МИНОБРНАУКИ РОССИИ

Ярославский государственный университет им. П.Г. Демидова

Кафедра информационных и сетевых технологий

УТВЕРЖДАЮ
Декан факультета ИВТ
_____ Д.Ю. Чалый
«18» мая 2020 г.

Рабочая программа дисциплины
«Алгоритмизация и программирование»

Направление подготовки
09.03.03 Прикладная информатика

Направленность (профиль)
«Прикладная информатика в экономике»

Форма обучения
очная

Программа рассмотрена
на заседании кафедры
от «16» апреля 2020 года, протокол № 8

Программа одобрена НМК
факультета ИВТ
протокол № 7 от «17» мая 2020 года

Ярославль
2020

1. Цели освоения дисциплины

Целями дисциплины «Алгоритмизация и программирование» являются освоение теоретических основ современной информатики и основных алгоритмов, а также подходов к программированию на языке Python. Данный курс вырабатывает у студентов алгоритмическое мышление, умение применять основные концепции и классические алгоритмы современной информатики и эффективно решать возникающие задачи на практике. Также курс вырабатывает у студентов практические навыки использования современных языковых средств для решения прикладных задач обработки данных, которые могут быть опубликованы в вебе, а также хранения, обработки и поиска текстовой и другой информации в иных хранилищах данных.

2. Место дисциплины в структуре образовательной программы

Дисциплина «Алгоритмизация и программирование» относится к базовой части ОП бакалавриата.

Для освоения данной дисциплиной студенты должны обладать знаниями по математике и информатике в объеме школьной программы, проявлять настойчивость, целеустремленность и инициативу в процессе обучения.

Полученные в рамках дисциплины «Алгоритмизация и программирование» знания необходимы для развития алгоритмического мышления, развития навыков решения сложных задач, изучения профильных курсов по программированию.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

Процесс изучения дисциплины направлен на формирование следующих элементов компетенций в соответствии с ФГОС ВО, ОП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция	**Индикатор достижения компетенции (код и формулировка)**	**Перечень планируемых результатов обучения**
Общепрофессиональные компетенции		
ОПК-2 Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности;	ОПК-2.1 Обладает навыками выбора современных программных средств для решения прикладных задач в профессиональной деятельности.	Знать: - функционал стандартной библиотеки языка Python; - функционал библиотек для анализа данных. Уметь: – использовать функции стандартной библиотеки языка Python; - использовать функции и классы библиотек для анализа данных; Владеть: – навыками выбора современных программных библиотек для решения поставленных задач; – навыками реализации решений прикладных задач с использованием стандартных библиотек и библиотек анализа данных.

ОПК-2 Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности;	ОПК-2.2 разрабатывает и реализует алгоритмы решения прикладных задач с использованием оригинальных алгоритмов и программных средств с использованием современных систем программирования	Знать: - способы представления комбинаторных структур с помощью структур данных; - основные концепции методов анализа производительности алгоритмов; Уметь: – использовать переборные алгоритмы, в основе которых лежат различные комбинаторные модели; - проводить анализ производительности алгоритмов; Владеть: – навыками решения переборных задач; – навыками оценки производительности программного кода.
ОПК-7. Способен разрабатывать алгоритмы и программы, пригодные для практического применения;	ОПК-7.1 демонстрирует знания методов алгоритмизации, языков и технологий программирования, пригодных для практического применения. ОПК-7.2 демонстрирует умение применять методы алгоритмизации, языки и технологии программирования при решении профессиональных задач	Знать: – основные структуры данных на примере языка Python; – основные алгоритмические подходы к решению прикладных задач. Уметь: – использовать основные алгоритмические структуры данных для решения практических задач; – использовать наиболее популярные алгоритмы для решения практических задач. Владеть: – навыками решения задач с использованием основных структур данных.
ОПК-7. Способен разрабатывать алгоритмы и программы, пригодные для практического применения;	ОПК-7.3 демонстрирует навыки: программирования, отладки и тестирования прототипов программно-технических комплексов задач	Знать: – основы объектно-ориентированного программирования; – принципы декомпозиции задачи на составляющие; – способы применения различных методов отладки и тестирования программ. Уметь: – моделировать несложные задачи предметной области с помощью объектно-ориентированного подхода; – проводить декомпозицию задач; – планировать тестирование разработанного программного кода.

		Владеть: – навыками разработки решений на основе объектно-ориентированного подхода; – разрабатывать простые юнит-тесты для разработанных программ.

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 10 зачетных единиц, 360 акад. часов.

Дисциплина реализуется с применением дистанционных образовательных технологий (ДОТ), предоставляемых образовательной площадкой МООК ЯрГУ им. П.Г. Демидова (DemidOnline).

Дисциплина преподается с использованием онлайн курсов:

- «Программирование на языке Python для начинающих», размещённого по ссылке:
  https://demidonline.uniyar.ac.ru/courses/course-v1:DemidOnline+PrgPytUn002x+2020/course/
- «Обработка данных на языке Python для начинающих», размещённого по ссылке:
  https://demidonline.uniyar.ac.ru/courses/course-v1:DemidOnline+DataPytUn001x+2020/course/

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации (по семестрам)
			Контактная работа					самостоятельная работа	
			лекции	практические	лабораторные	консультации	аттестационные испытания		
1	Раздел 1. Константы, переменные, условный оператор	1	6		6	0,5		8	Тест Контест Итоговый контест по Python
2	Раздел 2. Глобальные переменные и цикл while	1	4		4	0,5		8	Тест Контест Итоговый контест по Python
3	Раздел 3. Строки, списки и цикл for	1	4		4	1		8	Тест Контест Итоговый контест по Python

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации (по семестрам)
4	Раздел 4. Словари, кортежи и файлы	1	4		4	1		8	Тест Контест Итоговый контест по Python
5	Раздел 5. Классы, объекты и ссылки	1	4		4	1		8	Тест Итоговый контест по Python
6	Раздел 6. Переборные алгоритмы	1	4		4	1		8,7	Тест Контест Итоговый контест по Python
7	Раздел 7. Графовые алгоритмы	1	4		4	1		10	Тест Контест Итоговый контест по Python
8	Раздел 8. Алгоритмические задачи		6		6	1		10	Тест Контест Итоговый контест по Python
	Всего за 1 семестр		34		34	7	0,3	68,7	Зачет
9	Раздел 9. Анализ быстродействия алгоритмов.	2	10		10	2		8	Контрольная работа Экзамен
10	Раздел 10. Основные подходы к разработке прикладного программного обеспечения.	2	8		8	2		8	Контрольная работа Экзамен
11	Раздел 11. Разработка сложного прикладного программного обеспечения с использованием сложных алгоритмических подходов.	2	10		10	2		10	Контест Итоговый контест Экзамен
12	Раздел 12. Введение в объектно-ориентированное программирование	2	6		6	1		7	Проверочная работа
							36		Экзамен
	Всего за 2 семестр		34		34	7	36	33	

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации (по семестрам)
13	Раздел 13. Введение в задачи анализа данных. Получение данных из различных источников.	3	4		4	2		6	Индивидуальное задание №1 Индивидуальное задание №2 Итоговое задание по обработке данных
14	Раздел 14. Обработка данных с использованием библиотек numpy и pandas.	3	5		5	2		8,7	Индивидуальное задание №3 Индивидуальное задание №4 Итоговое задание по обработке данных
15	Раздел 15. Средства визуализации данных в Python.	3	4		4	2		8	Индивидуальное задание №5 Итоговое задание по обработке данных
16	Раздел 16. Средства для проведения воспроизводимых исследований.	3	4		4	1		8	Итоговое задание по обработке данных
	Всего за 3 семестр		17		17	7	0,3	30,7	Зачет
	Всего		85		85	21	36,6	132,4	

Содержание разделов дисциплины:

Раздел 1. Константы, переменные, условный оператор

1.0. Программы на языке Python и среда разработки PyCharm

1.1. Работа с целочисленными и вещественными константами

1.2. Вычисления с использованием переменных

1.3. Использование функций

1.4. Еще больше функций: программные модули

1.5. Логический тип данных и условный оператор

1.6. Строковый тип данных

1.7. Исправление синтаксических ошибок в программах

Раздел преподается с использованием онлайн-курса «Программирование на языке Python для начинающих»

