## 1. Списки. Коротка теорія

Списки в Python - впорядковані змінювані колекції об'єктів довільних типів.
Щоб використовувати списки, їх потрібно створити. Створити список можна декількома способами. Наприклад, можна обробити будь ітеруємий об'єкт (наприклад, рядок) вбудованою функцією list:

```
list ('список')
```

Результат: ['с', 'п', 'и', 'с', 'о', 'к']

Список можна створити і за допомогою літералу:

```
s = [] # Порожній список
```

```
s = [ 'с', 'п', [ 'исок'], 2]
```

Як видно з прикладу, список може містити будь-яку кількість будь-яких об'єктів (в тому числі і вкладені списки), або не містити нічого. І ще один спосіб створити список - це генератори списків. Генератор списків – спосіб побудувати новий список, застосовуючи вираз до кожного елементу послідовності. Генератори списків дуже схожі на цикл for.

```
s = [c * 3 for c in 'list']
```

Результат: [ 'lll', 'iii', 'sss', 'ttt']

## 2. Введення і виведення даних при роботі зі списками

**Введення даних в список:**

Текстові дані, що розділяються пропусками, можна ввести в список таким чином:

```
s = input().split()
```

Цілі числа, що розділяються пропусками, можна ввести в список за допомогою list comprehension:

```
s = [int(x) for x in input().split()]
```

Для дійсних чисел:

```
s = [float(x) for x in input().split()]
```

**Виведення даних списку:**

Виведення даних списку через пропуск:

```
print(*s)
```

## 3. Функції і методи списків

Коли список створено, з ним можна проводити різні дії. Для списків доступні основні вбудовані функції, а також методи списків.

Таблиця методів списків:

| Метод | Що робить |
|---|---|
| list.append(x) | Додає елемент в кінець списку<br>*(Англ. дієслово "append" - "додавати")* |
| list.extend(L) | Розширює список list, додаючи в кінець все елементи списку L *(Англ. дієслово "extend" - "розширювати")* |
| list.insert(i, x) | Вставляє на i-ий елемент значення x<br>*(Англ. "insert" - "вставляти")* |
| list.remove(x) | Видаляє перший елемент у списку, який має значення x. ValueError, якщо такого елемента не існує<br>*(Англ. "remove" - "видалати")* |
| list.pop([i]) | Видаляє i-ий елемент і повертає його. Якщо індекс не вказано, видаляється останній елемент<br>*(Англ. "pop out" - "вискакувати")* |
| list.index(x, [start [, end]]) | Повертає положення першого елемента зі значенням x (при цьому пошук ведеться від start до end)<br>*(Англ. "index" - "індекс")* |
| list.count(x) | Повертає кількість елементів зі значенням x<br>*(Англ. "count" - "рахувати")* |
| list.sort([key=функция]) | Сортує список на основі функції<br>*(Англ. "sort" - "сортувати")* |
| list.reverse() | Обертає список<br>*(Англ. "reverse" - "зворотний")* |
| list.copy() | Копія списку<br>*(Англ. "copy" - "копіювати")* |
| list.clear() | Очищує список<br>*(Англ. "clear" - "очищати")* |

Потрібно відзначити, що методи списків, на відміну від строкових методів, змінюють сам список, а тому результат виконання не потрібно записувати в змінну.

## 4. Приклади роботи зі списками

Пошук максимального елемента:

```
lst = [5, 1, 7, 9, 2, 1, 3]
print (max(lst))
```

Результат: 9

Пошук мінімального елемента:

```
lst = [5, 1, 7, 9, 2, 1, 3]
print (min(lst))
```

Результат: 1

Пошук суми елементів:

```
lst = [5, 1, 7, 9, 2, 1, 3]
print (sum(lst))
```

Результат: 28

Пошук кількості елементів:

```
lst = [5, 1, 7, 9, 2, 1, 3]
print (len(lst))
```

Результат: 7

Пошук середнього арифметичного елементів:

```
lst = [5, 1, 7, 9, 2, 1, 3]
print (sum(lst)/len(lst))
```

Результат: 4.0

Сортування списку за зростанням:

```
lst = [5, 1, 7, 9, 2, 1, 3]
lst.sort()
print (lst)
```

Результат: [1, 1, 2, 3, 5, 7, 9]

Сортування списку за спаданням:

```
lst = [5, 1, 7, 9, 2, 1, 3]
lst.sort(reverse = True)
print (lst)
```

Результат: [9, 7, 5, 3, 2, 1, 1]

## 5. Казка «Два зоопарки»

У місті Макондо є зоопарк. В ньому живуть такі звірі:
Вовк, ведмідь, страус, мавпа, слон, лама, тигр, пума, поні, папуга.
Запишемо це як список, назву списку візьмемо по назві міста, де знаходиться зоопарк.

```
makondo = ['вовк', 'ведмідь', 'страус', 'мавпа', 'слон', 'лама', 'тигр', 'поні', 'пума']
```

