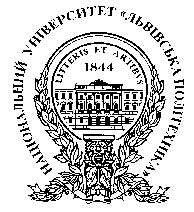
МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

Національний Університет “Львівська політехніка”



Лабораторна робота № 7

***ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ.***

***СТРУКТУРНЕ ПРОГРАМУВАННЯ МОВОЮ PYTHON (частина1).***

Виконав:

студент групи ПРЛс-11

Форманюк А.А.

Прийняв:

Дупак Б.П.

Львів 2015

**МЕТА РОБОТИ**

* Вивчення основ програмування на мові *Python*.
* Вивчення основ структурного програмування мовою *Python*.
* Повторення та закріплення знань отриманих при виконанні попередніх лабораторних робіт.
* Покращення загальних навичок у програмуванні.

**КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

**Виконання даної лабораторної роботу дозволить знайти відповіді на такі основні питання:**

1. Яким чином писати гарно структуровані, чіткі, сприйнятливі (читабельні) програми, придатні для неодноразового використання?
2. Яким чином працюють основні конструкційні блоки, а саме: цикл, функція, присвоювання?
3. Які можуть бути пастки при програмуванні на Python та як їх уникати?

## *1. Основи програмування*

### *1.1 Присвоювання*

Присвоювання – найпростіше поняття програмування, але навіть і йому властиві певні тонкощі. Розглянемо приклад:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> foo = 'Monty'**  **>>> bar = foo**  **#1**  **>>> foo = 'Python'**  **#2**  **>>> bar**  **'Monty'** | |

Коли записується bar = foo (рядок програми #1), то значення змінної foo (стрічка 'Monty') присвоюється змінній bar. Таким чином, bar є копією foo. Якщо змінити foo на нову стрічку 'Python' (рядок програми #2), то змінної bar ці зміни не торкнулася.

За допомогою операції присвоєння відбувається копіювання значення виразу іншій змінній, хоча насправді, значення структурованого об’єкту, наприклад такого, як список, це є посилання на цей об’єкт. В наступному прикладі (рядок програми #1) відбувається присвоєння посилання на значення змінної foo новій змінній bar. Якщо відбуваються зміни в foo (рядок програми #2) то ці зміни також торкаються і bar.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> foo = ['Monty', 'Python']**  **>>> bar = foo** **#1**  **>>> foo[1] = 'Bodkin'** **#2**  **>>> bar**  **['Monty', 'Bodkin']** | |

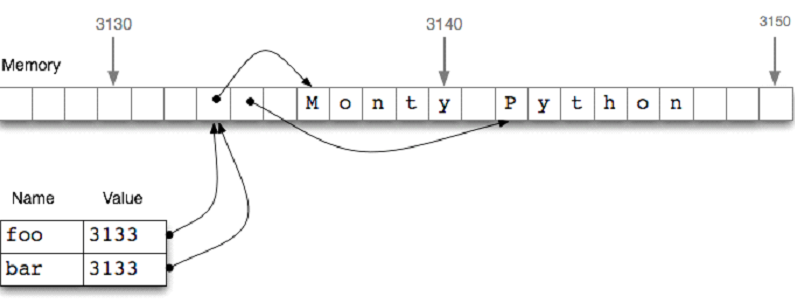


Рис.1: Операція присвоювання для списків в оперативній пам’яті. Два об’єкти foo та bar, типу список, посилаються на те саме місце в оперативній пам’яті. Зміна foo модифікує bar і навпаки.

Рядок bar = foo означає не копіювання вмісту змінної а тільки копіювання її посилання на відповідний об’єкт. Для глибшого розуміння розглянемо, яким чином списки збережені в оперативній пам’яті. На рис.1. показано, що список foo, це посилання на об’єкт збережений за адресою 3133. Коли відбувається операція присвоювання bar = foo , то копіюється це посилання на об’єкт 3133.

