Лабораторне заняття №7

з навчальної дисципліни

Спеціалізовані мови програмування

Python (advanced)

на тему:

ВІДОБРАЖЕННЯ

Мета роботи

Ознайомитися з особливостями відображеннями в мові програмування Python 3

Хід роботи

- Самостійно на ПК реалізувати програмний код наведений нижче

ПРИКЛАД №1

"""
Для змінюваних послідовностей характерні операції присвоювання і видалення елементів за індексом або зрізом. Вони реалізуються за допомогою спеціальних методів __setitem__ и __delitem__.
"""

```
my_list = [1, 4, 3, 5, 7]
print(my_list)

my_list[0] = 10
print(my_list)

my_list[3:] = range(10)
print(my_list)

del my_list[0]
print(my_list)

del my_list[:]
print(my_list)
```

ПРИКЛАД №2

```
sequence = [1, 3, 5]
print(sequence)

# Додавання елемента в кінець
sequence.append(3)  # sequence[len(sequence):] = [3]
print(sequence)

# Додавання в кінець елементів з ітерабельного об'єкта
sequence.extend(range(3))  # sequence[len(sequence):] = range(3)
print(sequence)

# Додавання елемента по індексу
sequence.insert(1, 8)  # sequence[1:1] = [8]
print(sequence)

# Видалення першого входження елемента
sequence.remove(3)  # del sequence[sequence.index(3)]
print(sequence)

# Видалення останнього елемента
sequence.pop()  # del sequence[-1]
print(sequence)

# Видалення елемента по індексу
sequence.pop(4)  # del sequence[4]
print(sequence)

# Видалення всіх елементів
sequence.clear()  # del sequence[:]
print(sequence)
```

ПРИКЛАД №3

```
sequence = list(range(10))
print(sequence)

# Неповне копіювання
copy = sequence.copy()  # copy = sequence[:]
print(copy)

# Розворот в зворотному порядку
sequence.reverse()
print(sequence)
```

ПРИКЛАД №4

"""

Спискові включення дозволяють лаконічно описувати списки, які будуються за певними виразами, фільтрувати їх і застосовувати функції до їх елементів. Їх синтаксис аналогічний синтаксису виразів-генераторів, але тут використовіються квадратні дужки.

"""

```
# Створення списку
my_list = [x ** 2 for x in range(10)]
print(my_list)

# Вибір парних і непарних чисел зі списку
print([x for x in my_list if x % 2 == 0])
print([x for x in my_list if x % 2 == 1])
```

ПРИКЛАД №5

"""

Метод sort () сортує список, модифікуючи його. Також існує вбудована функція sorted, яка сортує будь-який ітеруємий об'єкт, не модифікуючи його, і повертає список.

"""

```
my_list = [4, 3, 2, 0, 17, -5, 3]
list_copy = my_list[:]
print(my_list)

my_list.sort()
print(my_list)

list_copy.sort(reverse=True)
print(list_copy)

print()

string = 'Lorem ipsum dolor sit amet.'
print(string)
sorted_string = sorted(string)
print(sorted_string)
print(''.join(sorted_string))
```

ПРИКЛАД №6

```
# Операція множення часто використовується зі списками
# для ініціалізації списку заданою кількістю однакових елементів

some_list = [0] * 10
print(some_list)

print()
```

ПРИКЛАД №7

```
def print_matrix(matrix):
    """Функція виводу елементів матриці"""
    for row in matrix:
        print(' '.join(str(element) for element in row))

# Створення матриці 5x5 (неправильно)
matrix_done_wrong = [[0] * 5] * 5
# Пока что выводится правильно
print_matrix(matrix_done_wrong)
print()

# Зміна одного елемента
matrix_done_wrong[1][3] = 8
# Змінилися відповідні елементи у всіх рядках
print_matrix(matrix_done_wrong)
print()
```

ПРИКЛАД №8

```
# Створення матриці 5x5 (правильно)
matrix_done_right = [[0] * 5 for _ in range(5)]
# Вивод елементів
print_matrix(matrix_done_right)

print()

# Зміна одного елемента
matrix_done_right[1][3] = 8
print_matrix(matrix_done_right)
```

