#1

Розробіть функцію get\_days\_from\_today(date), яка повертатиме кількість днів від поточної дати, де параметр date - це рядок формату '2020-10-09' (рік-місяць-день).

Підказки:

* Параметр date розбити на рік, місяць та день можна використовуючи метод рядків split.
* datetime приймає аргументи типу int, використовуйте перетворення типів.
* ігноруйте години, хвилини та секунди для вашої дати, важливі повні дні.
* кількість днів ви можете отримати відніманням з поточної дати, заданої в змінній date (без часу).

Наприклад: Якщо поточна дата - '5 травня 2021', то виклик get\_days\_from\_today("2021-10-09") поверне нам -157.

from datetime import datetime

*def* get\_days\_from\_today(*date*):

    get\_data = *date*.split("-")

    get\_data2 = datetime(

*year*=int(get\_data[0]), *month*=int(get\_data[1]), *day*=int(get\_data[2]))

    current\_datetime = datetime.now()

    difference = current\_datetime.date() - get\_data2.date()

    return difference.days

#2

Напишіть функцію визначення кількості днів у конкретному місяці. Ваша функція повинна приймати два параметри: month - номер місяця у вигляді цілого числа в діапазоні від 1 до 12 і year - рік, що складається із чотирьох цифр. Перевірте, чи функція коректно обробляє місяць лютий високосного року.

from datetime import date

*def* get\_days\_in\_month(*month*, *year*):

    first\_day = date(*year*=*year*, *month*=*month*, *day*=1)

    if *month* == 12:

        last\_day = date(*year*=*year*+1, *month*=*month*-11, *day*=1)

    else:

        last\_day = date(*year*=*year*, *month*=*month*+1, *day*=1)

    diff = last\_day - first\_day

    return diff.days

print(get\_days\_in\_month(12, 1971))

#3.

Напишіть функцію get\_str\_date(date), яка перетворюватиме дату з бази даних у форматі ISO '2021-05-27 17:08:34.149Z' у вигляді наступного рядка 'Thursday 27 May 2021' - день тижня, число, місяць та рік. Перетворене значення функція повертає під час виклику.

from datetime import datetime

# дату з бази даних у форматі ISO '2021-05-27 17:08:34.149Z'

# у вигляді наступного рядка 'Thursday 27 May 2021' - день тижня, число, місяць та рік

*def* get\_str\_date(*date*):

    list = *date*.split("-")

    list[2] = list[2][0:2]

    date\_new = datetime(*year*=int(list[0]), *month*=int(

        list[1]), *day*=int(list[2]))

    res = date\_new.strftime('%A %d %B %Y')

    return res

print(get\_str\_date('2021-05-27 17:08:34.149Z'))

#4

Щоб виграти головний приз лотереї, необхідний збіг кількох номерів на лотерейному квитку з числами, що випали випадковим чином і в певному діапазоні під час чергового тиражу. Наприклад, необхідно вгадати шість чисел від 1 до 49 чи п'ять чисел від 1 до 36 тощо.

Напишіть функцію, яка випадково підбиратиме набір чисел для лотерейного квитка. Серед цих чисел не має бути дублікатів.

Формат функції get\_numbers\_ticket(min, max, quantity), де параметри:

* min - мінімальне значення діапазону, не може бути менше 1
* max - максимальне значення діапазону, не може бути більше 1000
* quantity - кількість чисел у наборі (має бути min < quantity < max)

Функція повинна повернути перелік випадкових чисел за зростанням. Якщо порушено умови обмежень на параметри функції, тоді повернути пустий список.

import random

# quantity - кількість чисел у наборі(має бути min < quantity < max)

# Функція повинна повернути перелік випадкових чисел за зростанням.

# Якщо порушено умови обмежень на параметри функції, тоді повернути пустий список

*def* get\_numbers\_ticket(*min*, *max*, *quantity*) -> list:

    if *min* < *quantity* < *max* and *min* > 0 and *max* <= 1000:

        res = random.sample(range(*min*, *max*+1), *k*=*quantity*)

        res.sort()

        return res

    else:

        return []

print(get\_numbers\_ticket(*min*=-1, *max*=49, *quantity*=6))

#5

**ЗАВДАННЯ. ВИБІР ПЕРЕМОЖЦІВ РОЗІГРАШУ**

**ЗАВДАННЯ**

Ви проводите розіграш кавоварок Bosch серед зареєстрованих користувачів вашої інтернет-сторінки.

