

Практична робота № 3

Тема: Робота зі списками та рядками в Python.

Мета: Ознайомитися з основними принципами роботи зі списками та рядками. Отримати практичні навички використання функцій та методів обробки спискових та рядкових даних.

Література:

Васильєв О.М. Програмування мовою Python – Тернопіль: Навчальна книга – Богдан, 2019. – 504с. стор. 227-234, 282-298

Зміст роботи:

Завдання 1.

Створіть список з шести цілих чисел. Додайте у створений список число -5 розмістивши його на другу позицію у списку.

Знайдіть мінімальний та максимальний елемент списку.

До отриманого списку додайте елемент у вигляді списку [1,2,3] починаючи з третього елементу.

В кінці списку додайте новий елемент в якому відобразіть ваше прізвище та ім'я.

Визначте кількість елементів списку.

Результати виведіть на екран.

Завдання 2.

На складі зберігається 20 видів товарів. До списку С занесено назву товару, до списку А занесено кількість одиниць кожного товару, до списку В - ціни цих товарів. Обчисліть загальну вартість товарів на складі та середню ціну. Визначте якого товару найбільше на складі.

Завдання 3.

Сформуйте список із 25 елементів що знаходяться в діапазоні від – 50 до 50. Всі додатні елементи записати у список А1, від'ємні в А2. Отримані списки роздрукувати.

Завдання 4.

Створіть список за зразком

`users = ["Mark", "Tom", "Bob", "Alice", "Tom", "Bill", "Tom", "Alex", "Shaun", "Mark"]`. Визначте скільки раз зустрічаються елементи Tom, Mark, Alice, John. Вилучіть зі списку третій елемент та елемент Tom. Результати виведіть на екран.

Завдання 5.

Створіть текстову змінну в якій напишіть коротку інформацію про себе(10 речень). Підрахуйте кількість символів «а» у тексті.

Виведіть на екран створений рядок розміщуючи по одному реченню у екранному рядку.

Завдання 6.

Створіть текстову змінну `str` в яку запишіть своє ім'я, групу в якій навчаєтесь, спеціальність. З створеного рядка виведіть на екран тільки назву групи. В рядку `str` замініть своє ім'я на прізвище і виведіть змінений рядок на екран.

Виконайте розподіл рядка по пробілу та обчисліть кількість слів у вашому рядку.

Завдання 7.

Створіть довільний список та перетворіть його на рядок використовуючи роздільник на власний розсуд.

Проілюструйте кодом твердження: «Замість списку в метод `join` можна передати простий рядок, тоді роздільник вставлятиметься між символами цього рядка.»

Завдання 8.

Скласти програму, яка для кожного з натуральних чисел від 1 до 10 включно виведе квадрат числа, якщо воно парне, і куб числа, якщо воно непарне.

Методичні рекомендації

Для звернення до елементів списку треба використовувати індекси, які представляють номер елемента в списку. Індекси починаються з нуля. Тобто другий елемент буде мати індекс 1. Для звернення до елементів з кінця можна використовувати негативні індекси, починаючи з -1. Тобто у останнього елемента буде індекс -1, у передостаннього - -2 і так далі.

```
numbers = [1, 2, 3, 4, 5]
print (numbers [0]) # 1
print (numbers [2]) # 3
print (numbers [-3]) # 3
numbers [0] = 125 # змінюємо перший елемент списку
print (numbers [0]) # 125
```

Якщо необхідно створити список, в якому повторюється одне і те ж значення кілька разів, то можна використовувати символ зірочки *. Наприклад, визначимо список з шести п'ятірок:

```
numbers = [5] * 6 # [5, 5, 5, 5, 5, 5]
print (numbers)
```

Щоб додати елемент застосовуються методи

append () – додає елемент в кінець списку,
insert() – додає елемент у вказану позицію,
extend() – додає групу елементів в кінець списку.

