IT's TINKOFF

Разбор заданий отборочного тура трека Backend на Python

Съешь же ещё этих мягких французских булок, да выпей чаю

Текст задачи

Необходимо написать функцию, определяющую, является ли предложение панграммой.

Панграмма – это предложение, которое содержит каждую отдельную букву алфавита по крайней мере один раз.

Например, предложение:

`Съешь же ещё этих мягких французских булок, да выпей чаю.`

является панграммой, потому что в нем используются все буквы алфавита от А до Я по крайней мере один раз.

Ограничения

- Функция должна принимать на вход строку;
- Функция должна возвращать булевое значение;
- Регистр букв не должен учитываться;
- Все символы кроме кириллицы должны игнорироваться.

Входные параметры

`Съешь же ещё этих мягких французских булок, да выпей чаю.`

Выходные параметры

`True`

IT's TINKOFF

Пример реализации

```python
def is_pangram(text: str) -> bool:
    """Определить, является ли предложение панграммой."""

    alphabet = set("абвгдеёжзийклмнопрстуфхцчшщъыьэюя")

    # Приводим text к нижнему регистру.
    for letter in text.lower():
        # При наличии буквы в text, удаляем букву из alphabet.
        if letter in alphabet:
            alphabet.remove(letter)

    # Если в alphabet не осталось букв, то text - панграмма.
    return len(alphabet) == 0


if __name__ == "__main__":
    input_str = input()
    print(is_pangram(input_str))
```

Камуфляж и шпионаж

Текст задачи

Необходимо написать функцию, которая декодирует и расшифровывает строку.

Ограничения

- Функция должна принимать на вход строку;
- Функция должна возвращать строку.

Входные параметры

РґРѕС‰СѓСЏ СЂСЉРЅР¶СѓР±, Ps С‘РѕС€СЊС‡С‰ СЂСЉРѕС‘СѓР±.

IT's TINKOFF

Выходные параметры

Хакер пляшет, а чайник плачет.

Подсказка

о - к - н - а

Пример реализации

```python
def decrypt(text: str) -> str:
    """Расшифровка текста."""

    # Декодируем строку по подсказке: о - к - н - а
    # Подсказка намекает на кодировку в Windows.
    decoded_text = text.encode("cp1251").decode("utf-8")

    alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"
    # Находим сдвиг по подсказке: о - к - н - а
    # Необходимо сдвигать только буквы в нижнем регистре.
    # а -> н: нопрстуфхцчшщъыьэюяабвгдеёжзийклм
    # н -> к: клмнопрстуфхцчшщъыьэюяабвгдеёжзий
    # к -> о: опрстуфхцчшщъыьэюяабвгдеёжзийклмн
    shifted_alphabet = "опрстуфхцчшщъыьэюяабвгдеёжзийклмн"

    # Применяем сдвиг к входной строке.
    trans_table = str.maketrans(shifted_alphabet, alphabet)
    return decoded_text.translate(trans_table)


if __name__ == "__main__":
    input_text = input()
    print(decrypt(input_text))
```

IT's TINKOFF

Помоги буквам найти свою пару

Текст задачи

Написать функцию, которая находит все уникальные символьные комбинации из заданной строки.

Например, для строки

`abc`

уникальными комбинациями являются

`a, ab, abc, ac, acb, b, ba, bac, bc, bca, c, ca, cab, cb, cba`

Ограничения

- Функция должна принимать на вход строку, для которой нужно найти уникальные комбинации;
- Функция должна возвращать список всех уникальных комбинаций в заданной строке;
- Функция должна возвращать результат в отсортированном виде;
- На выход необходимо отдать строку вида: `a, ab, abc`

Входные параметры

`abc`

Выходные параметры

`a, ab, abc, ac, acb, b, ba, bac, bc, bca, c, ca, cab, cb, cba`

Пример реализации

```python
from itertools import permutations


def find_combinations(text: str) -> list[str]:
    """Найти уникальные символьные комбинации."""

    combinations: set[str] = set()
    for i in range(1, len(text)+1):
```

```
        # Применяем функцию permutations для нахождения перестановок длины i.
        combinations.update("".join(pm) for pm in permutations(text, i))

    return sorted(combinations)


if __name__ == "__main__":
    input_str = input()
    print(", ".join(find_combinations(input_str)))
```

Немного аналитики

Текст задачи

Необходимо написать функцию, которая получает всех администраторов от 35 лет, которые зарегистрировались в системе после 3 января 2013 года 12:00 по MSK.

Ограничения

- Функция должна принимать на вход JSON в виде строки;
- Функция должна возвращать JSON в виде строки.

