Задания к курсу по основам Python

Урок 1. Знакомство с Python

1. Поработайте с переменными, создайте несколько, выведите на экран, запросите у пользователя несколько чисел и строк и сохраните в переменные, выведите на экран.

2. Пользователь вводит время в секундах. Переведите время в часы, минуты и секунды и выведите в формате чч:мм:сс. Используйте форматирование строк.

3. Узнайте у пользователя число n. Найдите сумму чисел n + nn + nnn. Например, пользователь ввёл число 3. Считаем 3 + 33 + 333 = 369.

4. Пользователь вводит целое положительное число. Найдите самую большую цифру в числе. Для решения используйте цикл while и арифметические операции.

5. Запросите у пользователя значения выручки и издержек фирмы. Определите, с каким финансовым результатом работает фирма (прибыль — выручка больше издержек, или убыток — издержки больше выручки). Выведите соответствующее сообщение. Если фирма отработала с прибылью, вычислите рентабельность выручки (соотношение прибыли к выручке). Далее запросите численность сотрудников фирмы и определите прибыль фирмы в расчете на одного сотрудника.

6. Спортсмен занимается ежедневными пробежками. В первый день его результат составил a километров. Каждый день спортсмен увеличивал результат на 10 % относительно предыдущего. Требуется определить номер дня, на который общий результат спортсмена составить не менее b километров. Программа должна принимать значения параметров a и b и выводить одно натуральное число — номер дня.

Например: a = 2, b = 3.

Результат:

1-й день: 2  
2-й день: 2,2  
3-й день: 2,42  
4-й день: 2,66  
5-й день: 2,93  
6-й день: 3,22

Ответ: на 6-й день спортсмен достиг результата — не менее 3 км.

### Урок 2. Встроенные типы и операции с ними

### 1. Создать список и заполнить его элементами различных типов данных. Реализовать скрипт проверки типа данных каждого элемента. Использовать функцию type() для проверки типа. Элементы списка можно не запрашивать у пользователя, а указать явно, в программе.

### 2. Для списка реализовать обмен значений соседних элементов, т.е. Значениями обмениваются элементы с индексами 0 и 1, 2 и 3 и т.д. При нечетном количестве элементов последний сохранить на своем месте. Для заполнения списка элементов необходимо использовать функцию input().

### 3. Пользователь вводит месяц в виде целого числа от 1 до 12. Сообщить к какому времени года относится месяц (зима, весна, лето, осень). Напишите решения через list и через dict.

### 4. Пользователь вводит строку из нескольких слов, разделённых пробелами. Вывести каждое слово с новой строки. Строки необходимо пронумеровать. Если в слово длинное, выводить только первые 10 букв в слове.

### 5. Реализовать структуру «Рейтинг», представляющую собой не возрастающий набор натуральных чисел. У пользователя необходимо запрашивать новый элемент рейтинга. Если в рейтинге существуют элементы с одинаковыми значениями, то новый элемент с тем же значением должен разместиться после них.

#### **Подсказка. Например, набор натуральных чисел: 7, 5, 3, 3, 2.**

#### **Пользователь ввел число 3. Результат: 7, 5, 3, 3, 3, 2.**

#### **Пользователь ввел число 8. Результат: 8, 7, 5, 3, 3, 2.**

#### **Пользователь ввел число 1. Результат: 7, 5, 3, 3, 2, 1.**

#### **Набор натуральных чисел можно задать непосредственно в коде, например, my\_list = [7, 5, 3, 3, 2].**

### 6. \* Реализовать структуру данных «Товары». Она должна представлять собой список кортежей. Каждый кортеж хранит информацию об отдельном товаре. В кортеже должно быть два элемента — номер товара и словарь с параметрами (характеристиками товара: название, цена, количество, единица измерения). Структуру нужно сформировать программно, т.е. запрашивать все данные у пользователя.

#### **Пример готовой структуры:**

#### **[**

(1, {“название”: “компьютер”, “цена”: 20000, “количество”: 5, “eд”: “шт.”}),

(2, {“название”: “принтер”, “цена”: 6000, “количество”: 2, “eд”: “шт.”}),

(3, {“название”: “сканер”, “цена”: 2000, “количество”: 7, “eд”: “шт.”})

#### **]**

### Необходимо собрать аналитику о товарах. Реализовать словарь, в котором каждый ключ — характеристика товара, например название, а значение — список значений-характеристик, например список названий товаров.

