In [1]:

```
print(list ('список'))# обраюлька итерируемого объекта функцией list  в список
```

```
['с', 'п', 'и', 'с', 'о', 'к']
```

In [5]:

```
#можно создать и при помощи литерала
s = [] # пустой список
l = ['с', 'п', ['исок'], 2] # список из букв, строки (вложенный список) и цифры
print (s)
print(l)
r = ['с', 'п', list('исок'), 2] # применение функции list для созжания подсписка
print (r)
```

```
[]
['с', 'п', ['исок'], 2]
['с', 'п', ['и', 'с', 'о', 'к'], 2]
```

In [9]:

```
# создание списка с помощью генераторов
#Генератор списков - способ построить новый список,
#применяя выражение к каждому элементу последовательности. Генераторы списков очень пох
ожи на цикл for.
c = [c * 3 for c in 'list']
print(c)
# более сложная конструкция генератора создания списков
c = [c * 3 for c in 'list' if c != 'i']
print(c)
c = [c + d for c in 'list' if c != 'i' for d in 'spam' if d != 'a']
print(c)
```

```
['lll', 'iii', 'sss', 'ttt']
['lll', 'sss', 'ttt']
['ls', 'lp', 'lm', 'ss', 'sp', 'sm', 'ts', 'tp', 'tm']
```

In [12]:

```
#  можно создать копию списка используя :, в данном случае создается копия объекта
a = ['П','И','Э']
print (a)
b = a[:]
print (b)
# применение = выполняет формирование ссылки к объекту
a = ['П','И','Э']
a = ['П','И','Э']
b = a
print (b)
a[1] = 1
print (a)
print (b)
```

```
['П', 'И', 'Э']
['П', 'И', 'Э']
['П', 'И', 'Э']
['П', 1, 'Э']
['П', 1, 'Э']
```

In [24]:

```
# рассмотрим применеие функций для рабты со списками
#Добавление элемента в список осуществляется с помощью метода append().
a = []
a.append(3) # добавили в пустой список a  3
a.append("hello") # добавили в список a  hello
print(a)


# удаление элементов списка с помощью метода remove
b = [2, 3, 5]
print(b)
b.remove(3)
print(b)
# если требуется удалить элемент списка по индексу используем метод del
b = [2, 3, 5]
print(b)
del a[0]
print(a)
# возможно применение отрицательных индексов
v = ['1','2','3']
print(v)
del v[-1]
print(v)
# возможно получение подсписка
t =[1,2,3,4,5,6,7,8,9,0]
print (t)
s = t[1:4]
print(s)
```

```
[3, 'hello']
[2, 3, 5]
[2, 5]
[2, 3, 5]
['hello']
['1', '2', '3']
['1', '2']
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
[2, 3, 4]
```

In [26]:

```
# методы списков
# list.append(x) Добавляет элемент в конец списка. Ту же операцию можно сделать так a[l
en(a):] = [x].
a = [1, 2]
a.append(3)
print(a)
a[len(a):] = [4]
print(a)
```

```
[1, 2, 3]
[1, 2, 3, 4]
```

In [28]:

```
# list.extend(L)
#Расширяет существующий список за счет добавления всех элементов из списка L. Эквивалентно команде a[len(a):] = L
a = [1, 2]
l = [3, 4]
a.extend(l)
print (a)
a[len(a):] = l
print (a)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4, 3, 4]
```

In [30]:

```
# list.insert(i, x)
#  добавление элемента списка по индексу
a = [1, 2]
a.insert(3, 6)
print(a)
a.insert(len(a), 9)
print (a)
```

```
[1, 2, 6]
[1, 2, 6, 9]
```

In [32]:

```
#list.remove(x)
# удаляет первое вхождение элемента в списке
a = [1, 1, 2,1,3, 2]
print(a)
a.remove(2)
print(a)
```

```
[1, 1, 2, 1, 3, 2]
[1, 1, 1, 3, 2]
```

In [34]:

```
#list.pop([i])
#Удаляет элемент из позиции i и
#возвращает его. Если использовать метод без аргумента, то будет удален последний элемент из списка.
a = [1, 1, 2,1,3, 2]
print(a)
a.pop(3)
print(a)
a.pop()
print(a)
```

```
[1, 1, 2, 1, 3, 2]
[1, 1, 2, 3, 2]
[1, 1, 2, 3]
```

In [36]:

```
#list.clear()
#Удаляет все элементы из списка. Эквивалентно del a[:].
a = [1, 1, 2,1,3, 2]
print(a)
a.clear()
print(a)
b = [1, 1, 2,1,3, 2]
print(b)
del b[:]
print(b)
```

```
[1, 1, 2, 1, 3, 2]
[]
[1, 1, 2, 1, 3, 2]
[]
```

