Словник. Кортеж

Мета роботи: ознайомитися з такими об’єктами як словники, кортежі мови Python. *Об’єкт дослідження* – словник, кортеж.

ПЛАН

1. Словник
2. Кортеж

Завдання

1. Вивчити теоретичні основи написання алгоритмів з використанням словника і кортежу. Опрацювати приклади.

2. Відповідно до свого варіанту

- визначити умови;
- за допомогою формул описати варіанти виконання необхідний дій;
- написати програму, яка розв’язує завдання;
- організувати введення даних з клавіатури, виведення у консоль;
- для подання необхідної інформації реалізувати певну структуру даних, під час визначення якої використовувати будь-які комбінації вбудованих об'єктів (кортежі, словники, списки, рядки, числа); спробуйте звернутися до окремих елементів словника;
- реалізувати а) режим виведення на екран всіх значень словника; б) додавання (видалення) нового запису до (зі) словника; в) режим перегляду вмісту словника за відсортованими ключами (перетворити об’єкт подання ключів в список або скористатися функцією *sorted*, застосувавши її до словника, або об’єкту, який повертає метод *keys*);

Варіанти «Організація пошуку та вибору інформації»

1. Задано кількість очок, набраних кожною з n=9 команд–учасниць першості з футболу(перелік очок і команд задано у порядку зайнятих ними місць, тобто в порядку зменшення кількості набраних очок,жодна пара команд–учасниць не набрала однакову кількість очок). З'ясувалося, що в перелік забули включити ще одну, десяту, команду. Скласти програму, яка визначає: а) місце, яке зайняла зазначена команда (кількість набраних нею очок відомо, відомо також, що вона не стала чемпіоном і не зайняла останнє місце); б) назви команд, які набрали менше очок, ніж ця команда.

2. Задано дані про n=10 моменти часу однієї доби: години (значення від 0 до 23) і хвилини (від 0 до 59). Скласти програму, яка порівнює два будь-яких моменти часу за їх умовним порядковим номером і визначає, який з моментів був у цю добу раніше.

3. Задано дати кожної з n=10 подій, які відбулися після 1930 року: рік, номер місяця і число. Скласти програму, яка порівнює дві будь-яких події за

часом (визначає, яка з подій відбулася пізніше). Подію можна подати у вигляді рядку.

4. Задано дані про n=10 моменти часу однієї доби: години (значення від 0 до 23), хвилини (від 0 до 59) і секунди (від 0 до 59). Скласти програму, яка порівнює два будь-яких моменти часу (визначає, який з моментів відбувся у задану добу раніше).

5. Задано прізвища всіх n=10 співробітників фірми та їх адреси. Скласти програму, яка визначає, чи працюють у фірмі люди з прізвищем: Кузін, Куравльов, Кудін, Кульков або Кубиків. У разі позитивної відповіді надрукувати їх адреси.

6. Задано назви n=10 країн із загальною інформацією про них (площа, населення) і про частини світу, де вони розташовані. Скласти програму, яка визначає, чи є серед заданих країн ті, що знаходяться в Африці або в Азії. У разі позитивної відповіді надрукувати їх назви.

7. Задано дані про n=10 учнів класу: прізвище, ім’я, по батькові, дата народження (рік, номер місяця і число). Скласти програму, яка визначає, чи є в класі учні, у яких сьогодні день народження, і якщо так, то вивести їх ім'я та прізвище.

8. Задано прізвища та номери телефонів n=10 осіб у записнику. Скласти програму, яка визначає, чи є в записнику: а) телефон певної особи, і, якщо є, друкує номер її телефону; б) інформація про людину із заданим номером телефону, і, якщо є, вивести прізвище цієї людини.

9. Задано дані про n=10 учнів кількох шкіл, які займаються в районному Будинку творчості учнів (прізвище, ім'я, адреса, номер школи і клас). Скласти програму, яка визначає прізвище, ім'я та адресу учнів, що навчаються у визначеній школі в старших (10-11) класах, ці дані записати в окремий масив–список з елементами типу «ключ: кортеж».

10. Задано дані про кількість опадів, які випали за кожен день місяця, і про температуру повітря в ці дні. Скласти програму, що визначає, яка кількість опадів випала у вигляді снігу і яка – у вигляді дощу (вважати, що дощ іде, якщо температура повітря вище 0° С).

11. Задано дані про потужність двигуна (в кінських силах – к.с.) і вартість n=10 легкових автомобілів. Скласти програму, яка визначає загальну вартість автомобілів, у яких потужність двигуна перевищує 100 к. с.

12. Задано дані про вік і стать кожної з n=10 осіб. Скласти програму, яка визначає загальну кількість чоловіків.

13. Задано дані про кількість учнів у кожному з n=6 навчальних закладів і про тип закладів (школа, технікум або училище). Скласти програму, яка визначає загальну кількість учнів шкіл.

14. Задано дані про ціну і тираж кожного з n=5 журналів. Скласти програму, яка визначає середню вартість журналів, тираж яких менше 10 000 примірників.

15. Задано дані про вартість і «вік» кожної з n=10 моделей легкових автомобілів. Скласти програму, яка визначає середню вартість автомобілів, «вік» яких перевищує 6 років.

16. Задано дані про зріст і стать кожної з n=10 осіб. Скласти програму, яка визначає середній зріст чоловіків.

17. Задано дані про вартість кожної з n=5 моделей автомобілів і про їхній тип (легковий або вантажний). Скласти програму, яка визначає середню вартість легкових автобілів.

18. Задано дані про оцінки кожного з n=10 учнів класу з дванадцяти предметів. Скласти програму, яка визначає середню оцінку кожного учня і всього класу. Вивести прізвища учнів, у яких середня оцінка вище середньої в класі.

19. Задано дані у вигляді (маса; об'єм) n предметів, які виготовлені з різніх матеріалів. Скласти програму, яка визначає максимальну щільність матеріалу. *Підказка*: щільність = маса / об'єм.

20. Задано дані про чисельність населення (в мільйонах жителів) і площа (в тисячах квадратних км) n=10 держав. Скласти програму, яка визначає назву держави з максимальною щільністю населення.

21. Задано дані про оцінки кожного з n=10 учнів класу з десяти предметів. Скласти програму, яка визначає прізвище одного з учнів, який має: а) найбільшу суму оцінок; б) найменшу суму оцінок.

