«М.А. Гендельманның 110 жылдығына арналған «Сейфуллин оқулары – 19» халықаралық ғылыми-практикалық конференциясының материалдары = Материалы международной научно-практической конференции «Сейфуллинские чтения – 19», посвященной 110 - летию М.А. Гендельмана» - 2023.- Т. II, Ч.I.- С. 329-332.

УДК 519.7

МЕТОДЫ СОРТИРОВКИ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

Курочкина А.П., Оспанова А.С., студенты 2 курса
НАО «Казахский агротехнический исследовательский университет им. С.Сейфуллина, г. Астана

Для анализа данных в настоящее время широко пользуются возможности языков высокого уровня программирования, к ним относятся C, C++, Perl, Python и т.д.

Анализ данных на языке программирования Python можно осуществлять на основе интерпретатора самого языка и на основе математических и научных библиотек как NumPy или Pandas.

В нашей научной статье мы рассматриваем анализ данных, как методы сортировки данных. При этом практический рассмотрим использование методов сортировки в списках и в структурах данных (множества, списки, словари, кортежи), на языке программирования Python, на основе интерпретатора самого языка Python.

Методы сортировки на основе функции sorted()

Функция sorted() сортирует элементы данного итерируемого объекта в определенном порядке (по возрастанию или по убыванию) и возвращает его в виде списка [1].

На рисунке 1 приводим практические примеры использования метода сортировки на основе функции sorted() в списках и структурных данных Pythonкак список и кортеж.

```
# список
агрономия_list = ['ягоды', 'почва', 'деревья',
'плоды', 'цветы']
print(sorted(агрономия_list))

# строка
направление_string = 'агрономия'
print(sorted(направление_string))

# кортеж
буквы_tuple = ('б', 'д', 'а', 'в', 'г')
print(sorted(буквы_tuple))
```

Рисунок 1 – Примеры использования метода sorted()

Метод - сортировка по убыванию sorted()

Функция sorted() принимает "reverse = True" параметр в качестве необязательного аргумента [2].

Параметр "reverse = True" сортирует итерируемый объект в порядке убывания в структурных данных, как показано на рисунке 2, во множестве, словаре, кортеже и списке.

```
# Множества
py_set = {'яблоки', 'сливы', 'капуста',
 'груши', 'арбуз'}
print(sorted(py_set, reverse=True))

# Словарь
py_dict = {'e': 1, 'a': 2, 'u': 3, 'o': 4, 'i': 5}
print(sorted(py_dict, reverse=True))

# Кортеж
frozen_set = frozenset(('e', 'a', 'u', 'o', 'i'))
print(sorted(frozen_set, reverse=True))
```

Рисунок 2 - Параметр «reverse = True» при сортировке структур данных

Метод сортировки с помощью одного ключа

В сортировке с использованием функции list.sort() и sorted() можно использовать параметр key для указания функции, которая будет вызываться на каждом элементе до сравнения[1].

Значение key должно быть неопределенной заданной нами функцией, принимающей один аргумент и возвращающей ключ для сортировки.

При таком подходе метод сортировки производится быстрее, так как функция-ключ вызывается один раз для каждого элемента.

Рассмотрим пример программного кода метода сортировки с помощью одного ключа:

```
# задаем ключ
def ключ(elem):
    return elem[0]
# random list
a = [(2, 2), (3, 4), (4, 1), (1, 3)]
# сортировка по ключу
сортировка_list = sorted(a, key=ключ)
# вывод сортировки списка
print('сортировка:', сортировка_list)
```

В практическом примере заданная нами неопределенная функция "ключ" является параметром key.

Метод сортировки несколькими ключами

Даны метод расмотрим на примере программнного кода, где переменой "студенты_list" присвоен список с данными 5 студентов. Данные в базе данных помещаем во вложенном списке в кортежи.

Список мы должны отсортировать так чтобы студент с самыми высокими оценками был в начале. В случае, если у студентов одинаковые оценки, их необходимо отсортировать так, чтобы младший участник оказался первым. Для такого типа сортировки используем несколько ключей, возвращая кортеж вместо числа.

