**21.8 Практическая работа**

Цели практической работы

* Отработать навык использования рекурсивного метода для решения задач, поработать со стеком вызовов.
* Научиться находить и исправлять возможные проблемы, которые могут возникнуть при передаче изменяемых типов данных в качестве параметров функции.
* Научиться отличать именованные аргументы и значения по умолчанию, уметь использовать их при написании функций, чтобы избежать повторов и писать более гибкий код.
* Отработать применение аргументов \*args и \*\*kwargs для решения задач.

Что входит в работу

Задача 1. Challenge 2.

Задача 2. Свой zip 2.

Задача 3. Ряд Фибоначчи.

Задача 4. Поиск элемента 2.

Задача 5. Ускоряем работу функции.

Задача 6. Глубокое копирование.

Задача 7. Продвинутая функция sum.

Задача 8. Список списков 2.

Задача 9. Ханойские башни.

Задача 1. Challenge 2

**Что нужно сделать**

Вдохновившись мотивацией Антона, ваш друг тоже решил поставить перед собой задачу, но не сильно связанную с математикой, а именно: написать функцию, которая выводит все числа от 1 до num без использования циклов. Помогите другу реализовать такую функцию.

Пример работы программы:

Введите num: 10

1

2

3

4

5

6

7

8

9

10

**Что оценивается**

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Input содержит корректные приглашения для ввода.
* Основной функционал описан в отдельной функции(-ях).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

Задача 2. Свой zip 2

**Что нужно сделать**

Написав аналог функции zip на собеседовании, вы вспомнили, что этот код можно сделать даже лучше, и резко вырвали листок с кодом из рук работодателя, оставив его в недоумении.

Напишите функцию, которая будет являться аналогом функции zip и сделайте так, чтобы программа работала с любыми итерируемыми типами данных.

* Нельзя пользоваться циклами (за исключением генераторов), условными операторами и функциями определения типа (isinstance или type).
* Функция должна возвращать  список с кортежами, в котором n-ый кортеж состоит из n-ых элементов всех последовательностей. Например, десятый кортеж будет содержать десятый элемент каждой из переданных последовательностей
* Основной код оставьте пустым или закомментированным (используйте его только для тестирования).

**Что оценивается**

* Результат вычислений корректен.
* Основной функционал описан в отдельной функции(-ях).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

Задача 3. Ряд Фибоначчи

**Что нужно сделать**

Числа Фибоначчи — это ряд чисел, в котором каждое следующее число равно сумме двух предыдущих: 1, 1, 2, 3, 5, 8, 13…

Пользователь вводит num\_pos. Напишите функцию, которая выводит число, стоящее на позиции num\_pos в ряде Фибоначчи.

**Пример 1:**

Введите позицию числа в ряде Фибоначчи: 6

Число: 8

**Пример 2:**

Введите позицию числа в ряде Фибоначчи: 10

Число: 55

**Что оценивается**

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Input содержит корректные приглашения для ввода.
* Основной функционал описан в отдельной функции(-ях).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

Задача 4. Поиск элемента 2

**Что нужно сделать**

Пользователь вводит искомый ключ. Если он хочет, то может ввести максимальную глубину — уровень, до которого будет просматриваться структура.

Напишите функцию, которая находит заданный пользователем ключ в словаре и выдаёт значение этого ключа на экран. По умолчанию уровень не задан. В качестве примера можно использовать такой словарь:

site = {  
    'html': {  
        'head': {  
            'title': 'Мой сайт'  
        },  
        'body': {  
            'h2': 'Здесь будет мой заголовок',  
            'div': 'Тут, наверное, какой-то блок',  
            'p': 'А вот здесь новый абзац'  
        }  
    }  
}

**Пример 1:**

Введите искомый ключ: head

Хотите ввести максимальную глубину? Y/N: n

Значение ключа: {'title': 'Мой сайт'}

**Пример 2:**

Введите искомый ключ: head

Хотите ввести максимальную глубину? Y/N: y

Введите максимальную глубину: 1

Значение ключа: None

**Что оценивается**

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Input содержит корректные приглашения для ввода.
* Основной функционал описан в отдельной функции(-ях).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