#### **Пример:**

#### **{**

“название”: [“компьютер”, “принтер”, “сканер”],

“цена”: [20000, 6000, 2000],

“количество”: [5, 2, 7],

“ед”: [“шт.”]

#### **}**

### Урок 3. Функции

### 1. Реализовать функцию, принимающую два числа (позиционные аргументы) и выполняющую их деление. Числа запрашивать у пользователя, предусмотреть обработку ситуации деления на ноль.

### 2. Реализовать функцию, принимающую несколько параметров, описывающих данные пользователя: имя, фамилия, год рождения, город проживания, email, телефон. Функция должна принимать параметры как именованные аргументы. Реализовать вывод данных о пользователе одной строкой.

### 3. Реализовать функцию my\_func(), которая принимает три позиционных аргумента, и возвращает сумму наибольших двух аргументов.

### 4. Программа принимает действительное положительное число x и целое отрицательное число y. Необходимо выполнить возведение числа x в степень y. Задание необходимо реализовать в виде функции my\_func(x, y). При решении задания необходимо обойтись без встроенной функции возведения числа в степень.

#### **Подсказка: попробуйте решить задачу двумя способами. Первый — возведение в степень с помощью оператора \*\*. Второй — более сложная реализация без оператора \*\*, предусматривающая использование цикла.**

### 5. Программа запрашивает у пользователя строку чисел, разделенных пробелом. При нажатии Enter должна выводиться сумма чисел. Пользователь может продолжить ввод чисел, разделенных пробелом и снова нажать Enter. Сумма вновь введенных чисел будет добавляться к уже подсчитанной сумме. Но если вместо числа вводится специальный символ, выполнение программы завершается. Если специальный символ введен после нескольких чисел, то вначале нужно добавить сумму этих чисел к полученной ранее сумме и после этого завершить программу.

### 6. Реализовать функцию int\_func(), принимающую слово из маленьких латинских букв и возвращающую его же, но с прописной первой буквой. Например, print(int\_func(‘text’)) -> Text.

### Продолжить работу над заданием. В программу должна попадать строка из слов, разделенных пробелом. Каждое слово состоит из латинских букв в нижнем регистре. Сделать вывод исходной строки, но каждое слово должно начинаться с заглавной буквы. Необходимо использовать написанную ранее функцию int\_func().

### Урок 4. Полезные инструменты

### 1. Реализовать скрипт, в котором должна быть предусмотрена функция расчета заработной платы сотрудника. В расчете необходимо использовать формулу: (выработка в часах \* ставка в час) + премия. Для выполнения расчета для конкретных значений необходимо запускать скрипт с параметрами.

### 2. Представлен список чисел. Необходимо вывести элементы исходного списка, значения которых больше предыдущего элемента.

### ****Подсказка:**** элементы, удовлетворяющие условию, оформить в виде списка. Для формирования списка использовать генератор.

### Пример исходного списка: [300, 2, 12, 44, 1, 1, 4, 10, 7, 1, 78, 123, 55].

### Результат: [12, 44, 4, 10, 78, 123].

### 3. Для чисел в пределах от 20 до 240 найти числа, кратные 20 или 21. Необходимо решить задание в одну строку.

#### **Подсказка: использовать функцию range() и генератор.**

### 4. Представлен список чисел. Определить элементы списка, не имеющие повторений. Сформировать итоговый массив чисел, соответствующих требованию. Элементы вывести в порядке их следования в исходном списке. Для выполнения задания обязательно использовать генератор.

### Пример исходного списка: [2, 2, 2, 7, 23, 1, 44, 44, 3, 2, 10, 7, 4, 11].

### Результат: [23, 1, 3, 10, 4, 11]

### 5. Реализовать формирование списка, используя функцию range() и возможности генератора. В список должны войти четные числа от 100 до 1000 (включая границы). Необходимо получить результат вычисления произведения всех элементов списка.

### ****Подсказка:**** использовать функцию reduce().

### 6. Реализовать два небольших скрипта:

#### **а) итератор, генерирующий целые числа, начиная с указанного,**

#### **б) итератор, повторяющий элементы некоторого списка, определенного заранее.**

#### **Подсказка: использовать функцию count() и cycle() модуля itertools. Обратите внимание, что создаваемый цикл не должен быть бесконечным. Необходимо предусмотреть условие его завершения.**

### Например, в первом задании выводим целые числа, начиная с 3, а при достижении числа 10 завершаем цикл. Во втором также необходимо предусмотреть условие, при котором повторение элементов списка будет прекращено.