In [41]:

```
#list.index(x[, start[, end]])
# возвращает индекс элемента в списке
a = [1, 1, 2,1,3, 2]
print(a)
print(a.index(3))
print(a.index(1,2,4)) # ищем 1 в диапазоне от 2 до 4 элементов списка
```

```
[1, 1, 2, 1, 3, 2]
4
3
```

In [42]:

```
#list.count(x)
# возвращает количество вхождений элементов в список
a=[1, 2, 2, 3, 3]
print(a.count(2))
```

```
2
```

In [47]:

```
#list.sort(key=None, reverse=False)
#Сортирует элементы в списке по возрастанию. Для сортировки в обратном порядке использу
йте флаг reverse=True.
#Дополнительные возможности открывает параметр key, за более подробной информацией обра
титесь к документации.
a = [1, 4, 2, 8, 1]
print (a)
a.sort()
print(a)
a.sort(reverse=True)
print(a)
```

```
[1, 4, 2, 8, 1]
[1, 1, 2, 4, 8]
[8, 4, 2, 1, 1]
```

In [48]:

```
#list.reverse()
# изменяет порядок элементов в списке
a = [1, 3, 5, 7]
print(a)
a.reverse()
print(a)
```

```
[1, 3, 5, 7]
[7, 5, 3, 1]
```

In [49]:

```
#list.copy()
#Возвращает копию списка. Эквивалентно a[:].
a = [1, 7, 9]
b = a.copy()
print(a)
print(b)
b[0] = 8
print(a)
print(b)
```

```
[1, 7, 9]
[1, 7, 9]
[1, 7, 9]
[8, 7, 9]
```

In [52]:

```
#List Comprehensions чаще всего на русский язык переводят как  абстракция списков или с
писковое включение,
#является частью синтаксиса языка, которая предоставляет простой способ построения спис
ков.
#Проще всего работу list comprehensions показать на примере. Допустим вам необходимо со
здать
#список целых чисел от 0 до n, где n предварительно задается. Классический способ решен
ия
#данной задачи выглядел бы так:
n = int(input())
a=[]
for i in range(n):
        a.append(i)
print(a)
```

```
6
[0, 1, 2, 3, 4, 5]
```

In [54]:

```
#Использование list comprehensions позволяет сделать это значительно проще:
n = int(input())
a = [i for i in range(n)]
print(a)
#или вообще вот так, в случае если вам не нужно больше использовать n:
a = [i for i in range(int(input()))]
print(a)
```

```
7
[0, 1, 2, 3, 4, 5, 6]
7
[0, 1, 2, 3, 4, 5, 6]
```

In [57]:

```
#List Comprehensions как обработчик списков
#В языке Python есть две очень мощные функции для работы с коллекциями: map и filter.
#Они позволяют использовать функциональный стиль программирования, не прибегая к помощи циклов,
#для работы с такими типами как list, tuple, set, dict и т.п.
#Списковое включение позволяет обойтись без этих функций.
#Приведем несколько примеров для того, чтобы понять о чем идет речь.


#Пример с заменой функции map.
#Пусть у нас есть список и нужно получить на базе него новый, который содержит элементы первого,
#возведенные в квадрат. Решим эту задачу с использованием циклов:
a = [1, 2, 3, 4, 5, 6, 7]
b = []
for i in a:
    b.append(i**2)
print('a = {}\nb = {}'.format(a, b))

#задача, решенная с использованием map
v = [1, 2, 3, 4, 5, 6, 7]
r = list(map(lambda x: x**2, v))
print('v = {}\nr = {}'.format(v, r))
```

```
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
v = [1, 2, 3, 4, 5, 6, 7]
r = [1, 4, 9, 16, 25, 36, 49]
```

In [63]:

```
# использование метода filter
# Построим на базе существующего списка новый, состоящий только из четных чисел без мет
ода  filter:
a = [1, 2, 3, 4, 5, 6, 7]
b = []
for i in a:
    if i%2 == 0:
        b.append(i)
print('a = {}\nb = {}'.format(a, b))
# решение задачи с использованием метода  filter
v = [1, 2, 3, 4, 5, 6, 7]
r = list(filter(lambda x: x % 2 == 0, v))
print('v = {}\nr = {}'.format(v, r))
#Решение через списковое включение
l = [1, 2, 3, 4, 5, 6, 7]
k = [i for i in l if i % 2 == 0]
print('l = {}\nk = {}'.format(l, k))
```

```
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
v = [1, 2, 3, 4, 5, 6, 7]
r = [2, 4, 6]
l = [1, 2, 3, 4, 5, 6, 7]
k = [2, 4, 6]
```

