Тема 1.3. Списки

Тема 1.3.1

 # Список - нумерованный набор объектов. Изменяемый тип данных:

 # по индексу можно не только получить элемент, но и изменить его.

 # Списки поддерживают операции: обращение к элементу по индексу, получение среза, конкатенацию(+), повторение(*), проверку на вхождение

 # Создание пустого списка

```
l = list()
print(l)
input()       # пауза ждем ввод
```

 # элементы списка внутри квадратных скобок

```
arr = [1, 'str', 34, 20]
print('Это список arr:', arr)
```

 # дополнение списка методом append()

```
arr = []
arr.append(5)
arr.append('Лондон')
arr.append(35)
print('Это список arr:', arr)
```

 # преобразовать строку в список

```
z = 'Крокодил'
print('Это строка z:', z)

l = list('строка')
print('Это список l:', l)
l_new = []
l_new = list(z)
print('Это список l_new:', l_new)
```

 # групповое присваивание. изменение элемента списка

```
x = y = [1, 2]
print('Это список x:', x, 'Это список y:', y)
y[1] = 100
print('Это список x:', x, 'Это список y:', y)

x, y = [1, 5], [1, 5]
print('Это список x:', x, 'Это список y:', y)
y[1] = 100
print('Это список x:', x, 'Это список y:', y)
```

 # Проверить, на какой объект ссылается переменная

```
x = y = [1, 2]
print('Это список x:', x, 'Это список y:', y)
print(x is y)
x, y = [1, 5], [1, 5]
print('Это список x:', x, 'Это список y:', y)
print(x is y)
```

Задание 1.

1.Создать пустой список. Вывести на печать.

2.Создать список с конкретными элементами (например, названия 4-х городов: Париж, Москва, Лондон, Барселона . Вывести на печать.

3.Дополнить список городов новым городом методом append()

4.Ввести с экрана строку. Преобразовать строку в список. Вывести на печать.

5.Создать 2 списка, используя групповое присваивание. Вывести на печать. Изменить 1-й элемент 2-го списка. Проверить, как изменились оба списка.

Тема 1.3.2

 # создание поверхностной копии списка

 # проверка списков на равенство

 # вложенность списков

 # создание полной копии списка

```
import copy
x = [1, 2, 3, 4, 5]
print('Это список x', x)

y = list(x)
print('Это список y', y)

z = x[:]
print('Это список z', z)
print(x is y)
print(x is z)

y[1] = 100
z[0] = 200

print('Это список x', x)
print('Это список y', y)
print('Это список z', z)

# создание вложенного списка
x = [1, [2, 3, 4, 5]]
print('Это список x:', x)
y = list(x)
print(y is x)
y[1][1] = 100
print('Это список x', x)
print('Это список y', y)

# создание полной копии списка функцией deepcopy() из модуля copy
x = [1, [2, 3, 4, 5]]
y=copy.deepcopy(x)
y[1][1] = 100
```

```
print('Это список x', x)

print('Это список y', y)


x = [1, 2]

print('Это список x', x)

y = [x, x]

print('Это список y', y)


z = copy.deepcopy(y)

print(z[0] is x)

print(z[1] is x)

print(z[0] is z[1])
```

Задание 2.

1. Создать список. Создать его поверхностную копию. Проверить списки на равенство
2. Создать вложенный список.
3. Создать полную копию списка функцией deepcopy() из модуля copy