ПРИКЛАД №9

```
"""
Кортежі - це незмінні послідовності, як правило, використовується, щоб зберігати різнотипні дані (або однотипні, але логічно представляють різні сутності). Представлені класом tuple.
"""
# Пустий кортеж
empty_tuple = ()
print(empty_tuple)
# Кортеж із одного елемента
singleton_tuple = (8,)
print(singleton_tuple)
# Кортеж із декількох елементов
some_tuple = (3, 2, 1, 8)
print(some_tuple)
# Або, теж саме
some_tuple = 3, 2, 1, 8
print(some_tuple)
# Список кортежів
coordinates = [(8, 3), (2, 0), (3, 4), (0, 0)]
print(coordinates)
# Кортеж кортежів
triangle = ((0, 0), (4, 0), (0, 3))
print(triangle)
```

ПРИКЛАД №10

```
# Змінні, об'єднані в кортеж, можуть стояти в лівій частині присвоєння або заголовку циклу for. Тоді їм
# присвоюються відповідні значення ітерабельного об'єкта.

a, b, c = 1, 2, 3
print(a)
print(b)
print(c)

# В список rest будуть поміщені елементи послідовності, що залишилися
a, b, *rest = range(10)
print(a)
print(b)
print(rest)

# Поміняти місцями значення двох змінних
print(a, b)
a, b = b, a
print(a, b)

# Список кортежей
tuples = [(x, y) for x in range(3) for y in range(3)]
# Ітерування списку
for t in tuples:
    print(t)
# Ітерування з розпаковкою
for x, y in tuples:
    print(x, y)
```

ПРИКЛАД №11

```
#Вбудована функція zip повертає об'єкт-ітератор, який повертає кортежі, що складаються з відповідних
#елементів заданих послідовностей. Кількість елементів, які повертає ітератор, дорівнює довжині
#найменшою з послідовностей.

nodes = ['node1', 'node2', 'node3']
weights = [1, 7, 5, 5, 9, 3]

for node, weight in zip(nodes, weights):
    print('The weight of node', node, 'is', weight)

# Вбудована функція enumerate повертає об'єкт-ітератор, який повертає пари індексів і значень
# послідовності Тобто, поведінка enumerate (seq) аналогічно zip (range (seq), seq))

for index, node in enumerate(nodes):
    print('nodes[{}] = {}'.format(index, node))
```

ПРИКЛАД №12

```
# Функція може мати довільну кількість аргументів. Після всіх позиційних
#параметрів функції або замість них (але перед тими, які передбачається
#використовувати як іменовані) в її сигнатурі можна вказати спеціальний
#аргумент з символом * перед ім'ям. Тоді фактичні параметри зберігаються в
#кортежі з цим ім'ям.

def multiply(*numbers):
    result = 1
    for number in numbers:
        result *= number
    return result


print(multiply(2, 3))
print(multiply(1, 9, 7, 8))
```

```
# Також існує і зворотна можливість. Якщо при виконанні функції
# перед ім'ям ітерабельного об'єкта поставити символ *, то його елементи
# розпаковуються в позиційні аргументи.

def print_person(name, age, address):
    print(name, 'is', age, 'years old and lives at', address)


data = [
    ('John', 23, '18 Spring Lane'),
    ('Kate', 18, '20 Victory Str'),
    ('Vasiliy', 20, '323 Green Ave'),
]

for person in data:
    print_person(*person)
```

ПРИКЛАД №14

"""
Послідовності однакових типів можна порівнювати. Порівняння відбуваються в лексикографічному порядку: послідовність меншої довжини менше, ніж послідовність більшої довжини, якщо ж їх довжини рівні, то результат порівняння дорівнює результату порівняння перших відмінних елементів.
"""

```
print('abc' < 'ab')
print('abc' < 'abcd')

words = ['lorem', 'ipsum', 'dolor', 'sit', 'amet']
print(sorted(words))
```

Завдання на самостійну роботу

Оформити звіт

Заняття закінчено.
Дякую за увагу!