22. Задано дані про оцінки кожного з n=10 учнів класу з чотирьох предметів. Скласти програму, яка визначає прізвище одного з учнів, який має максимальну суму оцінок.

23. Задано дані про бали, набрані кожним з n=10 спортсменів-п'ятиборців в кожному з п'яти видів спорту. Скласти програму, яка визначає прізвище спортсмена-переможця змагань.

24. Задано дані про кількість очок, набраних кожною з n=10 команд-учасниць першості з футболу. Жодна пара команд не набрала однакову кількість очок. Скласти програму, яка визначає: а) назву команди, яка стала чемпіоном; б) назву команд, які посіли друге і третє місця; в) назву команд, які зайняли перше і друге місця, не використовуючи при цьому два оператори циклу (два проходи по масиву).

25. Задано дані про зріст кожного з n=15 учнів класу. Немає жодної пари учнів з однаковим зростом. Скласти програму, яка визначає: а) прізвища найвищого і найнижчого учнів класу; б) прізвища двох учнів, які є найвищими без урахування найвищого учня класу; в) прізвища двох учнів, які є найвищими в класі, не використовуючи при цьому два оператори циклу (два проходи по масиву).

26. Задано дані про n=10 співробітників фірми (прізвище, зарплата і стать). Скласти програму, яка визначає: а) прізвище особи, яка має найбільшу зарплату (вважати, що такий є лише один); б) прізвища чоловіка і жінки, які мають найменшу зарплату (вважати, що такі є і вони єдині в своїй групі співробітників).

27. Задано дані про n=10 співробітників фірми: прізвище, вік і відношення до військової служби (військовозобов'язаний чи ні). Скласти програму, яка визначає: а) прізвище наймолодшого за віком співробітника серед військовозобов'язаних (вважати, що такий є і він один); б) прізвища найстарших за віком людей серед військовозобов'язаних і серед невійськовозобов'язаних (вважати, що такі є і вони єдині в своїй групі).

28. Задано дані про розклад поїздів, які проходять через певну станцію: номер поїзда, призначення (звідки куди, наприклад, Москва – Омськ), час (години та хвилини) прибуття, час (години та хвилини) відправлення. Година та хвилина – цілі, додатні числа; значення годин не перевищує «23», значення хвилин – «59». Загальна кількість поїздів n=10. Поїзди приходять кожен день. Скласти програму, яка визначає, які поїзди (номер і призначення) стоять у визначений момент часу на станції.

29. Задано дані про про багаж (кількість речей і загальна вага багажу) n=10 пасажирів. Скласти програму, яка визначає: а) кількість пасажирів, які мають більше двох речей; б) чи є хоч один пасажир, багаж якого складається з однієї речі вагою менше 25 кг; в) число пасажирів, кількість речей яких перевершує середнє число речей всіх пасажирів; г) номер багажу, в якому середня вага однієї речі відрізняється від загальної середньої ваги однієї речі не більше ніж на 0,5 кг.

30. Задано дані про про зріст n=10 юнаків класу, впорядковані за зменшенням (немає жодної пари учнів, які мають однаковий зріст). На початку навчального року до класу вступив новий учень (його зріст не співпадає із зростом жодного з учнів класу, перевищує зріст самого низького учня і менше зросту найвищого). Скласти програму, яка визначає: а) прізвища всіх учнів, зріст яких менше росту «новенького»; б) прізвище учня, після якого слід записати прізвище «новенького», щоб впорядкованість не порушилася; в) прізвище учня, зріст якого найменше відрізняється від росту «новенького». У завданнях (а) і (б) умовний оператор не використовувати.

Контрольні запитання

1. Що таке словник? Назвіть його основні властивості.
2. Що таке кортеж? Назвіть його основні властивості.

Аудиторна робота

Приклад 12.1.

1. Створення словників

```
# наведені приклади створюють однакові словники
a=dict(one=1, two=2, three=3)
b={'one': 1, 'two': 2, 'three': 3}
c=dict(zip(['one', 'two', 'three'], [1, 2, 3]))
d=dict([('two', 2), ('one', 1), ('three', 3)])
e=dict({'three': 3, 'one': 1, 'two': 2})
print(a == b == c == d == e);print(a);print()
# Використання генераторів словників
print({string: string.upper() for string in ('one','two',
```

```
'three')})
```

Результат

```
True
{'three': 3, 'two': 2, 'one': 1}
{'three': 'THREE', 'two': 'TWO', 'one': 'ONE'}
```

2. Операції зі словниками

```
phonebook = {
    'Jack': '032-846',
    'Guido': '917-333',
    'Mario': '120-422',
    'Mary': '890-532',  # остання кома ігнорується}
# len(d) – кількість елементів
print(len(phonebook),'entries found');print()
```

d[key] – отримання значення з ключем *key*. Якщо ключ не існує, відображення реалізує спеціальний метод *__missing __ (self, key)*: якщо ключ не існує і метод *__missing__* не визначений, видається виняток KeyError

```
try:
    print('Mary:', phonebook['Mary'])
    print('Lumberjack:', phonebook['Lumberjack'])
except KeyError as e:
    print('No entry for', *e.args)
print()
```

d[key]=value – змінити значення або створити пару ключ-значення, якщо ключ не існує

```
phonebook['Lumberjack'] = '000-777'
```

key in d, key not in d – перевірка наявності ключа в відображенні

```
for person in ('Guido', 'Mary', 'Ahmed'):
    if person in phonebook:
        print(person, 'is in the phonebook')
    else:
        print('No entry found for', person)
print()
```

iter(d) – теж саме, що iter(d.keys())

```
print('People in the phonebook:')
for person in phonebook:
    print(person)
print()
```

copy() – створити неповну копію словника

```
phonebook_copy = phonebook.copy()
print('Phonebook:', phonebook)
print('Phonebook copy:', phonebook_copy);print()
```

clear() – видалити всі елементи зі словника

```
phonebook_copy.clear()
```

```
print('Phonebook:', phonebook)
print('Phonebook copy:', phonebook_copy);print()
```