Тема 1.3.3. Операции над списками

```
# Операции над списками.

# позиционное присваивание переменным значений элементов списка


# получить количество элементов списка

arr = [1, 2, 3, 4, 7]

z = len(arr)

print(z)

# получить последний элемент списка

last = arr[len(arr)-1]

print(last)


arr = [1, 2, 3, 4, 8]
```

```
print(arr[-1],arr[len(arr)-1])

# изменить элемент по индексу
arr = [1, 2, 3, 4, 7]
print('Это список arr:', arr)
arr[0] = 750
arr[3] = 30
print('Это список arr:', arr)

# извлечение среза
arr = [1, 2, 3, 4, 5]
m = arr[:]   # поверхностная копия списка
print(m)
print(m is arr)

# вывести элементы в обратном порядке
arr = [1, 2, 3, 4, 5]
print(arr[::-1])

# вывести список без первого и последнего элементов
print(arr[1:]) # без первого элемента
print(arr[:-1]) # без последнего элемента

# получить первые два элемента списка
print(arr[0:2])

# вывести фрагмент от 2-го до 4-го элементов списка
print(arr[1:4])

# изменить или вырезать фрагменты списка
arr = [1, 2, 3, 4, 5]
print(arr)
```

```
arr[1:3] = [6, 7]  # изменить значения элементов 1 и 2
print(arr)

arr[1:3] = []  # удалить  элементы 1 и 2
print(arr)

# соединить два списка
arr1 = [1, 2, 3, 4, 5]
arr2 = [6, 7, 8, 9]
arr3 = arr1 + arr2
print(arr3)

# добавить элементы в список
arr1 = [1, 2, 3, 4, 5]
arr1+ = [12, 15, 17]
print(arr1)

arr1 = [3, 4, 5]
arr2 = [1, 2, 3]
arr1+ = arr2
print(arr1)

# повторить список несколько раз   - *
arr = [1, 2, 3]
print(arr*3)

# проверка на вхождение
print(2 in arr, 8 in arr)

z = 5
print(z in arr)
```

Задание 3. Операции над списками

1.Создать список из 5 элементов. Вывести на печать. Определить и вывести на печать количество элементов списка.

2.Определить и вывести на печать первый и последний элементы списка.

3. Изменить 3-й и 5-й элементы списка по индексу. Вывести измененный список на печать.

4.Создать поверхностную копию списка. Вывести на печать.

5. Вывести на печать элементы списка в прямом и обратном порядке.

6. Вывести на печать полный список, список без первого элемента, список без последнего элемента.

7. Вывести на печать первые два элемента списка

8.Вывести на печать фрагмент списка от 1-го до 3-го элементов

9.Изменить значения элементов 3 и 5. Вывести измененный список на печать

10. Удалить 2-й и 4-й элементы списка. Вывести измененный список на печать

11. Создать 2 списка, вывести на печать, соединить списки и вывести новый список на печать.

12. Добавить новые элементы в список. Вывести измененный список на печать

13. Вывести на печать список, повторив его 4 раза, используя операцию *

14. Создать список из 6 элементов. Проверить, содержатся ли в нем значения 2, 7, 5,значение переменной n.

Тема 1.3. 4. Многомерные списки.

 # создание вложенного списка

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

 # доступ к элементу многомерного списка

```
arr = [[1, ['a', 'b'], 3], [4, 5, 6], [7, 8, 9]]
print(arr)
print(arr[0][1][0])
```

```
mass = [[1, {'a': 10,'b': ['s', 5]}]]
print(mass[0][1]['b'][0])
```

 # перебор элементов списка

```
for i in arr: print(i)
```

arr = [1, 2, 3, 4] **# элементы имеют неизменяемый тип - число**

```
for i in arr: i+ = 10
print(arr)
```

arr = [[1, 2], [3, 4]] **# элементы имеют изменяемый тип - список**

```
for i in arr: i[1]+ = 10
print(arr)
```

arr = [1, 2, 3, 4, 7] **# элементы имеют неизменяемый тип - число**

```
for i in range(len(arr)):
    arr[i]+ = 10
print(arr)
```

```
arr = [1, 2, 3, 4]
for i, elem in enumerate(arr):
    arr[i]* = 2
print (arr)
```

```
arr = [1, 2, 3, 4]
i, c = 0, len(arr)
while i<c:
    arr[i]* = 2
    i+ = 1
print(arr)
```

Задание 4. Многомерные списки

1.Создать вложенный список. Вывести на печать

2.Вывести на печать отдельные элементы списка, используя их индексы.

3.Вывести на печать список поэлементно, используя цикл for

4.Увеличить каждый элемент списка в три раза, используя цикл for. Вывести измененный список на печать. Сделать то же самое, используя цикл while

Тема 1.3. 5. Генераторы списков и выражения-генераторы

```
# каждый элемент списка умножить на 2
arr = [1, 2, 3, 4]
arr = [i*2 for i in arr]
print(arr)
# получить четные элементы списка и умножить их на 10
arr = [1, 2, 3, 4]
arr = [i*10 for i in arr if I % 2 == 0]
print(arr)

# или можно так:

arr = []
for i in [1, 2, 3, 4]:
   if I % 2 == 0:
      arr.append(i*10)
print(arr)

# получить четные элементы вложенного списка и умножить их на 10
arr = [[1, 2], [3, 4], [5, 6]]
arr = [j*10 for i in arr for j in i if j % 2 == 0]
print(arr)

# или можно так:
arr = []
for i in [[1, 2], [3, 4], [5, 6]]:
   for j in i:
      if j % 2 == 0:
```

```
        arr.append(j*10)
print(arr)


# просуммировать четные числа в списке
arr = [2, 5, 8, 16, 7]
z = sum((i for i in arr if I % 2 == 0))
print(z)
```

Задание 5.