### 7. Реализовать генератор с помощью функции с ключевым словом yield, создающим очередное значение. При вызове функции должен создаваться объект-генератор. Функция должна вызываться следующим образом: for el in fact(n). Функция отвечает за получение факториала числа, а в цикле необходимо выводить только первые n чисел, начиная с 1! и до n!.

#### **Подсказка: факториал числа n — произведение чисел от 1 до n. Например, факториал четырёх 4! = 1 \* 2 \* 3 \* 4 = 24.**

### Урок 5. Работа с файлами

### 1. Создать программно файл в текстовом формате, записать в него построчно данные, вводимые пользователем. Об окончании ввода данных свидетельствует пустая строка.

### 2. Создать текстовый файл (не программно), сохранить в нем несколько строк, выполнить подсчет количества строк, количества слов в каждой строке.

### 3. Создать текстовый файл (не программно), построчно записать фамилии сотрудников и величину их окладов. Определить, кто из сотрудников имеет оклад менее 20 тыс., вывести фамилии этих сотрудников. Выполнить подсчет средней величины дохода сотрудников.

### 4. Создать (не программно) текстовый файл со следующим содержимым:

#### **One — 1**

#### **Two — 2**

### Three — 3

#### **Four — 4**

### Необходимо написать программу, открывающую файл на чтение и считывающую построчно данные. При этом английские числительные должны заменяться на русские. Новый блок строк должен записываться в новый текстовый файл.

### 5. Создать (программно) текстовый файл, записать в него программно набор чисел, разделенных пробелами. Программа должна подсчитывать сумму чисел в файле и выводить ее на экран.

### 6. Необходимо создать (не программно) текстовый файл, где каждая строка описывает учебный предмет и наличие лекционных, практических и лабораторных занятий по этому предмету и их количество. Важно, чтобы для каждого предмета не обязательно были все типы занятий. Сформировать словарь, содержащий название предмета и общее количество занятий по нему. Вывести словарь на экран.

#### **Примеры строк файла:**

Информатика: 100(л) 50(пр) 20(лаб).  
Физика: 30(л) — 10(лаб)  
Физкультура: — 30(пр) —

#### **Пример словаря:**

{“Информатика”: 170, “Физика”: 40, “Физкультура”: 30}

### 7. Создать (не программно) текстовый файл, в котором каждая строка должна содержать данные о фирме: название, форма собственности, выручка, издержки.

#### **Пример строки файла: firm\_1 ООО 10000 5000.**

### Необходимо построчно прочитать файл, вычислить прибыль каждой компании, а также среднюю прибыль. Если фирма получила убытки, в расчет средней прибыли ее не включать.

### Далее реализовать список. Он должен содержать словарь с фирмами и их прибылями, а также словарь со средней прибылью. Если фирма получила убытки, также добавить ее в словарь (со значением убытков).

#### **Пример списка: [{“firm\_1”: 5000, “firm\_2”: 3000, “firm\_3”: 1000}, {“average\_profit”: 2000}].**

### Итоговый список сохранить в виде json-объекта в соответствующий файл.

#### **Пример json-объекта:**

[{"firm\_1": 5000, "firm\_2": 3000, "firm\_3": 1000}, {"average\_profit": 2000}]

#### **Подсказка: использовать менеджеры контекста.**

### Урок 6. Объектно-ориентированное программирование

### 1. Создать класс TrafficLight (светофор) и определить у него один атрибут color (цвет) и метод running (запуск). Атрибут реализовать как приватный. В рамках метода реализовать переключение светофора в режимы: красный, желтый, зеленый. Продолжительность первого состояния (красный) составляет 7 секунд, второго (желтый) — 2 секунды, третьего (зеленый) — на ваше усмотрение. Переключение между режимами должно осуществляться только в указанном порядке (красный, желтый, зеленый). Проверить работу примера, создав экземпляр и вызвав описанный метод.

### Задачу можно усложнить, реализовав проверку порядка режимов, и при его нарушении выводить соответствующее сообщение и завершать скрипт.

