Лабораторная работа 2.5 Работа с кортежами в языке Python

Цель работы: *приобретение навыков по работе с кортежами при написании программ с помощью языка программирования Python версии 3.x.*

Ход работы

Что такое кортеж (tuple) в Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список. Как вы наверное знаете, список – это изменяемый тип данных. Т. е. если у нас есть список *a = [1, 2, 3]* и мы хотим заменить второй элемент с 2 на 15, то мы может это сделать, напрямую обратившись к элементу списка.

```
>>> a = [1, 2, 3]
>>> print(a)
[1, 2, 3]
>>> a[1] = 15
>>> print(a)
[1, 15, 3]
```

С кортежем мы не можем производить такие операции, т. к. элементы его изменять нельзя.

```
>>> b = (1, 2, 3)
>>> print(b)
(1, 2, 3)
>>> b[1] = 15
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    b[1] = 15
TypeError: 'tuple' object does not support item assignment
```

Зачем нужны кортежи в Python?

Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них – это обезопасить данные от случайного изменения. Если мы получили откуда-то массив данных, и у нас есть желание поработать с ним, но при этом непосредственно менять данные мы не собираемся, тогда, это как раз тот случай, когда кортежи придутся как нельзя кстати. Используя их в данной задаче, мы дополнительно получаем сразу несколько бонусов – во-первых, это экономия места. Дело в том, что кортежи в памяти занимают меньший объем по сравнению со списками.

```
>>> lst = [10, 20, 30]
>>> tpl = (10, 20, 30)
>>> print(lst.__sizeof__())
32
>>> print(tpl.__sizeof__())
24
```

Во-вторых – прирост производительности, который связан с тем, что кортежи работают быстрее, чем списки (т. е. на операции перебора элементов и т. п. будет тратиться меньше времени). Важно также отметить, что кортежи можно использовать в качестве ключа у словаря.

Создание, удаление кортежей и работа с его элементами

Создание кортежей

Для создания пустого кортежа можно воспользоваться одной из следующих команд.

```
>>> a = ()
>>> print(type(a))
<class 'tuple'>
>>> b = tuple()
>>> print(type(b))
<class 'tuple'>
```

Кортеж с заданным содержанием создается также как список, только вместо квадратных скобок используются круглые.

```
>>> a = (1, 2, 3, 4, 5)
>>> print(type(a))
<class 'tuple'>
>>> print(a)
(1, 2, 3, 4, 5)
```

При желании можно воспользоваться функцией *tuple()*.

```
>>> a = tuple([1, 2, 3, 4])
>>> print(a)
(1, 2, 3, 4)
```

Определять кортежи очень просто, сложности могут возникнуть только с кортежами, содержащими ровно один элемент. Если мы просто укажем значение в скобках, то Python подумает, что мы хотим посчитать арифметическое выражение со скобками:

```
not_a_tuple = (42)  # 42
```

Чтобы сказать Python, что мы хотим создать именно кортеж, нужно поставить после элемента кортежа запятую:

```
tuple = (42,)  # (42,)
```

Доступ к элементам кортежа

Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса. Но, как уже было сказано – изменять элементы кортежа нельзя!

```
>>> a = (1, 2, 3, 4, 5)
>>> print(a[0])
1
>>> print(a[1:3])
(2, 3)
>>> a[1] = 3
Traceback (most recent call last):
  File "<pyshell#24>", line 1, in <module>
    a[1] = 3
TypeError: 'tuple' object does not support item assignment
```

Удаление кортежей

Удалить отдельные элементы из кортежа невозможно.

```
>>> a = (1, 2, 3, 4, 5)
>>> del a[0]
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    del a[0]
TypeError: 'tuple' object doesn't support item deletion
```

Но можно удалить кортеж целиком.

```
>>> del a
>>> print(a)
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    print(a)
NameError: name 'a' is not defined
```

Преобразование кортежа в список и обратно

На базе кортежа можно создать список, верно и обратное утверждение. Для превращения списка в кортеж достаточно передать его в качестве аргумента функции tuple().

```
>>> lst = [1, 2, 3, 4, 5]
>>> print(type(lst))
<class 'list'>
>>> print(lst)
[1, 2, 3, 4, 5]
>>> tpl = tuple(lst)
>>> print(type(tpl))
<class 'tuple'>
>>> print(tpl)
(1, 2, 3, 4, 5)
```

Обратная операция также является корректной.

```
>>> tpl = (2, 4, 6, 8, 10)
>>> print(type(tpl))
<class 'tuple'>
>>> print(tpl)
(2, 4, 6, 8, 10)
>>> lst = list(tpl)
>>> print(type(lst))
<class 'list'>
>>> print(lst)
[2, 4, 6, 8, 10]
```