1. Создать список из 7 элементов, вывести на печать.
2. Все элементы списка, кратные 3, увеличить в два раза. Вывести измененный список на печать.
3. Вычислить сумму всех элементов списка, кратных двум, вывести на печать.
4. Вычислить сумму всех элементов списка, вывести на печать.
5. Записать в другой массив все элементы первого, кратные двум

Тема 1.3.6.

 # перебор элементов списка без циклов

 # lambda, map, zip, reduce

```
old_list = ['1', '2', '3', '4', '5', '6', '7']
new_list = []
for item in old_list:
   new_list.append(int(item))

print (new_list)
input()

old_list = ['1', '2', '3', '4', '5', '6', '7']
new_list = list(map(int, old_list))
print(new_list)

def miles_to_kilometers(num_miles):
   """ Converts miles to the kilometers """
```

```
    return num_miles * 1.6

mile_distances = [1.0, 6.5, 17.4, 2.4, 9]
kilometer_distances = list(map(miles_to_kilometers, mile_distances))
print (kilometer_distances)

# то же с использованием lambda-выражения
mile_distances = [1.0, 6.5, 17.4, 2.4, 9]
kilometer_distances = list(map(lambda x: x * 1.6, mile_distances))

print (kilometer_distances)
# сложить соответствующие элементы  списков
l1 = [1, 2, 3]
l2 = [4, 5, 6]

new_list = list(map(lambda x,y: x + y, l1, l2))
print (new_list)

# фильтрация элементов последовательности
# функция, передаваемая в filter должна возвращать значение True / False,
# чтобы элементы корректно отфильтровались

mixed = ['мак', 'просо', 'мак', 'мак', 'просо', 'мак', 'просо', 'просо', 'просо', 'мак']
zolushka = list(filter(lambda x: x == 'мак', mixed))
print(zolushka)

# Вычисление суммы всех элементов списка при помощи reduce:
from functools import reduce
items = [1, 2, 3, 4, 5]
sum_all = reduce(lambda x, y: x + y, items)

print (sum_all)
```

 # Вычисление наибольшего элемента в списке при помощи reduce:

```
from functools import reduce

items = [1, 24, 17, 14, 9, 32, 2]

all_max = reduce(lambda a,b: a if (a > b) else b, items)


print (all_max)
```

 # Функция zip объединяет в кортежи элементы из последовательностей переданных в качестве аргументов.

 # zip прекращает выполнение, как только достигнут конец самого короткого списка.

```
a = [1, 2, 3]

b = "xyz"

c = (None, True)


res = list(zip(a, b, c))

print (res)
```

Задание 6.

1. Создать список, вывести на печать.
2. Используя цикл for и append, создать новый список из первого и вывести на печать.
3. Создать новый список из первого, используя map, и вывести на печать.
4. Увеличить все элементы списка, кратные двум, в 4 раза, вывести список на печать.
5. Создать 2 списка, каждый из 4 элементов. Сложить соответствующие элементы двух списков и создать новый список. Вывести на печать. Использовать map и lambda.
6. Создать список с неоднократно повторяющимся элементом. Создать новый список, в который отфильтровать повторяющийся в первом списке элемент, используя функцию filter.
7. Создать новый список. Вычислить сумму всех элементов списка при помощи reduce, вывести на печать.
8. Создать новый список. Вычислить наибольший элемент в списке при помощи reduce, вывести на печать.
9. Создать последовательности: 2 списка и 2 строки. Используя функцию zip, объединить в кортеж элементы из этих последовательностей, переданных в качестве аргументов. Вывести результат на печать.