Задача 5. Ускоряем работу функции

**Что нужно сделать**

У нас есть функция, которая делает определённые действия с входными данными:

1. берёт факториал от числа;
2. результат делит на куб входного числа;
3. получившееся число возводит в 10-ю степень.

def calculating\_math\_func(data):  
    result = 1  
    for index in range(1, data + 1):  
        result \*= index  
    result /= data \*\* 3  
    result = result \*\* 10  
    return result

Однако каждый раз нам приходится делать сложные вычисления, хотя входные и выходные данные одни и те же. И тут наши знания тонкостей Python должны нам помочь.

Оптимизируйте функцию так, чтобы высчитывать факториал для одного и того же числа только один раз.

Подсказка: вспомните, что происходит с изменяемыми данными, если их выставить по умолчанию в параметрах функции.

**Что оценивается**

* Результат вычислений корректен.
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

Задача 6. Глубокое копирование

**Что нужно сделать**

Вы сделали для заказчика структуру сайта по продаже телефонов:

site = {  
    'html': {  
        'head': {  
            'title': 'Куплю/продам телефон недорого'  
        },  
        'body': {  
            'h2': 'У нас самая низкая цена на iPhone',  
            'div': 'Купить',  
            'p': ‘Продать'  
        }  
    }  
}

Заказчик рассказал своим коллегам на рынке, и они тоже захотели такой сайт, только для своих товаров. Вы посчитали, что это лёгкая задача, и быстро принялись за работу.

Напишите программу, которая запрашивает у клиента, сколько будет сайтов, а затем запрашивает название продукта и после каждого запроса выводит на экран активные сайты.

Условия: структуру сайта нужно описать один раз, копипасту никто не любит.

Подсказка: используйте рекурсию.

**Пример:**

Сколько сайтов: 2

Введите название продукта для нового сайта: iPhone

Сайт для iPhone:

site = {  
    'html': {  
        'head': {  
            'title': 'Куплю/продам iPhone недорого'  
        },  
        'body': {  
            'h2': 'У нас самая низкая цена на iPhone',  
            'div': 'Купить',  
            'p': ‘Продать'  
        }  
    }  
}

Введите название продукта для нового сайта: Samsung

Сайт для iPhone:

site = {  
    'html': {  
        'head': {  
            'title': 'Куплю/продам iPhone недорого'  
        },  
        'body': {  
            'h2': 'У нас самая низкая цена на iPhone',  
            'div': 'Купить',  
            'p': ‘Продать'  
        }  
    }  
}  
  
Сайт для Samsung:   
site = {  
    'html': {  
        'head': {  
            'title': 'Куплю/продам Samsung недорого'  
        },  
        'body': {  
            'h2': 'У нас самая низкая цена на Samsung',  
            'div': 'Купить',  
            'p': ‘Продать'  
        }  
    }  
}

**Что оценивается**

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Input содержит корректные приглашения для ввода.
* Основной функционал описан в отдельной функции(-ях).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

Задача 7. Продвинутая функция sum

**Что нужно сделать**

Как вы знаете, в Python есть полезная функция sum, которая умеет находить сумму элементов списков. Но иногда базовых возможностей функций не хватает для работы и приходится их усовершенствовать.

Напишите свою функцию sum, которая должна быть более гибкой, чем стандартная функция sum. Она должна уметь:

* складывать числа из списка списков;
* складывать числа из набора параметров.

Основной код оставьте пустым или закомментированным (используйте его только для тестирования).

Примеры вызовов функции:

sum([[1, 2, [3]], [1], 3])

Ответ в консоли: 10

sum(1, 2, 3, 4, 5)

Ответ в консоли: 15

**Что оценивается**

* Результат вычислений корректен.
* Весь функционал описан в отдельной функции.
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

Задача 8. Список списков 2

**Что нужно сделать**

Вы уже работали с многомерными списками и решали задачи, где с помощью list comprehensions «выпрямляли» многомерные списки в один. Однако такой фокус не пройдёт, если у элементов разные уровни вложенности и этих списков неограниченное количество.