Розглянемо ще декілька прикладів. Створимо змінну empty і присвоємо їй значення пустого списку. В наступному рядку використаємо цю змінну декілька раз.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> empty = []**  **>>> nested = [empty, empty, empty]**  **>>> nested**  **[[], [], []]**  **>>> nested[1].append('Python')**  **>>> nested**  **[['Python'], ['Python'], ['Python']]** | |

Бачимо, що зміна одного вкладеного списку привела до зміни і всіх інших. Це відбулося тому, що ці всі три елементи списку насправді є посиланнями на одне і те саме місце (на той самий список) .

***Виконати самостійно***. Створити список списків: nested = [[]] \* 3. Змінити один з елементів списку і прослідкувати, як змінилися інші елементи списку. Використовувати вбудовану функцію id() для знаходження числового значення, яке ідентифікує об’єкт – ідентифікатора (адреса об’єкту) id(nested[0]), id(nested[1]), id(nested[2]).

В наступному прикладі показано, як присвоювання нового значення елементу списку не приводить до зміни його інших елементів:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> nested = [[]] \* 3**  **>>> nested[1].append('Python')**  **>>> nested[1] = ['Monty']**  **>>> nested**  **[['Python'], ['Monty'], ['Python']]** | |

Початковий список містить три посилання на єдиний пустий об’єкт типу список. В результаті модифікації цього об’єкту додаванням до нього 'Python' список містить ті самі три посилання але вже на єдиний об’єкт ['Python'] типу список. В подальшому відбувається перезаписування одного з цих посилань на посилання на новий об’єкт ['Monty']. У списку відбулася модифікація одного з трьох посилань на об’єкт ['Python'].

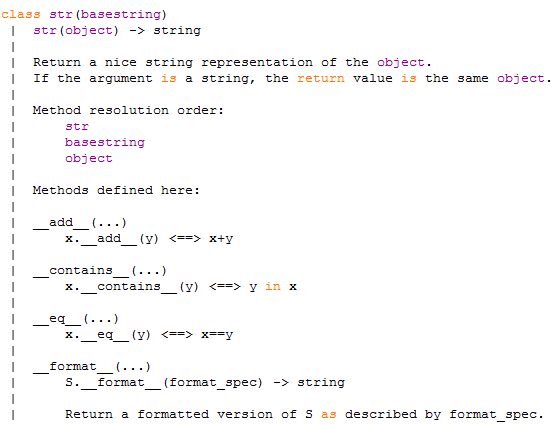
Потрібно чітко розрізняти модифікацію об’єкту через посилання на об’єкт і перезаписування посилання на об’єкт.

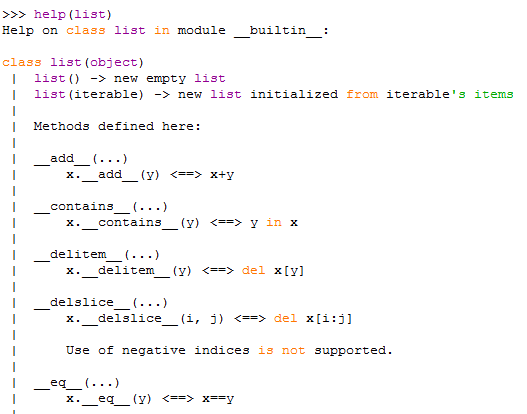
Для копіювання вмісту списку foo до нового списку bar можна записати bar = foo[:] . В цьому випадку відбувається копіювання посилань на об’єкт. Для копіювання структури без копіювання посилань на об’єкт потрібно скористатися функцією copy.deepcopy() модуля copy. (deepcopy будує новий складний об’єкт і тоді вставляє в нього копії об’єктів, які знайдені в оригіналі.)