Раздел 2. Глобальные переменные и цикл while

2.1. Обработка исключений

2.2. Области видимости: локальные и глобальные переменные

2.3. Пример: решение квадратного уравнения

2.4. Цикл while

2.5. Решение типовых задач с помощью цикла while

2.6. Анализ циклов while

Раздел преподается с использованием онлайн-курса «Программирование на языке Python для начинающих»

Раздел 3. Строки, списки и цикл for

3.1. Функции для работы со строковыми значениями

3.2. Основы работы со списками

3.3. Основные функции для работы со списками

3.4. Разбиение строк и объединение элементов списка в строку

3.5. Цикл for

3.6. Ошибки при работе со строками, списками и циклами

Раздел преподается с использованием онлайн-курса «Программирование на языке Python для начинающих»

Раздел 4. Словари, кортежи и файлы

4.1. Работа с файлами

4.2. Кортежи

4.3. Словари

4.4. Множества

4.5. Особые ситуации при работе со словарями и множествами

Раздел преподается с использованием онлайн-курса «Программирование на языке Python для начинающих»

Раздел 5. Классы, объекты и ссылки

5.1. Модель данных Python: объекты и ссылки

5.2. Объекты и ссылки в программном коде

5.3. Классы

5.4. Специальные методы классов

5.5. Итераторы

Раздел преподается с использованием онлайн-курса «Программирование на языке Python для начинающих»

Раздел 6. Переборные алгоритмы

6.1. Введение в переборные алгоритмы

6.2. Характеристика списка, множества и словаря

6.3. Перебор кортежей

6.4. Перебор кортежей в Python

6.5. Перебор перестановок

6.6. Перебор перестановок в Python

6.7. Перебор сочетаний

6.8. Перебор сочетаний в Python

6.9. Комбинаторный поиск

Раздел преподается с использованием онлайн-курса «Программирование на языке Python для начинающих»

Раздел 7. Графовые алгоритмы

7.1. Введение в теорию графов

7.2. Как задать граф в коде на Python

7.3. Стек и очередь

7.4. Алгоритмы для обхода графов в глубину и в ширину

7.5. Реализация обхода графов в глубину и в ширину на Python

7.6. Раскраска графов

7.7. Реализация алгоритма раскраски графа на Python

7.8. Работа с деревьями на Python

Раздел преподается с использованием онлайн-курса «Программирование на языке Python для начинающих»

Раздел 8. Алгоритмические задачи

8.1. Рекурсия

8.2. Ошибочные ситуации в рекурсивных функциях

8.3. Алгоритмы "Разделяй и властвуй"

8.4. Реализация алгоритмов "Разделяй и властвуй" на Python

8.5. Динамическое программирование

Раздел преподается с использованием онлайн-курса «Программирование на языке Python для начинающих»

Раздел 9. Анализ быстродействия алгоритмов.

9.1. Экспериментальный анализ быстродействия алгоритмов.

9.2. Асимптотический подход к анализу быстродействия алгоритмов.

9.3. Анализ быстродействия алгоритмов «Разделяй и властвуй», алгоритмов динамического программирования, графовых алгоритмов.

Раздел 10. Основные подходы к разработке прикладного программного обеспечения.

10.1. Основы тестирования программного обеспечения.

10.2. Методы разработки надежного программного обеспечения. Инварианты. Конструкция assert. Обработка исключения.

10.3. Модули в Python.

10.4. Стандарты оформления кода и средства для проверки стандартов – Pylint.

Раздел 11. Разработка сложного прикладного программного обеспечения с использованием сложных алгоритмических подходов.

11.1. Алгоритмы на графах, построение остовного дерева, система непересекающихся множеств.

11.2. Алгоритмы на строках. Поиск подстроки в строке.

11.3. Решение сложных задач методом динамического программирования.

Раздел 12. Введение в объектно-ориентированное программирование

12.1. Использование классов в проектировании ПО

12.2. Основные понятия объектно-ориентированного программирования: инкапсуляция, наследование и полиморфизм.

12.3.

Раздел 13. Введение в задачи анализа данных. Получение данных из различных источников

13.1 Введение в науки о данных

13.2 Работа с файлами как с источниками данных

13.3 Работа с источниками данных в веб

Раздел преподается с использованием онлайн-курса «Обработка данных на языке Python для начинающих»

Раздел 14. Обработка данных с использованием библиотек numpy и pandas

14.1 Библиотека numpy. Введение

14.2 Создание numpy-массивов

14.3 Индексирование numpy-массивов

14.4 Проведение вычислений с numpy-массивами

14.5 Введение в библиотеку pandas

14.6 Класс Series в библиотеке pandas

14.7 Создание объектов класса DataFrame в pandas

14.8 Индексирование pandas-объектов

14.9 Проведение вычислений в библиотеке pandas

Раздел преподается с использованием онлайн-курса «Обработка данных на языке Python для начинающих»

Раздел 15. Средства визуализации данных в Python

15.1 Введение в библиотеку matplotlib

15.2 Стили графиков в matplotlib

15.3 Настройка отображения осей в matplotlib

15.4 Работа с подграфиками в matplotlib

15.5 Работа с аннотациями графиков в matplotlib

Раздел преподается с использованием онлайн-курса «Обработка данных на языке Python для начинающих»

Раздел 16. Средства для проведения воспроизводимых исследований

16.1 Методология воспроизводимых исследований

16.2 Использование наборов открытых данных

16.3 Использование средств Jupyter Notebook

16.4 Использование средств Google Colab

16.5 Использование средств Anaconda Cloud

Раздел преподается с использованием онлайн-курса «Обработка данных на языке Python для начинающих»

5. Образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

В процессе обучения используются следующие образовательные технологии:

Вводная лекция – дает первое целостное представление о дисциплине и ориентирует студента в системе изучения данной дисциплины. Студенты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

Академическая лекция (или лекция общего курса) – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Требования к академической лекции: современный научный уровень и насыщенная информативность, убедительная аргументация, доступная и понятная речь, четкая структура и логика, наличие ярких примеров, научных доказательств, обоснований, фактов.

Лабораторное занятие – занятие, посвященное освоению конкретных умений и навыков и закреплению полученных на лекции знаний в оборудованной компьютерной аудитории.

Консультации – вид учебных занятий, являющийся одной из форм контроля самостоятельной работы студентов. На консультациях по просьбе студентов рассматриваются наиболее сложные моменты при освоении материала дисциплины, преподаватель отвечает на вопросы студентов, которые возникают у них в процессе самостоятельной работы.

В процессе обучения используются следующие технологии электронного обучения и дистанционные образовательные технологии:

Электронный учебный курс «Алгоритмизация и программирование» в LMS Электронный университет Moodle ЯрГУ, в котором:

- представлены задания для самостоятельной работы обучающихся по темам дисциплины;
- осуществляется проведение отдельных мероприятий текущего контроля успеваемости студентов;
- представлены правила прохождения промежуточной аттестации по дисциплине;
- представлен список учебной литературы, рекомендуемой для освоения дисциплины;
- представлена информация о форме и времени проведения консультаций по дисциплине в режиме онлайн;
- посредством форума осуществляется синхронное и (или) асинхронное взаимодействие между обучающимися и преподавателем в рамках изучения дисциплины.

