ЛЕКЦИЯ 4 – СПИСКИ, СЛОВАРИ, ЦИКЛЫ И МАССИВЫ

4.1 Списки и кортежи

Список представляет собой упорядоченную последовательность элементов. Он очень гибкий и является одним из самых используемых типов в Python. Элементы списка не обязательно должны быть одного типа.

Объявить список довольно просто. Внутрь квадратных скобок помещаются элементы списка, разделённые запятой:

```
a = [67, 5, 90, 20, 30]
b = ['Маша', 'Ваня', 'Лена', 'Марина', 'Арнольд']
print(a)
print(b)
aa = a[:]  # Автономная копия списка
aaa = list(a)  # Автономная копия списка
aaaa = a.copy()  # Автономная копия списка
aaa.reverse()  # Сортировка списка в обратном порядке
print(a[0])  # Первый элемент 67
print(a[2])  # Третий элемент 90
print(b[1])  # второй элемент Ваня
b.append('Дима')  # Добавление элемента в конец списка
print(b[-1])  # Печать последнего элемента Дима
a.sort()  # Сортировка списка
print(a)  # [5, 20, 30, 67, 90]
print(aa)  # [67, 5, 90, 20, 30]
print(aaa)  # [30, 20, 90, 5, 67]
print(aaaa)  # [67, 5, 90, 20, 30]
```

```
[67, 5, 90, 20, 30]
['Маша', 'Ваня', 'Лена', 'Марина', 'Арнольд']
67
90
Ваня
Дима
[5, 20, 30, 67, 90]
[67, 5, 90, 20, 30]
[30, 20, 90, 5, 67]
[67, 5, 90, 20, 30]
```

Так же как и список, кортеж (tuple) является упорядоченной последовательностью элементов. Вся разница заключается в том, что кортежи неизменяемы. Кортежи используются для защиты данных от перезаписи и обычно работают быстрее, чем списки, т.к. их нельзя изменять.

Для создания кортежа нужно поместить внутрь круглых скобок элементы, разделённые запятой:

```
t = (5, 'program', 1 + 3j)
print("t[0] =", t[0])
print("t[1] =", t[1])
print("t[2] =", t[2])
for i in t:
    print(i, end=" --- ")
```

```
t[0] = 5
t[1] = program
t[2] = (1+3j)
5 --- program --- (1+3j) ---
```

Списки имеют большой набор функций:

- append , extend - добавление;
- insert - вставка;
- index - найти индекс первого вхождения конкретного элемента;
- count - подсчет повторов элемента;
- remove , del - удаление элемента;
- sort - сортировка;
- reverse - реверс;
- pop - извлечение элемента;
- len - длина списка;
- max - максимальный элемент;
- min - минимальный элемент;
- оператор in - проверка элемента на вхождение.

4.2 Словари

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному, называется словарем или ассоциативным массивом. Соответствующая структура данных в языке Python 3 называется dict.

Каждый элемент словаря состоит из двух объектов: ключа и значения. В примере ниже ключом является название страны, а значением является название столицы. Ключ идентифицирует элемент словаря, значение является данными, которые соответствуют данному ключу. Значения ключей — уникальны, двух одинаковых ключей в словаре быть не может.

```
# Создадим пустой словарь Capitals
Capitals = dict()

# Заполним его несколькими значениями
Capitals['Russia'] = 'Moscow'
Capitals['Ukraine'] = 'Kiev'
Capitals['USA'] = 'Washington'

Countries = ['Russia', 'France', 'USA', 'Russia']

for country in Countries:
    # Для каждой страны из списка проверим, есть ли она в словаре Capitals
    if country in Capitals:
        print('Столица страны ' + country + ': ' + Capitals[country])
    else:
        print('В базе нет страны с названием ' + country)
```

```
Столица страны Russia: Moscow
В базе нет страны с названием France
Столица страны USA: Washington
Столица страны Russia: Moscow
```