Деструктуризация

Обращение по индексу, это не самый удобный способ работы с кортежами. Дело в том, что кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит — очень непросто. Но есть способ лучше! Как мы кортеж собираем, так его можно и разобрать:

```
name_and_age = ('Bob', 42)

(name, age) = name_and_age
name  # 'Bob'
age   # 42
```

Именно таким способом принято получать и сразу разбирать значения, которые возвращает функция (если таковая возвращает несколько значений, конечно):

```
(quotient, modulo) = div_mod(13, 4)
```

Соответственно кортеж из одного элемента нужно разбирать так:

```
(a,) = (42,)
a  # 42
```

Если же после имени переменной не поставить запятую, то синтаксической ошибки не будет, но в переменную `a` кортеж запишется целиком, т. е. ничего не распакуется. **Всегда помните о запятых!**

Кортежи, множественное присваивание и обмен значениями

Благодаря тому, что кортежи легко собирать и разбирать, в Python удобно делать такие вещи, как множественное присваивание. Смотрите:

```
(a, b, c) = (1, 2, 3)
a  # 1
b  # 2
c  # 3
```

Используя множественное присваивание, можно провернуть интересный трюк: обмен значениями между двумя переменными. Вот код:

```
a = 100
b = 'foo'

(a, b) = (b, a)
a  # 'foo'
b  # 100
```

Строку `(a, b) = (b, a)` нужно понимать как "присвоить в `a` и `b` значения из кортежа, состоящего из значений переменных `b` и `a`".

Операции над кортежами

Создание кортежа из итерированного объекта

Кортеж можно создать с помощью операции tuple(). Эта операция принимает параметром итерированный объект, которым может быть другой кортеж, список, строка.

Например

```
# Операция tuple()
# 1. Создание кортежа из слова 'Hello'
d = tuple('Hello'); # d = ('H', 'e', 'l', 'l', 'o')

# 2. Создание кортежа из списка
# Заданный список
lst = [2, "abc", 3.88]

# Создать кортеж
e = tuple(lst) # e = (2, 'abc', 3.88)

# 3. Создание кортежа из другого кортежа
f = tuple((3, 2, 0, -5)) # f = (3, 2, 0, -5)
```

Операция *T[i:j]*. Взятие среза в кортеже

С помощью операции взятия среза можно получить другой кортеж. Общая форма операции взятия среза для кортежа следующая

```
T2 = T1[i:j]
```

здесь

- *T2* – новый кортеж, который получается из кортежа *T1*;
- *T1* – исходный кортеж, для которого происходит срез;
- *i*, *j* – соответственно нижняя и верхняя границы среза. Фактически берутся ко вниманию элементы, лежащие на позициях *i*, *i*+1, ..., *j*-1. Значение *j* определяет позицию за последним элементом среза.

Операция взятия среза для кортежа может иметь модификации такие же как и для списков.

Например

```
# Операция [i:j] - взятие среза
# 1. Кортеж, содержащий целые числа
A = ( 0, 1, 2, 3)
item = A[0:2]   # item =   (0, 1)
```

```
# 2. Кортеж, содержащий список
A = ( 2.5, ['abcd', True, 3.1415], 8, False, 'z')
item = A[1:3] # item = (['abcd', True, 3.1415], 8)

# 3. Кортеж, содержащий вложенный кортеж
A = (3, 8, -11, "program")
B = ("Python", A, True)
item = B[:3] # item = ('Python', (3, 8, -11, 'program'), True)
item = B[1:] # item = ((3, 8, -11, 'program'), True)
```

Конкатенация `+`

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом +. В простейшем случае для конкатенации двух кортежей общая форма операции следующая

```
T3 = T1 + T2
```

где

- *T1*, *T2* – кортежи, для которых нужно выполнить операцию конкатенации. Операнды *T1*, *T2* обязательно должны быть кортежами. При выполнении операции конкатенации для кортежей, использовать в качестве операндов любые другие типы (строки, списки) запрещено;
- *T3* – кортеж, который есть результатом.

Например

```
# Кортежи. Конкатенация +
# Конкатенация двух кортежей
A = (1, 2, 3)
B = (4, 5, 6)
C = A + B # C =   (1, 2, 3, 4, 5, 6)

# Конкатенация кортежей со сложными объектами
D = (3, "abc") + (-7.22, ['a', 5]) # D = (3, 'abc', -7.22, ['a', 5])

# Конкатенация трех кортежей
A = ('a', 'aa', 'aaa')
B = A + (1, 2) + (True, False) # B = ('a', 'aa', 'aaa', 1, 2, True, False)
```

Повторение `*`

Кортеж может быть образован путем операции повторения, обозначаемой символом *****. При использовании в выражении общая форма операции следующая

```
T2 = T1 * n
```

здесь

- *T2* – результирующий кортеж;
- *T1* – исходный кортеж, который нужно повторить *n* раз;
- *n* – количество повторений кортежа *T1*.