Массовые открытые онлайн-курсы, размещенные на платформе DemidOnline:

«Программирование на языке Python для начинающих», размещённого по ссылке: https://demidonline.uniyar.ac.ru/courses/course-v1:DemidOnline+PrgPytUn002x+2020/course/

«Обработка данных на языке Python для начинающих», размещённого по ссылке: https://demidonline.uniyar.ac.ru/courses/course-v1:DemidOnline+DataPytUn001x+2020/course/

В рамках этих курсов:

- представлены онлайн-лекции по отдельным темам дисциплины;
- представлены задания для самостоятельной работы обучающихся по темам дисциплины;
- осуществляется проведение отдельных мероприятий текущего контроля успеваемости студентов;

6. Перечень лицензионного и (или) свободно распространяемого программного обеспечения, используемого при осуществлении образовательного процесса по дисциплине

В процессе осуществления образовательного процесса по дисциплине используются:

В процессе осуществления образовательного процесса используются:

– для формирования текстов материалов для промежуточной и текущей аттестации – программы Microsoft Office, издательская система LaTeX;

- свободно распространяемая среда программирования Python 3;
- свободно распространяемая оболочка IDE PyCharm Community Edition или PyScripter;
- свободно распространяемое ПО Pylint;
- информационная система "БУКИ-NEXT" (АБИС "Буки-Next");

7. Перечень современных профессиональных баз данных и информационных справочных систем, используемых при осуществлении образовательного процесса по дисциплине (при необходимости)

В процессе осуществления образовательного процесса по дисциплине используются:

Автоматизированная библиотечно-информационная система «БУКИ-NEXT» http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php

8. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

а) основная:

1. Доусон М. Программируем на Python. СПб.: Питер, 2015. – 416 с.
2. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск. Вильямс, 2012. – 824 с.
3. Кнут Д. Э. Искусство программирования. Том 2. Получисленные алгоритмы. Вильямс, 2011. – 832 с.
4. Кнут Д. Э. Искусство программирования. Том 1. Основные алгоритмы. Вильямс, 2015 г. – 720 с.
5. Кнут Д. Э. Искусство программирования. Том 4А. Комбинаторные алгоритмы. Вильямс, 2015 г. – 960 с.
6. Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение: учебник для вузов. - СПб.: Питер, 2001.-736с.

7. Р. И. Кабаков «R в действии. Анализ и визуализация данных на языке R» // ДМК Пресс, 2014. 580 с.
8. С. Мастицкий, В. Шитиков «Статистический анализ и визуализация данных с помощью R» // ДМК Пресс, 2015. 496 с.

б) дополнительная:

1. Саммерфильд М. Python на практике. ДМК Пресс, 2014. – 338 с.
2. Лутц М. Python: карманный справочник. Вильямс, 2015. – 320 с.

в) ресурсы сети «Интернет»

1. Электронная библиотека учебных материалов ЯрГУ (http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php).

2.Информационная система "Единое окно доступа к образовательным ресурсам" (http://www.edu.ru (раздел Учебно-методическая библиотека) или по прямой ссылке http://window.edu.ru/library).

3. Электронно-библиотечная система «Университетская библиотека online» (www.biblioclub.ru).

4. Timus Online Judge. Архив задач с проверяющей системой (acm.timus.ru).

5. Яндекс.Контест. Архив задач с проверяющей системой (contest.yandex.ru).

9. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине включает в свой состав специальные помещения:

- учебные аудитории для проведения занятий лекционного типа;
- учебные аудитории для проведения практических занятий (семинаров);
- учебные аудитории для проведения групповых и индивидуальных консультаций;
-помещения для самостоятельной работы;
-помещения для хранения и профилактического обслуживания технических средств обучения.

Специальные помещения укомплектованы средствами обучения, служащими для представления учебной информации большой аудитории.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду организации, а также материалам онлайн курсов, размещённых на образовательной онлайн площадке ЯрГУ им. П.Г. Демидова (DemidOnline).

Автор(ы) :
Зав. кафедрой
информационных и сетевых технологий, к.ф.-м.н. Д.Ю. Чалый

Доцент каф.
информационных и сетевых технологий, к.ф.-м.н. О.Б. Лавровская

Заведующий кафедрой ИСТ _____ Д.Ю. Чалый

Приложение №1 к рабочей программе дисциплины
«Алгоритмизация и программирование»
Фонд оценочных средств
для проведения текущего контроля успеваемости и промежуточной аттестации студентов по дисциплине

1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

1.1. Контрольные задания и иные материалы, используемые в процессе текущей аттестации

Задания для тестов
(проверяют сформированность ОПК-2, ОПК-7, индикаторы ОПК-2.1, ОПК-2.2, ОПК-7.1, ОПК-7.3)

Тесты проводятся в рамках online-курса «Программирование на языке Python для начинающих». В тестах как правило 15 вопросов. Максимальный балл за правильный ответ составляет 1 балл. На каждый вопрос дается одна попытка ответа.

На прохождение теста дается 1,5 часа.

Итоги прохождения теста оцениваются по следующим правилам:

- количество набранных баллов от 13 до 15 соответствует оценке «отлично»;

- количество набранных баллов от 9 до 12 соответствует оценке «хорошо»;

- количество набранных баллов от 5 до 9 соответствует оценке «удовлетворительно»;

- количество баллов меньше 5 соответствует оценке «неудовлетворительно».

Примерные вопросы тестов:

Тест №1.

(проверяет сформированность ОПК-2, индикаторы ОПК-2.1)

Вопрос 1.

Что напечатает следующая программа:

```
str1 = "Привет"
str2 = 'мир'
salute = str1+str2
print(salute,"!")
```

2. *Отметьте только один овал.*

- ◯ Привет мир!
- ◯ Приветмир!
- ◯ Приветмир !
- ◯ Привет мир !

Вопрос 2.

Какое значение примет переменная res после выполнения следующей программы:

```
a = 10
b = '2'
res = a + int(b) + 3
```

3. *Отметьте только один овал.*

- ◯ 15
- ◯ 105
- ◯ 10int3
- ◯ Программа выведет сообщение об ошибке.
- ◯ ab3

Вопрос 3.

Как можно использовать оператор [], чтобы напечатать символ 'y', взятый из строки x = 'From chaly@uniyar.ac.ru'?

4. *Отметьте все подходящие варианты.*

- ☐ print(x[8])
- ☐ print(x[y])
- ☐ print(x[15])
- ☐ print(x[9])
- ☐ print(x[14])

Вопрос 4.

Как использовать оператор для создания среза [:] для получения подстроки 'uniyar' из строки x = 'From chaly@uniyar.ac.ru'?

5. *Отметьте только один овал.*

- ◯ s = x[11:18]
- ◯ s = x[10:17]
- ◯ s = x[10:18]
- ◯ s = x[11:17]

Вопрос 5.

Что напечатает следующий оператор:

```
print(len('apple'*5))
```

6. *Отметьте только один овал.*

- -1
- apple apple apple apple apple
- Ничего не напечатает так как в операторе есть синтаксическая ошибка.
- 25

Вопрос 6.

Какая из следующих строковых функций убирает пробельные символы в начале и в конце строки?

8. *Отметьте только один овал.*

- split()
- strip()
- lstrip()
- rltrim()

Вопрос 7.

Пусть у нас есть список my_great_list = ['pi', 3.14, 'e', 2.78]. Какие из следующих утверждений являются верными?

9. *Отметьте все подходящие варианты.*

- [ ] my_great_list[1] равно 'pi'
- [ ] В Python нельзя определять такие списки.
- [ ] my_great_list[-1] является ошибочным использованием списка.
- [ ] my_great_list[2]+my_great_list[4] равно 5.92
- [ ] my_great_list[2] равно 'e'
- [ ] len(my_great_list) вернет 4

Вопрос 8.

Какие из следующих выражений изменяют содержимое списка my_list?

10. *Отметьте все подходящие варианты.*

- [ ] my_list.reverse()
- [ ] my_list.append(3.14)
- [ ] [1, 2, 3] + my_list
- [ ] other_list.extend(my_list)
- [ ] my_list.extend(["some", "stuff"])

Вопрос 9.

Какой оператор надо вставить вместо ######, чтобы найти произведение элементов произвольного списка numbers?

```
numbers = ...
######
for n in numbers:
    product *= n
print(product)
```

12. *Отметьте только один овал.*

- product = numbers[1]
- product = 0
- product = 1
- product = numbers[0]
- product = []

Вопрос 10.

Следующая функция реализует простой способ, позволяющий обратить строку. Какой оператор необходимо вставить вместо ######?

```
def reverse_string(s):
    """Возвращает строку в обратном порядке"""
    ######
    for char in s:
        result = char + result
    return result

print(reverse_string("Hello!")) # На экран выведет строку '!olleH'
```

14. *Отметьте только один овал.*

- result = 0
- result = " "
- result = ""
- result = []

Тест №2.