Методы словарей:

- clear() - очищает словарь;
- copy() - возвращает копию словаря;
- fromkeys(seq[, value]) - создает словарь с ключами из seq и значением value (по умолчанию None);
- get(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None);

- items() - возвращает пары (ключ, значение);
- keys() - возвращает ключи в словаре;
- pop(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение);
- popitem() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены;
- setdefault(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ с значением default (по умолчанию None);
- update([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!);
- values() - возвращает значения в словаре.

```
Countries = {'Russia': 'Moscow', 'Ukraine': 'Kiev', 'USA': 'Washington',
'Kazakhstan': 'Astana'}

print(Countries)

key1 = 'USA'
key2 = 'us'
if key1 in Countries:
    del Countries[key1]

try:
    del Countries[key2]
except KeyError:
    print('Нет элемента с ключом "' + key2 + '"  в словаре')

print(Countries)
```

```
{'Ukraine': 'Kiev', 'Russia': 'Moscow', 'Kazakhstan': 'Astana',
'USA': 'Washington'}
Нет элемента с ключом "us"  в словаре
{'Ukraine': 'Kiev', 'Russia': 'Moscow', 'Kazakhstan': 'Astana'}
```

```
# Преобразование списка-кортеджа в словарь
users = (
    ("+111123455", "Tom"),
    ("+384767557", "Bob"),
    ("+958758767", "Alice")
)
users_dict = dict(users)
print(users)
print(users_dict)

# получаем элемент с ключом "+111123455"
print(users_dict["+111123455"])  # Tom

# установка значения элемента с ключом "+384767557"
users_dict["+384767557"] = "Bob Smith"
print(users_dict["+384767557"])  # Bob Smith
```

```
(('+111123455', 'Tom'), ('+384767557', 'Bob'), ('+958758767',
'Alice'))
{'+958758767': 'Alice', '+384767557': 'Bob', '+111123455': 'Tom'}
Tom
Bob Smith
```

4.2.1 Сортировка словарей

```
# Пример подсчета одинаковых символов в текстовом файле с использованием словаря
my_dict = dict()

my_text = open(u'D:/text.txt', 'r').read()

for c in my_text:
    if c in my_dict:
        my_dict[c] = my_dict[c] + 1
    else:
        my_dict.update({c: 1})

for w in sorted(my_dict, key=my_dict.get, reverse=True):
    print(w, my_dict[w])
```

```
L 9
o 9
f 8
p 8
```