Список зареєстрованих користувачів — це словник такого типу:

participants = {

"603d2cec9993c627f0982404": "test@test.com",

"603f79022922882d30dd7bb6": "test11@test.com",

"60577ce4b536f8259cc225d2": "test2@test.com",

"605884760742316c07eae603": "vitanlhouse@gmail.com",

"605b89080c318d66862db390": "elhe2013@gmail.com",

}

Ключ словника — це унікальний ідентифікатор бази даних MongoDB, а значення &mdash це email користувача. Вам необхідно випадково відібрати декілька переможців розіграшу.

Створіть функцію get\_random\_winners(quantity, participants), яка повертатиме список унікальних ідентифікаторів бази даних зі словника participants в кількості quantity. Це буде список переможців

Вимоги:

* Отримайте перелік ключів словника. (Після виконання методу keys() використовуйте перетворення типів)
* Перемішайте отриманий список за допомогою методу shuffle
* Виберіть випадкових переможців, використовуючи метод sample.
* Якщо передана кількість переможців більша за кількість користувачів (quantity > len(participants)) — поверніть порожній список.

Наприклад: виклик get\_random\_winners(2, participants) може повернути список з випадковим набором ідентифікаторів як: ['60577ce4b536f8259cc225d2', '605b89080c318d66862db390'].

import random

*def* get\_random\_winners(*quantity*, *participants*):

    if *quantity* > len(*participants*):

        return []

    else:

        participants\_clear = list(*participants*.keys())

        random.shuffle(participants\_clear)

        res = random.sample(participants\_clear, *k*=*quantity*)

        print(res)

participants = {

    "603d2cec9993c627f0982404": "test@test.com",

    "603f79022922882d30dd7bb6": "test11@test.com",

    "60577ce4b536f8259cc225d2": "test2@test.com",

    "605884760742316c07eae603": "vitanlhouse@gmail.com",

    "605b89080c318d66862db390": "elhe2013@gmail.com",

}

get\_random\_winners(2, participants)

#6

Створіть функцію decimal\_average(number\_list, signs\_count), яка обчислюватиме середнє арифметичне типу Decimal з кількістю значущих цифр signs\_count. Параметр number\_list — список чисел

##### Увага

Не забувайте приводити всі числа у списку до типу `decimal`

Приклад:

* виклик функції decimal\_average([3, 5, 77, 23, 0.57], 6) поверне 21.714
* виклик функції decimal\_average([31, 55, 177, 2300, 1.57], 9) поверне 512.91400

from decimal import Decimal, getcontext

# обчислюватиме середнє арифметичне типу Decimal з кількістю значущих цифр signs\_count.

# Параметр number\_list — список чисел

*def* decimal\_average(*number\_list*, *signs\_count*):

    getcontext().prec = *signs\_count*

    res = 0

    for item in *number\_list*:

        res += Decimal(item)

    return res/(len(*number\_list*))

print(decimal\_average([31, 55, 177, 2300, 1.57], 9))

#7

У нас є іменований кортеж для зберігання котів у змінній Cat. На першому місці у нас кличка котика nickname, потім його вік age та ім'я власника кота owner.

Напишіть функцію convert\_list(cats), яка працюватиме у двох режимах.

Якщо функція convert\_list приймає у параметрі cats список іменованих кортежів

[Cat("Mick", 5, "Sara"), Cat("Barsik", 7, "Olga"), Cat("Simon", 3, "Yura")]

То функція поверне наступний список словників:

[

{"nickname": "Mick", "age": 5, "owner": "Sara"},

{"nickname": "Barsik", "age": 7, "owner": "Olga"},

{"nickname": "Simon", "age": 3, "owner": "Yura"},

]

І в той же час, якщо функція convert\_list приймає в параметрі cats список словників, то результатом буде зворотна операція та функція поверне список іменованих кортежів.