(проверяет сформированность ОПК-7, индикаторы ОПК-7.3)

Вопрос 1.

Какая из функций обязана содержать объявление global point чтобы изменить значение глобальной переменной point?

```
point = [0, 0]

def function1():
    point[0] += 1
    point[1] += 1

def function2():
    point = [50, 50]
```

2. *Отметьте все подходящие варианты.*

- [ ] function1
- [ ] function2

Вопрос 2.

Выберите правильные утверждения, которые касаются следующего фрагмента когда:

```
lst = [1, 2, 3, 5, 7, 11]
t = tuple([[30, 40], lst, [20]])
lst.append(25)
l2 = t[2]
l2.append(10)
```

4. *Отметьте все подходящие варианты.*

- [ ] После выполнения этого кода в позиции t[2] будет список [20].
- [ ] После выполнения этого кода длина списка в позиции t[1] будет равна шести.
- [ ] Эта программа будет выполняться с ошибками.
- [ ] В результате выполнения этого кода в позиции t[1] будет список [1, 2, 3, 5, 7, 11, 25]
- [ ] После выполнения этого кода длина списка в позиции t[2] будет равна двум.

Вопрос 3.

Рассмотрим следующую программу:

```
a = ["green", "blue", "white", "black"]
b = a
c = list(a)
d = c
a[3] = "red"
c[2] = a[1]
b = a[1 : 3]
b[1] = c[2]
```

После того, как этот код отработает, на сколько объектов типа список будут ссылаться переменные?

Вопрос 4.

Перечислите через пробел элементы списка lst после того, как отработает вот такой код:

```
def function1(a):
    b = list(a)
    b.pop()
    a.pop()
    return a[0]+1

def function2(b):
    for i in range(len(b)):
        b[i] += 1

lst = [ 1, 2, 5, 7, 9 ]
function2(lst)
function1(lst)
```

Вопрос 5.

Что будет выведено на экран в результате работы следующей программы?

```
class Account:
    def __init__(self, id):
        self.id = id
        id = 666

acc = Account(123)
print(acc.id)
```

Вопрос 7.

Какая разница между понятием функции и понятием метода?

9. *Отметьте все подходящие варианты.*

- [ ] Методы имеют обязательный параметр self, а функции нет.
- [ ] Функции определяются вне классов, а методы являются частью определения класса.
- [ ] Между функцией и методом нет никакой разницы.
- [ ] Методы определяются в стандартной библиотеке Python, а функции определяются в вашем собственном коде.

Вопрос 8.

Рассмотрим определение класса, который мы написали на лекции:

```
class Bird:
    def __init__(self, bird_type = RED):
        if bird_type in [ RED, CHUCK, BOMB ]:
            self.bird = bird_type
        else:
            self.bird = RED

    def bird(self):
        return self.bird

    def __str__(self):
        if self.bird == RED:
            return "RED"
        elif self.bird == CHUCK:
            return "CHUCK"
        elif self.bird == BOMB:
            return "BOMB"
```

Для чего нужен параметр self?

Вопрос 9.

Допустим, мы написали класс Primes, который генерирует последовательность простых чисел (т.е. таких чисел, которые делятся только сами на себя и на единицу):

```
class Primes:
    def __init__(self):
        self.n = 2
        self.a = 7

    def __gcd__(self, x, y):
        while y > 0:
            rem = x % y
            x = y
            y = rem
        return x

    def nextG(self):
        g = self.__gcd__(self.n, self.a)
        self.n = self.n + 1
        self.a = self.a + g
        return g

    def nextPrime(self):
        g = self.nextG()
        while g == 1:
            g = self.nextG()
        return g

    def __str__(self):
        return str((self.n, self.a))

primes = Prime()
```

Последняя строка листинга определяет переменную primes в качестве объекта класса Prime. Как корректно вызвать метод nextPrime, чтобы получить следующее простое число?

Вопрос 10.

```
class Point2D:
    def __init__(self, x = 0, y = 0):
        self.x = x
        self.y = y

    def translate(self, deltax = 0, deltay = 0):
        """Сместить точку по оси x на deltax
            и по оси y на deltay."""
        self.x += deltax
        self.y += deltay
```

Какие из следующих фрагментов кода являются корректной инициализацией объекта класса Point2D и использованием метода translate?

```
# Фрагмент А
point1 = Point2D(3, 9)
point2 = Point2D()
point2.translate(20, 4)

# Фрагмент Б
Point2D = (3, 9)
Point2D.translate(5, −2)

# Фрагмент В
point = Point2D([3, 9])
point.translate(5, −2)

# Фрагмент Г
point = Point2D(7.0, 13.5)
point.translate(8.16, 12.1)
```

Тест №3

(проверяет сформированность ОПК-2, ОПК-2.2)

Вопрос 1.

Монетка может падать либо орлом, либо решкой. Закодируем при помощи нуля орла, а при помощи единицы решку. Мы хотим перебрать все варианты десятикратного подбрасывания монетки и посчитать, в скольких из них нет выпадений двух орлов подряд. Какую модель переборного алгоритма мы можем использовать?

Вопрос 2.

Назовем четное натуральное число n четнопростым, если n не раскладывается в произведение двух четных чисел (например, 16 - не является четнопростым, а 10 - четнопростое). Дано число n, необходимо найти все возможные четнопростые a и b, такие что a ∗ b = n. Какой из вариантов перебора наиболее оптимален?

4. *Отметьте только один овал.*

- ◯ A = {2, 4, 6,..., n}, производим перебор по множеству X = A^2.
- ◯ A = {1, 2,..., n}, производим перебор по множеству X = A^2.
- ◯ A = {2, 4, 6,..., n/2}, производим перебор по множеству A.
- ◯ A = {1, 2,..., n/2}, производим перебор по множеству X = A^2.

Вопрос 3.

Допустим, мы хотим найти максимум функции f(x), определенной на множестве векторов x = (x1, x2, x3, . . . , x7), где каждый элемент кортежа x принадлежит множеству {−2, −1, 0, 1, 2}. Сколько всего векторов нам придется перебрать?

Вопрос 4.

Предположим, что компьютерный пароль содержит 8 символов из множества {a, b, c,..., z}. Сколько различных паролей можно сформировать из этого множества?

Вопрос 5.

Допустим, мы генерируем в лексикографическом порядке кортежи длины 4 из элементов множества A = {1, 2, 3,..., 15}. Какой кортеж будет идти следующим за кортежем (1, 15, 14, 15)?

Вопрос 6.

Мы генерируем все n-кортежи из элементов множества A. Выберите правильные утверждения:

8. *Отметьте все подходящие варианты.*

- [ ] Если A = {0, 10, 100, 1000, 10000}, то самым первым при генерации в лексикографическом порядке должен быть кортеж (0, 0, 0, 10).
- [ ] Если A = {1, 5, 7, 9}, то следующие кортежи находятся в лексикографическом порядке: (1, 5, 7, 9), (1, 5, 9, 1), (1, 5, 9, 5), (1, 5, 9, 7).
- [ ] При посещении кортежей в лексикографическом порядке мы посещаем каждый кортеж один раз.
- [ ] При посещении кортежей в лексикографическом порядке мы посещаем каждый кортеж несколько раз.

Вопрос 7.

Пусть дано множество A = {2, 3, 5, 7, 11}. Вспомним индийский алгоритм лексикографической генерации перестановок. Выберите верные утверждения:

14. *Отметьте все подходящие варианты.*

- [ ] Следующей после перестановки (7, 11, 2, 5, 3) будет перестановка (7, 11, 5, 2, 3).
- [ ] Для получения следующей за перестановкой (11, 7, 3, 5, 2) необходимо увеличить элемент 2.
- [ ] Чтобы получить следующую перестановку после (3, 5, 11, 7, 2) необходимо увеличить элемент 5.
- [ ] Для получения перестановки после (11, 7, 2, 3, 5) алгоритм поменяет местами элементы 5 и 2.
- [ ] Для получения следующей перестановки после (11, 5, 3, 7, 2) алгоритм поменяет местами элементы 3 и 7.
- [ ] Следующей после перестановки (2, 7, 11, 5, 3) будет перестановка (2,11, 3, 5, 7).

