

#	Задание	Требования
1	Напишите код на Python, который предлагает пользователю ввести свой рост, вес, возраст через запятую. На выходе программа должна вывести ИМТ (индекс массы тела) и возраст в виде: Ваш ИМТ - «числовая значение» (интерпретация значения по следующей таблице) Ваша возрастная категория - «наименование категории по таблице»	Переменные должны быть названы в соответствии со значениями, которые в них содержатся. Т.е. вес - weight, возраст - age или age и so. Программа должна понимать, в сантиметрах вводит пользователь значение или в метрах (для метров используется точка, например 1.80). Формула для вычисления ИМТ: $\text{вес} / (\text{в метрах})^2$ Программа должна выполнить следующие проверки: <ul style="list-style-type: none">• Пользователь ввел все нужные параметры.• Введенные значения должны быть цифровыми.• Проверка деления на 0. Если какая-либо проверка не выполняется, программа должна вернуть предупреждение для пользователя, и запросить ввод данных повторно. Следная таблица ИМТ: <ul style="list-style-type: none">• до 16 (включительно): <u>Выдаваемый дефицит массы тела</u>• от 16 до 18,5 (включительно): <u>Недостаточная (дефицит) масса тела</u>• от 18,5 до 25 (включительно): <u>Норма</u>• от 25 до 30 (включительно): <u>Избыточная масса тела (предожирение)</u>• от 30 до 35 (включительно): <u>Ожирение первой степени</u>• от 35 до 40 (включительно): <u>Ожирение второй степени</u>• от 40 и более: <u>Ожирение третьей степени (морбидное)</u> Таблица возрастных категорий: <ul style="list-style-type: none">• до 1 года (включительно): <u>Младенцы</u>• от 1 года до 10 лет (включительно): <u>Дети</u>• от 10 лет до 18 лет (включительно): <u>Подростки</u>• от 18 лет до 60 лет (включительно): <u>Взрослые</u>• от 60 лет: <u>Пожилые</u>
2	Написать код на Python, который предлагает ввести пользователю дату в формате dd.mm.yyyy, на выходе необходимо получить: dd.mm.yyyy - «обычный/искомого год» <«год» год (также добавит наименование года, обезьяны или собаки, например) <«» квартал <«месяц» месяц (прописью) <«день» день <«сколько прошло суток от начала года» <«день недели» день недели	При выполнении данной задачи не следует пользоваться специализированными модулями Python для работы с датами. Программа должна проверить, что пользователь ввел дату в корректном формате. Если формат некорректный, программа должна выдать предупреждение и запросить повторный ввод данных. Числовое значение - от начала времени (01.01.1970) при условии, что время берется в виде 00:00:00. Если при запуске скрипта добавить аргумент, то возьмется случайная дата (от 1970 года) без возможности вводить ее пользователю.
3	Написать код на Python, который предлагает получить n-ое значение числа Фибоначчи и вычислить факториал n!. Само значение N задается пользователем, после чего на экран выводится вся последовательность чисел Фибоначчи в виде итераций, число итераций и вычисленный факториал. Например, N = 5. 1 итерация: 2 2 итерация: 3 3 итерация: 5 4 итерация: 8 5 итерация: 13 Выполнено 5 итераций. Факториал 5! = 120	Первые значения 1 и 1 не учитываются при расчете. Об этом необходимо уведомить пользователя с помощью вывода на экран во время запуска приложения. Должна быть проверка на допустимость значения. При N < 1 на экране выводится ошибка.
4	Написать код на Python, который предлагает сыграть с пользователем в угадайку. Программа загадывает число от 1 до 100, пользователю дана возможность ввода. Если пользователь не угадывает, программа сообщает ему об этом. Скрипт не завершается до тех пор, пока пользователь не угадает значение.	Проверка на "попадание" должна находиться в отдельной функции, в которую передается два аргумента - загаданное число и ввод от пользователя. На выходе функции должна быть строка с пояснением, куда двигаться пользователю. Необходимо использовать ориентир, состоящий из 2 параметров: 1) Расстояние от точки, в которой находится игрок до цели. Абсолютная величина, выражается в следующих единицах: "холодно", "тепло", "горячо", "жарко" (спасибо можно дополнить своими значениями). 2) Направление движения игрока. Относительная величина, в которой необходимо будет учитывать связь между прошлым введенным значением и нынешним. Выражается в следующих единицах: "холоднее", "горячее". Лог игры должен записываться в файл в папку Logs, которая должна лежать рядом с файлом скрипта. Скрипт должен проверять существование папки Logs, если ее нет - он должен создать ее. В имени папки обязательно должны присутствовать имя файла python-скрипта и время начала игры. Лог обязательно должен создаваться в начале игры и дополняться по ходу действия. В лог должны присутствовать: 1) Время начала игры; 2) Введенные пользователем значения; 2) Время окончания игры; 3) Количество попыток, сделанных пользователем.