Для визначення типу параметра cats використовуйте функцію [isinstance](https://docs.python.org/3/library/functions.html" \l "isinstance).

import collections

Cat = collections.namedtuple("Cat", ["nickname", "age", "owner"])

cats = [

    {"nickname": "Mick", "age": 5, "owner": "Sara"},

    {"nickname": "Barsik", "age": 7, "owner": "Olga"},

    {"nickname": "Simon", "age": 3, "owner": "Yura"},

]

# cats = [Cat("Mick", 5, "Sara"), Cat(

#     "Barsik", 7, "Olga"), Cat("Simon", 3, "Yura")]

*def* convert\_list(*cats*):

    res = []

    for item in *cats*:

        if isinstance(item, tuple):

            res.append(dict(item.\_asdict()))

        if isinstance(item, dict):

            res.append(Cat(\*\*item))

    return res

#8

Є список IP адрес:

IP = [

"85.157.172.253",

...

]

Реалізуйте дві функції. Перша get\_count\_visits\_from\_ip за допомогою Counter повертатиме словник, де ключ це IP, а значення – кількість входжень у вказаний список.

Приклад:

{

'85.157.172.253': 2,

...

}

Друга функція get\_frequent\_visit\_from\_ip повертає кортеж з найбільш часто уживаним в списку IP і кількістю його появ в списку.

Пример:

('66.50.38.43', 4)

from collections import Counter

*def* get\_count\_visits\_from\_ip(*ips*):

    count = Counter(*ips*)

    return count

*def* get\_frequent\_visit\_from\_ip(*ips*):

    count = Counter(*ips*)

    res = count.most\_common(1)

    return res[0]

ips = ['66.50.38.43', '66.50.38.43', '66.50.38.43', '66.50.38.43',

       '76.98.129.245', '76.98.129.245', '76.98.129.245',

       '85.157.172.253', '85.157.172.253', '173.37.214.238',

       '173.37.214.238', '143.231.49.229', '27.137.126.114',

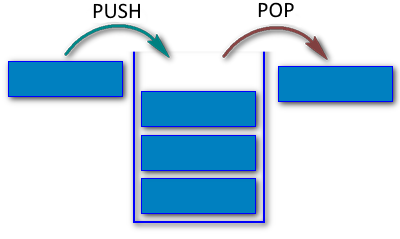
       '248.95.93.236']

print(get\_count\_visits\_from\_ip(ips))

print(get\_frequent\_visit\_from\_ip(ips))

#9

LIFO (англ. last in, first out, "останнім прийшов - першим пішов") - спосіб організації даних або іншими словами Стек (Stack). У структурованому лінійному списку, організованому за принципом LIFO, елементи можуть додаватися та вибиратися тільки з одного кінця, що називається «вершиною списку». Структура LIFO може бути проілюстрована наступним малюнком.



За допомогою колекції deque реалізуйте структуру даних LIFO. Створіть змінну lifo, що містить колекцію deque. Обмежте розмір за допомогою константи MAX\_LEN. Функція push додає значення element на початок списку lifo. Функція pop дістає та повертає перше значення зі списку lifo.

from collections import deque

MAX\_LEN = 2

lifo = deque(*maxlen*=MAX\_LEN)

*def* push(*element*):

    lifo.appendleft(*element*)

*def* pop():

    return lifo.popleft()

#10

from collections import deque

MAX\_LEN = 5

fifo = deque(*maxlen*=MAX\_LEN)

*def* push(*element*):

    fifo.append(element)

*def* pop():

    return fifo.popleft()

За допомогою колекції deque реалізуйте структуру даних FIFO. Створіть змінну fifo, що містить колекцію deque. Обмежте розмір за допомогою константи MAX\_LEN. Функція push додає значення element до кінця списку fifo. Функція pop дістає та повертає перше значення зі списку fifo.