### 2. Реализовать класс Road (дорога), в котором определить атрибуты: length (длина), width (ширина). Значения данных атрибутов должны передаваться при создании экземпляра класса. Атрибуты сделать защищенными. Определить метод расчета массы асфальта, необходимого для покрытия всего дорожного полотна. Использовать формулу: длина \* ширина \* масса асфальта для покрытия одного кв метра дороги асфальтом, толщиной в 1 см \* чи сло см толщины полотна. Проверить работу метода.

### Например: 20м \* 5000м \* 25кг \* 5см = 12500 т

### 3. Реализовать базовый класс Worker (работник), в котором определить атрибуты: name, surname, position (должность), income (доход). Последний атрибут должен быть защищенным и ссылаться на словарь, содержащий элементы: оклад и премия, например, {"wage": wage, "bonus": bonus}. Создать класс Position (должность) на базе класса Worker. В классе Position реализовать методы получения полного имени сотрудника (get\_full\_name) и дохода с учетом премии (get\_total\_income). Проверить работу примера на реальных данных (создать экземпляры класса Position, передать данные, проверить значения атрибутов, вызвать методы экземпляров).

### 4. Реализуйте базовый класс Car. У данного класса должны быть следующие атрибуты: speed, color, name, is\_police (булево). А также методы: go, stop, turn(direction), которые должны сообщать, что машина поехала, остановилась, повернула (куда). Опишите несколько дочерних классов: TownCar, SportCar, WorkCar, PoliceCar. Добавьте в базовый класс метод show\_speed, который должен показывать текущую скорость автомобиля. Для классов TownCar и WorkCar переопределите метод show\_speed. При значении скорости свыше 60 (TownCar) и 40 (WorkCar) должно выводиться сообщение о превышении скорости.

### Создайте экземпляры классов, передайте значения атрибутов. Выполните доступ к атрибутам, выведите результат. Выполните вызов методов и также покажите результат.

### 5. Реализовать класс Stationery (канцелярская принадлежность). Определить в нем атрибут title (название) и метод draw (отрисовка). Метод выводит сообщение “Запуск отрисовки.” Создать три дочерних класса Pen (ручка), Pencil (карандаш), Handle (маркер). В каждом из классов реализовать переопределение метода draw. Для каждого из классов методы должен выводить уникальное сообщение. Создать экземпляры классов и проверить, что выведет описанный метод для каждого экземпляра.

### Урок 7. ООП. Продвинутый уровень

### 1. Реализовать класс Matrix (матрица). Обеспечить перегрузку конструктора класса (метод \_\_init\_\_()), который должен принимать данные (список списков) для формирования матрицы.

#### **Подсказка: матрица — система некоторых математических величин, расположенных в виде прямоугольной схемы.**

#### **Примеры матриц вы найдете в методичке.**

### Следующий шаг — реализовать перегрузку метода \_\_str\_\_() для вывода матрицы в привычном виде.

### Далее реализовать перегрузку метода \_\_add\_\_() для реализации операции сложения двух объектов класса Matrix (двух матриц). Результатом сложения должна быть новая матрица.

#### **Подсказка: сложение элементов матриц выполнять поэлементно — первый элемент первой строки первой матрицы складываем с первым элементом первой строки второй матрицы и т.д.**

### 2. Реализовать проект расчета суммарного расхода ткани на производство одежды. Основная сущность (класс) этого проекта — одежда, которая может иметь определенное название. К типам одежды в этом проекте относятся пальто и костюм. У этих типов одежды существуют параметры: размер (для пальто) и рост (для костюма). Это могут быть обычные числа: V и H, соответственно.

### Для определения расхода ткани по каждому типу одежды использовать формулы: для пальто (V/6.5 + 0.5), для костюма (2 \* H + 0.3). Проверить работу этих методов на реальных данных.

### Реализовать общий подсчет расхода ткани. Проверить на практике полученные на этом уроке знания: реализовать абстрактные классы для основных классов проекта, проверить на практике работу декоратора @property.

### 3. Реализовать программу работы с органическими клетками. Необходимо создать класс Клетка. В его конструкторе инициализировать параметр, соответствующий количеству клеток (целое число). В классе должны быть реализованы методы перегрузки арифметических операторов: сложение (\_\_add\_\_()), вычитание (\_\_sub\_\_()), умножение (\_\_mul\_\_()), деление (\_\_truediv\_\_()).Данные методы должны применяться только к клеткам и выполнять увеличение, уменьшение, умножение и обычное (не целочисленное) деление клеток, соответственно. В методе деления должно осуществляться округление значения до целого числа.