Тема 1.3. 7.

```
# функция filter() - проверка, фильтрация элементов последовательности
# функция reduce()

from functools import reduce
a = [1, -4, 6, 8, -10]
def func(x):
    if x > 0:
        return 1
    else:
        return 0
b = filter(func, a)
b = list(b)
print(b)

a = [-1, 0, 1, 0, 0, 1, 0, -1]
b = list(filter(None, a))
print(b)

s = ['a', '', 'd', 'cc', ' ']
ss = list(filter(None, s))
print(s)

def f_filter(elem):
    if elem<0: return False
    return True
arr = [-1, 2, -3, 4, 0, -20, 10]
arr = filter(f_filter, arr)
arr = list(arr)
print(arr)

t = (-1, 2, -3, 4, 0, -20, 10)   # кортеж
t = filter(f_filter, t)
```

```
t = list(t)
print(t)


# redice() получить сумму всех элементов
def f_sum(elem1, elem2):
    print("%s,%s" % (elem1, elem2))
    return elem1 + elem2
arr = [1, 2, 3, 4, 25]
summa = reduce(f_sum, arr)
print(summa)
```

Задание 7.

1. Написать функцию zz, которая определяет элемент, значение которого больше 20 и меньше 25.
2. Создать список, в котором есть несколько элементов, значения которых больше 20 и меньше 25.
3. Используя функцию zz и функцию filter() перенести в новый список из первого элементы, значения которых больше 20 и меньше 25.

Тема 1.3.8.

```
# добавление  элементов списка методы append, extend insert +
# метод append(<объект>)
arr = [1, 2, 3]
arr.append(4)                    # добавить число
print(arr)

arr.append([7, 8])               # добавить список
print(arr)

arr.append((9, 10))              # добавить кортеж

arr.append("str")                # добавить строку - можно только один аргумент!
print(arr)
```

 # метод extend(<Последовательность>)

```
arr = [1, 2, 3]

arr.extend([7, 8, 9, 10]) # добавить список
print(arr)

arr.extend((5, 710)) # добавить кортеж
print(arr)

arr.extend("dao")  # добавить буквы из строки
print(arr)
```

 # конкатенация

```
arr = [1, 2, 3]
arr+ = [3, 4, 5]
print(arr)
arr2 = [7, 8, 9]
arr+ = arr2
print(arr)
```

 # метод insert(<индекс>,<объект>) добавляет только один объект в указанную позицию

```
arr  = [1, 2, 3, 4, 5, 6]
arr.insert(0, 0)
print(arr)
arr.insert(-1, 20)
print(arr)
arr.insert(-5, 45)
print(arr)
arr.insert(4, 120)
```

```
print(arr)

# чтобы добавить несколько объектов, нужно присвоить значения срезу
arr = [1, 2, 3]
arr[:0] = [4, 5, 6]
print(arr)
arr[:0] = [45, 77, 34]
print(arr)


# удаление элементов списка методы pop, remove, оператор del


# метод pop удаляет элемент по индексу и возвращает его
# если индекс не указан, удаляется и возвращается последний элемент списка
arr = [1, 2, 3]
z = arr.pop(1)
print(arr)
print(z)
z=arr.pop()
print(arr)
print(z)

# метод remove(<значение>) удаляет первый элемент, содержащий указанное значение.
# метод изменяет список и ничего не возвращает, если значения нет - сообщение ValueeError

arr = [1, 2, 3, 1, 1]
print(arr)
arr.remove(1)
print(arr)

# чтобы удалить все повторяющиеся элементы списка, надо список преобразовать в множество,
# а затем множество обратно в список
```

```
arr = [1, 2, 3, 1, 1, 2, 2, 3, 3]
print(arr)
s = set(arr)   # преобразуем список в множество
print(s)
arr = list(s)    # преобразуем множество в список
print(arr)

# оператор del
arr = [1, 2, 3, 4, 5]
del arr[4]
print(arr)

del arr[:2] # удаляет первый и второй элементы списка
print(arr)
```

Задание 8.

