Практична робота № 1

Функції на мові Python

Варіанти завдань

1. Написати функцію, яка приймає один аргумент-список та набір ключових аргументів і повертає індекси тих числових елементів списку, які мають найменшу дробову частину, а також назву тих ключових аргументів, дробова частина яких найбільша.
2. Написати функцію, яка повертає ті елементи списку своїх аргументів, які є рядками і мають довжину, яка більша за сумарну довжину сусідніх аргументів-рядків (якщо сусідній елемент не є рядком, то вважати, що його довжина рівна 0).
3. Написати функцію `compound`, яка повертає список тих своїх аргументів, які можуть бути записані у вигляді конкатенації кількох інших аргументів. Наприклад, результатом виклику

   `compound("maker", "newsmaker", "step", "by", "stepbystep", "step", "news")`

   має бути `["newsmaker", "stepbystep"]`.
4. Написати функцію, яка для вхідного списку трійок чисел повертає підсписок тих трійок, які утворюють трикутники найбільшого периметру.
5. Написати функцію `flat(list_arg, depth)` для «спрямлення» вкладених колекцій заданої глибини.
   Наприклад, якщо `L = [1, [2, [3, range(4, 7), 7], 8], ([9], 10])`, то `flat(L)` має повертати `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`, `flat(L, 1)` має повертати `[1, 2, [3, range(4, 7), 7], 8, [9], 10]`, а `flat(L, 2)` — `[1, 2, 3, range(4, 7), 7, 8, 9, 10]`.
6. Написати функцію `deep_product`, яка обчислює добуток усіх чисел, які входять до складу її аргументів (кількість аргументів наперед невідома, нечислові неітеровані аргументи мають ігноруватися, елементи ітерованих аргументів мають оброблятися окремо, у аргументах-словниках потрібно проводити обробку значень). У випадку відсутності у складі аргументів чисел функція має повертати `None`.
7. Написати функцію, єдиним параметром якої є список натуральних чисел. На основі вхідного списку має обчислюватися 2-ий список, кожний елемент якого рівний сумі цифр відповідного елемента початкового списку. За тим самим принципом обчислюються 3-ий, 4-ий, … списки до тих пір, поки останній список не буде співпадати із своїм попередником. Функція має повертати цей список.
8. Написати функцію, яка видаляє ті елементи списку, які є рядками і утворені із різних символів (жодний символ у яких не повторюється).
9. Написати функцію `watcher`, яка може використовуватися для відслідковування викликів інших функцій, відслідковуючи кількість звертань до них. Наприклад, результатом виклику коду

```
def add(a, b, c):
    return a + b + c

traced_add = watcher(add)
```

```
traced_add(2, 3, 5)
traced_add(4, c=1, b=3)
traced_add('abc', 'd', 'ef')
```

може бути

```
add is called 1 time; add(2, 3, 5) returned 10.
add is called 2 times; add(4, c=1, b=3) returned 8.
add is called 3 times; add(abc, d, ef) returned abcdef.
```

10. Написати функцію `flat_map(func, *iterable_args)`, яка застосовує функцію func до кожного кортежу відповідних елементів із кожної із колекцій та спрямлає результат (на 1 рівень), якщо він не є рядком, і повертає результат у вигляді списку. При цьому елементи-словники верхнього рівня потрібно заміняти списками пар ключ-значення. Наприклад, фрагмент коду

```
functions = [
    lambda x, y: [x, [x, y]],
    lambda x, y: x * y,
    lambda x, y: {x: [x, y], y: [y, x]},
    lambda x, y: [y * x, y * (x - 1)]
]
for function in functions:
    print(flat_map(function, [3, 2], 'ab'))
```

має вивести на консоль

```
[3, [3, 'a'], 2, [2, 'b']]
['aaa', 'bb']
[(3, [3, 'a']), ('a', ['a', 3]), (2, [2, 'b']), ('b', ['b', 2])]
['aaa', 'aa', 'bb', 'b']
```

11. Написати генеруючу функцію, яка повертає спільні ключі усіх своїх аргументів-словників, яким відповідають однакові значення.

12. Написати генеруючу функцію `extract_str`, яка повертає ті рядки, які зустрічаються у складі її аргументів (кількість аргументів наперед невідома, аргументи можуть бути іменованими, неітеровані аргументи, які не є рядками, мають ігноруватися, елементи ітерованих аргументів мають оброблятися окремо, у аргументах-словниках потрібно проводити обробку як ключів, так і значень).

13. Написати функцію, яка для вхідного списку трикутників повертає список назв тих трикутників, які мають найбільшу площу (трикутник задається у вигляді словника, ключі якого відповідають назвам вершин, а значення — їх координатам).

14. Написати функцію, яка повертає об’єднання кількох словників. Якщо ключ спільний для кількох словників, то йому має відповідати множина усіх значень, які відповідають цьому ключу у словниках-аргументах.

15. Написати функцію `transform`, яка приймає лише іменовані аргументи (типу ключ=значення) та повертає словник, ключами якого є різні значення серед вхідних аргументів, а значеннями — список ключів вхідних аргументів, які

відповідають цим значенням, якщо їх не менше двох, або єдиний ключ із цим значенням. Наприклад, результатом виклику

```
transform(u=5, v=2, x=1, y=2, z=1)
```

має бути `{1: ['x','z'], 2: ['v','y'], 5: 'u'}`.