Пример.

```
# Кортежи. Повторение *
# Кортеж, который содержит простые числа
A = (1, 2, 3) * 3 # A = (1, 2, 3, 1, 2, 3, 1, 2, 3)

# Кортеж, который содержит вложенные объекты
B = ("ab", ["1", "12"])*2 # A=('ab', ['1','12'], 'ab', ['1','12'])
```

Обход кортежа в цикле

Элементы кортежа можно последовательно просмотреть с помощью операторов цикла *while* или *for*.

Например

```
# Обход кортежа в цикле
# 1. Цикл for
# Заданный кортеж
A = ("abc", "abcd", "bcd", "cde")

# Вывести все элементы кортежа
for item in A:
    print(item)

# 2. Цикл while
# Исходный кортеж - целые числа
A = (-1, 3, -8, 12, -20)

# Вычислить количество положительных чисел
i = 0
k = 0 # количество положительных чисел

while i < len(A):
    if (A[i]<0):
        k = k + 1
    i = i + 1

# Вывести результат
print("k = ", k)

# 3. Обход в цикле for
# Заданный кортеж, содержащий строки
A = ("abc", "ad", "bcd")

# Сформировать новый список из элементов кортежа A,
# в новом списке B, каждый элемент удваивается
B = [item * 2 for item in A]

print("A = ", A)
print("B = ", B)
```

Результат выполнения программы

```
abc
abcd
bcd
cde
k = 3
A = ('abc', 'ad', 'bcd')
B = ['abcabc', 'adad', 'bcdbcd']
```

Операция `in`. Проверка вхождения элемента в кортеж

```
# Проверка вхождения элемента в кортеж
# Оператор in
# Заданный кортеж, который содержит строки
A = ("abc", "abcd", "bcd", "cde")

# Ввести элемент
item = str(input("s = "))

if (item in A):
    print(item, " in ", A, " = True")
else:
    print(item, " in ", A, " = False")
```

Результат выполнения программы

```
s = abc
abc in   ('abc', 'abcd', 'bcd', 'cde') = True
```

Методы работы с кортежами

Метод *index()*. Поиск позиции элемента в кортеже

Чтобы получить индекс (позицию) элемента в кортеже, нужно использовать метод index(). Общая форма вызова метода следующая

```
pos = T.index(item)
```

где

- *T* – кортеж, в котором осуществляется поиск;
- *pos* – позиция (индекс) элемента *item* в кортеже. Первому элементу соответствует позиция 0. Если элемента нет в кортеже, генерируется исключительная ситуация. Поэтому, перед использованием метода index() рекомендуется делать проверку на наличие элемента (с помощью операции in).

Пример. В примере, в перечне названий дней недели вычисляется порядковый номер дня.

```
# Метод index - определяет позицию (индекс) элемента в кортеже
# Заданный кортеж
A = ("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat")

# Запрос к вводу названия дня недели
day = str(input("Enter day: "))
```

```
# Корректно вычислить индекс
if day in A: # проверка, есть ли строка day в кортеже A
    num = A.index(day)
    print("Number of day = ", num + 1)
else:
    num = -1
    print("Wrong day.")
```

Результат работы программы

```
d = 1
```

Метод *count()*. Количество вхождений элемента в кортеж

Чтобы определить количество вхождений заданного элемента в кортеж используется метод count, общая форма которого следующая:

```
k = T.count(item)
```

здесь

- *T* – исходный кортеж;
- *k* – результат (количество элементов);
- *item* – элемент, количество вхождений которого нужно определить. Элемент может быть составным (строка, список, кортеж).

```
# Метод count - подсчет количества вхождений элемента в кортеж
# Заданный кортеж
A = ("ab", "ac", "ab", "ab", "ca", "ad", "jklmn")

d1 = A.count("ab")   # d1 = 3
d2 = A.count("jprst") # d2 = 0
d3 = A.count("ca")   # d3 = 1

print("d1 = ", d1)
print("d2 = ", d2)
print("d3 = ", d3)
```

Результат работы программы

```
d1 = 3
d2 = 0
d3 = 1
```

Пример 1. *Ввести кортеж A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран. Использовать в программе вместо списков кортежи.*

