## Циклы и условия. Практическое задание

### 1: Запросите от пользователя число, сохраните в переменную, прибавьте к числу 2 и выведите результат на экран. Если возникла ошибка, прочитайте ее, вспомните урок и постарайтесь устранить ошибку.

### 2: Используя цикл, запрашивайте у пользователя число, пока оно не станет больше 0, но меньше 10.

### После того, как пользователь введет корректное число, возведите его в степень 2 и выведите на экран.

### Например, пользователь вводит число 123, вы сообщаете ему, что число неверное, и говорите о диапазоне допустимых. И просите ввести заново.

### Допустим, пользователь ввел 2, оно подходит. Возводим его в степень 2 и выводим 4.

### 3: Создайте программу “Медицинская анкета”, где вы запросите у пользователя следующие данные: имя, фамилия, возраст и вес.

### Выведите результат согласно которому:

* Пациент в хорошем состоянии, если ему до 30 лет и вес от 50 и до 120 кг,
* Пациенту требуется заняться собой, если ему более 30 и вес меньше 50 или больше 120 кг
* Пациенту требуется врачебный осмотр, если ему более 40 и вес менее 50 или больше 120 кг.
* Все остальные варианты вы можете обработать на ваш вкус и полет фантазии.

### (Формула не соответствует реальной действительности и здесь используется только ради примера)

Примечание: при написание программы обратите внимание на условия в задаче и в вашем коде. Протестируйте программу несколько раз и убедитесь, что проверки срабатывают верно. В случае ошибок, уточните условия для той или иной ситуации.

Пример: Вася Пупкин, 29 год, вес 90 - хорошее состояние  
Пример: Вася Пупкин, 31 год, вес 121 - следует заняться собой  
Пример: Вася Пупкин, 31 год, вес 49 - следует заняться собой  
Пример: Вася Пупкин, 41 год, вес 121 - следует обратится к врачу!  
Пример: Вася Пупкин, 41 год, вес 49 - следует обратится к врачу!

## Тема: встроенные типы и операции с ними

## Практическое задание

### 1: Даны два произвольные списка. Удалите из первого списка элементы присутствующие во втором списке.

Примечание. Списки создайте вручную, например так:

my\_list\_1 = [2, 5, 8, 2, 12, 12, 4]  
my\_list\_2 = [2, 7, 12, 3]

### 2: Дана дата в формате dd.mm.yyyy, например: 02.11.2013. Ваша задача — вывести дату в текстовом виде, например: второе ноября 2013 года. Склонением пренебречь (2000 года, 2010 года)

### 3: Дан список заполненный произвольными целыми числами.

### Получите новый список, элементами которого будут только уникальные элементы исходного.

Примечание. Списки создайте вручную, например так:

my\_list\_1 = [2, 2, 5, 12, 8, 2, 12]

### В этом случае ответ будет:

[5, 8]

Практическое задание: игра «Угадай число»

1. В этой игре человек загадывает число, а компьютер пытается его угадать.

В начале игры человек загадывает число от 1 до 100 в уме или записывает его на листок бумаги. Компьютер начинает его отгадывать предлагая игроку варианты чисел. Если компьютер угадал число, игрок выбирает “победа”. Если компьютер назвал число меньше загаданного, игрок должен выбрать “загаданное число больше”. Если компьютер назвал число больше, игрок должен выбрать “загаданное число меньше”. Игра продолжается до тех пор пока компьютер не отгадает число.

Пример игры:

Допустим, пользователь загадал число 42

`15

35

96  
<  
37

74  
<  
52  
<  
42  
=`

Примечание: Знаки “=”, “>” и “<” пользователь вводит с клавиатуры для общения с компьютером. Вы можете использовать этот способ или придумать свой.

Функции. Практическое задание

1: Создайте функцию, принимающую на вход имя, возраст и город проживания человека. Функция должна возвращать строку вида «Василий, 21 год(а), проживает в городе Москва»

2: Создайте функцию, принимающую на вход 3 числа и возвращающую наибольшее из них.

3: Давайте опишем пару сущностей player и enemy через словарь, который будет иметь ключи и значения:

* name - строка полученная от пользователя,
* health = 100,
* damage = 50. ### Поэкспериментируйте с значениями урона и жизней по желанию. ### Теперь надо создать функцию attack(person1, person2). Примечание: имена аргументов можете указать свои. ### Функция в качестве аргумента будет принимать атакующего и атакуемого. ### В теле функция должна получить параметр damage атакующего и отнять это количество от health атакуемого. Функция должна сама работать со словарями и изменять их значения.

4: Давайте усложним предыдущее задание. Измените сущности, добавив новый параметр - armor = 1.2 (величина брони персонажа)

Теперь надо добавить новую функцию, которая будет вычислять и возвращать полученный урон по формуле damage / armor

Следовательно, у вас должно быть 2 функции:

1. Наносит урон. Это улучшенная версия функции из задачи 3.
2. Вычисляет урон по отношению к броне.

Примечание. Функция номер 2 используется внутри функции номер 1 для вычисления урона и вычитания его из здоровья персонажа.

## Модули и библиотеки. Практическое задание

### 1: Создайте модуль (модуль - программа на Python, т.е. файл с расширением .py). В нем создайте функцию создающую директории от dir\_1 до dir\_9 в папке из которой запущен данный код. Затем создайте вторую функцию удаляющую эти папки. Проверьте работу функций в этом же модуле.

### 2: Создайте модуль. В нем создайте функцию, которая принимает список и возвращает из него случайный элемент. Если список пустой функция должна вернуть None. Проверьте работу функций в этом же модуле.

Примечание: Список для проверки введите вручную. Или возьмите этот: [1, 2, 3, 4]

### 3: Создайте модуль main.py. Из модулей реализованных в заданиях 1 и 2 сделайте импорт в main.py всех функций. Вызовите каждую функцию в main.py и проверьте что все работает как надо.

Примечание: Попробуйте импортировать как весь модуль целиком (например из задачи 1), так и отдельные функции из модуля.

## Работа с файлами кодировки. Практическое задание

### 1: Создать модуль music\_serialize.py. В этом модуле определить словарь для вашей любимой музыкальной группы, например:

my\_favourite\_group = {  
‘name’: ‘Г.М.О.’,  
‘tracks’: [‘Последний месяц осени’, ‘Шапито’],  
‘Albums’: [{‘name’: ‘Делать панк-рок’,‘year’: 2016},  
{‘name’: ‘Шапито’,‘year’: 2014}]}

### С помощью модулей json и pickle сериализовать данный словарь в json и в байты, вывести результаты в терминал. Записать результаты в файлы group.json, group.pickle соответственно. В файле group.json указать кодировку utf-8.

### 2: Создать модуль music\_deserialize.py. В этом модуле открыть файлы group.json и group.pickle, прочитать из них информацию. И получить объект: словарь из предыдущего задания.