Входные параметры


```
[
  {
    "id": "6423376a984af552c76986f5",
    "email": "lucille_cervantes@acumentor.limited",
    "roles": ["owner"],
    "apiKey": "1ea51d2b-4844-4583-adc0-5e266ead741f",
    "profile": {
      "dob": "1989-08-08",
      "name": "Lucille Cervantes",
      "about": "In mollit ut culpa pariatur exercitation eu labore. Cupidatat du
is proident fugiat aliqua pariatur nisi fugiat.",
      "address": "47 Opal Court, Harold, Iowa",
      "company": "Acumentor",
      "location": {
        "lat": 76.251926,
        "long": 62.613692
      }
```

```
    },
    "username": "lucille89",
    "createdAt": "2010-07-07T14:06:36.065+00:00",
    "updatedAt": "2010-07-08T14:06:36.065+00:00"
  },
  {
    "id": "6423376a582728a816d1c31f",
    "email": "rodgers_galloway@animalia.tips",
    "roles": ["admin", "guest"],
    "apiKey": "958b47d2-2fc6-4f82-91f7-247b99b6eb1f",
    "profile": {
      "dob": "1971-06-28",
      "name": "Rodgers Galloway",
      "about": "Esse duis quis incididunt ut incididunt sint laborum labore Lore
m ipsum proident tempor Lorem. Est culpa nulla aliqua Lorem ad dolor ipsum enim.
",
      "address": "82 Jerome Avenue, Washington, Nevada",
      "company": "Animalia",
      "location": {
        "lat": 80.748957,
        "long": 96.703735
      }
    },
    "username": "rodgers94",
    "createdAt": "2014-10-10T14:56:25.410+03:00",
    "updatedAt": "2014-10-11T14:56:25.410+03:00"
  }
]
```


Выходные параметры


```
[
  {
    "username": "rodgers94",
    "email": "rodgers_galloway@animalia.tips",
    "name": "Rodgers Galloway",
    "age": 51
  }
]
```

IT's TINKOFF

Пример реализации

```python
import json
from datetime import date, datetime, timedelta, timezone
from typing import Any


THRESHOLD = datetime(2013, 1, 3, 12, tzinfo=timezone(timedelta(hours=3)))
MIN_AGE = 35
ROLE = "admin"


def filter_data(text: str) -> str:
    """Фильтрация данных."""

    # Для работы с данными приводим json-строку к словарю.
    data = json.loads(text)
    today = date.today()

    filtered_data: list[dict[str, Any]] = []
    for row in data:
        born = datetime.strptime(row["profile"]["dob"], "%Y-%m-%d").date()
        age = today.year - born.year - ((today.month, today.day) < (born.month,
born.day))

        # Проверяем элемент на совпадение условию из задания.
        if (
            datetime.datetime.fromisoformat(row["createdAt"]) >= THRESHOLD
            and ROLE in row["roles"]
            and age >= MIN_AGE
        ):
            filtered_data.append({
                "username": row["username"],
                "email": row["email"],
                "name": row["profile"]["name"],
                "age": age
            })

    # Переводим словарь в json-строку.
```

```
    return json.dumps(filtered_data)

if __name__ == "__main__":
    input_str = input()
    print(filter_data(input_str))
```

5 букв

Текст задачи

Необходимо написать функцию, которая проверяет правильность решения для игры 5 букв.

Ограничения

- Функция должна принимать на вход строку вида: `ответ_пользователя, правильный_ответ`
- Функция должна возвращать массив из 5 элементов, каждый элемент – это цифра отвечающая за совпадения: `-1: Буква отсутствует  0: Буква есть, но в другом месте  1: Буква на своём месте`
- Слова должны состоять строго из 5 букв;
- При проверке повторяющихся букв должно учитываться их количество. Например, если в ответе **ТЕКСТ** есть только одна буква **Е**, а пользователь отправил слово **ЕГЕРЬ**, то только одна буква **Е** должна быть помечена как **0**, остальные должны быть **-1**;
- На выход необходимо отдать строку вида: `1, 0, -1, 1, -1`

