

Внутри условных инструкций можно использовать любые инструкции языка Питон, в том числе и условную инструкцию. Получаем вложенное ветвление – после одной развилки в ходе исполнения программы появляется другая развилка. При этом вложенные блоки имеют больший размер отступа (например, 8 пробелов). Покажем это на примере программы, которая по данным ненулевым числам x и y определяет, в какой из четвертей координатной плоскости находится точка (x,y):

```
x = int(input())
y = int(input())
if x > 0:
```

```
        if y > 0:              # x > 0, y > 0
            print("Первая четверть")
        else:                  # x > 0, y < 0
            print("Четвертая четверть")
    else:
        if y > 0:              # x < 0, y > 0
            print("Вторая четверть")
        else:                  # x < 0, y < 0
            print("Третья четверть")
```

В этом примере мы использовали *комментарии* – текст, который интерпретатор игнорирует. Комментариями в Питоне является символ `#` и весь текст после этого символа до конца строки.

3. Операторы сравнения

Как правило, в качестве проверяемого условия используется результат вычисления одного из следующих операторов сравнения:

`<`

Меньше — условие верно, если первый операнд меньше второго.

`>`

Больше — условие верно, если первый операнд больше второго.

`<=`

Меньше или равно.

`>=`

Больше или равно.

`==`

Равенство. Условие верно, если два операнда равны.

`!=`

Неравенство. Условие верно, если два операнда неравны.

Например, условие `(x * x < 1000)` означает "значение `x * x` меньше 1000", а условие `(2 * x != y)` означает "удвоенное значение переменной `x` не равно значению переменной `y`".

Операторы сравнения в Питоне можно объединять в цепочки (в отличии от большинства других языков программирования, где для этого нужно использовать логические связки), например, `a == b == c` или `1 <= x <= 10`.

4. Тип данных bool

Операторы сравнения возвращают значения специального логического типа `bool`. Значения логического типа могут принимать одно из двух значений: `True` (истина) или `False` (ложь). Если преобразовать логическое `True` к типу `int`, то получится 1, а преобразование `False` даст 0. При обратном преобразовании число 0 преобразуется в `False`, а любое ненулевое число в `True`. При преобразовании `str` в `bool` пустая строка преобразовывается в `False`, а любая непустая строка в `True`.

4.1. Логические операторы

Иногда нужно проверить одновременно не одно, а несколько условий. Например, проверить, является ли данное число четным можно при помощи условия `(n % 2 == 0)` (остаток от деления `n` на `2` равен `0`), а если необходимо проверить, что два данных целых числа `n` и `m` являются четными, необходимо проверить справедливость обоих условий: `n % 2 == 0` и `m % 2 == 0`, для чего их необходимо объединить при помощи оператора `and` (логическое И): `n % 2 == 0 and m % 2 == 0`.

В Питоне существуют стандартные логические операторы: логическое И, логическое ИЛИ, логическое отрицание.

Логическое И является бинарным оператором (то есть оператором с двумя операндами: левым и правым) и имеет вид `and`. Оператор `and` возвращает `True` тогда и только тогда, когда оба его операнда имеют значение `True`.

Логическое ИЛИ является бинарным оператором и возвращает `True` тогда и только тогда, когда хотя бы один операнд равен `True`. Оператор “логическое ИЛИ” имеет вид `or`.

Логическое НЕ (отрицание) является унарным (то есть с одним операндом) оператором и имеет вид `not`, за которым следует единственный операнд. Логическое НЕ возвращает `True`, если операнд равен `False` и наоборот.

Пример. Проверим, что хотя бы одно из чисел a или b оканчивается на 0:

```
a = int(input())
b = int(input())
if a % 10 == 0 or b % 10 == 0:
    print('YES')
else:
    print('NO')
```

Проверим, что число a — положительное, а b — неотрицательное:

```
if a > 0 and not (b < 0):
```

Или можно вместо `not (b < 0)` записать `(b >= 0)`.

5. Каскадные условные инструкции

Пример программы, определяющий четверть координатной плоскости, можно переписать используя “каскадную“ последовательность операцией `if... elif... else`:

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
    print("Первая четверть")
elif x > 0 and y < 0:
    print("Четвертая четверть")
elif y > 0:
    print("Вторая четверть")
else:
    print("Третья четверть")
```

такой конструкции условия `if`, ..., `elif` проверяются по очереди, выполняется блок, соответствующий первому из истинных условий. Если все проверяемые условия ложны, то выполняется блок `else`, если он присутствует.

Команда if.

Самая простая команда if состоит из одного условия и одного действия.