### ****Сложение.**** Объединение двух клеток. При этом число ячеек общей клетки должно равняться сумме ячеек исходных двух клеток.

### ****Вычитание.**** Участвуют две клетки. Операцию необходимо выполнять только если разность количества ячеек двух клеток больше нуля, иначе выводить соответствующее сообщение.

### ****Умножение.**** Создается общая клетка из двух. Число ячеек общей клетки определяется как произведение количества ячеек этих двух клеток.

### ****Деление.**** Создается общая клетка из двух. Число ячеек общей клетки определяется как целочисленное деление количества ячеек этих двух клеток.

### В классе необходимо реализовать метод make\_order(), принимающий экземпляр класса и количество ячеек в ряду. Данный метод позволяет организовать ячейки по рядам.

### Метод должен возвращать строку вида \*\*\*\*\*\n\*\*\*\*\*\n\*\*\*\*\*..., где количество ячеек между \n равно переданному аргументу. Если ячеек на формирование ряда не хватает, то в последний ряд записываются все оставшиеся.

### Например, количество ячеек клетки равняется 12, количество ячеек в ряду — 5. Тогда метод make\_order() вернет строку: \*\*\*\*\*\n\*\*\*\*\*\n\*\*.

### Или, количество ячеек клетки равняется 15, количество ячеек в ряду — 5. Тогда метод make\_order() вернет строку: \*\*\*\*\*\n\*\*\*\*\*\n\*\*\*\*\*.

### Урок 8. ООП. Полезные дополнения

### 1. Реализовать класс «Дата», функция-конструктор которого должна принимать дату в виде строки формата «день-месяц-год». В рамках класса реализовать два метода. Первый, с декоратором @classmethod, должен извлекать число, месяц, год и преобразовывать их тип к типу «Число». Второй, с декоратором @staticmethod, должен проводить валидацию числа, месяца и года (например, месяц — от 1 до 12). Проверить работу полученной структуры на реальных данных.

### 2. Создайте собственный класс-исключение, обрабатывающий ситуацию деления на нуль. Проверьте его работу на данных, вводимых пользователем. При вводе пользователем нуля в качестве делителя программа должна корректно обработать эту ситуацию и не завершиться с ошибкой.

### 3. Создайте собственный класс-исключение, который должен проверять содержимое списка на наличие только чисел. Проверить работу исключения на реальном примере. Необходимо запрашивать у пользователя данные и заполнять список. Класс-исключение должен контролировать типы данных элементов списка.

### Примечание: длина списка не фиксирована. Элементы запрашиваются бесконечно, пока пользователь сам не остановит работу скрипта, введя, например, команду “stop”. При этом скрипт завершается, сформированный список выводится на экран.

#### **Подсказка: для данного задания примем, что пользователь может вводить только числа и строки. При вводе пользователем очередного элемента необходимо реализовать проверку типа элемента и вносить его в список, только если введено число. Класс-исключение должен не позволить пользователю ввести текст (не число) и отобразить соответствующее сообщение. При этом работа скрипта не должна завершаться.**

### 4. Начните работу над проектом «Склад оргтехники». Создайте класс, описывающий склад. А также класс «Оргтехника», который будет базовым для классов-наследников. Эти классы — конкретные типы оргтехники (принтер, сканер, ксерокс). В базовом классе определить параметры, общие для приведенных типов. В классах-наследниках реализовать параметры, уникальные для каждого типа оргтехники.

### 5. Продолжить работу над первым заданием. Разработать методы, отвечающие за приём оргтехники на склад и передачу в определенное подразделение компании. Для хранения данных о наименовании и количестве единиц оргтехники, а также других данных, можно использовать любую подходящую структуру, например словарь.

### 6. Продолжить работу над вторым заданием. Реализуйте механизм валидации вводимых пользователем данных. Например, для указания количества принтеров, отправленных на склад, нельзя использовать строковый тип данных.

#### **Подсказка: постарайтесь по возможности реализовать в проекте «Склад оргтехники» максимум возможностей, изученных на уроках по ООП.**

### 7. Реализовать проект «Операции с комплексными числами». Создайте класс «Комплексное число», реализуйте перегрузку методов сложения и умножения комплексных чисел. Проверьте работу проекта, создав экземпляры класса (комплексные числа) и выполнив сложение и умножение созданных экземпляров. Проверьте корректность полученного результата.