Входные параметры

ДОЖДЬ, ДЗЮДО

Выходные параметры

`1, 0, -1, 1, -1`

IT's TINKOFF

Пример реализации

```python
WORD_LEN = 5

def get_answer_mask(text: str) -> list[int]:
    """Получить маску."""

    # Разделяем строку на: ответ пользователя, правильный ответ.
    suggestion, answer = text.split(', ')

    mask = [-1] * WORD_LEN

    # Список позиций, в которых буквы либо отсутствуют, либо не на своем месте.
    positions: list[int] = []
    for i in range(WORD_LEN):
        # Если буквы на i позиции совпадают, то буква на своем месте.
        if suggestion[i] == answer[i]:
            mask[i] = 1
        else:
            positions.append(i)

    # Находим буквы, которые не на своем месте.
    for i in positions:
        for j in range(WORD_LEN):
            if mask[j] == -1 and suggestion[j] == answer[i]:
                mask[j] = 0
                break

    return mask


if __name__ == "__main__":
    input_str = input()
    print(', '.join(map(str, get_answer_mask(input_str))))
```

IT's TINKOFF

Сжатие самоизолирующихся последовательностей

Текст задачи

Написать функцию, которая будет принимать на вход массив чисел и отдавать сжатые последовательности в виде массива кортежей.

Ограничения

- Функция должна принимать на вход строку с числами вида: `1, 2, 2, 3, 4, 3, 3, 3`
- Функция должна возвращать массив кортежей с числами;
- Каждый элемент кортежа должен удовлетворять формату: `(x, y)  x - число из последовательности;  y - количество раз, которое это число встречается в последовательности.`
- На выход необходимо отдать строку вида: `(1, 1), (2, 2), (3, 1), (4, 1), (3, 3)`

Входные параметры

`1, 2, 2, 3, 4, 3, 3, 3`

Выходные параметры

`(1, 1), (2, 2), (3, 1), (4, 1), (3, 3)`

Пример реализации

```python
from itertools import groupby


def compress_sequence(text: str) -> list[tuple[int, int]]:
    """Сжать последовательность."""

    result: list[tuple[int, int]] = []
    for key, group in groupby(text.split(", ")):
        result.append((int(key), len(list(group))))

    return result


if __name__ == "__main__":
```

```
input_str = input()
print(str(compress_sequence(input_str))[1:-1])
```

Не трогай свечу

Текст задачи

Сгенерировать из цен свечи по интервалам:

```
1 минута (1min)
2 минуты (2min)
5 минут (5min)
```

Свеча агрегирует в себе цены на своем интервале. Например, для диапазона цен от `18:03:14` до `18:17:50` будет 4 пятиминутные свечи:

```
18:00:00 (все цены с временем 18:00:00 <= ts < 18:05:00)
18:05:00 (все цены с временем 18:05:00 <= ts < 18:10:00)
18:10:00 (все цены с временем 18:10:00 <= ts < 18:15:00)
18:15:00 (все цены с временем 18:15:00 <= ts < 18:20:00)
```

И 8 двухминутных свечей:

```
18:02:00 (все цены с временем 18:02:00 <= ts < 18:04:00)
18:04:00 (все цены с временем 18:04:00 <= ts < 18:06:00)
...
18:14:00 (все цены с временем 18:14:00 <= ts < 18:16:00)
18:16:00 (все цены с временем 18:16:00 <= ts < 18:18:00)
```

Если за какой-то интервал цен нет, то этот интервал для данного идентификатора остается пустым. Пример: если для идентификатора TSLA есть цены в интервале `[18:00:00, 18:00:59]` и `[18:02:00, 18:02:59]`, то у него должно быть только двухминутные свечи с временем `18:00:00` и `18:02:00`.

Ограничения

- На вход подаются строки с ценами по нескольким инструментам следующего вида (разделителем является запятая): `Идентификатор_инструмента,цена,время(в формате `rfc3339`)`
- Входные данные отсортированы по времени в порядке возрастания `TSLA,191.97,2023-04-11T12:04:30Z TCSG,32.49,2023-04-11T12:04:30Z`
- На выход необходимо отдать свечи, формат свечи (разделителем является запятая): `Идентификатор_инструмента,цена_открытия,максимальная_цена,минимальная_цена,цена_закрытия,время(в формате rfc3339, кратное интервалу),интервал`
- Свечи в выводе отсортированы сначала по идентификаторам, внутри идентификатора по интервалам, а внутри интервала отсортированы по времени;
- Время свечи должно быть кратно интервалу. Пример: время цены `18:01:35` -> время одноминутной свечи `18:01:00`;
- Свечи необходимо разделять символом перевода строки;
- Важно не забыть выгрузить последние, незакрытые свечи;
- В коде есть ошибка при чтении из `stdin`, необходимо исправить.

