# Лабораторне заняття №5

з навчальної дисципліни

Спеціалізовані мови програмування

**Python (advanced)**

на тему:

# ОСНОВНІ ФУНКЦІЇ РОБОТИ ЗІ СПИСКАМИ/РЯДКАМИ

# Мета роботи

Ознайомитися з основним функціоналом роботи зі списками в мові програмування Python 3

# Хід роботи

- Самостійно на ПК реалізувати програмний код наведений нижче

# ПРИКЛАД №1

```
# Список
my_list = [1, 2, 3]

# Ітерування
print('Iterating:')
for element in my_list:
    print(element)

print()

# Отримання доступу до елементів за допомогою цілочисельних ключів (індексація)
print('Indexing:')
print(my_list[0])
print(my_list[2])
print(my_list[-1])

print()

# Довжина послідовності
print('Length:', len(my_list))
```

# ПРИКЛАД №2

"""

Деякі ітеровані об'єкти мають певні загальні властивості. Ітеровані об'єкти, які підтримують ефективний доступ до елементів з використанням цілочисельних індексів через спеціальний метод ***__getitem __ ()*** і підтримують метод ***__len __*** (), який повертає кількість елеметов, називаються ***послідовностями***. Вже були розглянуті три типи даних, які є послідовностями.

"""

```
# Строка
string = "Lorem ipsum dolor sit amet, consectetur adipisicing elit."
# Итерирование
print('Iterating:')
for character in string:
    print(character)

print()

# Отримання доступу до елементів за допомогою цілочисельних ключів (індексація)
print('Indexing:')
print(string[0])
print(string[2])
print(string[-1])
print()

# Довжина послідовності
print('Length:', len(string))
```

# ПРИКЛАД №3

"""

Більшість послідовностей підтримують операції перевірки входження елемента ***in*** та ***not in.***
Для підтримки даної операції необхідно реалізувати спеціальний метод ***__contains__***

"""

```
# Список
my_list = [1, 3, 5, 7]
print(3 in my_list)
print(9 in my_list)
print(18 not in my_list)

print()

# Діапазон
my_range = range(2, 10)
print(3 in my_range)
print(5 not in my_range)

print()

# Для строк ця операція перевіряє входження підстроки
print('ips' in 'Lorem ipsum dolor sit amet.')

print()
```

# ПРИКЛАД №4

```
"""
Поширеною операцією є конкатенація послідовностей
"""

print('A ' + 'string')
print([2, 3] + [4, 5])
```

# ПРИКЛАД №5

```
print('*' * 80)

print([2, 3] * 10)
```

# ПРИКЛАД №6

"""
Також в послідовностях реалізовано метод ***count***,
який повертає кількість входжень елемента в послідовність.
"""

```
text = '''Lorem ipsum dolor sit amet, consectetur adipisicing elit. Temporibus
doloremque facere blanditiis, dolores porro a autem facilis repudiandae
ea commodi quidem, maxime aliquid aut quam similique? Ea magni accusantium
nam excepturi dignissimos.'''

print(text.count('o'))
```

# ПРИКЛАД №7

"""
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('ABCD' < 'abcd')

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

# ПРИКЛАД №8

```
some_str = 'Lorem ipsum'

print(len(some_str))
```

# ПРИКЛАД №9

```
some_str = 'Lorem ipsum dolor sit amet.'

print(some_str[0])
print(some_str[2])
print(some_str[6])

print()
print(some_str[2:5])
print(some_str[:5])
print(some_str[2:-2])
print()
print(some_str[:])
print(some_str[3:10:2])
print(some_str[:15:2])
print(some_str[:15:-2])
print()
print(some_str[::-2])
print(some_str[::-1])
print(some_str[2::-1])
print(some_str[-2:2:-1])
print()
```

# ПРИКЛАД №10

```
some_str  = '        Lorem ipsum dolor sit amet     '
test = 'sadasdasd'

print(some_str)

# find – виконує пошук в some_str підстроки "L" , повертає індекс  , або  -1
print(some_str.find('L'))

print(some_str.rfind('o')) # Пошук підрядка в рядку. Повертає номер останнього входження або -1
print(some_str.lower())
print(some_str.upper())
print(some_str.index('o')) # Пошук підрядка в рядку. Повертає номер першого входження або викликає ValueError

print(some_str.rindex('o')) # Пошук підрядка в рядку. Повертає номер останнього входження або викликає ValueError
```

# ПРИКЛАД №10 (продовження)

```
print(some_str.replace('e' , '--- ---'))
print(some_str.split(' ')) #       Розбиття рядка по роздільнику
print(some_str.isdigit()) #       Чи складається рядок з цифр
print(test.isalpha()) #            Чи складається рядок з літер

print(some_str.startswith('Lo'))
print(some_str.endswith('et'))

print(some_str.lstrip()) #Видалення пробілів на початку рядка
print(some_str.rstrip()) #Видалення пробілів в кінці рядка
print(some_str.strip())
```

# Завдання на самостійну роботу

# Оформити звіт

Заняття закінчено.
Дякую за увагу!