In [65]:

```
#Слайсы (срезы) списков
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(a)
# Получить первые пять элементов списка
print(a[0:5])
# Получить элементы с 3-го по 7-ой
print(a[2:7])
# Взять из списка элементы с шагом 2
print(a[::2])
# Взять из списка элементы со 2-го по 8-ой с шагом 2
print(a[1:8:2])
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4]
[2, 3, 4, 5, 6]
[0, 2, 4, 6, 8]
[1, 3, 5, 7]
```

In [69]:

```
#Слайсы можно сконструировать заранее, а потом уже использовать по мере необходимости.
 Это возможно сделать,
#в виду того, что слайс – это объект класса slice.
#Ниже приведен пример, демонстрирующий эту функциональность:
s = slice(0, 5, 1)
print(a[s])
s = slice(1, 8, 2)
print(a[s])
```

```
[0, 1, 2, 3, 4]
[1, 3, 5, 7]
```

In [75]:

```
#List Comprehensions… в генераторном режиме
import sys
a = [i for i in range(10)]
type(a)
print(sys.getsizeof(a))
b = (i for i in range(10))
type(b)
print(sys.getsizeof(b))
c = [i for i in range(10000)]
print(sys.getsizeof(c))
d = (i for i in range(10000))
print(sys.getsizeof(d))
```

```
192
120
87624
120
```

In [ ]:

```
Задача 1:
Напишите программу, на вход которой подаётся список чисел одной строкой.
Программа должна для каждого элемента этого списка вывести сумму двух его соседей.
Для элементов списка, являющихся крайними, одним из соседей считается элемент,
находящий на противоположном конце этого списка.
Например, если на вход подаётся список «1 3 5 6 10», то на выход ожидается список «13
6 9 15 7».
Если на вход пришло только одно число, надо вывести его же.
Вывод должен содержать одну строку с числами нового списка, разделёнными пробелом.
Задача 2:
Напишите программу, которая принимает на вход список чисел в одной строке и выводит на
экран в одну строку значения,
которые повторяются в нём более одного раза.
Выводимые числа не должны повторяться, порядок их вывода может быть произвольным.
Например: 4 8 0 3 4 2 0 3
Задача 3:
Выполните обработку элементов прямоогольной матрицы A, имеющей N строк и M столбцов.
Все элементы имеют целый тип. Дано целое число H.
Определите, какие столбцы имеют хотя бы одно такое число, а какие не имеют.
Задача 4:
Список задается пользователем с клавиатуры. Определите, является ли список симметричным
.
Задача 5:
Список задается пользователем с клавиатуры.
Определите, можно ли удалить из списка каких-нибудь два элемента так, чтобы новый списо
к оказался упорядоченным
Задача 6:
Список задается пользователем с клавиатуры.
Определите, сколько различных значений содержится в списке.
Задача 7:
Список задается пользователем с клавиатуры.
Удаление из списка элементов, значения которых уже встречались в предыдущих элементах
Задача 8:
Пользователь вводит упорядоченный список книг (заданной длины по алфавиту). Добавить но
вую книгу, сохранив
упорядоченность списка по алфавиту
Задача 9:
Дан список целых чисел. Упорядочьте по возрастанию только:
    а) положительные числа;
    б) элементы с четными порядковыми номерами в списке.
Задача 10:
Даны два списка. Определите, совпадают ли множества их элементов.
Задача 11:
Дан список. После каждого элемента добавьте предшествующую ему часть списка.
Задача 12:
Пусть элементы списка хранят символы предложения. Замените каждое вхождение слова "itma
threpetitor" на "silence".
Задача 13*:
Дан текстовый файл. Создайте двусвязный список, каждый элемент которого содержит
количество символов в соответствующей строке текста.
Задача 14*:
Создайте двусвязный список групп факультета. Каждая группа представляет собой односвязн
ый список студентов.
Задача 15*:
Дан список студентов. Элемент списка содержит фамилию, имя, отчество, год рождения, кур
с, номер группы,
оценки по пяти предметам. Упорядочите студентов по курсу, причем студенты одного курса
располагались в алфавитном порядке.
Найдите средний балл каждой группы по каждому предмету. Определите самого старшего студ
```

```
ента и самого младшего студентов.
Для каждой группы найдите лучшего с точки зрения успеваемости студента.
```

In [ ]:

```
#дополнительные методы работы со списками
#len(list): возвращает длину списка
#min(list): возвращает наименьший элемент списка
#max(list): возвращает наибольший элемент списка
```