**20.8 Практическая работа**

Цель задания

Отработать представление информации в Python в виде кортежей, а также закрепить работу со словарями и отработать использование новых методов словарей. Научиться:

* инициализировать кортежи и работать с ними при решении задач;
* использовать функцию enumerate для оптимизации работы кода;
* одновременно работать и с ключами, и со значениями словаря, получать их отображение с помощью метода;
* работать с составными ключами словарей;
* использовать функцию zip для решения задач.

Что входит в задание

Задача 1. Ревью кода.

Задача 2. Универсальная программа 2.

Задача 3. Функция.

Задача 4. Игроки.

Задача 5. Одна семья.

Задача 6. По парам.

Задача 7. Функция сортировки.

Задача 8. Контакты 3.

Задача 9. Протокол соревнований.

Задача 10. Своя функция zip.

Если работаете одновременно со значениями и индексами строки/списка/кортежа, используйте enumerate.

Если работаете одновременно с ключами и значениями словаря, используйте items.

Задача 1. Ревью кода

**Что нужно сделать**

Ваня работает middle-разработчиком на Python в IT-компании. Один кандидат на позицию junior-разработчика прислал ему код тестового задания.

В задании был словарь из трёх студентов. Необходимо:

1. Вывести на экран список пар «ID студента — возраст».
2. Написать функцию, которая принимает в качестве аргумента словарь и возвращает два значения: полный список интересов всех студентов и общую длину всех фамилий студентов.

Далее в основном коде вызывается функция, значения присваиваются отдельным переменным и выводятся на экран.

Ваня — очень придирчивый программист, и после просмотра кода он понял, что этого кандидата на работу не возьмёт, хотя он выдаёт верный результат. Вот код кандидата:

students = {  
    1: {  
        'name': 'Bob',  
        'surname': 'Vazovski',  
        'age': 23,  
        'interests': ['biology, swimming']  
    },  
    2: {  
        'name': 'Rob',  
        'surname': 'Stepanov',  
        'age': 24,  
        'interests': ['math', 'computer games', 'running']  
    },  
    3: {  
        'name': 'Alexander',  
        'surname': 'Krug',  
        'age': 22,  
        'interests': ['languages', 'health food']  
    }  
}  
  
  
  
  
def f(dict):  
    lst = []  
    string = ''  
    for i in dict:  
        lst += (dict[i]['interests'])  
        string += dict[i]['surname']  
    cnt = 0  
    for s in string:  
        cnt += 1  
    return lst, cnt  
  
  
  
  
pairs = []  
for i in students:  
    pairs += (i, students[i]['age'])  
  
  
  
  
my\_lst = f(students)[0]  
l = f(students)[1]  
print(my\_lst, l)

Перепишите этот код так, чтобы он был максимально pythonic и Ваня мало к чему мог придраться (только если очень захочется). Убедитесь, что программа верно работает Проверки на существование записей в словаре не обязательны, но приветствуются.

Результат работы программы:

Список пар "ID студента — возраст": [(1, 23), (2, 24), (3, 22)]

Полный список интересов всех студентов: {'running', 'computer games', 'math', 'languages', 'biology, swimming', 'health food'}

Общая длина всех фамилий студентов: 20

**Что оценивается**

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом — в видео 2.3).
* Новый код более оптимизирован и чист по стилю программирования (pythonic), чем старый.

Задача 2. Универсальная программа 2

**Что нужно сделать**

Спустя некоторое время заказчик попросил вас немного изменить скрипт для криптографии: теперь индексы элементов должны быть простыми числами.

Напишите функцию, которая возвращает список элементов итерируемого объекта (кортежа, строки, списка, словаря), у которых индекс — это простое число. Для проверки на простое число напишите отдельную функцию is\_prime. Основной код оставьте пустым (используйте его только для тестирования).

Дополнительно: сделайте так, чтобы основная функция состояла только из оператора return и так же возвращала список.

Пример вызова функции:

print(crypto([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]))

Ответ в консоли: [2, 3, 5, 7]

Пример вызова функции:

print(crypto('О Дивный Новый мир!'))

Ответ в консоли: ['Д', 'и', 'н', 'й', 'в', 'й', 'р']

**Что оценивается**

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

Задача 3. Функция

**Что нужно сделать**

Напишите функцию, которая на вход принимает кортеж и какой-то случайный элемент (его можно вводить с клавиатуры). Функция должна возвращать новый кортеж, начинающийся с первого появления элемента в нём и заканчивающийся вторым его появлением включительно.

