МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №8 по дисциплине основы программной инженерии

Выполнил:
Емельянова Яна
Александровна, 2 курс,
группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г

1. Работа с кортежами в языке Python

Примеры из методических указаний

Что такое кортеж (tuple) в Python?

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

if __name__ == '__main__':
    a = [1, 2, 3]
    print(a)
    a[1] = 15
    print(a)
    b = (1, 2, 3)
    print(b)
    b[1] = 15
```

```
[1, 2, 3]
[1, 15, 3]
(1, 2, 3)
Traceback (most recent call last):
  File "C:\Users\Evil\PycharmProjects\LB8\examples\e1.py", line 11, in <module>
    b[1] = 15
TypeError: 'tuple' object does not support item assignment

Process finished with exit code 1
```

Зачем нужны кортежи в Python?

e2.py

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

if __name__ == '__main__':
    lst = [10, 20, 30]
    tpl = (10, 20, 30)
    print(lst.__sizeof__())
    print(tpl.__sizeof__())

```

```
104
48

Process finished with exit code 0
```

Создание кортежей

e3.py

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

if __name__ == '__main__':
    a = ()
    print(type(a))
    b = tuple()
    print(type(b))
    a = (1, 2, 3, 4, 5)
    print(type(a))
    print(a)
    a = tuple([1, 2, 3, 4])
    print(a)
    not_a_tuple = (42)
    tpl = (42,)
    print(type(not_a_tuple))
    print(type(tpl))

```

```
<class 'tuple'>
<class 'tuple'>
<class 'tuple'>
(1, 2, 3, 4, 5)
(1, 2, 3, 4)
<class 'int'>
<class 'tuple'>

Process finished with exit code 0
```

Доступ к элементам кортежа

e4.py

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

if __name__ == '__main__':
    a = (1, 2, 3, 4, 5)
    print(a[0])
    print(a[1:3])
    a[1] = 3

```

e5.py

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

if __name__ == '__main__':
    a = (1, 2, 3, 4, 5)
    del a[0]
    del a
    print(a)

```

Удаление кортежей

e6.py

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

if __name__ == '__main__':
    lst = [1, 2, 3, 4, 5]
    print(type(lst))
    print(lst)
    tpl = tuple(lst)
    print(type(tpl))
    print(tpl)
    # Обратная операция также является корректной.
    tpl = (2, 4, 6, 8, 10)
    print(type(tpl))
    print(tpl)
    lst = list(tpl)
    print(type(lst))
    print(lst)

```

Преобразование кортежа в список и обратно

```
<class 'list'>
[1, 2, 3, 4, 5]
<class 'tuple'>
(1, 2, 3, 4, 5)
<class 'tuple'>
(2, 4, 6, 8, 10)
<class 'list'>
[2, 4, 6, 8, 10]

Process finished with exit code 0
```

Деструктуризация

e7.py

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

if __name__ == '__main__':
    name_and_age = ('Bob', 42)
    (name, age) = name_and_age
    print(f"name = {name}")
    print(f"age = {age}")
    (a,) = (42,)
    print(f"a = {a}")

```

```
name = Bob
age = 42
a = 42

Process finished with exit code 0
```

Кортежи, множественное присваивание и обмен значениями

e8.py

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

if __name__ == '__main__':
    (a, b, c) = (1, 2, 3)
    print(f"a = {a}")
    print(f"b = {b}")
    print(f"c = {c}")
    a = 100
    b = 'foo'
    (a, b) = (b, a)
    print(f"a = {a}")
    print(f"b = {b}")

```

```
a = 1
b = 2
c = 3
a = foo
b = 100

Process finished with exit code 0
```

Создание кортежа из итерированного объекта

e9.py

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

if __name__ == '__main__':
    # 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)
    print(f"d = {d}")
    print(f"e = {e}")
    print(f"f = {f}")

```

```
d = ('H', 'e', 'l', 'l', 'o')
e = (2, 'abc', 3.88)
f = (3, 2, 0, -5)

Process finished with exit code 0
```

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

e10.py

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

if __name__ == '__main__':
    # Операция [i:j] - взятие среза
    # 1. Кортеж, содержащий целые числа
    A = (0, 1, 2, 3)
    item = A[0:2]  # item = (0, 1)
    print(f"item = {item}")
    # 2. Кортеж, содержащий список
    A = (2.5, ['abcd', True, 3.1415], 8, False, 'z')
    item = A[1:3]  # item = (['abcd', True, 3.1415], 8)
    print(f"item = {item}")
    # 3. Кортеж, содержащий вложенный кортеж
    A = (3, 8, -11, "program")
    B = ("Python", A, True)
    item = B[:3]  # item = ('Python', (3, 8, -11, 'program'), True)
    print(f"item = {item}")
    item = B[1:]  # item = ((3, 8, -11, 'program'), True)
    print(f"item = {item}")