1. Создать список. Вывести на печать.
2. Используя метод append, добавить в список число, список, кортеж, строку. Вывести обновленный список на печать
3. Используя метод extend, добавить в список другой список, кортеж, буквы из строки. Вывести обновленный список на печать.
4. Добавить в список другой список, используя конкатенацию. Вывести список на печать
5. Используя метод insert(<индекс>,<объект>), добавить один объект в указанную позицию.
6. Создать список. Вывести на печать.Добавить в список несколько объектов. Вывести список на печать.
7. По индексу удалить из списка элемент, используя метод pop.
8. Используя метод remove, удаляет первый элемент, содержащий указанное значение.
9. Создать список, содержащий повторяющиеся элементы. Вывести на печать. Удалить из списка все повторяющиеся элементы. Вывести на печать.
10. Создать список, содержащий 5 элементов, вывести на печать. Используя оператор del, удалить из списка 3-й элемент, первый и последний элемент. Итоговый список вывести на печать.

Тема 1.3.9

```
# поиск элемента в списке
# in позволяет определить, есть ли элемент в списке, но не сообщает его позицию
arr = [1, 2, 3, 4, 5, 6]
```

```
print(2 in arr)  # возвращает True
print(7 in arr) # возвращает False

# метод index(<Значение>[,<Начало>[,<Конец>]])
# если Начало и Конец не указаны, поиск производится с начала списка
# если элемента в списке нет, возбуждается исключение ValueError

arr = [1, 2, 1, 2, 1]
print(arr.index(1))
print(arr.index(2))
print(arr.index(1, 1))
print(arr.index(1, 3, 5))

# метод count(<Значение>) - количество элементов с указанным значением
arr = [1, 2, 1, 1, 2, 2, 2, 2, 3]
print(arr.count(1))
print(arr.count(2))
print(arr.count(3))

z = 1
l = 2
m = 3
print(arr.count(z))
print(arr.count(l))
print(arr.count(m))

z = int(input('Введите число:'))
print(arr.count(z))

# функции max() и min()
arr = {7, 4, 3, 8, 10, 1, 25}
print(arr)
```

```
print(max(arr))

print(min(arr))


# функция any(<Последовательность>)

# возвращает True, если хоть один элемент возвращает True, иначе возвращает False

print(any([0, None]))    # False

print(any([0, None, 1]))  # True

print(any([]))         # False


# функция all(<Последовательность>)

# возвращает True, если все элементы возвращают True или нет элементов, иначе возвращает False

print(all([0, None]))    # False

print(all([0, None, 1]))  # False

print(all([]))         # True

print(all(["str", 10]))   # True
```

Задание 9

1. Создать список. Вывести на печать.
2. Определить, содержится ли в списке элемент, заданный конкретным значением, переменной. В зависимости от результата вывести сообщение.
3. Создать список, в котором некоторые элементы содержатся несколько раз. Вывести на печать.
4. Определить, сколько раз в списке содержится повторяющийся элемент. При поиске задать элемент конкретным значением, переменной, ввести значение в переменную с экрана
5. Создать список. Определить и вывести на печать максимальный и минимальный элементы списка.

Тема 1.3.10

```
# Переворачивание и перемешивание списка

# метод reverse() меняет порядок следования элементов списка на противоположный

# метод изменяет текущий список и ничего не возвращает

arr = [1, 2, 3, 4, 5, 6]

print(arr)

arr.reverse()
```

```
print(arr)
```

 # функция reversed(<Последовательность>) изменить порядок следования и получить новый список

```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(arr)
print(list(reversed(arr)))
```

```
for i in reversed(arr): print(i)  # вывод с помощью цикла
```

```
print([i for i in reversed(arr)]) # использование генератора списков
```

 # функция shuffle(<Список> [, <число от 0.0 до 1.0>]) перемешивание списка случайным образом

 # ф-я ничего не возвращает. Если второй параметр не указан, то исп-ся значение, возвращаемое ф-ей random()

```
import random
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
random.shuffle(arr)
print(arr)
```

Задание 10

1. Создать список. Вывести на печать. Используя метод reverse() изменить порядок следования элементов списка на противоположный и вывести на печать.
2. Используя функцию reversed(<Последовательность>) изменить порядок следования элементов в списке и получить новый список. Вывести на печать.
3. Используя функцию reversed(<Последовательность>) изменить порядок следования элементов в списке и получить новый список. Вывести на печать поэлементно, используя цикл for.
4. Создать список. Вывести на печать. Используя функцию shuffle(<Список> [, <число от 0.0 до 1.0>]) перемешать список случайным образом и вывести на печать

