**Лабораторная работа 3.1. Работа с массивами данных. Подпрограммы**

**Цель работы**: научиться работать с функциями и подпрограммами в Python, а также исследовать их преимущества, особенности применения и возможные недостатки. Закрепить навыки создания, вызова и применения различных видов функций, включая глобальные, локальные, рекурсивные и анонимные.

**Задачи**:

* Изучить теоретические основы подпрограмм и функций в Python:

- понятие подпрограмм;

- механизм их работы;

- область видимости и взаимодействие глобальных и локальных функций.

* Изучить особенности позиционных, ключевых параметров, упаковки и распаковки аргументов.
* Закрепить навыки использования рекурсии для решения практических задач.
* Познакомиться с концепцией анонимных функций (lambda) и их применением.
* Исследовать влияние побочных эффектов на работу функций.
* Решить практические задачи, применяя знания по теме.

**Примечание**

При выполнении заданий используйте заготовки решений: -> [Репозиторий](https://gitverse.ru/yuripetrov/python-course-tasks).

**Примечание**

При решении задач данного раздела обратите внимание:

* для каждой задачи данного раздела сначала создаются функции, а затем, используя их, выполняется решение задачи;
* ввод и вывод данных для функции необходимо обеспечить самостоятельно;
* для каждой написанной функции нужно:
  + избегать побочного эффекта;
  + использовать строки документации;
* при выводе вещественных результатов оставьте два знака после запятой.

В заданиях, где функции используются несколько раз, подумайте, насколько бы увеличился объем кода без их использования.

* Изучить теоретические основы раздела[Подпрограммы](https://www.yuripetrov.ru/edu/python/ch_05.html)

<https://www.yuripetrov.ru/edu/python/ch_05_01.html>

**Теоретические основы**

**3.1.1** Основные понятия и механизм работы

**3.1.1.1** Определение подпрограммы

**Подпрограмма (или функция)** — это именованный блок кода, который выполняет определённую задачу и может быть вызван из других частей программы. Основное назначение функций — избегать дублирования кода и повышать читаемость программ.

**3.1.1.2 Вызов подпрограммы**

Для вызова функции используется её имя, за которым следуют круглые скобки с аргументами (если они есть). Например:

def greet(name):

print(f"Hello, {name}!")

greet("Alice")

**3.1.1.3** Механизм работы

1. Определение функции с помощью ключевого слова **def**.

2. Передача аргументов и обработка параметров.

3. Выполнение тела функции.

4. Возврат значения (при необходимости) с помощью оператора **return**.

**3.1.1.4** Преимущества и недостатки

Преимущества:

- Уменьшение дублирования кода.

- Повышение читаемости и структурированности.

- Возможность многократного использования кода.

Недостатки:

- Сложность отладки, если функции сильно связаны.

- Возможные ошибки при неправильной работе с аргументами и областями видимости.

**3.1.2** Функции в Python

Python предоставляет множество возможностей для работы с функциями:

- Объявление с использованием ключевого слова **def**.

- Поддержка глобальных и локальных функций.

- Использование встроенных функций (например, **len**, **max**, **print**).

**3.1.3** Глобальные и локальные функции

**3.1.3.1** Параметры и аргументы

**3.1.3.1.1** Позиционные и ключевые параметры/аргументы

- Позиционные параметры: порядок аргументов имеет значение.

- Ключевые параметры: задаются с указанием имени аргумента.

Пример:

def greet(name, greeting="Hello"):

print(f"{greeting}, {name}!")

greet("Alice") # Позиционный аргумент

greet(name="Bob", greeting="Hi") # Ключевой аргумент

**3.1.3.1.2** Упаковка и распаковка аргументов

Python позволяет передавать произвольное количество аргументов через упаковку:

def sum\_numbers(\*args):

return sum(args)

print(sum\_numbers(1, 2, 3, 4))

**3.1.3.2** Область видимости

Переменные внутри функций имеют локальную область видимости, если не указано иное.