4.3 Циклы

```
mas = ['Ленин', 'Сталин', 'Хрущёв', 'Брежнев', 'Горбачёв', 'Путин']
mas.append('Медведев')

for x in mas:
    print('правил ' + x + ' а после него... ')

a = 1
while (a < 5):
    print(a, "^ 2 =", a * a, ' ', a ** 2)
    a = a + 1

print('Висит груша, нельзя скушать. Что это такое?')

s = ''
while ((s != 'Лампочка') and (s != 'лампочка')):
    s = input('Введите ответ и нажмите Enter: ')

print('Вы отгадали загадку!')
```

```
правил Ленин а после него...
правил Сталин а после него...
правил Хрущёв а после него...
правил Брежнев а после него...
правил Горбачёв а после него...
правил Путин а после него...
правил Медведев а после него...
1 ^ 2 = 1   1
2 ^ 2 = 4   4
3 ^ 2 = 9   9
4 ^ 2 = 16   16
Висит груша, нельзя скушать. Что это такое?
Введите ответ и нажмите Enter: лампочка
Вы отгадали загадку!
```

```
print('Введите стих, отделяя строки нажатием Enter, последней строкой введите слово Конец')
```

```
while (True):
    s = str(input())
    if ((s == 'Конец') or (s == 'конец')):
        break
    k = 0
    for x in s:  # Перебор по буквам слова
        if (x in 'аеёиоуыэюя'):  # Проверка буквы на гласную
            k = k + 1
    print(k)
```

```
Это мой новый стих
5
Красивый и новый
6
Да
1
Конец
```

```
month = ["январь", "февраль", 'март', 'апрель']
print(month[0])  # январь
print(month[0:2])  # ['январь', 'февраль']
print('Как прекрасны месяцы', " и ".join(month), "!")  # Как прекрасны месяцы январь
и февраль и март и апрель !
month.append("май")
month.remove("январь")
print()

# ["февраль", "март", "апрель", "май"]
for m in month:
    print(m, end=' ')  # февраль март апрель май
print()

for m in (reversed(month)):
    print(m, end=' ')  # май апрель март февраль
print()

for k, m in enumerate(reversed(month)):
    print(k, m, end=' ')  # 0 май 1 апрель 2 март 3 февраль
print()

for i in range(10):
    print(i, end=' ')  # 0 1 2 3 4 5 6 7 8 9
print()

for i in range(9, -1, -1):
    print(i, end=' ')  # 9 8 7 6 5 4 3 2 1 0
print()

for i in range(0, 10, 4):
    print(i, end=' ')  # 0 4 8
print()

for i in range(3, 10):
    print(i, end=' ')  # 3 4 5 6 7 8 9
print()

for i in range(9, -1, -2):
    print(i, end=' ')  # 9 7 5 3 1
print()
```

```
январь
['январь', 'февраль']
Как прекрасны месяцы январь и февраль и март и апрель !
```

```
февраль март апрель май
май апрель март февраль
0 май 1 апрель 2 март 3 февраль
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
0 4 8
3 4 5 6 7 8 9
9 7 5 3 1
```

4.4 Массивы

Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются матрицами или двумерными массивами. В языке программирования Python таблицу можно представить в виде **списка** строк, каждый элемент которого является в свою очередь списком, например, чисел.

При изменении переменных, указывающих на элементы массива, изменяется и сам массив.

```
a = [[1, 2, 3], [4, 5, 6]]  # Двумерный массив
print(a[0])  # [1, 2, 3]
print(a[1])  # [4, 5, 6]

b = a[0]
print(b)  # [1, 2, 3]
print(a[0][2])  # 3

a[0][1] = 7
print(a)  # [[1, 7, 3], [4, 5, 6]]
print(b)  # [1, 7, 3]

b[2] = 9
print(a[0])  # [1, 7, 9]
print(b)  # [1, 7, 9]
```

Для обработки и вывода списка, как правило, используют два вложенных цикла. Первый цикл перебирает номер строки, второй цикл бежит по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for i in range(len(a)):
    for j in range(len(a[i])):
        print(a[i][j], end=' ')
    print()
```

```
1 2 3 4
5 6
7 8 9
```

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for row in a:
    for elem in row:
        print(elem, end=' ')
    print()
```

```
1 2 3 4
5 6
7 8 9
```

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    for elem in row:
        s += elem  # Расчет суммы элементов массива
print(s)
```

```
45
```

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j] # Расчет суммы элементов массива
print(s)
```

```
45
```

```
n = 3
m = 4
a = []
for i in range(n):
    a.append([777] * m)
print(a)
```

```
[[777, 777, 777, 777], [777, 777, 777, 777], [777, 777, 777, 777]]
```

```
# в первой строке ввода идёт количество строк массива
n = int(input())
a = []
for i in range(n):
    row = input().split()
    for j in range(len(row)):
        row[j] = int(row[j])
    a.append(row)
print (a)
```

```
4
1 2 33 17 12 41
1 22 8
3 3
1 2 3
[[1, 2, 33, 17, 12, 41], [1, 22, 8], [3, 3], [1, 2, 3]]
```