Тема 1.3.11

 # выбор элементов случайным образом

 # функция choice(<Последовательность>) из модуля random

```
import random
```

```
print(random.choice(['soy', 'd', 'ser', 1, 5]))


arr =  ['der', 'farr', 4, 7, 8]
print(arr)
print(random.choice(arr))


# функция sample(<Последовательность, <Количество элементов>)
# сам список не изменяется


arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(arr)
arr2 = random.sample(arr, 4)
print(arr2)
```

Задание 11

1. Создать список. Вывести на печать. Выбрать элементы случайным образом, используя функцию choice(<Последовательность>) из модуля random. Вывести на печать.
2. Создать список. Вывести на печать. Выбрать элементы случайным образом, используя функцию sample(<Последовательность, <Количество элементов>) из модуля random. Вывести на печать.

Тема 1.3.12.

```
# сортировка списка метод sort() форматы:
# sort([cmp=None][,key=None][,reverse=False])
# sort([<Пользовательская функция>[,<Функция>[,<Порядок элементов>]]])
# метод изменяет список и ничего не возвращает


arr = [1, 4, 7, 2, 6, 10, 23, 15, 8]
print(arr)
arr.sort()  # сортировка по возрастанию
print(arr)


arr = [1, 4, 7, 2, 6, 10, 23, 15, 8]
```

```
print(arr)
arr.sort(reverse = True)  # сортировка по убыванию
print(arr)

# стандартная сортировка  важен регистр
arr = ["Карр", "Каркуша", "каркуша"]
print(arr)
arr.sort()
for i in arr:
    print(i)

# пользовательская сортировка
# чтобы регистр не учитывался, нужно указать ссылку в параметре key

arr = ["Карр", "Каркуша", "каркуша"]
print(arr)
arr.sort(key = str.lower)
for i in arr:
    print(i)

def f_sort(a,b):
    a1 = a.lower()
    b1 = b.lower()
    if a1 > b1: return 1
    if b1 < b1: return -1
    return 0

# функция sorted(<Последовательность>[,cmp=None][,key=None][,reverse=False]}

arr = [1, 4, 7, 2, 6, 10, 23, 15, 8]
print(arr)
arr_new=sorted(arr)
```

```
print(arr_new)


arr1 = sorted(arr, reverse = True)

print(arr1)


arr = ["Карр", "Каркуша", "каркуша"]

print(arr)

m = sorted(arr, key = str.lower)

for i in m:

   print(i)
```

Задание 12.

1. Создать список. Вывести на печать.
2. Отсортировать список по возрастанию, используя метод sort(). Вывести на печать.
3. Отсортировать список по убыванию, используя метод sort(). Вывести на печать.
4. Выполнить стандартную сортировку списка с учетом и без учета регистра
5. Выполнить пользовательскую сортировку списка с учетом и без учета регистра.
6. Создать список. Вывести на печать.
7. Отсортировать список по возрастанию, используя функцию sorted(). Вывести на печать.
8. Отсортировать список по убыванию, используя метод функцию sorted(). Вывести на печать.

Тема 1.3.13.

```
# заполнение списка числами функция range([<Начало>,]<Конец>[,<Шаг>])

# если шаг не указан, используется значение 1

arr = list(range(11))

print(arr)


arr = list(range(1, 16))

print(arr)


arr = list(range(15, 0, -1))

print(arr)


# получение списка со случайными значениями или случайными элементами из другого списка
```

 # функция sample(<Последовательность>,<Количество элементов>) из модуля random

```
import random

arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(arr)

arr_new = random.sample(arr, 3)

print(arr_new)

arr1 = random.sample(range(300), 5)

print(arr1)
```

 # преобразование списка в строку метод join():

 # <Строка>=<Разделитель>.join(<последовательность>)

```
arr = ["Строка 1", "Строка 2", "Строка 3"]

arr_new = "-".join(arr)

print(arr_new)
```

```
arr = ["Строка 1", "Строка 2", "Строка 3", str(34)]

arr_new = "-".join(arr)

print(arr_new)
```

```
arr = ["Строка 1", "Строка 2", "Строка 3", 34]

arr_new = "-".join(str(arr))

print(arr_new)
```

Задание 13.

1. Заполнить список числами, используя функцию range(). Вывести на печать.
2. Получить список со случайными значениями или случайными элементами из другого списка, используя функцию sample() из модуля random. Вывести на печать.
3. Преобразовать список в строку методом join(). Вывести на печать.