```
if условие:
    действие  # отступ в 4 пробела
```

Лабораторная работа 3
Цикл while

1. Цикл while

Цикл while (“пока”) позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Условие записывается до тела цикла и проверяется до выполнения тела цикла. Как правило, цикл whileиспользуется, когда невозможно определить точное значение количества проходов исполнения цикла.

Синтаксис цикла while в простейшем случае выглядит так:

```
while условие:
    блок инструкций
```

При выполнении цикла while сначала проверяется условие. Если оно ложно, то выполнение цикла прекращается и управление передается на следующую инструкцию после тела цикла while. Если условие истинно, то выполняется инструкция, после чего условие

проверяется снова и снова выполняется инструкция. Так продолжается до тех пор, пока условие будет истинно. Как только условие станет ложно, работа цикла завершится и управление передастся следующей инструкции после цикла.

Например, следующий фрагмент программы напечатает на экран квадраты всех целых чисел от 1 до 10. Видно, что цикл while может заменять цикл for ... in range(...):

```
i = 1
while i <= 10:
    print(i ** 2)
    i += 1
```

В этом примере переменная i внутри цикла изменяется от 1 до 10. Такая переменная, значение которой меняется с каждым новым проходом цикла, называется счетчиком. Заметим, что после выполнения этого фрагмента значение переменной i будет равно 11, поскольку именно при i == 11условие i <= 10 впервые перестанет выполняться.

Вот еще один пример использования цикла while для определения количества цифр натурального числа n:

```
n = int(input())
length = 0
while n > 0:
    n //= 10  # это эквивалентно n = n // 10
    length += 1
print(length)
```

В этом цикле мы отбрасываем по одной цифре числа, начиная с конца, что эквивалентно целочисленному делению на 10 (n //= 10), при этом считаем в переменной length, сколько раз это было сделано.

В языке Питон есть и другой способ решения этой задачи:length = len(str(i)).

2. Инструкции управления циклом

После тела цикла можно написать слово else: и после него блок операций, который будет выполнен *один раз* после окончания цикла, когда проверяемое условие станет неверно:

```
i = 1
while i <= 10:
    print(i)
    i += 1
else:
    print('Цикл окончен, i =', i)
```

Казалось бы, никакого смысла в этом нет, ведь эту же инструкцию можно просто написать *после* окончания цикла. Смысл появляется только вместе с инструкцией break. Если во время выполнения Питон встречает инструкцию break внутри цикла, то он сразу же прекращает выполнение этого цикла и выходит из него. При этом ветка else исполняться не будет. Разумеется, инструкцию break осмыленно вызывать только внутри инструкции if, то есть она должна выполняться только при выполнении какого-то особенного условия.

Приведем пример программы, которая считывает числа до тех пор, пока не встретит отрицательное число. При появлении отрицательного числа программа завершается. В первом варианте последовательность чисел завершается числом 0 (при считывании которого надо остановиться).

```
a = int(input())
while a != 0:
    if a < 0:
        print('Встретилось отрицательное число', a)
        break
    a = int(input())
else:
```

```
    print('Ни одного отрицательного числа не встретилось')
```

Во втором варианте программы сначала на вход подается количество элементов последовательности, а затем и сами элементы. В таком случае удобно воспользоваться циклом for. Цикл for также может иметь ветку else и содержать инструкции break внутри себя.

```
n = int(input())
for i in range(n):
    a = int(input())
    if a < 0:
        print('Встретилось отрицательное число', a)
        break
else:
    print('Ни одного отрицательного числа не встретилось')
```

Другая инструкция управления циклом — continue (продолжение цикла). Если эта инструкция встречается где-то посередине цикла, то пропускаются все оставшиеся инструкции до конца цикла, и исполнение цикла продолжается со следующей итерации
:

```
for i in range(3):
    for j in range(5):
        if j > i:
            break
        print(i, j)
```

Увлечение инструкциями break и continue не поощряется, если можно обойтись без их использования. Вот типичный пример *плохого использования*инструкции break (данный код считает количество знаков в числе).