Дан такой список:

nice\_list = [1, 2, [3, 4], [[5, 6, 7], [8, 9, 10]], [[11, 12, 13], [14, 15], [16, 17, 18]]]

Напишите рекурсивную функцию, которая раскрывает все вложенные списки, то есть оставляет только внешний список.

Ответ: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]

Подсказка: можно возвращать списки и срезы списков.

**Что оценивается**

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Основной функционал описан в отдельной функции(-ях).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

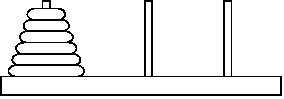
Задача 9. Ханойские башни

**Что нужно сделать**

Ниже представлен текст одной из классических головоломок и по совместительству алгоритмических задач по программированию. С ней знакомы многие, в том числе те, кто с программированием никак не связан.

Головоломка «Ханойские башни» состоит из трёх стержней, пронумерованных числами 1, 2, 3. На стержень 1 надета пирамидка из N дисков различного диаметра в порядке возрастания диаметра. Диски можно перекладывать с одного стержня на другой по одному, при этом диск нельзя класть на диск меньшего диаметра.

Необходимо переложить всю пирамидку со стержня 1 на стержень 3 за минимальное число перекладываний.



Напишите программу, которая решает головоломку — для данного числа дисков N печатает последовательность перекладываний в формате A B C, где:

A — номер перекладываемого диска;

B — номер стержня, с которого снимается данный диск;

C — номер стержня, на который надевается данный диск.

Например, строка 1 2 3 означает перемещение диска номер 1 со стержня 2 на стержень 3. В одной строке печатается одна команда. Диски пронумерованы числами от 1 до n в порядке возрастания диаметров.

Программа должна вывести минимальный (по количеству произведённых операций) способ перекладывания пирамидки из данного числа дисков.

Решение оформите в виде функции move (n, x, y), которая печатает последовательность перекладываний дисков для перемещения пирамидки высотой N со стержня номер X на стержень номер Y.

Пример:

Введите количество дисков: 2

Переложить диск 1 со стержня номер 1 на стержень номер 2

Переложить диск 2 со стержня номер 1 на стержень номер 3

Переложить диск 1 со стержня номер 2 на стержень номер 3

**Совет**: Для начала определите номер промежуточного колышка. Для этого подумайте о сумме трех чисел колышка.

**Что оценивается**

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Input содержит корректные приглашения для ввода.
* Основной функционал описан в отдельной функции(-ях).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).

Что оценивается в практической работе

* Практическая работа сдана через GitLab.
* Структура папок и файлов репозитория соответствует репозиторию python\_basic.
* Все задачи выполнены в соответствующих папках и файлах main.py.
* Описания коммитов осмыслены и понятны: 111, done, «я сделалъ» — неверно, added m15 homework, 14.3 fix: variables naming — верно.
* Использованы именованные индексы, не просто i (подробнее в видео 7.2).
* Использованы правильные числа, без дополнительных действий со стороны пользователя, без +1 (подробнее об этом в видео 7.4).
* Правильно оформлен input, без пустого приветствия для ввода (подробнее об этом в видео 2.3).
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом в видео 2.3).
* Есть пробелы после запятых и при бинарных операциях.
* Нет пробелов после имён функций и перед скобками: print (),input () — неверно, print() — верно.
* Правильно оформлены блоки if-elif-else, циклы и функции, отступы одинаковы во всех блоках одного уровня.
* Основной функционал описывается в отдельной функции(-ях).

Рекомендации

* Арифметические операции [PEP8](https://docs.python.org/3.7/reference/expressions.html#operator-precedence)остаются в приоритете. Необходимо вводить and, or.
* Руководство по стилю Python [PEP8 на английском языке](https://www.python.org/dev/peps/pep-0008/).
* Руководство по стилю Python [PEP8 на русском языке](https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html).
* [Список встроенных функций](https://docs.python.org/3.7/library/functions.html).