Решение: Напишем программу для решения поставленной задачи.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
```

```
if __name__ == '__main__':
    # Ввести кортеж одной строкой.
    A = tuple(map(int, input().split()))
    # Проверить количество элементов кортежа.
    if len(A) != 10:
        print("Неверный размер кортежа", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = 0
    for item in A:
        if abs(item) < 5:
            s += item

    print(s)
```

Выражение `tuple(map(int, input().split()))` позволяет ввести целочисленный массив одной строкой. Это работает, поскольку изначально строка, полученная с помощью функции `input()` разбивается на список подстрок с помощью метода `split()`. Затем к каждой из подстрок списка функция `map` применяет функцию `int` (точнее вызов `int` создает целое число на основании значения своего аргумента). Поскольку функция `map` возвращает генератор, то с помощью вызова `tuple` генератор преобразуется к кортежу.

Данная задача может быть также решена с помощью списковых включений следующим образом:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys


if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = sum(a for a in A if abs(a) < 5)
    print(s)
```

В этом примере показано использование списковых включений для расчета суммы, однако в отличие от выражения `[a for a in A ...]`, которое на выходе дает нам список, выражение `(a for a in A ...)` дает на выходе специальный объект **генератора**, а не кортеж. Для преобразования генератора в кортеж необходимо воспользоваться вызовом `tuple()`.

Аппаратура и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 10 с подключением к глобальной сети Интернет.

2. Операционная система Windows 10.
3. Система контроля версий Git.
4. Браузер для доступа к web-сервису GitHub, рекомендован к использованию Google Chrome.
5. Дистрибутив языка программирования Python, включающий набор популярных библиотек Anaconda.
6. Интегрированная среда разработки PyCharm Community Edition.

Указания по технике безопасности

При работе на ЭВМ без разрешения руководителя занятия запрещается:

- подавать (снимать) напряжение на ПЭВМ и электрические розетки с распределительного щита;
- включать и выключать блоки питания ПЭВМ и мониторы;
- извлекать ПЭВМ из защитного кожуха;
- устранять неисправности, возникшие в ходе выполнения лабораторной работы.

Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл `.gitignore` необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.
6. Создайте проект PyCharm в папке репозитория.
7. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.
8. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.
9. Приведите в отчете скриншоты работы программ решения индивидуального задания.
10. Зафиксируйте сделанные изменения в репозитории.
11. Добавьте отчет по лабораторной работе в *формате PDF* в папку *doc* репозитория. Зафиксируйте изменения.
12. Выполните слияние ветки для разработки с веткой *main* / *master*.
13. Отправьте сделанные изменения на сервер GitHub.
14. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Индивидуальные задания

1. Известно количество очков, набранных каждой из 20 команд – участниц первенства по футболу. Перечень очков дан в порядке убывания (ни одна пара команд не набрала одинаковое количество очков). Определить, какое место заняла команда, набравшая n очков (естественно, что значение n имеется в перечне). Условный оператор не использовать.
2. В начале кортежа записано несколько равных между собой элементов. Определить количество таких элементов и вывести все элементы, следующие за последним из них. Условный оператор не использовать.
3. Известны оценки по геометрии каждого из 24 учеников класса. В начале списка перечислены все пятерки, затем все остальные оценки. Сколько учеников имеет по геометрии оценку «5»? Условный оператор не использовать.