**3.1.3.3** Возврат нескольких значений

Функции могут возвращать несколько значений, используя кортежи:

def get\_min\_max(numbers):

return min(numbers), max(numbers)

min\_val, max\_val = get\_min\_max([1, 2, 3, 4])

**3.1.3.4** Рекурсия

**Рекурсия** — это вызов функции внутри самой себя.

Пример:

def factorial(n):

if n == 1:

return 1

return n \* factorial(n - 1)

print(factorial(5))

**3.1.3.5** Строки документации

Документация функций оформляется с помощью строк:

def greet(name):

"""Функция для приветствия пользователя."""

print(f"Hello, {name}!")

**3.1.4** Анонимные функции

**Анонимные функции (или lambda-функции)** — это краткая запись функций, которые не требуют имени:

square = lambda x: x 2

print(square(4))

**3.1.5** Побочный эффект

**Побочные эффекты** — это изменения состояния программы вне функции (например, изменение глобальных переменных).

**Контрольные вопросы**

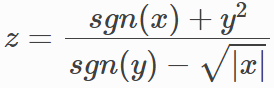
* + - Что такое подпрограмма? Какие преимущества она даёт?
    - Чем отличается глобальная область видимости от локальной?
    - Как работают позиционные и ключевые аргументы?
    - В чём заключаются особенности рекурсивных функций?
    - Что такое lambda-функция и как её применять?

**Варианты заданий**

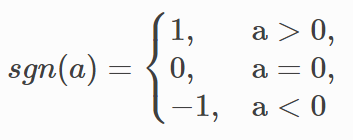
## [Тема Глобальные и локальные функции](https://www.yuripetrov.ru/edu/python/ch_05_02.html#id13)[¶](https://www.yuripetrov.ru/edu/python/ch_05_02.html#id4)

**№ 3.1.1**

Определите значение



где sgn - [сигнум-функция](https://ru.ruwiki.ru/wiki/Sgn" \t "_blank), а x и y - вещественные числа:



**№ 3.1.2**

Дан список температурных изменений в течение дня (целые числа). Известно, что измеряющее устройство иногда сбоит и записывает отсутствие температуры (значение **[None](https://www.yuripetrov.ru/edu/python/ch_03_01.html" \l "None" \o "None)**).

Выведите среднюю температуру за наблюдаемый промежуток времени, предварительно очистив список от неопределенных значений. Гарантируется, что хотя бы одно определенное значение в списке есть.

**№ 3.1.3**

Выведите все счастливые номера билетов в диапазоне от a до b (положительные целые числа, a<b), если известно, что счастливым считается номер, у которого количество четных цифр равно количеству нечетных.

**№ 3.1.4**

Дата характеризуется тремя натуральными числами: день, месяц и год. Учитывая, что год может быть високосным, реализуйте две функции, которые определяют вчерашнюю и завтрашнюю дату.

**№ 3.1.5**

В [задаче № 3.1.4](https://www.yuripetrov.ru/edu/python/ch_05_02.html#ch-05-02-04) замените функции получения вчерашней и завтрашней даты на одну:

**def** another\_date(day, month, year, delta=1):

**pass**

где delta - ключевой параметр, определяющий сколько дней необходимо добавить или вычесть (если аргумент отрицательный) из переданной даты. Функции, реализованные в [задаче № 5.2.4](https://www.yuripetrov.ru/edu/python/ch_05_02.html#ch-05-02-04) сделайте локальными для another\_date(), вызывая их внутри необходимое количество раз.

**№ 3.1.6**

Дан список из чисел.

Определите их [НОК](https://ru.ruwiki.ru/wiki/%D0%9D%D0%B0%D0%B8%D0%BC%D0%B5%D0%BD%D1%8C%D1%88%D0%B5%D0%B5_%D0%BE%D0%B1%D1%89%D0%B5%D0%B5_%D0%BA%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5) (наименьшее общее кратное) и [НОД](https://ru.ruwiki.ru/wiki/%D0%9D%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B9_%D0%BE%D0%B1%D1%89%D0%B8%D0%B9_%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C) (наибольший общий делитель).

**№ 3.1.7**

Даны n предложений. Определите, сколько из них содержат хотя бы одну цифру.