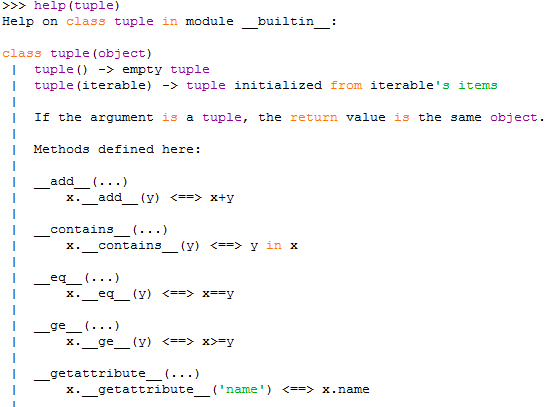
|  |  |
| --- | --- |
|  | **>>> foo = ['Monty', 'Python']**  **>>> bar=foo**  **>>> id(foo)**  **13276784**  **>>> id(bar)**  **13276784**  **>>> foo = ['Monty', 'Python']**  **>>> bar=foo[:]**  **>>> id(foo)**  **13238144**  **>>> id(bar)**  **13258544** |

**Завдання**

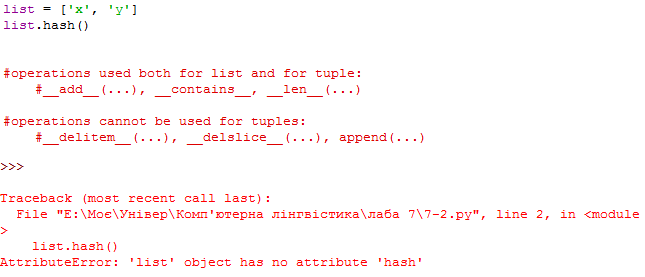
**Завдання 1.** Знайти в Python's help додаткову інформацію про послідовності. В інтерпретаторі, набрати по черзі help(str), help(list), та help(tuple). На екрані буде відображено повний список функцій властивих кожному з типів. Деякі функції мають спеціальні імена з подвійними підкреслюваннями. Кожній такій функції відповідає і інший запис показаний в документації. Наприклад x.\_\_getitem\_\_(y) відповідає x[y].

****

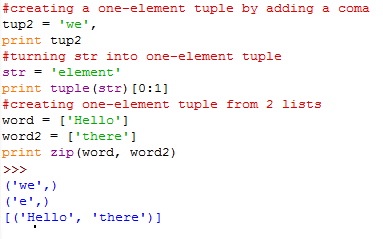
****

****

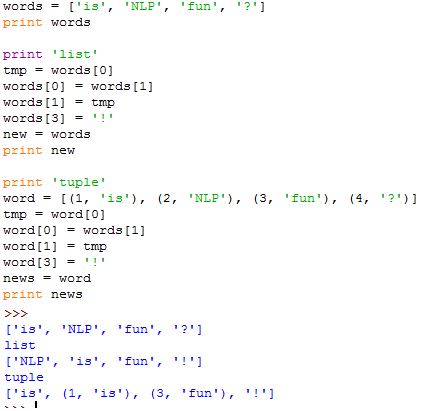
**Завдання 2.** Знайти три операції, які можна здійснювати і зі списками та із кортежами. Знайти три операції, які не можна здійснювати над кортежами. Знайдіть коли використання списку замість кортежу приводить до Python помилки.

****

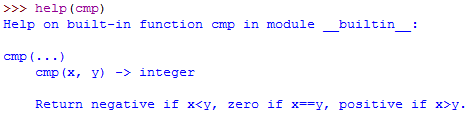
**Завдання 3.** Яким чином можна створити кортеж з одного елемента. Продемонструвати два різні способи.

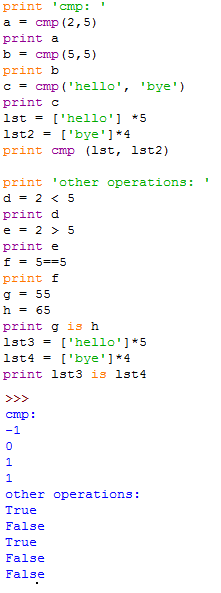
****

**Завдання 4.** Створити список words = ['is', 'NLP', 'fun', '?']. Використовуючи операції присвоювання подібні до words[1] = words[2] та тимчасову змінну tmp перетворити цей список в список ['NLP', 'is', 'fun', '!']. Здійснити аналогічні перетворення використовуючи присвоювання в кортежах.