```
n = int(input())
length = 0
while True:
    length += 1
    n //= 10
    if n == 0:
        break
print('Длина числа равна', length)
```

Гораздо лучше переписать этот цикл так:

```
n = int(input())
length = 0
while n != 0:
    length += 1
    n //= 10
print('Длина числа равна', length)
```

Впрочем, на Питоне можно предложить и более изящное решение:

```
n = int(input())
print('Длина числа равна', len(str(n)))
```

Лабораторная работа 4:

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.

5. Образовательные технологии

Процесс изложения учебного материала сопровождается систематическими (на каждом занятии) компьютерными презентациями и демонстрацией решения задач в интерактивном режиме с использованием мультимедийного оборудования, предусмотрено широкое использование в учебном процессе активных и интерактивных форм проведения занятий.

Предусмотрено общение и консультации с представителями российских и зарубежных компаний (из числа выпускников кафедры) как по электронной почте и скайпу, так и очные встречи.

Для эффективной реализации целей и задач ФГОС, для претворения компетентностного подхода в преподавании дисциплины «Технологии программирования и работы на ЭВМ»,используются следующие образовательные технологии и методы обучения:

Вид занятия	Технология	Цель	Формы и методы обучения
1	2	3	4

Лекции	Технология проблемного обучения.	Усвоение теоретических знаний, развитие мышления, формирование профессионального интереса к будущей деятельности.	Мультимедийные лекция-объяснение, лекция-визуализация, с привлечением формы тематической дискуссии, беседы, анализа конкретных ситуаций
Практические занятия	Технология проблемного, модульного, дифференцированного и активного обучения, деловая игра.	Развитие творческой и познавательной самостоятельности, обеспечение индивидуального подхода с учетом базовой подготовки.	Индивидуальный темп обучения. Инновационные интерактивные методы в обучении: использование Webресурсов для подготовки компьютерных презентаций, использование offline (электронная почта) для обмена информацией, консультаций с преподавателем, работа с электронными пособиями, возможность самотестирования
Лабораторные занятия	Технология проблемного, модульного, дифференцированного и активного обучения, деловая игра.	Организация активности студентов, обеспечение личностно деятельного характера усвоения знаний, приобретения навыков, умений.	Инновационные интерактивные методы в обучении. Постановка проблемных познавательных задач. Методы активного обучения: «круглый стол», игровое производственное.

Самостоятельная работа	Технологии концентрированного, модульного, дифференцированного обучения.	Развитие познавательной самостоятельности, обеспечение гибкости обучения, развитие навыков работы с различными источниками информации, развитие умений, творческих навыков.	Индивидуальные, групповые, интерактивные (в режимах online и offline).

6. Учебно-методическое обеспечение самостоятельной работы студентов

Виды и порядок выполнения самостоятельной работы.

1. Изучение конспектов лекций и рекомендованной литературы.
2. Подготовка к опросу на практических занятиях.
3. Решение задач и упражнений.
4. Подготовка к коллоквиуму и контрольным работам.
5. Поиск материала на интернет-форумах.
6. Подготовка к экзамену Примерное распределение времени самостоятельной работы студентов.

Вид самостоятельной работы	Примерная трудоёмкость, в.ч.	Формируемые компетенции
	очная	
Текущая СРС		
работа с лекционным материалом, с учебной литературой	14	ОПК-3 ПК-1, ПК-6.
опережающая самостоятельная работа (изучение нового материала до его изложения на занятиях)	14	ОПК-3 ПК-1, ПК-6.
самостоятельное изучение разделов дисциплины	14	ОПК-3
выполнение домашних заданий, домашних контрольных работ	14	ОПК-3 ПК-1, ПК-6.
подготовка к лабораторным работам, к практическим и семинарским занятиям	14	ОПК-3 ПК-1, ПК-6.
подготовка к контрольным работам, коллоквиумам, зачётам	36	ОПК-3 ПК-1, ПК-6.
Творческая проблемно-ориентированная СРС		
выполнение расчётно-графических работ	10	ОПК-3 ПК-1, ПК-6.
поиск, изучение и презентация информации по заданной проблеме, анализ научных публикаций по заданной теме	10	ОПК-3 ПК-1, ПК-6.
исследовательская работа, участие в конференциях, семинарах, олимпиадах	10	ОПК-3 ПК-1, ПК-6.
анализ данных по заданной теме, выполнение расчётов, составление схем и моделей на основе собранных данных	10	ОПК-3 ПК-1, ПК-6.
Итого СРС:	***144***	

Порядок контроля:

1. Опрос на лабораторном занятии
2. Проверка выполнения домашних заданий и контрольных работ
3. Коллоквиум

4. Зачет.