Вопрос 8.

Рассмотрим вот такой код:

```
def enumerate(A, n):
    result = []
    if n < 1:
        return result
    for a in A:
        result.append((a, ))
    while len(result[0]) < n:
        nextres = []
        for r in result:
            for a in A:
                newr = list(r)
                newr.append(a)
                nextres.append(tuple(newr))
        result = nextres
    return result
```

Выберите верные утверждения:

16. *Отметьте все подходящие варианты.*

- [ ] Элемент генерируемого множества имеет тип list.
- [ ] Этот код генерирует все кортежи из элементов множества A длины n.
- [ ] Если осуществить вызов функции enumerate([1, 10, 0], 2), то в сгенерированном множестве не будет элемента (10, 10).
- [ ] Элемент генерируемого множества имеет тип tuple.
- [ ] Все сгенерированные элементы хранятся в памяти и возвращаются как результат работы функции enumerate.
- [ ] Этот код генерирует все перестановки множества A длины n.

Вопрос 9.

Числа Фибоначчи можно посчитать с использованием рекурсивного алгоритма. Его реализует вот эта программа:

```
def fib(num):
    if num == 0:
        return 0
    elif num == 1:
        return 1
    else:
        return fib(num-1)+fib(num-2)
```

Пусть f(n) - это рекуррентное соотношение, которое задает количество вызовов функции fib. Какое из предложенных соотношений задает f(n)?

Внимание: f (n) задает не n-е число Фибоначчи, а количество вызовов функции fib при расчете этого числа.

Вы можете переписать эту программу и поэкспериментировать с небольшими значениями n.

4. *Отметьте только один овал.*

- f(0)=1; f(1)=1; f(n)=2f(n−1)+1
- f(0)=0; f(1)=1; f(n)=f(n−1)+f(n−2)
- f(0)=1; f(1)=1; f(n)=f(n−1)+f(n−2)+1
- f(0)=1; f(1)=1; f(n)=f(n−1)+1

Вопрос 10.

Количество рекурсивных вызовов, которые совершаются при расчете чисел Фибоначчи с использованием функции из предыдущего вопроса растет очень быстро при увеличении n. Для того чтобы ускорить эту функцию можно запоминать вычисленные значения в отдельной структуре данных, например, в словаре cache_dict, как это сделано в этой программе:

```
def cached_fib(n, cache_dict):
    if n in cache_dict:
        return cache_dict[n]
    else:
        n1 = cached_fib(n-1, cache_dict)
        n2 = cached_fib(n-2, cache_dict)
        cache_dict[n] = n1+n2
        return n1+n2
```

Если n>1, сколько вызовов функции cached_fib совершается во время вычисления выражения cached_fib(n, {0 : 0, 1 : 1})?

Вопрос 11.

Определите, сколько раз на экран будет выведена строка "Hop!" вот такой программой в зависимости от значения параметра n:

```
def printHop(n):
    if n == 0:
        return
    print("Hop!")
    printHop(n // 2)
```

Примерные задания для контестов

Контесты представляют собой интерактивные задания в системе Яндекс.Контест, которые позволяют автоматизированно проверять программы, которые студенты отправляют в качестве решений.

Обычно контест состоит из 3-6 задач. На решение задач дается одной недели.

Итоги прохождения контеста оцениваются по следующим правилам:

- все полностью решенные задачи соответствуют оценке «отлично»;

- более одной, но менее чем все полностью решенные задачи соответствуют оценке «хорошо»;

- одна полностью решенная задача соответствует оценке «удовлетворительно»;

- меньшее количество полностью решенных задач соответствуют оценке «неудовлетворительно».

Контест №1.

(проверяет сформированность ОПК-7, ОПК-7.1)

Индивидуальное задание представляет собой контест в системе Яндекс.Контест, который организуется специально для группы студентов. Примерные задачи:

Задача 1.

Компания мальчиков нарвала в саду яблок. Они хотят разделить яблоки между собой поровну, а то что останется, отдать маме на пирог. Определите, сколько яблок достанется каждому из них, а сколько они отдадут маме.

Формат ввода

В первой строке подается число N>0 - количество мальчиков в компании. В второй строке подается число M>=N - количество яблок.

Формат вывода

Вывести через пробел два целых числа: количество яблок, которое достанется каждому из мальчиков, и количество яблок, которые они отдадут маме.

Задача 2.

На вход вашей программе оператор Вася вводит два целых числа, A и B. Если A>B, то программа должна вывести их сумму, иначе вывести их произведение. Проблема заключается в том, что Вася печатает быстро и иногда печатает не только цифры, но и буквы. Ваша программа должна корректно обработать и случай ошибочного ввода.

Формат ввода

В первой строке подается значение A. Во второй строке подается значение B.

Формат вывода

Одно число, являющееся результатом вычислений.

В том случае, если в качестве хотя бы одного из значений было введено что-то отличающееся от целого числа, программа должна напечатать "Вася, внимательнее".

Задача 3.

На вход подаются три вещественных числа. Напишите программу, которая определяет, возможно ли составить треугольник с длинами сторон, равными этим числам. Если треугольник возможно составить, то определить, является он прямоугольным, тупоугольным или остроугольным.

Формат ввода

В первой строке подается первое вещественное число. Во второй строке вводится второе вещественное число. В третье строке вводится третье вещественное число.

Формат вывода

Если треугольник нельзя составить, вывести Нельзя. Если составляется прямоугольным треугольник, то вывести строку Прямоугольный, если составляется тупоугольный треугольник, то вывести строку Тупоугольный, иначе вывести строку Остроугольный.

Выводимая строка начинается с заглавной буквы, остальные буквы строчные.

Задача 4.

На вход подается натуральное число больше единицы. Напишите программу, которая определяет, является ли число простым.

Простое число - целое положительное число, имеющее ровно два различных натуральных делителя — единицу и самого себя.

Формат ввода

Натуральное число 1<N<10000.

Формат вывода

Напечатать Простое, если число является простым и Составное в противном случае.

Задача 5.

На обработку поступает натуральное число, не превышающее 10^9.

Нужно написать программу, которая выводит на экран сумму цифр числа, которые делятся на четыре. Если в числе нет таких цифр, требуется на экран вывести No.

Формат ввода

Натуральное число 0<=N<10^9.

Формат вывода

Сумма цифр числа, кратных 4. Если таких цифр нет, то No

Задача 6.

Дана строка, состоящая из строчных и прописных латинских букв. Найти в ней количество гласных букв. Гласными буквами в латинском алфавите считаются буквы «A», «E», «I», «O», «U», «Y».

Формат ввода

В единственной строке задана строка S. Длина строки не превышает 1000 символов.

Формат вывода

Одно число - количество гласных в строке S.

Задача 7.

Главный принцип торговли на бирже: дешевле купить и подороже продать. Петр разрабатывает новую алгоритмическую торговую систему, в которой ему надо определить самую выгодную сделку за предыдущий период времени торговли финансового актива.

Сделка однократная, сначала актив один раз покупается и через какое-то время продается.

Если сделок с положительным финансовым результатом нет, то вывести 0.0

Формат ввода

В первой строке подается одно число $2<N\leq100000$.

Далее следует одна строка, в которой задается последовательность цен на актив в течение интересующего Петра периода времени длины N действительных чисел $1.0\leq ai\leq500.0$.

Формат вывода

Надо вывести финансовый результат самой выгодной сделки с округлением до одного знака после запятой, либо 0.0, если такую сделку невозможно осуществить.

Контест №2

(проверяет сформированность ОПК-7, ОПК-7.1)

1. Разработайте набор классов для моделирования карточной колоды и отдельной карты.

2. Разработайте класс для моделирования отдельной персоны и нескольких персон, которые в совокупности составляют семью.
3. Разработать класс, который моделирует банковский счет и типовые операции по этому счету.

Контест №3.