4. Определить, есть ли в кортеже хотя бы одна пара одинаковых соседних элементов. В случае положительного ответа определить номера элементов первой из таких пар.
5. Если в кортеже есть хотя бы одна пара одинаковых соседних элементов, то напечатать все элементы, следующие за элементами первой из таких пар.
6. Дан кортеж целых чисел. Определить, есть ли в нем хотя бы одна пара соседних нечетных чисел. В случае положительного ответа определить номера элементов первой из таких пар.
7. Дан кортеж целых чисел. Если в нем есть хотя бы одна пара соседних четных чисел, то напечатать все элементы, предшествующие элементам последней из таких пар.
8. Определить, есть ли в кортеже хотя бы одна тройка соседних чисел, в которой средний элемент больше своих «соседей», т. е. предшествующего и последующего. В случае положительного ответа определить номера элементов первой из таких троек.
9. Если в кортеже есть хотя бы одна тройка соседних чисел, в которой средний элемент больше своих «соседей», т. е. предшествующего и последующего, то напечатать все элементы, предшествующие элементам последней из таких троек.
10. Определить, является ли кортеж упорядоченным по возрастанию. В случае отрицательного ответа определить номер первого элемента, нарушающего такую упорядоченность.
11. Имеются данные о сумме очков, набранных в чемпионате каждой из футбольных команд. Определить, перечислены ли команды в списке в соответствии с занятыми ими местами в чемпионате.
12. В начале кортежа записано несколько равных между собой элементов. Определить количество таких элементов и вывести все элементы, следующие за последним из них. Рассмотреть возможность того, что весь массив заполнен одинаковыми элементами. Условный оператор не использовать.
13. Известны оценки по информатике каждого ученика класса. В начале кортежа перечислены все пятерки, затем все остальные оценки. Сколько учеников имеет по информатике оценку «5»? Рассмотреть возможность случая, что такую оценку имеют все ученики. Условный оператор не использовать.
14. Фирме принадлежат два магазина. Известна стоимость товаров, проданных в каждом магазине за каждый день в июле и августе, которая хранится в двух массивах. Получить общую стоимость проданных фирмой товаров за два месяца.
15. Известно количество мячей, забитых футбольной командой за каждую игру в двух чемпионатах, которое хранится в двух кортежах. В каждом из чемпионатов команда сыграла 26 игр. Найти общее количество мячей, забитых командой в двух чемпионатах.
16. Известны данные о мощности двигателя (в лошадиных силах – л. с.) и стоимости 30 марок легковых автомобилей. Напечатать стоимость каждого из автомобилей, у которых мощность двигателя не превышает 80 л. с.
17. Известны данные о вместимости (в гигабайтах) и стоимости (в рублях) каждого из 22 типов жестких магнитных дисков (винчестеров). Напечатать вместимость тех винчестеров, которые стоят больше s рублей.
18. Имеется информация о количестве осадков, выпавших за каждый день месяца, и о температуре воздуха в эти дни. Определить, какое количество осадков выпало в виде снега и какое – в виде дождя. (Считать, что идет дождь, если температура воздуха выше 0 °C.)
19. Известны данные о численности населения (в миллионах жителей) и площади (в тысячах квадратных километров) 28 государств. Определить общую численность населения в «маленьких» государствах (чья площадь не превышает A тысяч квадратных километров).
20. Имеется информация о количестве осадков, выпавших за каждый день января и за каждый день марта. Определить, в каком из этих месяцев выпало больше осадков.
21. На плоскости даны 20 точек: $(x_1, y_1), (x_2, y_2), \ldots, (x_{20}, y_{20})$. Рассмотрим прямоугольники, содержащие эти точки, причем стороны прямоугольников параллельны или перпендикулярны координатным осям. Возьмем наименьший из них. Определить координаты противоположных углов такого прямоугольника – левого нижнего и правого верхнего.

22. Из элементов кортежа a сформировать кортеж b того же размера по правилу: если номер i элемента кортежа a четный, то $b_i = a_i^2$, в противном случае $b_i = 2 \cdot a_i$.
23. Из элементов кортежа m сформировать кортеж n того же размера по правилу: если номер i элемента кортежа m нечетный, то $n_i = i \times m_i$, в противном случае $n_i = m_i / i$.
24. Из элементов кортежа p сформировать кортеж q того же размера по правилу: элементы с номером i от 3-го по 10-й находятся по формуле $q_i = -p_i$, все остальные – по формуле $q_i = p_i \times i$.
25. Из элементов кортежа a, заполненного целыми числами, сформировать кортеж b того же размера по правилу: четные элементы кортежа a удвоить, нечетные оставить без изменения.
26. Даны два кортежа одного размера, в которых нет нулевых элементов. Получить третий кортеж, каждый элемент которого равен 1, если элементы заданных кортежей с тем же номером имеют одинаковый знак, и равен нулю в противном случае.

Содержание отчета и его форма

Отчет по лабораторной работе оформляется электронно в формате PDF, должен содержать ответы на контрольные вопросы, ссылку на репозиторий с которым выполнялась работа, скриншоты IDE PyCharm, скриншоты результатов работы программ.

Вопросы для защиты работы

1. Что такое списки в языке Python?
2. Каково назначение кортежей в языке Python?
3. Как осуществляется создание кортежей?
4. Как осуществляется доступ к элементам кортежа?
5. Зачем нужна распаковка (деструктуризация) кортежа?
6. Какую роль играют кортежи в множественном присваивании?
7. Как выбрать элементы кортежа с помощью среза?
8. Как выполняется конкатенация и повторение кортежей?
9. Как выполняется обход элементов кортежа?
10. Как проверить принадлежность элемента кортежу?
11. Какие методы работы с кортежами Вам известны?
12. Допустимо ли использование функций агрегации таких как `len()`, `sum()` и т. д. при работе с кортежами?
13. Как создать кортеж с помощью спискового включения.