Раздел (модуль, тема)	Вид самостоятельной работы - практическое содержание	Контрольные сроки (в нед.) и вид контроля	Уч.-мет. обеспечение (указаны источники из списка основной литературы)
1	2	3	4
4 семестр. Модуль 1. Основы программирования. Модуль 2. Строки и списки Модуль 3. Условия и циклы в Python	Строки и операции над ними. Строки: индексы и срезы. Методы строк. Комментарии в коде. F-строки в Python Списки и операции над ними. Списки: индексы и срезы. Условный оператор. Вложенный оператор if. Множественный оператор elif. Цикл for. Функция range() с одним параметром. Перегрузка range() с двумя параметрами. Перегрузка range() с 3 параметрами.	Контрольные сроки (в нед.) и вид контроля	материалы сайтов: Программа курса · Инди-курс программирования на Python · Stepik Программа курса · "Поколение Python": курс для начинающих · Stepik
5 семестр. Модуль 4. None, словари, множества и кортежи Модуль 5. Функции в Python. Модуль 6. Работа с модулями. Модуль 7. Работа с файлами	Кортежи (tuple). Операции и методы кортежей. Вид данных кортеж (tuple). Словарь. Знакомство с типом данных dict. Операции со словарями. Методы словаря.	8 и 17 недели обучения-контрольные работы. Проверка решенных задач. Проверка выполнения	материалы сайтов: Программа курса · Инди-курс программирования на Python · Stepik Программа курса · "Поколение Python": курс для начинающих · Stepik

	Списки и операции над ними. Списки: индексы и срезы. Определение и вызов функции. Инструкция def. Возвращаемое значение функции. Оператор return. Область видимости: локальная, глобальная и встроенная. Передача аргументов. Сопоставление аргументов по имени и позиции. Установка модулей в Python.		

Для обеспечения самостоятельной работы используется разработанный на кафедре пакет заданий и методических указаний. Самостоятельная работа студентов складывается из проработки лекционного материала и материалов форумов программирования, материала учебника, решения всех заданий из индивидуальных вариантов, решения рекомендуемых задач.

Примеры индивидуальных вариантов задач для самостоятельного выполнения:

Вариант 1

1. Выведите на экран все положительные делители натурального числа, введѐнного пользователем с клавиатуры.

2. Создайте два массива из 10 целых случайных чисел из отрезка [1;9] и третий массив из 10 действительных чисел. Каждый элемент с i-ым индексом третьего массива должен равняться отношению элемента из первого массива с i-ым индексом к элементу из второго массива с i-ым индексом. Вывести все три массива на экран (каждый на отдельной строке), затем вывести количество целых элементов в третьем массиве.

3. Создайте класс прямоугольников, описав в нѐм все необходимые свойства, подобрав им понятные имена и правильные типы данных.

Опишите в классе конструктор, позволяющий при создании нового объекта явно задать все его свойства. Если это необходимо, то проверьте допустимость их значений в конструкторе (например, в классе обыкновенных дробей нельзя создавать дробь с нулевым знаменателем). Создайте в классе метод, проверяющий равны ли два прямоугольника по площади. С использованием построенного класса создайте один прямоугольник со сторонами 3 и 8 и второй прямоугольник со сторонами 6 и 4. Проверьте с помощью созданного метода равны ли прямоугольники по площади и если да, то выведите соответствующее сообщение на экран.

Вариант 2

1. Выведите на экран все двузначные члены последовательности 2an-1+50, где a1=- 26.
2. Создайте массив из 11 случайных целых чисел из отрезка [-1;1], выведите массив на экран в строку. Определите какой элемент встречается в массиве чаще всего и выведите об этом сообщение на экран. Если два каких-то элемента встречаются одинаковое количество раз, то не выводите ничего.
3. Создайте класс углов отложенных против часовой стрелки от положительного направления оси абсцисс, описав в нѐм все необходимые свойства, подобрав им понятные имена и правильные типы данных.

Опишите в классе конструктор, позволяющий при создании нового объекта явно задать

все его свойства. Если это необходимо, то проверьте допустимость их значений в конструкторе (например, в классе обыкновенных дробей нельзя создавать дробь с нулевым знаменателем). Создайте в классе метод, проверяющий задают ли углы перпендикулярные прямые. С использованием построенного класса создайте угол в 10° и второй угол в 280°. Проверьте с помощью созданного метода задают ли углы перпендикулярные прямые и если да, то выведите соответствующее сообщение на экран.

Вариант 3

1. Создать программу, которая будет проверять попало ли случайно выбранное из отрезка [20;160] целое число в интервал (55;120) и сообщать результат на экран.