У місті Звягель вирішили створити зоопарк. Але тварин поки що в ньому немає.
Запишемо це як пустий список, назву списку візьмемо по назві міста, де знаходиться зоопарк

```
zvyahel = []
```

Визначимо, скільки тварин живе в кожному зоопарку, для цього скористаємося функцією визначення довжини списку - len()

```
print(len(makondo))
print(len(zvyahel))
```

Результат:
9
0

Давайте дізнаємося, хто живе в першій клітці зоопарку Макондо і хто живе у останній.

```
print(makondo[0])
print(makondo[-1])
```

Результат:
вовк
пума

Дізнаємося, чи є в зоопарку Макондо слон:

```
print('слон' in makondo)
```

Результат: True

Дізнаємося, в якій за номером клітці зоопарку Макондо живе слон:

```
print(makondo.index('слон'))
```

Результат: 4

Якщо слонів декілька, то програма знайде, в якій з кліток живе перший. Якщо хтось хоче перевірити, що буде, якщо пошукати тварину, якої немає в зоопарку, типу print(makondo.index('чупакабра')), то він побачить помилку Python: ValueError: 'чупакабра' is not in list
Запобігти перериванню роботи програми через таку помилку, можна, перевіривши чи є потрібний нам елемент у списку:

```
if 'чупакабра' in makondo:
    print(makondo.index('чупакабра'))
```

Тобто, спочатку треба розібратися, чи є чупакабра в зоопарку взагалі. А якщо є, то тоді вже дізнаватися, в якій клітці.

Громада міста Макондо вирішила відпустити на волю поні – нехай просто так гуляє вулицями міста. Нам треба знайти, в якій клітці живе поні, виключити її зі списку звірів і подивитися, які тварини залишилися у зоопарку Макондо. Тобто поні ми випускаємо і клітку де вона жила не зберігаємо, а здаємо в металобрухт:

```
print(makondo.index('поні'))
```

Результат: 7

```
makondo.pop(7)
print(makondo)
```

Результат:  ['вовк', 'ведмідь', 'страус', 'мавпа', 'слон', 'лама', 'тигр', 'пума']

Отже, як бачимо, поні в зоопарку Макондо вже немає. Далі події нашої історії починають відбуватися у славному Звягелі. У звягельскій зоопарк привезли два крокодили, Гену и Колю,  а ще бегемота і жирафа, розмістили їх в клітках послідовно, один за одним.  Давайте дізнаємося як розселилися тварини…

```
zvyahel.append('крокодил Гена')
zvyahel.append('крокодил Коля')
zvyahel.append('бегемот')
zvyahel.append('жираф')
print(zvyahel)
```

Результат: ['крокодил Гена', 'крокодил Коля', 'бегемот', 'жираф']

Все зрозуміло. І тут несподівано у Звягель привезли новенького крокодила Жорика і адміністрація зоопарку Звягеля вирішила поселити його, щоб він не плакав від нудьги, між тими крокодилами, що вже є, тобто між кліткою номер нуль і кліткою номер один:

```
zvyahel.insert(1, 'крокодил Жорик')
print(zvyahel)
```

Результат: ['крокодил Гена', 'крокодил Жорик', 'крокодил Коля', 'бегемот', 'жираф']

Крокодил Жорик дуже з того радий. В цей час громада міста Макондо вирішила подарувати звягельскому зоопарку ведмедя. У Звягелі дуже зраділи і вирішити побудувати нову клітку, спеціально для ведмедя. Нам треба знайти, в якій клітці зоопарку Макондо живе ведмідь, виключити його зі списку зоопарку Макондо, розмістити його в кінці списку зоопарку Звягеля. А ще подивитися, в якому зоопарку які тварини тепер живуть:

```
print(makondo.index('ведмідь'))
```
Результат: 1

```
zvyahel.append(makondo.pop(1))
print(makondo)
print(zvyahel)
```

Результат:
['вовк', 'страус', 'мавпа', 'слон', 'лама', 'тигр', 'пума']
['крокодил Гена', 'крокодил Жорик', 'крокодил Коля', 'бегемот', 'жираф', 'ведмідь']

Громаді міста Макондо дуже сподобалося дарувати подарунки. І вони вирішили подарувати Звягелю всіх своїх тварин. А собі потім нових десь назбирають.  Як це записати на Python:

```
zvyahel.extend(makondo)
makondo.clear()
print(makondo)
print(zvyahel)
```

Результат:
[]
['крокодил Гена', 'крокодил Жорик', 'крокодил Коля', 'бегемот', 'жираф', 'ведмідь', 'вовк', 'страус', 'мавпа', 'слон', 'лама', 'тигр', 'пума']

Ось така історія про два зоопарки. Хто дочитав – молодець!