(метод класу) **dict.fromkeys (sequence [, value])** – створює новий словник з ключами з послідовності *sequence* і заданим значенням (за замовчуванням None).

```
numbers_dict = dict.fromkeys(range(3), 42)
print(numbers_dict);print()
```

d.get(key[, default]) – безпечне отримання значення за ключем (ніколи не видає *KeyError*). Якщо ключ не знайдений, повертається значення default (за замовчуванням None).

```
for key in range(5):
    print('{}:'.format(key), numbers_dict.get(key, 0))
print()
```

d.items() – повертає об'єкт подання словника, який відповідає парам (двоелементним кортежам) вигляду (ключ, значення).

```
print('Items:', phonebook.items())
```

d.keys() – повертає об'єкт подання словника, який відповідає ключам словника.

```
print('Keys:', phonebook.keys())
```

d.values() – повертає об'єкт подання словника, який відповідає значенням

```
print('Values:', phonebook.values()); print()
```

d.pop(key[, default]) – якщо ключ *key* існує, метод *pop* видаляє елемент із словника і повертає його значення; якщо ключ не існує і задано значення *default*, то повертає дане значення, інакше видається виняток KeyError.

```
number = phonebook.pop('Lumberjack')
print('Deleted Lumberjack (was ' + number + ')')
print(phonebook);print()
```

```
>>> D={'toast': 4, 'muffin': 5, 'eggs': 3, 'ham': 1, 'spam': 2}
>>> D.pop('muffin') # видалення елементів словника за ключем
5
>>> D.pop('toast') # видаляє і повертає значення заданого ключа
4
>>> D
{'eggs': 3, 'ham': 1, 'spam': 2}
# видалення елементів списку за номером позиції
>>> L = ['aa', 'bb', 'cc', 'dd']
>>> L.pop() # видаляє і повертає останній елемент списку
'dd'
>>> L
['aa', 'bb', 'cc']
>>> L.pop(1) # видаляє і повертає елемент із заданої позиції
'bb'
>>> L
['aa', 'cc']
```

d.popitem() – видаляє довільну пару ключ-значення і повертає її. Якщо словник порожній, виникає виключення KeyError. Метод корисний для алгоритмів, які обходять словник, видаляючи вже оброблені значення (наприклад, деякі алгоритми, пов'язані з теорією графів).

```
person = phonebook.popitem()
print('Popped {} (phone: {})'.format(*person)); print()
```

d.setdefault(key[, default]) – якщо ключ *key* існує, повертає відповідне значення, інакше створює елемент з ключем *key* і значенням *default* (*default* за замовчуванням дорівнює None).

```
for person in ('Jack', 'Liz'):
    phone = phonebook.setdefault(person, '000-000')
    print('{}: {}'.format(person, phone))
print(phonebook);print()
```

d.update(mapping) – приймає інший словник або відображення, або ітерабельний об'єкт, який складається з ітерабельних об'єктів - пар ключ-значення, або іменовані аргументи. Додає відповідні елементи в словник, переписуючи елементи з існуючими ключами.

```
phonebook.update({'Alex': '832-438', 'Alice': '231-987'})
phonebook.update([('Joe', '217-531'), ('James', '783-428')])
phonebook.update(Carl='783-923', Victoria='386-486')
print(phonebook)
```

3. Використання елементів словника

Об'єкти, які повертають методи items (), keys (), values() - це об'єкти, які надають динамічне подання елементів словника (зміни у словнику автоматично відображаються і на цих об'єктах).

Операції з поданням словників:

iter (dictview) – отримання ітератора за ключами, значеннями або парами (ключ, значення); всі зображення словників при ітеруванні повертають елементи словника в однаковому порядку; при спробі змінити словник під час ітерування може виникнути виключення RuntimeError;

len (dictview) – кількість елементів в словнику;

x in dictview – перевірка існування ключа, значення або пари (ключ,-значення) в словнику.

```
dishes = {'eggs': 2, 'sausage': 1, 'bacon': 1, 'spam': 500}
keys = dishes.keys()
values = dishes.values()
# Ітерування
n = 0
for val in values:
    n += val
print(n)
# Ключі та значення ітерують у такому ж порядку
print(list(keys)); print(list(values))
# Об'єкти  динамічно відображають зміни в словнику
```

```
    del dishes['eggs'];del dishes['sausage']
    print(list(keys))
# Також вони підтримують операції із множинами
    print(keys & {'eggs', 'bacon', 'salad'})
    print(keys ^ {'sausage', 'juice'})
```

Результат

```
504
['bacon', 'eggs', 'sausage', 'spam']
[1, 2, 1, 500]
['bacon', 'spam']
{'bacon'}
{'juice', 'bacon', 'sausage', 'spam'}
```

4. Використання *collections.Counter*.

collections.Counter – це підклас *dict*, призначений для підрахунку об'єктів, які хешують; також її називають мультимножина. Елементи зберігають як ключі словника, а їх кількість – як значення.

```
    1)from collections import Counter
    2)
    counter = Counter();counter[1] += 1
    for i in range(3):
        print(counter[i])
    print()

    c=Counter('abcdeabcdabcaba')  # підрахунок кількості кожного
символа в рядку
    print(c.most_common(3))     # три найчастіших елементи
    print(sorted(c))            # всі унікальні елементи
    print(sorted(c.elements())) # всі елементи
    print(sum(c.values()))      # сума значень
    print(c['a'] )            # кількість літер 'a'

    for elem in 'shazam':      # додати нові літери
        c[elem] += 1
    print(c['a'])              # тепер у лічильнику сім літер 'a'
    del c['b']                 # видалити всі 'b'
    print(c['b'])

    d = Counter('simsalabim')  # створити новий лічильник
    c.update(d)                # додати його елементи до першого
    print(c['a'])              # тепер в ньому дев'ять 'a'

    c.clear()                   # очистити лічильник
    print(c)
    # Якщо рахунок елемента встановити «0» або зменшити до нуля,
    # він залишиться в лічильнику, поки не буде видалений явно
    c = Counter('aaabbc')
    c['b'] -= 2; print(c.most_common())
```

Результат

```
0
1
```

```
0
[('a', 5), ('b', 4), ('c', 3)]
['a', 'b', 'c', 'd', 'e']
['a', 'a', 'a', 'a', 'a', 'b', 'b', 'b', 'b', 'c', 'c', 'c', 'd',
'd', 'e']
15
5
7
0
9
Counter()
[('a', 3), ('c', 1), ('b', 0)]
```

5. Використання lambda-виразів

```
>>> key = 'got'
>>> {'already': (lambda: 2 + 2),
... 'got': (lambda: 2 * 4),
... 'one': (lambda: 2 ** 6)}[key]()
8
```