Если элемента нет вовсе — вернуть пустой кортеж.

Если элемент встречается только один раз — вернуть кортеж, который начинается с этого элемента и идёт до конца исходного.

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

Пример вызова функции:

print(slicer((1, 2, 3, 4, 5, 6, 7, 8, 2, 2, 9, 10), 2))

Ответ в консоли: (2, 3, 4, 5, 6, 7, 8, 2)

**Что оценивается**

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

Задача 4. Игроки

**Что нужно сделать**

У вас есть словарь игроков, которые участвовали в трёх видах спорта. В словаре хранятся пары «ФИ — очки»:

players = {

    ("Ivan", "Volkin"): (10, 5, 13),

    ("Bob", "Robbin"): (7, 5, 14),

    ("Rob", "Bobbin"): (12, 8, 2)

}

Один программист попросил нас для его базы прислать другой вариант хранения этой информации.

Напишите программу, которая объединяет ключ словаря со значением в один кортеж, и выведите результат на экран. Постарайтесь использовать как можно более эффективное решение.

Результат работы программы:

[('Ivan', 'Volkin', 10, 5, 13), ('Bob', 'Robbin', 7, 5, 14), ('Rob', 'Bobbin', 12, 8, 2)]

**Что оценивается**

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

Задача 5. Одна семья

**Что нужно сделать**

В одной базе данных хранится информация о членах нескольких разных семей. Хранение реализовано с помощью словаря с парами «ФИ — возраст».

Инициализируйте словарь членов семей (ФИ и возраст используйте любые) и напишите программу, которая запрашивает у пользователя фамилию и выводит на экран возраст всех членов одной семьи. Учтите, что, например, у двух людей разного пола фамилии отличаются окончанием. Поиск не должен быть регистрозависимым.

Пример:

Введите фамилию: Сидоров

Сидоров Никита 35

Сидорова Алина 34

Сидоров Павел 10

**Что оценивается**

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

Задача 6. По парам

**Что нужно сделать**

Напишите программу, которая инициализирует список из 10 случайных целых чисел, а затем делит эти числа на пары кортежей внутри списка. Выведите результат на экран.

Дополнительно: решите задачу несколькими способами.

Пример:

Оригинальный список: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Новый список: [(0, 1), (2, 3), (4, 5), (6, 7), (8, 9)]

**Что оценивается**

* Результат вычислений корректен.
* Формат вывода соответствует примеру.
* Переменные и функции имеют значащие имена, не только a, b, c, d (подробнее об этом — в видео 2.3).
* Для решения используются list comprehensions.

Задача 7. Функция сортировки

**Что нужно сделать**

Напишите функцию, которая сортирует по возрастанию кортеж, состоящий из целых чисел, и возвращает его отсортированным. Если хотя бы один элемент не является целым числом, то функция возвращает исходный кортеж.

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

Пример вызова функции:

print(tpl\_sort(6, 3, -1, 8, 4, 10, -5))

Ответ в консоли: (-5, -1, 3, 4, 6, 8, 10)

**Что оценивается**

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

Задача 8. Контакты 3

**Что нужно сделать**

Мы уже помогали Степану с реализацией телефонной книги на телефоне, однако внезапно оказалось, что книге не хватает ещё одной полезной функции: поиска.

Напишите программу, которая бесконечно запрашивает у пользователя действие, которое он хочет совершить: добавить контакт или найти человека в списке контактов по фамилии.

Действие «добавить контакт»: программа запрашивает имя и фамилию контакта, затем номер телефона, добавляет их в словарь и выводит на экран текущий словарь контактов. Если этот человек уже есть в словаре, то выводится соответствующее сообщение.

Действие «поиск человека по фамилии»: программа запрашивает фамилию и выводит все контакты с такой фамилией и их номера телефонов. Поиск не должен зависеть от регистра символов.

Пример работы программы:

Введите номер действия:

 1. Добавить контакт

 2. Найти человека

1

Введите имя и фамилию нового контакта (через пробел): Иван Сидоров

Введите номер телефона: 888

Текущий словарь контактов: {('Иван', 'Сидоров'): 888}

Введите номер действия:

 1. Добавить контакт

 2. Найти человека

1

Введите имя и фамилию нового контакта (через пробел): Иван Сидоров

Такой человек уже есть в контактах.