2. Пользователь вводит с клавиатуры натуральное число большее 3, которое сохраняется в переменную n. Если пользователь ввèл не подходящее число, то программа должна просить пользователя повторить ввод. Создать массив из n случайных целых чисел из отрезка [0;n] и вывести его на экран. Создать второй массив только из чѐтных элементов первого массива, если они там есть, и вывести его на экран.

3. Создайте класс прямоугольных треугольников, описав в нѐм все необходимые свойства, подобрав им понятные имена и правильные типы данных.

Опишите в классе конструктор, позволяющий при создании нового объекта явно задать все его свойства. Если это необходимо, то проверьте допустимость их значений в конструкторе (например, в классе обыкновенных дробей нельзя создавать дробь с нулевым знаменателем). Создайте в классе метод, вычисляющий длину высоты, опущенной на гипотенузу. С использованием построенного класса создайте треугольник с катетами 3 и 4. Вычислите с помощью метода и выведите на экран длину высоты опущенной на гипотенузу.

Вариант 4

1. Создайте программу, выводящую на экран первые 20 элементовпоследовательности 2 4 8 16 32 64 128 ….

2. Создать двумерный массив из 8 строк по 5 столбцов в каждой из случайных целых чисел из отрезка [10;99]. Вывести массив на экран.

3. Создайте класс комплексных чисел, описав в нѐм все необходимые свойства, подобрав им понятные имена и правильные типы данных.

Опишите в классе конструктор, позволяющий при создании нового объекта явно задать все его свойства. Если это необходимо, то проверьте допустимость их значений в конструкторе (например, в классе обыкновенных дробей нельзя создавать дробь с нулевым знаменателем). Создайте в классе метод, проверяющий являются ли два комплексных числа сопряженными. С использованием построенного класса создайте два комплексных числа: 3i+1 и 2i-1. Проверьте с помощью созданного метода являются ли числа сопряженными и если да, то выведите соответствующее сообщение на экран.

7. Фонд оценочных средств для проведения текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины

7.1. Типовые контрольные задания

Примерный перечень вопросов к промежуточному контролю или экзамену по всему изучаемому курсу:

1. Алгоритмы перевода чисел из одной системы счисления в другие. Перевод чисел в системах счисления с кратными основаниями
2. Арифметические действия в двоичной системе счисления
3. Этапы решения задач. Понятие алгоритма
4. Свойства и формы записи алгоритмов
5. Линейные алгоритмы. Ветвления в алгоритмах
6. Циклические алгоритмы
7. История развития языков программирования
8. Движущие силы развития языков программирования
9. Классификация языков программирования
10. Основные понятия алгоритмических языков программирования
11. Встроенный тип str**.** Методы объекта str**.**

12. print() и форматирование вывода.
13. Работа с файловой системой средствами Python.
14. Работа с файлами. Методы open(), close(), read(), write().
15. Модуль re**.** Синтаксис регулярных выражений, метасимволы.
16. Встроенные типы последовательностей list, tuple, range и их методы.
17. Встроенный объект dict и его методы.
18. Встроенные типы чисел -int, float, complex. Машинное представление чисел с плавающей точкой и целых. Преобразование типов при сравнении чисел.
19. Двоичное представление чисел. Неассоциативность операций в арифметике с плавающей запятой. Целые числа с произвольной точностью.
20. Множества. Встроенные типы set и frozenset.
21. Инструкции и синтаксис. Составные конструкции и обработка исключений
22. Инструкции if/else/elif, логические операторы и выражения сравнения
23. Циклы while и for в Python
24. Функции в Python. Основные понятия
25. Области видимости и пространство имен в Python.
26. Передача аргументов в функцию. Специальные режимы сопоставления аргументов.
27. Парадигма объектно-ориентированного программирования. Поддержка в Python функционального программирования.
28. Объекты. Динамическая типизация. Инкапсуляция.
29. Генерация объекта class. Новое пространство имен. Объект экземпляр класса.
30. Атрибуты класса. Атрибуты данных. Атрибуты-методы. Параметр self. Добавление атрибутов к классу во время исполнения программы.
31. Специальные методы и атрибуты классов. Методы __init__() и ___del__() в Python. Декораторы функций и декораторы классов. Инструменты интроспекции в Python. Метаклассы.
32. Абстрактные методы в Python. Классические классы и классы нового стиля.
33. Наследование. Базовый и производный класс. Построение производного класса.
34. Порождающие функции (функции-фабрики). Множественное наследование. Примеси (Mix-in)
35. Агрегация. Контейнеры. Иерархия наследования.
36. Полиморфизм. Подмена методов в производном классе. Доступ к методам базового класса.