(проверяет сформированность ОПК-7, ОПК-7.1)

Индивидуальное задание представляет собой контест в системе Яндекс.Контест, который организуется специально для группы студентов. Примерные задачи:

Задача 1.

Вводится число n – размер шахматной доски. Расставьте на доске n ферзей таким образом, чтобы ни один не бил другого. Выведите получившуюся доску.

Задача 2.

Вводится число k – количество предметов в рюкзаке, а также число m – максимальный вес предметов, которые вы можете поднять. В следующей строке вводится k чисел, задающие вес каждого из предметов, далее следует строка из k чисел, в которой задается ценность каждого из предметов. Найдите набор предметов максимальной ценности, которые умещаются в рюкзак.

Контест №4.

(проверяет сформированность ОПК-7, ОПК-7.1)

1. Разработайте структуру данных, которая подходит для хранения векторов произвольной длины и реализуйте функцию сложения векторов и скалярного произведения.
2. Разработайте класс, который позволяет хранить матрицы фиксированного размера и позволяет производить сложение и умножение матриц.
3. Реализовать метод Гаусса для решения систем линейных уравнений.
4. Реализовать с использованием метода Монте-Карло алгоритм, играющий в крестики-нолики.

Контест №5.

(проверяет сформированность ОПК-7, ОПК-7.1)

Задача 1.

Введение

На Марсианские конференции собираются ученые со всего Марса. Для простоты общения марсианские ученые берут себе вместо имен числительные. При регистрации каждый участник конференции указывает список своих друзей, с которыми он хочет посетить одну секцию работы конференции. Если это совсем молодой ученый (еще не успел завести себе друзей), то он при регистрации указывает пустой список. Напишите программу, определяющую количество участников секции в которой примет участие председатель конференции S.

Задание

Напишите программу, определяющую количество участников секции в которой примет участие председатель конференции S.

Исходные данные

Первая строка ввода содержит одно число S ($S \leqslant 100$) — имя Председателя, вторая строка содержит одно число N ($N \leqslant 100$) —- количество марсиан на конференции. Следующие N строк содержат списки друзей зарегистрированных марсиан от 1 до N. Помните, что отношение дружбы у марсиан всегда взаимное.

Ваше решение должно успеть выдать ответ за 10 секунд.

Задача 2.

Введение

На школьную научную конференцию собираются ученики из разных школ. При регистрации каждый участник конференции получил порядковый номер и указал список своих друзей, с которыми он хочет посетить одну секцию работы.

Задание

Напишите программу, определяющую количество участников секции в которой примет участие председатель конференции S.

Исходные данные

Первая строка ввода содержит одно число S ($S \leqslant 100$) — номер Председателя, вторая строка содержит одно число N ($N \leqslant 100$) — количество учеников на конференции. Следующие N строк содержат списки друзей зарегистрированных учеников от 1 до N. Отношение дружбы всегда взаимное.

Ваше решение должно успеть выдать ответ за 10 секунд.

Задача 3.

В физике и химии явлением перколяции (от лат. percolare, просачиваться, протекать) называется явление протекания или не протекания жидкостей через пористые материалы, электричества через смесь проводящих и непроводящих частиц и другие подобные процессы. Теория перколяции находит применение в описании разнообразных систем и явлений, в том числе таких, как распространение эпидемий и надежность компьютерных сетей.

Некоторые примеры задач, которые решаются через теорию перколяции:

- Сколько надо добавить медных опилок в ящик с песком, чтобы смесь начала проводить ток?
- Какой процент людей должен быть восприимчив к болезни, чтобы стала возможна эпидемия?

Мы будем моделировать нашу систему с помощью квадратной решетки размера N× N. Элементы решетки могут быть либо проводящими, либо непроводящими. Представьте себе, что мы льем жидкость сверху, в клетки из верхнего ряда решетки. При этом проводящие клетки заполнятся и жидкость будет распространяться дальше по соседним проводящим клеткам. Распространение жидкости идет либо по вертикали, либо по горизонтали.

Вам необходимо узнать, просочится ли жидкость сквозь нашу систему, т.е. распространится ли жидкость до нижнего ряда?

Условие с картинками можно посмотреть вот здесь - https://yadi.sk/i/03cMioC03Tebs9

Задача 4.

В некоторой стране n городов. Правительство решило электрифицировать все эти города. Для начала в k различных городах были построены электростанции. Другие города должны быть связаны с электростанциями линиями электропередач. Между любой парой городов i и j можно построить линию электропередач стоимостью cij рублей. После гражданской войны страна находится в глубоком кризисе, поэтому правительство решило построить всего лишь несколько линий электропередач. Конечно, после постройки линий должен существовать путь по ним от любого города до некоторого города с электростанцией. Найдите минимальную возможную стоимость постройки всех необходимых для этого линий электропередач.

Формат ввода

В первой строке записаны целые числа n и k (1 <= k <= n <= 100). Во второй строке записаны k различных целых чисел - номера городов с электростанциями. В следующих n строках записана таблица cij размера n× n, состоящая из целых чисел (0 <= cij <= 105). Гарантируется, что cij = cji, cij > 0 для $i \neq j$, cii = 0.

Формат вывода

Выведите минимальную стоимость электрификации всех городов.

Проверочная работа

(проверяет сформированность ОПК-7, ОПК-7.3)

1. Разработайте для каждой из задач из предыдущего индивидуального задания набор тестов, покрывающий крайние случаи каждой задачи.
2. Проверьте свое решение каждой из задач и предыдущего индивидуального задания с помощью средства Pylint и исправьте найденные ошибки оформления кода.

Задания для контрольной работы

(проверяет сформированность ОПК-2, ОПК-2.1)

В рамках контрольной работы предлагаются вопросы, которые позволяют проверить подготовку студентов по первым трем разделам.

1. Рассмотрим следующий фрагмент программы:

```
1  a, b = 10, "123"
2  ...
```

Какие из операторов можно использовать вместо точек в строке 2?

- ☐ `a += 10`
- ☐ `b = 10 / +(3-2)`
- ☐ `a = b*5`
- ☐ `a += b*3`
- ☐ `b = a + b`

2. Какие имена в следующей программе являются локальными, а какие глобальными?

```
1  import math
2  a, b, c = 2, 3, -1
3  def d(b, c):
4      r = b**2 - 4*a*c
5      return r
6  print((math.sqrt(d(b,c))-b)/(2*a), (-math.sqrt(d(b,c))-b)/(2*a))
```

	Глобальное	Локальное
a	☐	☐
b	☐	☐
c	☐	☐
d	☐	☐
r	☐	☐

3. Что выведет на экран следующая программа:

```
1  x=-10
2  for value in [3, 41, 12, 9, 10, 74, 15] :
3      if value < 10 :
4          x = x + value
5  print(x)
```

- ☐ 2
- ☐ 12
- ☐ 154
- ☐ 164

4. Программист написал программу, которая выводит на экран поздравление.

```
1  def date(day, month):
2      return str(day)+"."+str(month)
3  print('Сегодня'+date(13, "09"), ". Поздравляем с днем программиста!")
```

Что же в действительности будет выведено на экран?

- ☐ Сегодня13.09 . Поздравляем с днем программиста!
- ☐ Сегодня 13.09. Поздравляем с днем программиста!
- ☐ Сегодня 13.09 . Поздравляем с днем программиста!
- ☐ В программе есть ошибки, поэтому она ничего не выведет на экран.

5. Каким будет значение списка `lst` после выполнения вот такой программы:

```
1  lst = [ 1, 2, 5, 10 ]
2  b = lst
3  c = b[2]
4  c = 200
```

Ваш ответ: `lst=[` _____ `].`

элементы списка через запятую

---

6. Сколько всего указателей на объекты создается после выполнения следующего кода и на сколько объектов они указывают?

```
1  lst = [ [ 2 ], [ 11, 23 ] , [ 31 ] ]
```

Ваш ответ: создается _____ и _____.

количество указателей *количество объектов*

---

7. Следующая программа выводит на экран список. Что это за список?

```
1  lst = [ 1, 5, 8 ]
2  lst.append([ 135 ])
3  print(lst)
4  b = lst[3]
5  b.append(100)
6  print(lst)
```

Ваш ответ: _____.