Коли інтерпретатор створює словник, кожний з вкладених *lambda-виразів* генерує і залишає після себе функцію для подальшого використання - звернення за ключем витягує одну з цих функцій, а круглі дужки забезпечують виклик витягнутої функції. При такому підході словник перетворюється в більш універсальний засіб множинного вибору. Щоб реалізувати те ж саме без використання *lambda-виразів*, довелося б написати три окремі інструкції *def* за межами словника, в якому ці функції використовують, і посилатися на функції за їхніми іменами:

```
>>> def f1(): return 2 + 2
...
>>> def f2(): return 2 * 4
...
>>> def f3(): return 2 ** 6
...
>>> key = 'one'
>>> {'already': f1, 'got': f2, 'one': f3}[key]()
64
```

Приклад 12.2. Відомо дані про вартість кожного з *n* найменувань товарів: «число грн., число копійок». Скласти програму, яка порівнює вартість двох будь-яких найменувань товарів (визначає, який з товарів коштує найдорожче).

Крок 1. *Формуємо список кортежів із ціною товара* (a_i, b_i).

Нехай задано списки цілих випадкових чисел [a_1, ..., a_n] і [b_1, ..., b_n]. Написати програму формування списку [(a_1, b_1), ..., (a_n, b_n)]. Вивести на екран початкові та отриманий списки.

```
import random
n=10; a=[random.randint(0,25) for i in range(0,10)]
b=[random.randint(0,5) for j in range(0,10)]
print(a); print(b)
xy=zip(a,b); # кортеж
```

```
xy=list(xy); print(xy)
```

Крок 2. *Формуємо словник із назвою різних товарів та їх ціною.*

```
L=[]
for i in range(1, n+1):
 a='name'+str(i)
 L.append(a)
print(L)
D={k:v for (k,v) in zip(L,xy)}
print (D)
```

Крок 3. *Виконуємо пошук найдорожчого товару*

```
a=list(D.values()); print (a)
b=a[0][0];nomer_t=1
for i in range(1, n):
 if b<a[i][0]:
   b=a[i][0]
   nomer_t=i+1
print (b, 'грн.', nomer_t, 'товар')
```

Весь код

```
import random
n=10
a=[random.randint(0,25) for i in range(0,10)]
b=[random.randint(0,5) for j in range(0,10)]
print(a); print(b)
xy=zip(a,b); # кортеж
xy=list(xy); print(xy)

L=[]
for i in range(1, n+1):
 a='name'+str(i); L.append(a); print(L)
D={k:v for (k,v) in zip(L,xy)}
print ('СЛОВНИК')
print (D);a=list(D.values()); print (a)

b=a[0][0];nomer_t=1
for i in range(1, n):
 if b<a[i][0]:
   b=a[i][0]; nomer_t=i+1
print (b, 'грн.', nomer_t, 'товар')
```

Результат

```
СЛОВАРЬ
{'name8': (9, 0), 'name10': (13, 4), 'name4': (22, 0), 'name6': (25, 2), 'name1': (21, 0),
 'name2': (22, 0), 'name9': (2, 1), 'name5': (19, 5), 'name3': (25, 1), 'name7': (17, 0)}
[(9, 0), (13, 4), (22, 0), (25, 2), (21, 0), (22, 0), (2, 1), (19, 5), (25, 1), (17, 0)]
25 грн. 4 товар
```

б) додавання (видалення) нового запису до (зі) словника:

```
import random
n=5
a=[random.randint(0,25) for i in range(0,10)]
b=[random.randint(0,5) for j in range(0,10)]
```

```
xy=zip(a,b); # кортеж
xy=list(xy); # print(xy)
L=[]
for i in range(1, n+1):
 a='name'+str(i); L.append(a);
D={k:v for (k,v) in zip(L,xy)}
print ('СЛОВНИК'); print(D);
D['name11']=(22,0);print (D) # Додавання нового елемента в словник
del D['name11'] # видалення ключів з словника
print (D)
```

Результат

```
СЛОВАРЬ
{'name5': (16, 4), 'name2': (7, 0), 'name3': (16, 1), 'name1': (13, 1), 'name4': (17, 5)}
{'name11': (22, 0), 'name3': (16, 1), 'name1': (13, 1), 'name5': (16, 4), 'name4': (17, 5), 'name2': (7, 0)}
{'name3': (16, 1), 'name1': (13, 1), 'name5': (16, 4), 'name4': (17, 5), 'name2': (7, 0)}
```

Підказка:

```
D['name11']=(22,0) # Додавання нового елемента в словник
del D[key] # видалення ключів
D.pop(key [, default]) # видаляє ключ і повертає значення;
#якщо ключа немає, повертає default (за замовчуванням - виняток).
```

Приклад 12.3. Використання функції *zip*

```
import random
n=int(input('Кількість стовпчиків рядків  = '))
a=[random.randint(-9,9) for u1 in range(1000)]
m=[random.sample(a,n) for u in range(n)]
# де "n" - це кількість рядків, стовпчиків
answer='Матриця не ортонормована'
for i in m:
    print(i)
    for j in m:
        v = sum(x*y for x,y in zip(i,j))
        if (i==j and v==1) or (i!=j and v==0):
            answer='Матриця ортонормована'; break
print(answer)
```

Результат

```
Кількість стовпчиків/рядків  = 5
[2, -7, -6, 9, 3]
[5, 9, 1, -5, 7]
[-1, 6, 8, 0, -2]
[-3, -5, -8, -3, 5]
[9, -1, -9, -5, 3]
Матриця не ортонормована
```

Приклад 10.4. Нехай задано назви 10 країн із загальною інформацією про них (площа, населення, частина світу, де знаходиться країна). Сформувати словник із цими даними.