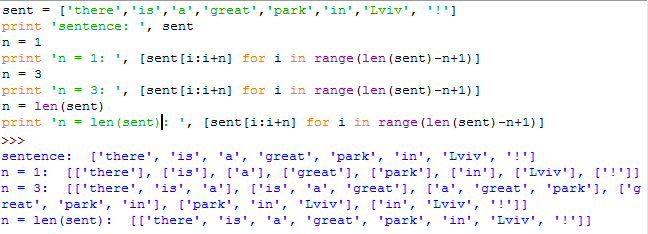
****

**Завдання 5.** Прочитати про вбудовану функцію здійснення порівнянь cmp, набравши help(cmp). Продемонструвати чим поведінка цієї функції відрізняється від поведінки операторів порівняння.

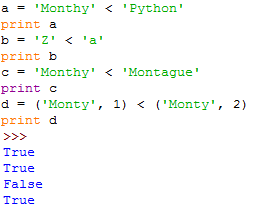
****



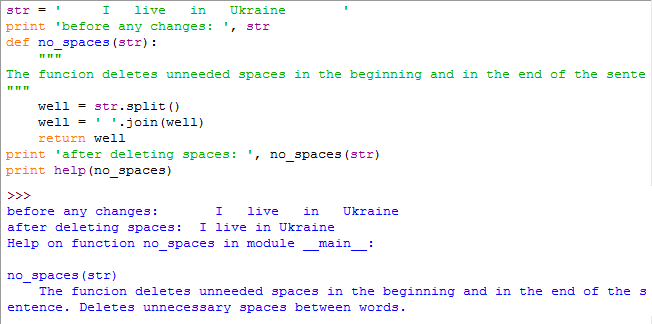
**Завдання 6.** Написати програму для коректного виділення в тексті n-грамів з врахуванням граничних випадків: n = 1, та n = len(sent)?



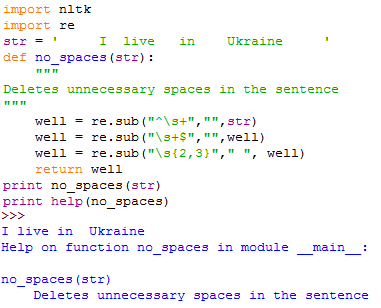
**Завдання 7.** Використати оператори нерівності для порівняння стрічок, наприклад. 'Monty' < 'Python'. Що станеться, якщо виконати 'Z' < 'a'? Порівняти стрічки,як мають однаковий префікс, наприклад 'Monty' < 'Montague'. Спробувати порівняти структуровані об’єкти ,наприклад. ('Monty', 1) < ('Monty', 2). Чи отримали очікувані результати?



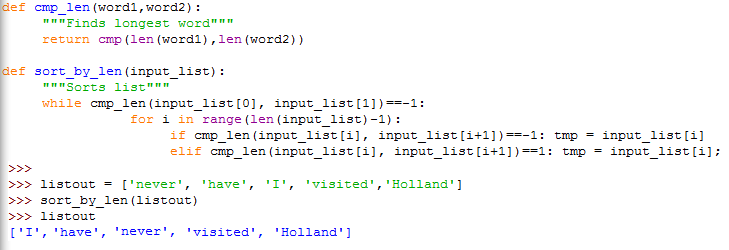
**Завдання 8.** Написати програму видалення пробілів на початку і в кінці стрічки та для видалення зайвих пробілів між словами. Використовувати split() та join(). Оформити у вигляді функції. Функція повинна містити повну стрічку документування.



**Завдання 9.** Написати програму видалення пробілів на початку і в кінці стрічки та для видалення зайвих пробілів між словами. Використовувати re.sub() . Оформити у вигляді функції. Функція повинна містити повну стрічку документування



**Завдання 10.** Написати програму сортування слів за їх довжиною. Визначити допоміжну функцію cmp\_len, яка буде використовувати функцію cmp для порівняння довжин слів. Функція повинна містити повну стрічку документування.



**Висновок:** на лабораторній роботі я вивчив основи програмування на мові *Python*, основи структурного програмування мовою *Python* і повторив та закріпив знання отримані при виконанні попередніх лабораторних робіт