4.4.1 Простой практический пример с массивом чисел

```
'''
ПРИМЕР 1 ДЛЯ ОДНОМЕРНОГО МАССИВА
'''

# Поменяем места минимальный и максимальный элементы в массиве (списке)
a = [12, 2, 44, 3, 4, 58, 6, 22, -55]  # Исходный список
print(a)

a_max = max(a)  # Найдем максимальное значение в массиве
a_min = min(a)  # Найдем минимальное значение в массиве
a_max_pos = a.index(a_max)  # Найдем позицию максимального значения в массиве
a_min_pos = a.index(a_min)  # Найдем позицию минимального значения в массиве
```

```
# Меняем значения максимума и минимума в массиве местами
a[a_max_pos] = a_min  # Поместим в место максимального элемента минимальное значение
a[a_min_pos] = a_max  # Поместим в место минимального элемента максимальное значение
print(a)

print()



'''
ПРИМЕР 2 ДЛЯ ДВУМЕРНОГО МАССИВА
'''

b_mass = [[12, 2, 334],
          [3, 33, 58],
          [-77, 22, -55]]  # Исходный список

b_max = b_min = b_mass[0][0]  # Задаем начальное значение максимума и минимума равным ячейке [0][0]
b_max_row = b_min_row = b_max_col = b_min_col = 0  # Задаем координаты максимума и минимума равными [0][0]

for row, r in enumerate(b_mass):  # Перебираем строки
    for col, cell in enumerate(r):  # Перебираем столбцы
        print(cell, end=' ')
        if (cell > b_max):  # Если текущая ячейка больше максимума, то изменяем данные о максумуме
            b_max = cell
            b_max_col = col
            b_max_row = row
        if (cell < b_min):  # Если текущая ячейка меньше минимума, то изменяем данные о минимуме
            b_min = cell
            b_min_col = col
            b_min_row = row
    print()

# Меняем значекния максимума и минимума в массиве местами
b_mass[b_max_row][b_max_col] = b_min  # Поместим в место максимального элемента минимальное значение
b_mass[b_min_row][b_min_col] = b_max  # Поместим в место минимального элемента максимальное значение

print()

for b in (b_mass):  # Вывод массива на экран
    print(b)

print()
```

```
[12, 2, 44, 3, 4, 58, 6, 22, -55]
[12, 2, 44, 3, 4, -55, 6, 22, 58]

12 2 334
3 33 58
-77 22 -55

[12, 2, -77]
[3, 33, 58]
[334, 22, -55]
```