1. Словник з використанням кортежу

```
table ={'Germany': ('Europe','154864','13189 KM^2'),
        'Ukraine': ('Europe', '467591','6248 KM^2'),
        'Egypt': ('Africa', '7891416','1488228 KM^2'),
        'Nigeria': ('Africa', '77804','99648 KM^2'),
        'China': ('Asia', '987654321','9137462 KM^2'),
        'USA': ('North America', '193764825','1230540 KM^2'),
```

```
      'Brazil': ('South America', '123456789', '894484 KM^2'),
      'Canada': ('North America', '456852', '69874 KM^2'),
      'India': ('Asia', '87864', '987697 KM^2'),
      'Finland': ('Europe', '28613','81468 KM^2')
        }
for cr in table: # Теж саме, що й for cr in table.keys()
  print(cr, '\t', table[cr])
```

2. Словник без використання кортежу (можна вносити зміни)

```
World={
 'Germany':{'cw':'Europe','ns': '154864','sq':'13189 KM^2'},
 'Ukraine':{'cw':'Europe','ns': '467591','sq':'6248 KM^2'},
 'Egypt':{'cw':'Africa', 'ns': '7891416','sq':'1488228 KM^2'},
 'Nigeria':{'cw':'Africa','ns': '77804','sq':'99648 KM^2'},
 'China':{'cw':'Asia','ns': '987654321','sq':'9137462 KM^2'},
 'USA':{'cw':'North America','ns':'193764825','sq':'1230540 KM^2'},
 'Brazil':{'cw':'South America','ns':'123456789','sq':'894484 KM^2'},
 'Canada':{'cw': 'North America','ns': '456852','sq':'69874 KM^2'},
 'India':{'cw': 'Asia','ns': '87864','sq':'987697 KM^2'},
 'Finland':{'cw': 'Europe','ns': '28613','sq':'81468 KM^2'}
        }
l=World.keys(); l=list(l); z=[]
for i in l:
    print(i, ':', World[i])
while True:
    n = str(input('Input name of country: '))
    ans = int()
    for i in range(0,10):
        ans = l[i]
        if n == l[i]:
            break
        else:
            i += 1; continue
    key1 = 'cw'; key2 = 'ns'; key3 = 'sq'
    print(n,'is  located  in',World[n][key1],'it  has',  World[n][key2],
'population and', World[n][key3])
    z.append({World[n][key1],World[n][key2],World[n][key3]})
    print(z)
    while True:
        try:
          answer=str(input('Do you want to change items ? (y/n) '))
        except:
          raise ValueError
        if answer == 'y':
          coun = str(input('change country: '))
          city = str(input('change part of world: '))
          word = str(input('change population : '))
          sq = str(input('change squer: '))
          World[coun][key1]=city;
          World[coun][key2] = word; World[coun][key3] = sq
        elif answer == 'n':
            for i in l:
                    print(i, ':', World[i])
            break
    break
```

Результат

```
Ukraine : {'cw': 'Europe', 'sq': '6248 KM^2', 'ns': '467591'}
Brazil : {'cw': 'South America', 'sq': '894484 KM^2', 'ns': '123456789'}
USA : {'cw': 'North America', 'sq': '1230540 KM^2', 'ns': '193764825'}
Canada : {'cw': 'North America', 'sq': '69874 KM^2', 'ns': '456852'}
Egypt : {'cw': 'Africa', 'sq': '1488228 KM^2', 'ns': '7891416'}
India : {'cw': 'Asia', 'sq': '987697 KM^2', 'ns': '87864'}
Germany : {'cw': 'Europe', 'sq': '13189 KM^2', 'ns': '154864'}
China : {'cw': 'Asia', 'sq': '9137462 KM^2', 'ns': '987654321'}
Nigeria : {'cw': 'Africa', 'sq': '99648 KM^2', 'ns': '77804'}
Finland : {'cw': 'Europe', 'sq': '81468 KM^2', 'ns': '28613'}
Input name of country: USA
USA is located in North America it has 193764825 population and 1230540 KM^2
[{'193764825', '1230540 KM^2', 'North America'}]
Do you want to change items ? (y/n) y
change country: USA
change part of world: 11
change population : 22
change squer: 33
Do you want to change items ? (y/n) n
Ukraine : {'cw': 'Europe', 'sq': '6248 KM^2', 'ns': '467591'}
Brazil : {'cw': 'South America', 'sq': '894484 KM^2', 'ns': '123456789'}
USA : {'cw': '11', 'sq': '33', 'ns': '22'}
Canada : {'cw': 'North America', 'sq': '69874 KM^2', 'ns': '456852'}
Egypt : {'cw': 'Africa', 'sq': '1488228 KM^2', 'ns': '7891416'}
India : {'cw': 'Asia', 'sq': '987697 KM^2', 'ns': '87864'}
Germany : {'cw': 'Europe', 'sq': '13189 KM^2', 'ns': '154864'}
China : {'cw': 'Asia', 'sq': '9137462 KM^2', 'ns': '987654321'}
Nigeria : {'cw': 'Africa', 'sq': '99648 KM^2', 'ns': '77804'}
Finland : {'cw': 'Europe', 'sq': '81468 KM^2', 'ns': '28613'}
```

3. БД країн у вигляді словника

```
World = {
 '1': {'co':'Germany','cw':'Europe','ns':'154864','sq':'13189 KM^2'},
 '2': {'co': 'Ukraine','cw': 'Europe','ns': '467591','sq':'6248 KM^2'},
 '3': {'co': 'Egypt','cw': 'Africa', 'ns': '7891416','sq':'1488228
KM^2'},
 '4': {'co': 'Nigeria','cw': 'Africa','ns': '77804','sq':'99648 KM^2'},
 '5': {'co': 'China','cw': 'Asia','ns': '987654321','sq':'9137462
KM^2'},
 '6': {'co': 'USA','cw': 'North America','ns': '193764825','sq':
'1230540 KM^2'},
 '7': {'co': 'Brazil','cw': 'South America','ns':
'123456789','sq':'894484 KM^2'},
 '8': {'co': 'Canada','cw': 'North America','ns': '456852','sq':'69874
KM^2'},
         '9': {'co': 'India','cw': 'Asia','ns': '87864','sq':'987697
KM^2'},
         '10':{'co': 'Finland','cw': 'Europe','ns': '28613','sq':'81468
KM^2'}
         }
l = World.keys();l = list(l); l.sort();
l.remove('10'); l.append('10')
for i in l: print(i, ':', World[i])
W_key = ('1','2','3','4','5','6','7','8','9','10')
key1 = 'cw'; key2 = 'ns'; key3 = 'sq'; key4 = 'co'
while True:
    n = str(input('Input name of country: '))
    ans = 0
    for i in range(0, 10):
        if n == World[W_key[ans]]['co']:
            break
        else:
            ans += 1; i += 1
            continue
    print(World[W_key[ans]][key4],'is located
in',World[W_key[ans]][key1],'it has',
```

```
World[W_key[ans]][key2],'population, and', World[W_key[ans]][key3])
```