здесь должен быть список

8. На прошлой лекции мы говорили об алгоритме, который генерирует все сочетания. Вот его код:

```
1  def next_choose(c, N, k):
2      if len(c) < k:
3          c.clear()
4          c.extend(range(k))
5          return tuple(range(k))
6      else:
7          l = list(c)
8          l.extend([N, 0])
9      j = 0
10     while l[j]+1 == l[j+1]:
11         c[j] = j
12         j += 1
13     if j == k:
14         return None
15     c[j] += 1
16     return tuple(c)
```

Выберите верные утверждения:

- ☐ Функция `next_choose` используется для генерации k-сочетаний из множества $\{0, \ldots, N-1\}$ без повторений.
- ☐ Текущее сочетание хранится в переменной `c`. По завершению работы функции в `c` находится следующее сочетание.
- ☐ Строки 10-12 используются одновременно и для поиска элемента, который надо увеличить, и для инициализации префикса сочетания.
- ☐ После вызова функции `next_choose([2, 3, 4, 5], 10, 4)` в переменной `c` будет список `[0, 1, 2, 6]`.
- ☐ Функция `next_choose` используется для генерации множества k-сочетаний из произвольного множества.
- ☐ Последним сочетанием, сгенерированным функцией `next_choose(..., 12, 5)` будет сочетание `[11, 11, 11, 11, 11]`.

9. Чтобы генерировать сочетания с повторениями можно использовать функцию `next_choose`, определенную в прошлом вопросе:

```
1  def next_choose_rep(c, N, k):
2      d = next_choose(c, N, k)
3      if d is None:
4          return None
5      d = list(d)
6      for i in range(len(d)):
7          d[i] -= 1
8      return tuple(d)
```

В этом коде две ошибки. Исправьте их:

Первая ошибка находится в строке [ ], должно быть _____ .
оператор

Вторая ошибка находится в строке [ ], должно быть _____ .
оператор

10. Отметьте верные утверждения:

- ☐ k-сочетаний без повторений строго больше, чем k-размещений без повторений.
- ☐ k-сочетаний без повторений столько же, сколько и $(n-k)$-сочетаний без повторений.

Список заданий к экзамену

Экзамен заключается в решении контеста в системе Яндекс.Контест. Задания аналогичны тем, которые даются в качестве индивидуальных заданий.

Примерные индивидуальные задания

В качестве индивидуальных заданий даются мини-проекты, которые позволяют оценить сформированность компетенций по разработке простых программ для задач анализа данных.

Задания реализованы в системе Яндекс.Контест, доступ в эту систему происходит через онлайн-курс «Обработка данных на языке Python для начинающих»

Индивидуальное задание №1. *Основы программирования на языке Python.* (проверяет сформированность ОПК-7, ОПК-7.1)

Анализ котировок акций является весьма увлекательным (и для кого-то прибыльным) занятием, которое в чем-то напоминает азартные игры. Однако нашей целью будет понять, как могут выглядеть данные и каким образом с ними можно работать.

Цель задания

- Найти, скачать и изучить набор данных;
- Вспомнить, как читать данные из файлов;
- Сгенерировать аналитические данные;
- Проанализировать данные: сколько мы могли бы заработать, если бы умели идеально играть на бирже;
- Визуализировать свои прибыли :)

Набор данных

Сайт Yahoo!Finance является хорошим местом чтобы начать изучение исторических данных о торгах на биржах в США. Каждой компании на бирже присваивают короткий

набор символов, который ее идентифицирует (по-другому этот набор еще называется тикер, ticker). Например, корпорация Apple имеет тикер AAPL, Microsoft - MSFT, Yandex - YNDX.

Индивидуальное задание №2. (проверяет сформированность ОПК-7, ОПК-7.1)

В это задании вашему вниманию предлагается набор данных в виде csv-файла, в котором содержатся данные о пассажирах с Титаника. Скачать этот набор данных можно по ссылке titanic_passengers.csv

Набор данных

Набор данных представляет собой csv-файл, в котором каждая строчка содержит данные на одного пассажира. Вот описание столбцов, которые представляют для нас интерес:

- PassengerId, уникальный в пределах файла идентификатор пассажира;
- Survived, 0 если пассажир не выжил и 1 если выжил после крушения;
- Pclass, класс (от 1 до 3), которым путешествовал пассажир;
- Last Name, фамилия пассажира;
- First Name, имя пассажира;
- Sex, пол пассажира (F — женский, M — мужской);
- Age, возраст пассажира;
- Fare, стоимость билета.

Аналитические данные

Для чтения исходных данных и подготовки аналитических данных используется стандартный модуль csv.

Индивидуальное задание №3. (проверяет сформированность ОПК-7, ОПК-7.1)

В этом задании вы попрактикуетесь в чтении Excel'евских файлов в формате xls и обработке данных с помощью библиотеки numpy. Скачать этот набор данных можно по ссылке olympicsalltime.xls

Набор данных

В файле с исходными данными находится единственный лист, содержащий большую таблицу с итогами Олимпийских игр для каждой страны:

- в первом столбце располагается название страны;
- в столбцах со второго по пятый находятся данные касательно летних олимпийских игр: количество игр, в которых участвовала страна; число завоеванных золотых, серебряных и бронзовых медалей;
- в столбцах с шестого по девятый находятся данные о зимних олимпийских играх.

Аналитические данные

Для чтения исходных данных и подготовки аналитических данных используется модуль xlrd. Вы можете использовать следующий функционал этого модуля:

Функция xlrd.open_workbook(filename) возвращает открытый файл с книгой Excel с именем filename.

Если вы открыли книгу book, то получить лист из этой книги можно с использованием метода book.sheet_by_index(idx), где idx - номер листа в книге по порядку (нумерация в Python начинается с 0).

На листе lst получить список строк можно при помощи метода lst.get_rows() (лучше преобразовать результат в список).

Каждая строка на листе является списком ячеек. Чтобы получить значение, записанное в ячейке cell, можно использовать атрибут cell.value.

Больше примеров можно найти по адресу https://habrahabr.ru/post/99923/.

Вашей задачей является формирование numpy-массива как итоговое представление аналитических данных.

Индивидуальное задание №4. (проверяет сформированность ОПК-7, ОПК-7.1)

В этом задании вы будете практиковаться в чтении xml-файлов и их обработке с помощью библиотеки pandas. Для этого в Python существует библиотека lxml. Скачать этот набор данных можно по ссылке movies.xml.zip

Набор данных

Вам будет предложен xml-файл, который представляет фрагмент выгрузки информации о фильмах из базы данных IMDB.

Этот файл имеет следующую структуру (чтобы в этом убедиться, можете открыть его в самом простом текстовом редакторе):

<?xml version='1.0' encoding='utf-8'?>
<collection>
<movie> <!--- Описание одного фильма --->
<title>'Breaker' Morant </title> <!--- Название фильма --->
<year>1980&60;/year> <!--- Год выпуска --->
<cast num="42"> <!--- Список актеров, num - сколько актеров снималось в фильме --->
<actor>Brown, Bryan (I)</actor> <!--- Имя актера --->
<actor>Henderson, Dick (II)</actor>
<actor>Gray, Ian (I)</actor>
...
</cast>
</movie>
<movie> <!--- Следующий фильм --->
...
</movie>
...
</collection>

внешним является тег </collection>, в котором описывается вся коллекция фильмов целиком;

отдельный фильм описывается в теге </movie>;

внутри каждого тега </movie> вложены сущности, описываемые тегами </title> (название фильма), </year> (год выпуска фильма) и </cast> (список актеров, которые играли роли в этом фильме);

у тега </cast> есть параметр num, в котором указывается длина списка актеров, игравших в фильме.

Аналитические данные

Для чтения исходных данных и подготовки аналитических данных используется модуль lxml. Вы можете использовать следующий функционал этого модуля:

Загрузить файл и построить DOM-дерево можно при помощи функции etree.parse(filename), где filename - имя xml-файла.

Как только вы построили DOM-дерево doc, получить корневой элемент можно при помощи метода doc.getroot().