4.4.2 Практический пример с массивом чисел

```
# Имеется двумерный массив 4x5. Реализовать возможность заполнения его
# случайными числами. Реализовать команду выполнить задание, которая выполняет:
# Если во втором столбце стоят две единицы, то уменьшить макс. элемент первой
# строки в два раза, а все единицы в таблице заменить нулями.

import random  # импортируем модуль random для генерации случайных чисел

# Функция генерирует nxm массив случайных чисел до max_value, у которого
# стандартное значение 20
def random_array(n, m, max_value=20):
    array = []  # инициализируем массив
    # Цикл for. Оператор range выдает диапазон чисел, в данном случае
    # от 0 до n-1
    for i in range(0, n):
        sub_array = []  # инициализируем подмассив
        # Если передать range один аргумент, то нижняя граница 0, в данном
        # случае диапазон чисел будет от 0 до m-1
        for j in range(m):
            # Генерируем слуйчаное число от 0 до 19 и добавляем его в подмассив
            number = random.randint(0, max_value)
            sub_array.append(number)
        # Добавляем полученный подмассив в основной массив
        array.append(sub_array)
    return array  # возвращаем массив из случайных чисел

def print_array(array):  # функция выводит массив в удобочитаемой форме
    print()  # переход на новую строку
    # Циклу for также можно давать массивы, тогда перебирается каждый элемент
    for i in array:
        # Так как массив состоит из подмассивов, тогда каждый элемент тоже
        # можно перебрать используя цикл for
        for j in i:
            print("%d\t" % j, end='')  # выводим каждое значение и табуляцию
        print()  # переход на новую строку

def main():
    array = random_array(4, 5)  # заполняем массив случайными числами
    print_array(array)  # выводим массив на экран
    # Бесконечный цикл while, который закончится только при помощи break
    while True:
        print  # переход на новую строку
        print("1. Заполнить массив случайными числами;")
        print("2. Выполнить задание;")
        print("3. Выход.")
        # Получаем ввод команды от пользователя
        key = input('Введите команду (1, 2 или 3): ')
        if key == '1':  # если команда 1, то заполняем массив заново
            array = random_array(4, 5)
            print_array(array)
            # После этого условия цикл начнется с начала
        elif key == '2':  # если команда 2, то проверяем условие
            print()  # переход на новую строку
            one_count = 0  # переменая для подсчета единиц во втором столбце
            for i in array:  # перебираем каждую строку
                if i[1] == 1:  # если второй элемент строки равен 1
                    one_count += 1  # ведем подсчет единиц
            if one_count != 2:  # если число единиц не равно 2
                print("Число единиц во втором столбце - %d." % one_count)
                print("Задание не будет выполнено.")
                # Далее цикл начнется сначала
            else:  # число единиц равно двум
                print("Количество единиц во втором столбце равно двум.")
```

```
                max_value = -1  # переменная для хранения макс. элемента
                for i in array[0]:  # перебираем значения первой строки
                    if i > max_value:  # если элемент больше текущего макс.
                        max_value = i  # приравниваем макс. значению элемента
                # Уменьшаем значение макс. элемента первой строки в два раза.
                # Оператор index() возвращает позицию указанного элемента, если
                # он присутствует, а если нет, то поднимает исключение (ошибку)
                array[0][array[0].index(max_value)] = max_value / 2
                # Заменяем все единицы на нули
                for i in range(4):  # перебираем каждую строку
                    # Цикл нужен, так как возможно 1 не единственная в строке
                    while True:
                        # оператор try обрабатывает исключения (ошибки)
                        try:
                            index = array[i].index(1)  # находим положение 1
                            array[i][index] = 0  # изменяем на 0
                        except ValueError:  # если в строке нет единицы
                            break  # выход из цикла
                # Выводим на экран результаты
                print("Макс. элемент первой строки был уменьшен в два раза;")
                print("Все единицы были заменены на нули.")
                print_array(array)
                break  # выход из цикла
        elif key == '3':
            exit(0)  # выходим из программы
if __name__ == '__main__':
    main()
```

4.4.3 Практический пример с массивом (списком) слов

```
# Дан текст, удалить из него все слова, содержащие нечетное количество символов

def main():
    # Получаем ввод текста от пользователя
    print("Введите текст:")
    text = input()
    # Инициализируем переменные
    word = ""  # здесь будем хранить текущее слово
    words = []  # в данном массиве будем сохранять слова
    text = text.strip()  # Убираем пробелы с обоих концов текста
    # Перебираем каждую букву введенного текста, для отбора отдельных слов
    for c in text:
        if c != ' ':
            # Добавляем к переменной word текущую букву, если она не равна
            # пробелу. После чего сразу переходим к следующей итерации цикла
            word += c
            continue
        # Если ход программы дошел до сюда, это значит, что текущая буква
        # пробел. Добавляем word в words и обнуляем временную переменную
        words.append(word)
        word = ""
    # По выходу из цикла в переменной хранится последнее слово
    words.append(word)
    # Готовим переменную для вывода на экран
    result = ""
    for w in words:
        # Добавляем слово к результату, если длина слова кратна двум,
        # тем самым удаляя все не кратные двум
        if len(w) % 2 == 0:
            result = result + w + " "
    print()
    print(result)  # вывод результата на экран
```

```
if __name__ == '__main__':
    main()
```

Язык Python включает много уже определенных, т. е. встроенных в него, функций

Язык Python включает т. е. встроенных