Результат

```
1 : {'ns': '154864', 'cw': 'Europe', 'co': 'Germany', 'sq': '13189 KM^2'}
2 : {'ns': '467591', 'cw': 'Europe', 'co': 'Ukraine', 'sq': '6248 KM^2'}
3 : {'ns': '7891416', 'cw': 'Africa', 'co': 'Egypt', 'sq': '1488228 KM^2'}
4 : {'ns': '77804', 'cw': 'Africa', 'co': 'Nigeria', 'sq': '99648 KM^2'}
5 : {'ns': '987654321', 'cw': 'Asia', 'co': 'China', 'sq': '9137462 KM^2'}
6 : {'ns': '193764825', 'cw': 'North America', 'co': 'USA', 'sq': '1230540 KM^2'}
7 : {'ns': '123456789', 'cw': 'South America', 'co': 'Brazil', 'sq': '894484 KM^2'}
8 : {'ns': '456852', 'cw': 'North America', 'co': 'Canada', 'sq': '69874 KM^2'}
9 : {'ns': '87864', 'cw': 'Asia', 'co': 'India', 'sq': '987697 KM^2'}
10 : {'ns': '28613', 'cw': 'Europe', 'co': 'Finland', 'sq': '81468 KM^2'}
Input name of country: USA
USA is located in North America it has 193764825 population, and 1230540 KM^2
Input name of country:
```

Теоретична частина

1. СЛОВНИКИ

Списки – це впорядковані набори об'єктів, на відміну від них елементи в словниках зберігають і витягують за допомогою ключа (а не зсуву, який визначає їхню позицію). Словники (як вбудований тип даних) можуть замінити різні алгоритми пошуку і структур даних. Іноді словники грають роль записів і таблиць символів, здатні служити для подання розріджених (здебільшого порожніх) структур даних. Нижче наведено основні характеристики словників в Python:

1. Доступ до елементів за певним ключем, а не за індексом.
2. Неупорядковані колекції довільних об'єктів.
3. Змінна довжина, гетерогенність і довільна кількість рівнів вкладеності.
4. Відносяться до категорії «змінюваних відображень».
5. Таблиці посилань на об'єкти (хеш-таблиці).

Якщо списки – це масиви посилань на об'єкти, які підтримують можливість доступу до елементів за їхніми позиціями, то словники – це невпорядковані таблиці посилань на об'єкти, які підтримують доступ до елементів за ключем. Усередині словники реалізовані як хеш-таблиці (структури даних, які забезпечують високу швидкість пошуку), спочатку невеликого розміру і збільшуються за необхідності. Інтерпретатор Python використовує оптимізовані алгоритми хешування для забезпечення максимально високої швидкості пошуку ключів. Подібно списками, словники зберігають посилання на об'єкти (а не їх копії).

У табл. 12.1 наведено деякі операції, які найчастіше використовують над словниками (щоб отримати повний перелік операцій, скористайтеся функцією *dir*(dict) або *help*(dict), де *dict* – це ім'я типу). При визначенні у вигляді літералів словники записують як послідовність пар «*key:value*», розділених комами, укладених у фігурні дужки {}.

Списки і словники збільшуються в розмірах по-різному (словники зазвичай доповнюють за допомогою операції привласнення за новими ключами під час виконання програми, такий підхід не годиться для списків, які зазвичай розширюються за допомогою методу *append*).

Таблиця 12.1

Літерали словників та операції

Операция	*Интерпретация*
D = {}	Порожній словник
D = {‘spam’: 2, ‘eggs’: 3}	Словник із двох елементів
D={‘food’:{‘ham’: 1,‘egg’:2}}	Вкладення
D = dict(name=’Bob’, age=40) D = dict(zip(keyslist, valslist)) D = dict.fromkeys([‘a’, ‘b’])	Альтернативні способи створення словників: іменовані аргументи, застосування функції zip, списки ключів
D[‘eggs’] D[‘food’][‘ham’]	Доступ до елементу за ключем
‘eggs’ in D	Перевірка на входження: перевірка наявності ключа
 D.keys() D.values() D.items() D.copy() D.get(key, default) D.get (key [, default]) D1.update(D2) D.pop (key [, default]) D.popitem () D.clear () D.update ([other]) D.setdefault (key [, default])	Методи: визначає список ключів, визначає список значень, визначає пару (ключ, значення). копіювання, отримання значення за замовчуванням , повертає значення ключа; якщо ключа немає, повертає default (за замовчуванням None), злиття, видаляє ключ і повертає значення; якщо ключа немає, повертає default (за замовчуванням видає виняток), видаляє і повертає пару (ключ, значення). Якщо словник порожній, видає виняток KeyError, очищує словник оновлює словник, додаючи пари (ключ, значення) з other, повертає значення ключа, але якщо його немає, створює ключ зі значенням default (за замовчуванням None). .
len(D)	Довжина (кількість елементів)
D[key] = 42 del D[key]	Додавання / редагування ключів, видалення ключів
list(D.keys()) D1.keys() & D2.keys()	Подання словника
D = {x: x*2 for x in range(10)}	Генератори словників
classmethod dict.fromkeys (seq [, value])	створює словник з ключами з seq і значенням value (за замовчуванням None)

Порожній словник в літеральному поданні – це пуста пара дужок {}. Словники можуть вкладатися в середину інших словників, списків або кортежів. Доступ до елементів словника здійснюють за ключем, а для доступу до елементів вкладених словників використовують об'єднання серії індексів (ключів в квадратних дужках) в ланцюжок.