Для видалення - методи

remove () - вилучає елемент з певним значенням,
pop () – вилучає елемент за індексом,
clear () – видаляє всі елементи зі списку.

range (end): створюється набір чисел від 0 до числа end

Крім того, якщо нам необхідний послідовний список чисел, то для його створення зручно використовувати функцію **range()**, яка має три форми:

range (start, end): створюється набір чисел від числа start до числа end

range (start, end, step): створюється набір чисел від числа start до числа end з кроком step

```
numbers = list(range(10))
print(numbers) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers = list(range(2, 10))
print(numbers) # [2, 3, 4, 5, 6, 7, 8, 9]
numbers = list(range(10, 2, -2))
print(numbers) # [10, 8, 6, 4]
```

Для сортування за зростанням застосовується метод **sort ()**:

```
users = ["Tom", "Bob", "Alice", "Sam", "Bill"]
```

Якщо необхідно відсортувати дані в зворотному порядку, то ми можемо після сортування застосувати метод **reverse ()**:

```
users.sort()
print(users)    # ["Alice", "Bill", "Bob", "Sam", "Tom"]
```

Вбудовані функції Python `min ()` і `max ()` дозволяють знайти мінімальне і максимальне значення відповідно:

```
numbers = [9, 21, 12, 1, 3, 15, 18]
```

```
print(min(numbers))    # 1
print(max(numbers))    # 21
```

!!!!!!! При виклику методів необхідно пам'ятати, що рядки в Python відносяться до категорії незмінних послідовностей, тобто всі функції і методи можуть лише створювати новий рядок.

Службові символи (екрановані послідовності)

```
\\    # \
\'    # '
\"    # "
\0    # символ Null (не є ознакою кінця рядка)
\a    # код дзвінка
\b    # повернення (Backspace)
\f    # ознака кінця сторінки
\n    # ознака кінця рядка
\r    # повернення каретки
\t    # горизонтальна табуляція
\v    # вертикальна табуляція
```

Оператори для роботи з рядками.

Оператор (операція)	Використання в програмі	пояснення
+	s1+s2	Конкатенация
*	s*2	Повторение
[]	s[i]	Обращение к символу строки s по индексу i
[:]	s[i:j]	Вытягивание подстроки из позиции i до позиции j

Для пошуку підрядка в рядку в Python застосовується метод **find ()**, який повертає індекс першого входження підрядка в рядок і має три форми:

`find (str)`: пошук підрядка `str` ведеться з початку рядка до її кінця

`find (str, start)`: параметр `start` задає початковий індекс, з якого буде проводитися пошук

`find (str, start, end)`: параметр `end` задає кінцевий індекс, до якого буде йти пошук

Якщо підрядок не знайдено, метод повертає `-1`:

```
d=sp.find("два")
print(sp[39:50])
```

Для заміни в рядку однієї підрядка на іншу застосовується метод **`replace ()`**:

`replace (old, new)`: замінює підрядок `old` на `new`

`replace (old, new, num)`: параметр `num` вказує, скільки входжень підрядка `old` треба замінити на `new`

Метод **`split ()`** розбиває рядок на список підрядків в залежності від роздільника. Як роздільник може виступати будь-який символ або послідовність символів. Даний метод має такі форми:

`split ()`: як роздільник використовується пропуск

`split (delimiter)`: як роздільник використовується `delimiter`

`split (delimiter, num)`: параметр `num` вказує, скільки входжень `delimiter`

використовується для поділу. Частина, що залишилася рядка додається в список без поділу на підрядка.

Для перетворення списку в рядок та з'єднання рядків використовується метод **`join ()`**: він об'єднує список рядків. Причому поточний рядок, у якій викликається даний метод, використовується як роздільник:

```
sentence = "-".join(words)
```

```
ince = " | ".join(words)
```

Замість списку в метод `join` можна передати простий рядок, тоді роздільник вставлятиметься між символами цього рядка:

```
word = "hello"
```

```
joined_word = "|".join(word)
```

```
print(joined_word)    # h|e|l|l|o
```

Контрольні запитання.

1. Дайте визначення списку, рядку.
2. Поясніть в чому полягає відмінність рядків та списків.
3. Як здійснити доступ до елементів рядка, списку?
4. Назвіть та поясніть методи роботи зі списками.
5. Назвіть та поясніть методи роботи з рядками.
6. Які операції можна виконувати над рядками ?