37. Специфика разработки программных средств.
38. Жизненный цикл программного средства.
39. Понятие качества программного средства.
40. Обеспечение надежности - основной мотив разработки программного средства. Методы борьбы со сложностью.
41. Обеспечение точности перевода.
42. Преодоление барьера между пользователем и разработчиком.
43. Обеспечение контроля правильности принимаемых решений.
44. Порядок разработки программного модуля.
45. Структурное программирование и пошаговая детализация.
46. Понятие о псевдокоде.
47. Контроль программного модуля.
48. Стратегия проектирования тестов. Заповеди отладки.
49. Автономная отладка и тестирование программного модуля.
50. Комплексная отладка и тестирование программного средства.
51. Документация, создаваемая в процессе разработки программных средств
52. Пользовательская документация программных средств.
53. Документация по сопровождению программных средств.
54. Назначение аттестации программного средства.
55. Испытания и оценка качества программного средства.

7.2. Темы рефератов:

1) Метод пошаговой детализации в программировании.
2) Концепции традиционного программирования.
3) Современные технологии программирования.
4) Технология структурного программирования.
5) Испытания и сертификация программных средств.
6) Документирование программных средств.
7) Тестирование программных средств.
8) Внутреннее проектирование и разработка программных средств.
9) Системный анализ и проектирование программных средств.
10) Жизненный цикл программных средств.
11) Сопровождение и конфигурационное управление ПС.
12) Технология сборочного программирования.
13) Экстремальное программирование.
14) Особенности современных методологий и технологий разработки ПС.
15) Направления развития и модели концепции открытых систем.
16) Технология объектно-ориентированного программирования
17) Технология применения CASE- систем.
18) Состав, структура и функциональные особенности CASE- средств.
19) Особенности и возможности Intranet-технологии.
20) Инструментальные средства создания Intranet-приложений.
21) Технологии параллельного программирования.
22) Компонентные технологии и разработка распределенного ПО.
23) Руководство программным проектом.
24) Рефакторинг программного обеспечения.
25) Быстрая разработка программного обеспечения.

7. Фонд оценочных средств для проведения текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины.

Типовые контрольные задания

Вариант 1

1. Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры. Найти максимальный элемент. Вывести массив на экран в обратном порядке.

2. В массиве действительных чисел все нулевые элементы заменить на среднее арифметическое всех элементов массива.

Вариант 2

1. Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры. Найти минимальный элемент. Вывести индекс минимального элемента на экран.

2. Дан массив целых чисел. Переписать все положительные элементы во второй массив, а остальные - в третий.

Вариант 3

1. В одномерном числовом массиве D длиной n вычислить сумму элементов с нечетными индексами. Вывести на экран массив D, полученную сумму.

2. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Вариант 4

1. Дан массив целых чисел. Найти максимальный элемент массива и его порядковый номер.

2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Вариант 5

1. Дан одномерный массив из 10 целых чисел. Вывести пары отрицательных чисел, стоящих рядом.

2. Дан целочисленный массив размера 10. Создать новый массив, удалив все одинаковые элементы, оставив их 1 раз.

Вариант 6

1. Дан одномерный массив из 10 целых чисел. Найти максимальный элемент и сравнить с ним остальные элементы. Вывести количество меньших максимального и больших максимального элемента.

2. Одномерный массив из 10-и целых чисел заполнить с клавиатуры, определить сумму тех чисел, которые >5.

Вариант 7

1. Дан массив целых чисел. Найти сумму элементов с четными номерами и произведение элементов с нечетными номерами. Вывести сумму и произведение.

2. Переставить в одномерном массиве минимальный элемент и максимальный.

Вариант 8

1. Найдите сумму и произведение элементов списка. Результаты вывести на экран.

2. В массиве действительных чисел все нулевые элементы заменить на среднее арифметическое всех элементов массива.

Вариант 9

1. Дан одномерный массив, состоящий из N вещественных элементов. Ввести массив с клавиатуры. Найти и вывести минимальный по модулю элемент. Вывести массив на экран в обратном порядке.

2. Даны массивы A и B одинакового размера 10. Вывести исходные массивы. Поменять местами их содержимое и вывести в начале элементы преобразованного массива A, а затем - элементы преобразованного массива B.