У элемента дерева node есть два полезных атрибута: node.tag (xml-тег элемента) и node.text (текст, который находится между открывающим и закрывающим тегом, включая пробелы и переводы строки).

Рекурсивно обойти дерево начиная с узла node можно при помощи такой конструкции:

for item in node.iter():

обрабатываем узел item

Индивидуальное задание №5. (проверяет сформированность ОПК-7, ОПК-7.1)

Мы возвращаемся к набору данных о выживших пассажирах на Титанике. Скачать этот набор данных в формате csv можно по ссылке titanic_passengers.csv

Набор данных

Набор данных представляет собой csv-файл, в котором каждая строчка содержит данные на одного пассажира. Вот описание столбцов, которые представляют для нас интерес:

- PassengerId, уникальный в пределах файла идентификатор пассажира;
- Survived, 0 если пассажир не выжил и 1 если выжил после крушения;
- Pclass, класс (от 1 до 3), которым путешествовал пассажир;
- Last Name, фамилия пассажира;
- First Name, имя пассажира;
- Sex, пол пассажира (F --- женский, M --- мужской);
- Age, возраст пассажира;
- Fare, стоимость билета.

Аналитические данные

Для чтения исходных данных и подготовки аналитических данных используется стандартный модуль csv.

Задача

Очень часто при решении задач обработки данных приходится сопоставлять данные из разных источников. Вам надо будет провести сопоставление данных из csv-файла и из веб.

Вот эта статья в Wikipedia содержит еще один список пассажиров Титаника: https://en.wikipedia.org/wiki/Passengers_of_the_RMS_Titanic.

Перед вами стоит проблема понимания, существуют ли расхождения в двух наборах данных и если да, то какие это расхождения. Для этого выполните следующие задачи:

- напишите код, который читает csv-файл;
- напишите код, который автоматически скачивает страницу из Wikipedia (используйте модуль requests);
- напишите код, позволяющий распарсить скачанную html-страницу вытаскивает из нее необходимые данные (модуль lxml + вспомните XPath);
- мы понимаем, что страница в Wikipedia может измениться со времени выхода этого курса, поэтому для ответа на вопросы вам необходимо использовать файл titanic.html.zip.

Список заданий к зачету в 3 семестре

. (проверяет сформированность ОПК-7, ОПК-7.1, ОПК-7.3)

В случае успешного выполнения заданий для индивидуальной работы, зачет выставляется автоматом. В противном случае зачет ставится на основании выполнения задания в компьютерном классе.

Все задания используют модули pandas, numpy, matplotlib и стандартные модули python. Засчитываются только решения, в которых используется функционал этих модулей, а не написание решения на обычном python с нуля.

1. Напишите программу, которая преобразует numpy-массив в объект типа Series.

2. Создайте два объекта типа Series с числовыми значениями и создайте программу, которая складывает и вычитает поэлементно один объект и другой.

3. Напишите программу, которая отсортирует значения объекта типа Series.

Пример входа	Пример выхода
0 10 1 20 2 5 3 15 4 100 dtype: int64	2 5 0 10 3 15 1 20 4 100 dtype: int64

4. Напишите программу, которая вставляет столбец в Dataframe.

Пример входа	Пример выхода
col2 0 10 1 20 2 5 3 15 4 100 dtype: int64	col1 col2 0 1 10 1 2 20 2 3 5 3 5 15 4 8 100 dtype: int64

5. Напишите программу, которая на в существующий Dataframe вставляет новый столбец, значения в котором представляют квадратный корень от суммы чисел в первых двух столбцах (построчно).

6. Напишите программу, которая отбирает в существующем Dataframe’е все строки, в первом столбце которых записано четное значение, а во втором столбце — нечетное.

7. Напишите программу, которая скачивает с сайта cbr.ru исторические курсы доллара и визуализирует его изменение за введенный промежуток времени.

2. Перечень компетенций, этапы их формирования, описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкалы оценивания

2.1. Шкала оценивания сформированности компетенций и ее описание

Оценивание уровня сформированности компетенций в процессе освоения дисциплины осуществляется по следующей трехуровневой шкале:

Пороговый уровень - предполагает отражение тех ожидаемых результатов, которые определяют минимальный набор знаний и (или) умений и (или) навыков, полученных студентом в результате освоения дисциплины. Пороговый уровень является обязательным уровнем для студента к моменту завершения им освоения данной дисциплины.

Продвинутый уровень - предполагает способность студента использовать знания, умения, навыки и (или) опыт деятельности, полученные при освоении дисциплины, для решения профессиональных задач. Продвинутый уровень превосходит пороговый уровень по нескольким существенным признакам.

Высокий уровень - предполагает способность студента использовать потенциал интегрированных знаний, умений, навыков и (или) опыта деятельности, полученных при освоении дисциплины, для творческого решения профессиональных задач и самостоятельного поиска новых подходов в их решении путем комбинирования и использования известных способов решения применительно к конкретным условиям. Высокий уровень превосходит пороговый уровень по всем существенным признакам.

Приложение №2 к рабочей программе дисциплины «Алгоритмизация и программирование»

Методические указания для студентов по освоению дисциплины

Основной формой изложения учебного материала по дисциплине «Алгоритмизация и программирование» являются лекции, причем в достаточно большом объеме. Это связано с тем, что в основе информатики и программирования лежит особый математический аппарат, с помощью которого решаются довольно сложные и громоздкие задачи. По большинству тем предусмотрены практические занятия, на которых происходит закрепление лекционного материала путем применения его к конкретным задачам и отработка практических навыков программирования.

Для успешного освоения дисциплины очень важно решение достаточно большого количества задач, как в аудитории, так и самостоятельно в качестве домашних заданий. Примеры решения задач разбираются на лекциях и практических занятиях, при необходимости по наиболее трудным темам проводятся дополнительные консультации. Основная цель решения задач – помочь усвоить фундаментальные понятия и основы информатики. Для решения всех задач необходимо знать и понимать лекционный материал. Поэтому в процессе изучения дисциплины рекомендуется регулярное повторение пройденного лекционного материала. Материал, законспектированный на лекциях, необходимо дома еще раз прорабатывать и при необходимости дополнять информацией, полученной на консультациях, практических занятиях или из учебной литературы. Полный список заданий для самостоятельной работы по темам (разделам) дисциплины приведен в ЭУК в LMS Moodle «Алгоритмизация и программирование», а также в онлайн-курсах на платформе DemidOnline. Вопросы, возникающие в процессе или по итогам решения этих задач, можно задать на консультациях или в форуме (чате) в ЭУК в LMS Moodle.

Большое внимание должно быть уделено выполнению домашней работы. В качестве заданий для самостоятельной работы дома студентам предлагаются задачи, аналогичные разобранным на лекциях и практических занятиях или немного более сложные, которые являются результатом объединения нескольких базовых задач.

Для проверки и контроля усвоения теоретического материала, приобретенных практических навыков работы с аппаратом современной информатики, в течение обучения проводятся мероприятия текущей аттестации в виде контрольной работы в 1-ом семестре и самостоятельных работ во всехсеместрах изучения дисциплины. Также проводятся консультации (при необходимости) по разбору заданий для самостоятельной работы, которые вызвали затруднения.

В конце первого семестра изучения дисциплины студенты сдают зачет, в конце первого года курса – экзамен, а в конце третьего семестра — зачет. Зачет по итогам первого семестра выставляется по итогам тестирования и краткого собеседования по его результатам. Экзамен принимается в компьютерной аудитории, где студентам предлагаются контесты, состоящие из 3-5 задач по тематике курса. На самостоятельную подготовку к экзамену выделяется 3 дня, во время подготовки к экзамену предусмотрена групповая консультация.

Освоить вопросы, излагаемые в процессе изучения дисциплины «Алгоритмизация и программирование» самостоятельно студенту крайне сложно. Это связано со сложностью изучаемого материала и большим объемом курса. Поэтому посещение всех аудиторных занятий является совершенно необходимым. Без упорных и регулярных занятий в течение семестра сдать зачет и экзамен по итогам изучения дисциплины студенту практически невозможно.