Коли інтерпретатор створює словник, він зберігає елементи в довільному порядку – щоб отримати значення назад, необхідно вказати ключ, з яким це значення було асоційоване. У звичайному випадку спочатку створюють словник, а потім виконують операції збереження нових ключів і звернення до

елементів за ключем:

```
>>> D={'spam': 2, 'ham': 1, 'eggs': 3} # Створення словника
>>> D['spam'] # витягування значення за ключем
2
>>> D # Випадковий порядок розташування
{'eggs': 3, 'ham': 1, 'spam': 2}
```

Тут змінній *D* присвоєно словник, в якому ключу '*spam*' відповідає цілочисельне значення «2». Для доступу до елементів словника використовують той самий синтаксис з квадратними дужками, що і при вилученні елементів списків, але в даному випадку доступ здійснюють за ключем. Остання інструкція в наведеному прикладі означає: порядок проходження ключів в словнику практично завжди відрізняється від початкового. Для забезпечення максимально високої швидкості пошуку за ключем (для хешування) ключі повинні розташовуватися в пам'яті в іншому порядку. Саме тому операції, які передбачають наявність встановленого порядку проходження елементів зліва направо (наприклад, витяг зрізу, конкатенація), непридатні до словників – вони дозволяють отримувати значення лише за ключами, а не за індексами.

Вбудована функція *len* може працювати і зі словниками – вона повертає кількість елементів в словнику (або довжину списку ключів). Оператор *in* перевірки входження дозволяє перевірити наявність ключа, а метод *keys* повертає всі ключі, наявні в словнику, у вигляді списку. Останній зручно використати для послідовної обробки словників, але при цьому ви не повинні робити будь-які припущення про порядок проходження ключів в списку. Оскільки результатом виклику методу *keys* є список, його завжди можна відсортувати:

```
>>> len(D) # кількість елементі словника
3
>>> 'ham' in D # перевірка на входження
True
>>> list(D.keys()) # створює новий список ключів
['eggs', 'ham', 'spam']
```

Оператор перевірки на входження *in* можна використати для роботи з рядками і списками, але точно так само і для роботи зі словниками. Це можливо завдяки тому, що словники визначають ітератори, які забезпечують покроковий обхід списків ключів. Виклик методу *keys* укладений у виклик функції list - list(D.keys()): метод *keys* повертає ітератор, а не список. Виклик функції list примусово виконує обхід всіх значень ітератора, що дозволяє вивести їх всі відразу. Ключі в словниках слідують в довільному порядку, який може змінюватися від версії до версії, тому не потрібно турбуватися, якщо у вас ключі будуть виведені в порядку, відмінному від того, що наведено тут.

Зміна словників. Словники, як і списки, відносяться до категорії змінних об'єктів, тому їх можна змінювати, збільшувати, зменшувати безпосередньо, не створюючи нові словники: щоб змінити або створити новий запис у словнику,

досить виконати операцію присвоювання за ключем. Інструкцію *del* також можна застосовувати до словників – вона видаляє значення, пов'язане з ключем, який грає роль індексу. Зверніть увагу на наявність вкладеного списку в наступному прикладі (значення для ключа '*ham*'). Всі типи-колекції в Python можуть вкладатися один в одного в довільному порядку:

```
>>> D
{'eggs': 3, 'ham': 1, 'spam': 2}
>>> D['ham'] = ['grill', 'bake', 'fry'] # зміна елементу
>>> D
{'eggs': 3, 'ham': ['grill', 'bake', 'fry'], 'spam': 2}
>>> del D['eggs'] # видалення елементу
>>> D
{'ham': ['grill', 'bake', 'fry'], 'spam': 2}
>>> D['brunch'] = 'Bacon' # додавання нового елементу
>>> D
{'brunch': 'Bacon', 'ham': ['grill', 'bake', 'fry'] , 'spam': 2}
```

Операція присвоювання за існуючим ключем словника призводить до зміни асоційованого з ним значення. Словники допускають виконання присвоювання за новим ключем (який раніше був відсутній), в результаті створюють новий елемент словника, як показано в попередньому прикладі для ключа '*brunch*'. Цей прийом не можна застосовувати до списків, бо в цьому випадку інтерпретатор виявляє вихід за межі списку і генерує повідомлення про помилку. Щоб збільшити розмір списку, необхідно використати такі інструменти списків, як метод append або присвоювання зрізу.

Методи словників забезпечують виконання різних операцій. Наприклад, методи словників values і items повертають список значень елементів словника і кортежі пар (*key*, *value*) відповідно:

```
>>> D = {'spam': 2, 'ham': 1, 'eggs': 3}
>>> list(D.values())
[3, 1, 2]
>>> list(D.items())
[('eggs', 3), ('ham', 1), ('spam', 2)]
```

Такі списки зручно використовувати в циклах, коли необхідно виконати обхід елементів словника. Спроба звернутися до неіснуючого елементу словника призводить до появи помилки, однак метод *get* в таких випадках повертає значення за замовчуванням (None або вказане значення). За допомогою цього методу можна реалізувати отримання значень за замовчуванням і уникнути появи помилки при зверненні до неіснуючого ключа:

```
>>> D.get('spam') # Ключ існує в словнику
2
>>> print(D.get('toast')) # Ключ не існує в словнику
None
>>> D.get('toast', 88)
88
```

Метод *update* реалізує операцію конкатенації для словників, при цьому

він не має ніякого відношення до впорядкування елементів зліва направо (для словників таке впорядкування не має сенсу). Він об'єднує ключі і значення одного словника з ключами і значеннями іншого, переписуючи значення з однаковими ключами:

```
>>> D
{'eggs': 3, 'ham': 1, 'spam': 2}
>>> D2 = {'toast':4, 'muffin':5}
>>> D.update(D2)
>>> D
{'toast': 4, 'muffin': 5, 'eggs': 3, 'ham': 1, 'spam': 2}
```

Словники мають набагато більшу кількість методів, ніж перераховано в табл. 12.1. Щоб отримати повний список, звертайтеся до інструкцій з Python.

Використання словників як «записів». Словники в Python здатні грати різні ролі. Вони здатні замінити реалізацію алгоритмів пошуку в структурах (бо операція індексування за ключем вже є операцією пошуку) і можуть подавати різні типи структурованої інформації. наприклад, словники – це один із багатьох способів опису властивостей елементів в програмах, тобто вони можуть грати ту ж роль, яку відіграють «структури» і «записи» в інших мовах програмування.

2. КОРТЕЖІ

Кортежі – це прості групи об'єктів. Вони діють як списки, за винятком того, що не допускають безпосередньої зміни (вони є незмінними) і в літеральній формі записуються як послідовність елементів в круглих дужках. Нижче коротко розглянуто їх властивості. Кортежі:

1. *Це впорядковані колекції об'єктів довільних типів*. Подібно рядкам і спискам, кортежі є колекціями об'єктів, впорядкованих за позицією (тобто вони забезпечують впорядкування свого вмісту зліва направо). Подібно списками, вони можуть містити об'єкти будь-якого типу.