Вариант 10

1. Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран это значение, иначе сообщение об их отсутствии.

2. Дан одномерный массив из 15 элементов. Элементам массива меньше 10 присвоить нулевые значения, а элементам больше 20 присвоить 1. Вывести на экран монитора первоначальный и преобразованный массивы в строчку.

Вариант 11

1 Найти наибольший элемент списка, который делиться на 2 без остатка и вывести его на экран.

2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из четных чисел исходного массива, меньше 10, или сообщить, что таких чисел нет. Полученный массив вывести в порядке возрастания элементов.

Вариант 12

1. Найти наименьший нечетный элемент списка и вывести его на экран.

2. Даны массивы A и B одинакового размера 10. Поменять местами их содержимое и вывести вначале элементы преобразованного массива A, а затем - элементы преобразованного массива B.

Вариант 13

1. Дан одномерный массив целых чисел. Проверить, есть ли в нем одинаковые элементы. Вывести эти элементы и их индексы.

2. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Вариант 14

1. Найти максимальный элемент численного массива и поменять его местами с минимальным.

2. Программа заполняет одномерный массив из 10 целых чисел числами, считанными с клавиатуры. Определить среднее арифметическое всех чисел массива. Заменить элементы массива большие среднего арифметического на 1.

Вариант 15

1. Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран эти значения.

2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Методические материалы, определяющие процедуру оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Общий результат складывается из текущего контроля - 50% и промежуточного контроля -50%.
Текущий контроль по дисциплине включает:

- посещение занятий - 10 баллов,
- выполнение текущих практических заданий - 40 баллов,

- выполнение домашних (аудиторных) контрольных работ - 50 баллов.

Промежуточный контроль по дисциплине включает:

- устный опрос - 50 баллов,
- письменная контрольная работа - 50 баллов.

8. Перечень основной и дополнительной учебной литературы, необходимойдля освоения дисциплины

а) адрес сайта курса http://eor.dgu.ru/Files/20221006fiit66-22.pdf

б) основная литература:

1. Васильев А.Н. Python на примерах [Электронный ресурс]: практический курс по программированию/ Васильев А.Н.- Электрон. текстовые данные- Санкт-Петербург: Наука и Техника, 2017.- 432 с.- Режим доступа: http://www.iprbookshop.ru/73043.html.- ЭБС «IPRbooks» (дата обращения: 07.06.2022)

2. Сузи Р.А. Язык программирования Python [Электронный ресурс]: учебное пособие/ Сузи Р.А.- Электрон. текстовые данные- Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020.- 350 с.- Режим доступа: http://www.iprbookshop.ru/97589.html.- ЭБС «IPRbooks» (дата обращения: 07.06.2022)

3. Информатика [Электронный ресурс]: учебное пособие. - Электрон.текстовые данные. - Ставрополь: Северо-Кавказский федеральный университет, 2016. - 178 с. - 2227-8397. - Режим доступа: http://www.iprbookshop.ru/66024.html (дата обращения: 07.06.2022)

4. Никифоров С.Н. Информатика. Часть 3. Прикладное программирование [Электронный ресурс]: учебное пособие / С.Н. Никифоров. - Электрон.текстовые данные. - СПб.: Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2016. - 128 с. - 978-5-9227-0743-5. - Режим доступа: http://www.iprbookshop.ru/74384.html (дата обращения: 07.06.2022)

5. Прохорова О.В. Информатика [Электронный ресурс]: учебник / О.В. Прохорова. - Электрон.текстовые данные. - Самара: Самарский государственный архитектурно-строительный университет, ЭБС АСВ, 2013. - 106 с. - 978-5-9585-0539-5. - Режим доступа: http://www.iprbookshop.ru/20465.html (дата обращения: 07.06.2022)

6. Шелудько В.М. Основы программирования на языке высокого уровня Python [Электронный ресурс]: учебное пособие/ Шелудько В.М.- Электрон.текстовые данные- Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017.- 146 с.- Режим доступа: http://www.iprbookshop.ru/87461.html.- ЭБС «IPRbooks» (дата обращения: 07.06.2022)

в) дополнительная литература:

1. Роганов Е.А. Основы информатики и программирования [Электронный ресурс]/ Роганов Е.А.- Электрон. текстовые данные.- М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.- 392 с.- Режим па:http://www.iprbookshop.ru/73689.html .- ЭБС «IPRbooks»

2. Информатика : [учеб. для экон. специальностей вузов / Н.В.Макарова, Л.А.Матвеев, В.Л.Бройдо и др.]; под ред. Н.В.Макаровой. - 3-е изд., перераб. - М. : Финансы и статистика, 2007, 2005, 2001. - 765 с. : ил. ; 26 см. - Библиогр. в конце гл. - Предм. указ.: с. 748-758. - Рекомендовано МО РФ. - ISBN 978-5-279-02202-1 : 369-60. Местонахождение: Научная библиотека ДГУ.