**№ 3.1.8**

Дана строка s и символ k. Реализуйте функцию, рисующую рамку из символа k вокруг данной строки, например:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Текст в рамке\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**№ 3.1.9**

Напишите функции для перевода числа (N∈[2;16]):

* из 10-й системы счисления в N-ю;
* из N-й системы счисления в 10-ю.

**№ 3.1.10**

Для введенного предложения выведите статистику символ=количество. Регистр букв не учитывается.

**№ 3.1.11**

Используя [шифр Цезаря](https://ru.ruwiki.ru/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F) (достаточно только букв русского алфавита, знаки препинания не изменяются), зашифруйте, а затем расшифруйте введенную строку.

**№ 3.1.12**

В вагоне-[купе](https://ru.ruwiki.ru/wiki/%D0%9A%D1%83%D0%BF%D0%B5_(%D0%B2%D0%B0%D0%B3%D0%BE%D0%BD)) имеется некоторое количество купе, в каждом из которых по 4 места. Разработчик хранит информацию о занятости одного купе в виде словаря:

{1: 'м', 2: **None**, 3: **None**, 4: 'ж'}

где:

* ключ определяет номер места (нечетные номера - нижние места, четные - верхние);
* значение может быть одно из трех: **[None](https://www.yuripetrov.ru/edu/python/ch_03_01.html" \l "None" \o "None)**, "м" и "ж", если место не занято, занято мужчиной или женщиной соответственно.

Информация о занятости всего вагона хранится как список указанных словарей.

Определите:

* список полностью свободных купе;
* список свободных мест в вагоне;
* список свободных нижних или верхних мест;
* список свободных мест в купе с исключительно мужской компанией;
* список свободных мест в купе с исключительно женской компанией.

**Совет**

Купе считается с исключительно мужской компанией, если:

* занято хотя бы одно место;
* все несвободные места заняты мужчинами.

**№ 3.1.13**

Дан список с результатами голосования на выборах в виде:

[1, 3, 2, 2, 2, 5, -1, ...]

где номер определяет голос за партию из списка:

1. Партия №1.

2. Партия №2.

3. Партия №3.

4. Партия №4.

5. Партия №5.

-1. Испорченный бланк.

Подведите итоги выборов, выведя на экран список партий в соответствии с убыванием количества полученных голосов и их процентным соотношением:

1. Партия №2 | 1111 | 58.21%

2. Партия №4 | 999 | 38.14%

...

## [Тема Упаковка и распаковка аргументов](https://www.yuripetrov.ru/edu/python/ch_05_02.html#id14)

**№ 3.1.14**

Напишите функцию, которая принимает неограниченное количество числовых аргументов и возвращает кортеж из двух списков:

* отрицательных значений (отсортирован по убыванию);
* неотрицательных значений (отсортирован по возрастанию).

**№ 3.1.15**

[Валовой внутренний продукт](https://ru.ruwiki.ru/wiki/%D0%92%D0%B0%D0%BB%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B2%D0%BD%D1%83%D1%82%D1%80%D0%B5%D0%BD%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82) может определяться по расходам:

Y=C+I+G+Ex−Im,

где:

* C - конечное потребление;
* I - валовое накопление капитала (инвестиции в фирму, то есть покупка станков, оборудования, запасов, места производства);
* G - государственные расходы;
* Ex - экспорт;
* Im - импорт.

Все величины - целые числа.

Выполните:

* введите с клавиатуры все 5 значений формулы и сохраните их в список и словарь вида
* {'c': 3, 'i': 1, ..., 'im': 2}
* напишите функцию для вычисления ВВП, которая содержит 5 параметров, соответствующих формуле;
* используя распаковку аргументов, вызовите функцию с данными введенных списка и словаря и выведите их на экран.

## [Тема Рекурсия](https://www.yuripetrov.ru/edu/python/ch_05_02.html#id15)

**№ 3.1.16**

Составьте две функции для возведения числа в степень: один из вариантов реализуйте в рекурсивном стиле.

**№ 3.1.17**

Дано натуральное число. Напишите рекурсивные функции для определения:

* суммы цифр числа;
* количества цифр в числе.