Пример

Входные параметры

```
TSLA,191.97,2023-04-11T12:04:30Z
TCSG,32.49,2023-04-11T12:04:30Z
TSLA,192.50,2023-04-11T12:05:15Z
TCSG,32.17,2023-04-11T12:05:15Z
TSLA,191.30,2023-04-11T12:05:53Z
TCSG,31.84,2023-04-11T12:05:53Z
TSLA,191.60,2023-04-11T12:06:39Z
TCSG,31.53,2023-04-11T12:06:39Z
```

Выходные параметры

```
TCSG,32.49,32.49,32.49,32.49,2023-04-11T12:04:00Z,1min
TCSG,32.17,32.17,31.84,31.84,2023-04-11T12:05:00Z,1min
TCSG,31.53,31.53,31.53,31.53,2023-04-11T12:06:00Z,1min
TCSG,32.49,32.49,31.84,31.84,2023-04-11T12:04:00Z,2min
TCSG,31.53,31.53,31.53,31.53,2023-04-11T12:06:00Z,2min
TCSG,32.49,32.49,32.49,32.49,2023-04-11T12:00:00Z,5min
TCSG,32.17,32.17,31.53,31.53,2023-04-11T12:05:00Z,5min
TSLA,191.97,191.97,191.97,191.97,2023-04-11T12:04:00Z,1min
```

```
TSLA,192.50,192.50,191.30,191.30,2023-04-11T12:05:00Z,1min
TSLA,191.60,191.60,191.60,191.60,2023-04-11T12:06:00Z,1min
TSLA,191.97,192.50,191.30,191.30,2023-04-11T12:04:00Z,2min
TSLA,191.60,191.60,191.60,191.60,2023-04-11T12:06:00Z,2min
TSLA,191.97,191.97,191.97,191.97,2023-04-11T12:00:00Z,5min
TSLA,192.50,192.50,191.30,191.60,2023-04-11T12:05:00Z,5min
```

Пример реализации

```python
from collections import defaultdict
from datetime import datetime, timedelta
from sys import stdin


def generate_candles(text: str) -> str:
    """Сгенерировать свечи."""

    intervals = {"1min": 1, "2min": 2, "5min": 5}
    candels: dict[tuple[str, str, datetime], list[float]] = defaultdict(list)
    for line in text.splitlines():
        pos, price, time = line.split(",")
        time = datetime.strptime(time, "%Y-%m-%dT%H:%M:%SZ")

        # Формируем свечи.
        for interval, minutes in intervals.items():
            interval_start = time - timedelta(
                minutes=time.minute % minutes,
                seconds=time.second,
                microseconds=time.microsecond,
            )
            candels[(pos, interval, interval_start)].append(float(price))

    result: list[str] = []
    for key in sorted(candels):
        pos, interval, interval_start = key
        prices = candels[key]
        result.append(
            f"{pos},{prices[0]:.2f},{max(prices):.2f},{min(prices):.2f},{prices[
-1]:.2f},"
            f"{interval_start.strftime('%Y-%m-%dT%H:%M:%SZ')},{interval}"
```

```
        )

    return "\n".join(result)


if __name__ == "__main__":
    # Ошибка была в том, что строка читалась до \n.
    input_str = stdin.read()
    print(generate_candles(input_str))
```

Жан-Клод ВАМ БАН!

Текст задачи

Есть список запретных слов и словосочетаний: **чай, горячий кофе.**

Эти слова и словосочетания могут встречаться в тексте: * в разных падежах: **чай, чая, чаю** * в разных формах: **чайный, чайного** * в комбинации с другими словами: **горячий и вкусный кофе**

Необходимо написать функцию, которая находит запретные слова в тексте.

Ограничения

- Функция должна принимать на вход строку;
- Функция должна возвращать список запретных слов в тексте;
- Функция должна возвращать результат в отсортированном виде;
- Если запретных слов нет, необходимо возвращать пустой список;
- На выход необходимо отдать строку вида:

```
горячий кофе, чай
```

Входные параметры

```
Потом пили чай с конфетами и орешками, разговаривали.
```

IT's TINKOFF

Выходные параметры

чай

Пример реализации

```python
import re


def find_ban_words(text: str) -> list[str]:
    """Найти запретные слова."""

    # Находим необходимые слова с помощью регулярок.
    ban_words: list[str] = []
    for match in re.finditer(r"\w+|[^\w\s]+", text):
        if re.match(r"ча[й|я|ю|йный|его]", match.group()):
            ban_words.append("чай")

    for match in re.finditer(
        r"горяч[ий|его|ему|им|ем].*кофе", text
    ):
        ban_words.append("горячий кофе")

    return sorted(ban_words)


if __name__ == "__main__":
    input_str = input()
    print(", ".join(find_ban_words(input_str)))
```