Два кортежа в такой структуре данных можно сравнить по элементно. Сравнение начинается с первого элемента. Если есть элементы равные т.е. одинакового значения, то сравнивается второй элемент и так далее.

Код программы по сортировке списка студентов по полученным баллам, когда производим сортировку данных методом использования нескольких ключей будет выглядеть так:

```
# Данные: (Имена студентов, баллы до 100 , возраст)
студенты_list = [
    ('Дима', 75, 19),
    ('Арман', 90, 19),
    ('Жанар', 85, 20),
    ('Светлана', 90, 21),
    ('Бауржан', 95, 17)
]
```

```
def sorter(item):
    # первым сравиваються балы, затем возраст)
    баллы = 100 - item[1]
    возраст= item[2]
    return (баллы, возраст)
Сортировка_list = sorted(студенты_list, key=sorter)
print(Сортировка_list)
```

Метод сортировки на основе функции list.sort() по первому столбцу.

Метод list.sort() - изменяет исходный список.Если элементы списка сами представляют собой списки, т. е. являются вложенными списками, то сортировка будет происходить по первым элементам вложенных списков, то есть в случае матрицы по первому столбцу [2].

На этот случай sort() принимает необязательный аргумент key, в котором передается другая функция. Этой другой функции передается очередной элемент списка. Она может отсортировать, что угодно и вернуть. Поэтому происходит произвольная сортировка.

Например, пользовательская функция может возвращать из переданного ей элемента, представляющего собой вложенный список, любой элемент этого вложенного списка. В свою очередь функция sort() будет сортировать по тем значениям, которые ей возвращаются**.**

Приведем программу, в которой список представляет собой не большую базу данных:

```
a = [['Арман',20,170,63],  ['Айбек',18,168,65],
['Дима',21,185,90], ['Бауржан',22,167,64]]
n = input('Сортировать по имени (1),возрасту (2), росту (3), весу (4): ')
n = int(n)-1
t = input('По возрастанию (0), по убыванию (1): ')
t = int(t)
a.sort(key=lambda i: i[n], reverse=t)
for i in a:
    print("%7s %3d %4d %3d"
    % (i[0],i[1],i[2],i[3]))
```

Допустим, каждый элемент содержит сведения о спортсмене: имя, возраст, рост и вес. На основе данного программного кода, пользователь может заказать сортировку по любому полю по возрасту, росту и весу в вложенных списках или массивах.

Функции list.sort() и sorted() имеют различие при использовании в методах сортировки. Как видно в наших примерах функция sorted() используется при проведении сортировки во всех структурных данных. Функция list.sort() используется только при сортировке списков или массивов данных.

В научных исследованиях при описании и обработке статистических данных экономике, математике, агрономии и т.д. особая роль отводиться к анализу данных на языках программирования, где использования языка программирования Pythonдает хорошие возможности сортировки, фильтрации и поиска больших данных благодаря используемым в нем методам, алгоритмам и библиотекам [3].

Таким образом, мы рассмотрели виды сортировки данных в структурах данных как списки, кортежи и словари, которые можно использовать при создании баз данных на языке программирования Python. Нами предлагаются практические методы сортировки на основе интерпретатора самого языка Python, что удобно использовать без использования возможностей специальных библиотек как NumPy или Pandas.

Список используемой литературы

1 Васильев А.Б. Python на примерах. [Текст]:Практический курс по программированию. / Васильев А.Б. -СПб: Наука и техник, 2019. - 432 с.
2 Stewart, J. M.PythonforScientists. Second ed. [Text] / J. M. Stewart. – Cambridge University: Press, -2017. - 272 p.
3 Nurpeisova A.A., Smailova L.K., Akimova B.Z., BorisovaE.V., Niyazbekova S.U. Condition and Prospects of Innovative Development of the Economy in Kazakhstan. //Socio-economic Systems: Paradigms for the Future. Springer International Publishing. 2021, 314p. 1773–1779