3. Галатенко, Владимир Антонович. Основы информационной безопасности : учеб. пособие для студентов вузов, обуч. по специальности 351400 "Прикл. информ." / Галатенко, Владимир Антонович. - 4-е изд. - М. : Изд-во Интернет-Ун-та Информ. Технологий: БИНОМ. Лаб. знаний, 2016, 2008, 2006. - 205 с. - (Основы информационных технологий). - Рекомендовано УМО. - ISBN 978-5-94774-821-5 : 230-00. Местонахождение: Университетская библиотека

4. Гагарина, Лариса Геннадьевна. Технология разработки программного обеспечения : [учеб. пособие] / Гагарина, Лариса Геннадьевна, Е. В. Кокорева ; под ред. Л.Г.Гагариной. - М. : ФОРУМ: ИНФРА-М, 2009, 2008. - 399 с. - (Высшее образование). - Допущено УМО. - ISBN 978-5-8199-0342-1 (ИД "ФОРУМ") : 246-84.

9. Перечень рекомендуемых ресурсов информационно-телекоммуникационной сети «Интернет»

1. ЭБС IPRbooks: http://www.iprbookshop.ru/ Лицензионный договор № 2693/17от

02.10.2017г. об оказании услуг попредоставлению доступа. Доступ открыт с с 02.10.2021 г. до 02.10.2022 по подписке(доступ будет продлен).
2. Электронно-библиотечная система «Университетская библиотека лайн» www.biblioclub.ru договор № 55_02/16 от 30.03.2016 г. об оказании информационных услуг. (доступ продлен до сентября 2022 года).
3. Moodle [Электронныйресурс]: система виртуального обучением: [базаданных] / Даг. гос. ун-т. - Махачкала, г. - Доступ из сети ДГУ или, после регистрации из сети ун-та, из любой точки, имеющей доступ в интернет. - URL: http://moodle.dgu.ru/ (дата обращения: 22.03.2022).
4. Доступ к электронной библиотеки на http://elibrary.ru основании лицензионного соглашения между ФГБОУ ВПО ДГУ и «ООО» «Научная Электронная библиотека» от 15.10.2003. (Раз в 5 лет обновляется лицензионное соглашение)
5. Национальная электронная библиотека https://нэб.рф/. Договор 101/НЭБ/101/НЭБ/1597 от 1.08.2017г. Договор действует в течении 1года с момента его подписания.
6. Федеральный портал «Российское образование» http://www.edu.ru/ (единое окно доступа к образовательным ресурсам).
7. Федеральное хранилище «Единая коллекция цифровых образовательных ресурсов» http://school-collection.edu.ru/
8. Российский портал «Открытого образования» http://www.openet.edu.ru
9. Сайт образовательных ресурсов Даггосуниверситета http://edu.icc.dgu.ru
10. Информационные ресурсы научной библиотеки Даггосуниверситета http://elib.dgu.ru (доступ через платформу Научной электронной библиотеки elibrary.ru).
11. Федеральный центр образовательного законодательства http://www.lexed.ru
12. https://www.msu.ru/resources/msu-ws1.html#mm - доступ к ресурсам мехмата МГУ.
13. https://www.msu.ru/resources/msu-ws1.html#cm - доступ к ресурсам ВМК МГУ.

10. Методические указания для обучающихся по освоению дисциплины

1) Припроведении аттестации студентов важно всегда помнить, что систематичность, объективность, аргументированность - главные принципы, на которых основаны контроль и оценка знаний студентов. Проверка, контроль и оценка знаний студента, требуют учета его индивидуального стиля в осуществлении учебной деятельности. Знание критериев оценки знаний обязательно для преподавателя и студента. Помимо выполнения заданий на лабораторных занятиях рекомендуется самостоятельно решить упражнения, предложенных к каждой лекции.

2) Планирование времени на самостоятельную работу, необходимого на изучение настоящей дисциплины, студентам лучше всего осуществлять на весь семестр, предусматривая при этом регулярное повторение пройденного материала. Самостоятельная работа студентов заключается в решении всех разобранных на занятиях упражнений, материала учебника и соответствующих форумов интернет, решения всех заданий из индивидуальных практических заданий, решения рекомендуемых задач, подготовки к сдаче промежуточных отчетов и экзамена и дополнительной работы в компьютерном классе самостоятельно. 11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень программного обеспечения и информационных справочных систем Для проведения полноценных занятий необходимо следующее программное обеспечение: Операционная система Windows 7, 8.1 и 10, JDK, Microsoft Visual Studio Express, NetBeans, Ubuntu Linux.

12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Лекции по дисциплине читаются в классе, оборудованном проектором, к каждому занятию имеются презентации, лабораторные работы проходят в компьютерном классе, оборудованном необходимым аппаратными и программными средствами. Часть лекций предоставляется студенту в электронном формате. Практические занятия проводятся в компьютерных классах с современным аппаратным и программным обеспечением.