2. *Забезпечують доступ до елементів за зсувом*. Подібно рядкам і списками, доступ до елементів кортежів здійснюють за зсувом (а не за ключем) – вони підтримують всі операції, які засновані на використанні зсуву, такі як індексування і витяг зрізу.

3. *Відносяться до категорії незмінних послідовностей*. Подібно рядкам і спискам, кортежі є послідовностями і підтримують багато операцій над послідовностями. Однак, подібно рядкам, кортежі є незмінними об'єктами, тому вони не підтримують жодну з операцій безпосередньої зміни, які застосовуються до списків.

4. *Мають фіксовану довжину, гетерогенні і підтримують довільну кількість рівнів вкладеності*. Оскільки кортежі є незмінними об'єктами, не можна змінити розмір кортежу, минаючи процедуру створення копії. З іншого боку, кортежі можуть зберігати інші складові об'єкти (тобто списки, словники та інші кортежі), а отже, підтримують довільну кількість рівнів вкладеності.

Масиви посилань на об'єкти. Подібно списками, кортежі простіше подати у вигляді масиву посилань на об'єкти, – кортежі зберігають покажчики

(посилання) на інші об'єкти, а операція індексування над кортежами виконується швидко. У табл. 12.2 наведено часто використовувані операції над кортежами. У програмному коді кортежі записують як послідовність об'єктів, розділених комами, укладену в круглі дужки. Порожні кортежі визначаються як пара порожніх круглих дужок ().

Таблиця 12.2

Літерали кортежів та операції

Операція	Інтерпретація
()	Порожній кортеж
T = (0,)	Кортеж з одного елементу (не вираз)
T = (0, ‘Ni’, 1.2, 3)	Кортеж із чотирьох елементів
T = 0, ‘Ni’, 1.2, 3	Ще один кортеж із чотирьох елементів Як виняток при визначенні кортежів інтерпретатор дозволяє опускати дужки, якщо синтаксично конструкція інтерпретується однозначно. Єдине місце, де круглі дужки є обов'язковими, - при передачі кортежів функціям у вигляді литералов.
T = (‘abc’, (‘def’, ‘ghi’))	Вкладені кортежі
T = tuple(‘spam’)	Створення кортежа із ітерованого об'єкта
T[i] T[i][j] T[i:j] len(T)	Індекс, індекс індексу, зріз, довжина
T1 + T2 T * 3	Конкатенація, повторення
for x in T: print(x) ‘spam’ in t2 [x ** 2 for x in T]	Обхід в циклі, перевірка входження
T.index(‘Ni’) T.count(‘Ni’)	Методы кортежів: пошук, підрахунок входжень

Розглянемо кортежі в дії. Як зазначено в табл. 12.2, кортежі не володіють методами, які є у списків (наприклад, кортежі не мають методу *append*). Проте кортежі підтримують звичайні операції над послідовностями, які застосовуються до рядків і до списків:

```
>>> (1, 2) + (3, 4) # Конкатенація
(1, 2, 3, 4)
>>> (1, 2) * 4 # Повторення
(1, 2, 1, 2, 1, 2, 1, 2)
>>> T = (1, 2, 3, 4) # Індексування, витяг зрізу
>>> T[0], T[1:3]
(1, (2, 3))
```

Особливості синтаксису визначення кортежів: коми і круглі дужки.

Другий і четвертий рядки в табл. 12.2 заслуговують додаткових пояснень. Щоб інтерпретатор розрізняв кортеж від простого виразу в дужках, то якщо необхідно отримати кортеж з одним елементом, потрібно додати кому після цього елемента, перед круглою дужкою, яка закриває:

```
>>> x = (40); x # Ціле число
40
>>> y = (40,); y # Кортеж, який містить ціле число
(40,)
```

Перетворення, методи і незмінюваність. Операції +, * і вилучення зрізу при застосуванні до кортежів повертають нові кортежі, кортежі мають скорочений набір методів. Якщо, наприклад, необхідно впорядкувати вміст кортежу, його спочатку слід перетворити в список, щоб перетворити в змінний об'єкт і отримати доступ до методу сортування або задіяти функцію *sorted*, яка приймає об'єкти будь-яких типів послідовностей (і не тільки):

```
>>> T = ('cc', 'aa', 'dd', 'bb')
>>> tmp = list(T) # Створити список із елементів кортежу
>>> tmp.sort();tmp # Відсортувати список
['aa', 'bb', 'cc', 'dd']
>>> T = tuple(tmp); T # Створити кортеж із елементів списку
('aa', 'bb', 'cc', 'dd')
>>> sorted(T) # або використати  вбудовану функцію sorted
['aa', 'bb', 'cc', 'dd']
```

де *list* і *tuple* – вбудовані функції, які використовують для перетворення в список і потім обернено в кортеж, ці функції створюють нові об'єкти, але завдяки їм створюється ефект перетворення. Для перетворення кортежів можна також використовувати генератори списків. Нижче з кортежу створюють список, причому попутно до кожного елементу додають число 20:

```
>>> T = (1, 2, 3, 4, 5)
>>> L = [x + 20 for x in T]; L
[21, 22, 23, 24, 25]
```

Генератори списків є операціями над послідовностями – вони створюють нові списки, але їх можна використати для обходу вмісту будь-яких об'єктів послідовностей, включаючи кортежі, рядки та інші списки. Кортежі мають обмежений набір методів – *index* і *count*, які діють так само, як методи списків:

```
>>> T = (1, 2, 3, 2, 4, 2) # Методи кортежів
>>> T.index(2) # Перше входження знаходиться в позиції 2
1
>>> T.index(2, 2) # наступне входження за позицією 2
3
>>> T.count(2) # визначити кількість двійок у кортежі
3
```

Правило незмінності застосовують тільки до самого кортежу, але не до об'єктів, які він містить. Наприклад, список всередині кортежу може змінюватися як зазвичай:

```
>>> T = (1, [2, 3], 4)
>>> T[1] = 'spam' # помилка: неможна змінити сам кортеж
TypeError: object doesn't support item assignment
>>> T[1][0] = 'spam'; T
# припустимо: вкладений змінюваний об'єкт можна змінити
```

```
(1, ['spam', 3], 4)
```