Текущий словарь контактов: {('Иван', 'Сидоров'): 888}

Введите номер действия:

 1. Добавить контакт

 2. Найти человека

1

Введите имя и фамилию нового контакта (через пробел): Алиса Петрова

Введите номер телефона: 999

Текущий словарь контактов: {('Иван', 'Сидоров'): 888, ('Алиса', 'Петрова'): 999}

Введите номер действия:

 1. Добавить контакт

 2. Найти человека

2

Введите фамилию для поиска: Сидоров

Иван Сидоров 888

Введите номер действия:

 1. Добавить контакт

 2. Найти человека

…….

**Что оценивается**

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

Задача 9. Протокол соревнований

**Что нужно сделать**

Вы продолжаете развиваться в геймдеве, и в этот раз вам пришло действительно внушительное техническое задание с описанием правил игр, входными и выходными данными. Вот как оно выглядит:

*Здравствуйте! Мы собираемся устраивать соревнования по [данные засекречены] и хотим, чтобы вы написали эффективную программу, которая составляла бы нам протокол и определяла победителя и призёров. О логике работы программы ниже.*

*Правила соревнований*

1. Участники имеют право играть несколько раз. Количество попыток одного участника не ограничивается.
2. Окончательный результат участника определяется по одной игре, лучшей для этого участника.
3. Более высокое место в соревнованиях занимает участник, показавший лучший результат.
4. При равенстве результатов более высокое место занимает участник, раньше показавший лучший результат.

*Как проходят соревнования*

В ходе соревнований заполняется протокол, каждая строка которого описывает одну игру и содержит результат участника и его игровое имя. Протокол формируется в реальном времени по ходу проведения чемпионата, поэтому строки в нём расположены в порядке проведения игр: чем раньше встречается строка в протоколе, тем раньше закончилась соответствующая этой строке игра.

Напишите программу, которая по данным протокола определяет победителя и призёров. Гарантируется, что в чемпионате участвует не менее трёх игроков.

*Описание входных данных*

Первая строка содержит число N — это общее количество строк протокола. Каждая из следующих N строк содержит записанные через пробел результаты участника (целое неотрицательное число) и игровое имя (имя не может содержать пробелов). Строки исходных данных соответствуют строкам протокола и расположены в том же порядке, что и в протоколе.

Гарантируется, что в соревнованиях не менее трёх участников.

*Описание выходных данных*

Программа должна вывести имена и результаты трёх лучших игроков по форме, приведённой ниже в примере.

*Пример входных и выходных данных*

Сколько записей вносится в протокол? 9

Записи (результат и имя):

1-я запись: 69485 Jack

2-я запись: 95715 qwerty

3-я запись: 95715 Alex

4-я запись: 83647 M

5-я запись: 197128 qwerty

6-я запись: 95715 Jack

7-я запись: 93289 Alex

8-я запись: 95715 Alex

9-я запись: 95715 M

Итоги соревнований:

1-е место. qwerty (197128)

2-е место. Alex (95715)

3-е место. Jack (95715)

**Что оценивается**

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

Задача 10. Своя функция zip

**Что нужно сделать**

В самом конце собеседования вас неожиданно спросили: «Расскажите, что делает функция zip?». Чтобы произвести впечатление, вы решили не только рассказать про неё, но и написать её аналог.

Даны строка и кортеж из чисел. Напишите программу, которая создаёт генератор из пар кортежей «символ — число». Затем выведите на экран сам генератор и кортежи.

Пример:

Строка: abcd

Кортеж чисел: (10, 20, 30, 40)

Результат:

<generator object <genexpr> at 0x00000204A4234048>

('a', 10)

('b', 20)

('c', 30)

('d', 40)

Дополнительно: создайте полный аналог функции zip — сделайте так, чтобы программа работала с любыми итерируемыми типами данных.

**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).

Что оценивается в задании

* Домашнее задание сдано 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 имеют значащие имена, не только a, b, c, d (подробнее об этом — в видео 2.3).
* Присутствуют пробелы после запятых и при бинарных операциях.
* Отсутствуют пробелы 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.
* При одновременной работе со значениями и индексами строки/списка/кортежа используется enumerate.
* При одновременной работе с ключами и значениями словаря используется items.

Советы и 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](https://www.python.org/dev/peps/pep-0008/) на английском языке.
* Руководство по стилю Python [PEP8](https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html) на русском языке.
* [Список встроенных функций.](https://docs.python.org/3.7/library/functions.html)