5	Написать код на Python, который оцифрует с из папки несколько заранее подготовленных файлов с животны. При запуске пользователю предлагается ввести путь до этих файлов. После чтения программа должна вывести оцифрованные текстовые файлы в виде списка животных и предложить выбор: 1) Вывести более подробную информацию о животном. 2) Скорректировать информацию. 3) Сохранить и выйти 4) Выйти без сохранения. Программа выполняется до тех пор, пока не выбраны пункты 3-4. Если выбран пункт 1, то сначала предлагается выбрать животное из списка, а затем выдается вся информация из файла, после чего пользователю снова предлагается выбор из 4 пунктов главного меню, описанного выше. Если выбран пункт 2, то сначала пользователю предлагается выбрать сначала животное, затем необходимый пункт для изменения, а после уже вводить корректировки. После нажатия Enter программа уточняет: Применить корректировки (y/n) с выводением на экран значения ДО изменения и ПОСЛЕ. Если выбрано "n", то вернуться в меню корректировки, если "y", то в главное меню.	Файлы необходимо подготовить самостоятельно, в формате txt. Достаточно 2-3 файлов, содержащие следующую информацию о животных: 1) Наименование 2) Вид 3) Вид 4) Средний продолжительность жизни 5) Средний рост 6) Средняя масса тела 7) Место обитания (на деревьях, под водой, под землей и т.д.) 8) Страна или страны обитания, указанных через запятую. Формат в котором будут напасаны данные - на усмотрение пользователя. Создать общий класс для всех животных. Для каждого конкретного животного создается экземпляр класса, параметры каждого животного записываются в атрибуты экземпляра. Применить инкапсуляцию: атрибуты создаваемого экземпляра должны быть protected или private, взаимодействие с ними должно осуществляться через специальные методы (getter, setter). При имплементации методов реализовать проверки корректности данных: 1. для параметров 4-6 должны быть введены ТОЛЬКО цифры (иначе вернуть ошибку типа данных, но программу продолжить), 2. для остальных параметров - только РУССКОЕ Буаны, значения может присутствовать только в 8 пункте. Также программа должна отслеживать, в каком экземпляре класса было произведено изменение. И при выборе пользователем "Сохранить и выйти" перезаписывать только те файлы, экземпляры классов которых были изменены.
6.	Необходимо разработать приложение, которое будет получать данные о прогнозе погоды из API https://openweathermap.org/ и складывать данные в Vertica. В приложении должен присутствовать следующий функционал: <ul style="list-style-type: none">• обращение к API• подключение к Vertica, сохранение данных в таблицу• логирование результатов работы, использовать в программе разные уровни логирования, хранить логи в папке logs (если ее нет - создавать), каждый запуск приложения записывать в отдельный лог с уникальным именем• считывание конфигурации из файла settings.ini. Файл может выглядеть следующим образом <pre>[[Блок кода]]</pre> <pre>[DATABASE] host=... port=... db_name=... user=... password=... [API] url=... token=... [LOGGING] log_level=...</pre> <ul style="list-style-type: none">• считывание аргументов командной строки для конфигурировании настроек при запуске (возможность указывать другую схему и имя таблицы в БД, сменить логирование на stdout вместо файла).	Работа с Vertica: 1. Установить Vertica. 2. Создать схему и таблицу (название придумать самостоятельно). Таблица должна содержать агрегированные данные из прогноза погоды. Примерный набор атрибутов: CITY, COUNTRY_CODE, TEMP_MIN, TEMP_MAX, PROCESSED_OTTM (можно дополнить другим атрибутами, которые уместно агрегировать до города, например HUMIDITY_MIN, HUMIDITY_MAX и пр). 3. Для работы с Vertica в Python использовать библиотеку vertica-python (https://github.com/vertica/vertica-python). Работа с API 1. Зарегистрироваться на сервисе и скачать свой токен (использовать free-версию) <u>Необходимо учитывать, что он активен всего на 60 дней</u> 2. Для работы с API использовать библиотеку ruopen (https://github.com/ruopen/ruopen) 3. Небольшая инструкция, как установить библиотеку и получить токен: https://pythonhowtoprogram.com/how-to-use-weather-api-to-get-weather-data-in-python-3/ . 4. Примеры работы с API можно найти на странице официальной документации ruopen: https://pythonweatherdoc.com/latest/3/code-examples.html 5. Вначале необходимо выполнить скачивание статического справочника с городами (http://bulk.openweathermap.org/sample/city_list.json.gz) 6. Затем для городов из справочника через функцию forecast_get_url можно получить информацию о прогнозе за каждые 3 часа на неделю вперед (default=day доступен только в платной подписке, поэтому нужно использовать interval=3h). 7. Необходимо помнить, что не все методы из API доступны во free-редакции, кроме того в ней сильно лимитировано кол-во запросов в минуту (можно получить ошибку 401), поэтому количество городов, для которых будет запрашиваться прогноз погоды, не должно быть большим (не более 50). Оформление кода: 1. Программу реализовать в виде отдельных модулей, каждый из которых выполнит свою функцию. 2. Использовать объектно-ориентированный подход. Создать следующие классы: <ul style="list-style-type: none">- класс, осуществляющий логирование. Так как логирование необходимо практически во всех модулях, можно, например, реализовать его в виде класса-миксина (LoggingMixin). Либо использовать другой подход на усмотрение разработчика.- класс по работе с БД (DatabaseClass).- класс по работе с API OpenWeather (WeatherClass).- класс для получения настроек (SettingsClass). 3. Главный файл приложения должен содержать вызов функции main, в которой будет реализовано создание экземпляров классов в нужном порядке, и вызов методов для реализации работы приложения. В нем же можно реализовать парсинг аргументов CLI.