```

```
item = (0, 1)
item = (['abcd', True, 3.1415], 8)
item = ('Python', (3, 8, -11, 'program'), True)
item = ((3, 8, -11, 'program'), True)

Process finished with exit code 0
```

Конкатенация

e11.py

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

if __name__ == '__main__':
    # Кортежи. Конкатенация +
    # Конкатенация двух кортежей
    A = (1, 2, 3)
    B = (4, 5, 6)
    C = A + B  # C = (1, 2, 3, 4, 5, 6)
    print(f"C = {C}")
    # Конкатенация кортежей со сложными объектами
    D = (3, "abc") + (-7.22, ['a', 5])  # D = (3, 'abc', -7.22, ['a', 5])
    print(f"D = {D}")
    # Конкатенация трех кортежей
    A = ('a', 'aa', 'aaa')
    B = A + (1, 2) + (True, False)  # B = ('a', 'aa', 'aaa', 1, 2, True, False)
    print(f"B = {B}")

```

```
C = (1, 2, 3, 4, 5, 6)
D = (3, 'abc', -7.22, ['a', 5])
B = ('a', 'aa', 'aaa', 1, 2, True, False)

Process finished with exit code 0
```

Повторение

e12.py

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

if __name__ == '__main__':
    # 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'])
    print(f"A = {A}")
    print(f"B = {B}")

```

```
A = (1, 2, 3, 1, 2, 3, 1, 2, 3)
B = ('ab', ['1', '12'], 'ab', ['1', '12'])

Process finished with exit code 0
```

Обход кортежа в цикле

e13.py

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

if __name__ == '__main__':
    # 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)

```

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

Process finished with exit code 0
```

Операция in. Проверка вхождения элемента в кортеж

e14.py

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

if __name__ == '__main__':
    # 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")

```

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

Process finished with exit code 0
```

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

Process finished with exit code 0
```

Метод index(). Поиск позиции элемента в кортеже

e15.py

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

if __name__ == '__main__':
    # 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.")

```

```
Enter day: Sun
Number of day =  1

Process finished with exit code 0
```

```
Enter day: hehereh
Wrong day.

Process finished with exit code 0
```

Метод count(). Количество вхождений элемента в кортеж

e16.py

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

if __name__ == '__main__':
    # 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)

```

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

Process finished with exit code 0
```

1.1 Пример 1 (рис. 1, 2).

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

import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    A = tuple(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)

```

Рисунок 1 – Код примера

```
1 2 3 4 5 6 7 8 9 19
10

Process finished with exit code 0
```

Рисунок 2 – Пример работы программы

Решение индивидуальных заданий

Вариант 9

1.2 Индивидуальное задание №1 (рис. 3, 4)

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


if __name__ == '__main__':
    a = tuple(map(int, input().split()))
    i = 0
    for index, el in enumerate(a):
        k = a.count(el)
        if k >= 2:
            if len(a) > index + 1:
                if a[index + 1] == el:
                    i = index + 1
                    break
    if i:
        i += 1
        print(a[i:])
    else:
        print("There are no pairs of similar elements.")
```

Рисунок 3 – Код программы

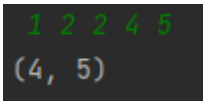

Рисунок 4 – Пример работы программы

2. Ответы на контрольные вопросы

1. Что такое кортежи в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список.

2. Каково назначение кортежей в языке 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. 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.

3. Как осуществляется создание кортежей?

Для создания пустого кортежа можно воспользоваться одной из следующих команд:

a = ()

b = tuple()

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

4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса.

```
a = (1, 2, 3, 4, 5)

 print(a[0])
```

5. Зачем нужна распаковка (деструктуризация) кортежа?

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
```

6. Какую роль играют кортежи в множественном присваивании?

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

```
(a, b) = (b, a)
```

7. Как выбрать элементы кортежа с помощью среза?

Общая форма операции взятия среза для кортежа следующая:

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

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.

8. Как выполняется конкатенация и повторение кортежей?

T3 = T1 + T2 – Конкатенация

T2 = T1 * n – Повторение

9. Как выполняется обход элементов кортежа?

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

10. Как проверить принадлежность элемента кортежу?

С помощью оператора «in»

11. Какие методы работы с кортежами Вам известны?

index() и count()

12. Допустимо ли использование функций агрегации таких как len() , sum() и т. д. при работе с кортежами?

Да, т.к они не изменяют элементы в самом кортеже.

13. Как создать кортеж с помощью спискового включения.

A = tuple(map(int, input().split()))