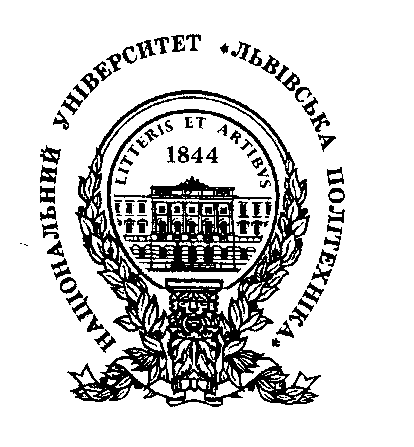
Міністерство освіти та науки України

Національний університет «Львівська Політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра САПР



Лабораторна робота №7

на тему:

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

Виконала:

студентка ПРЛм-11

Михайлів Р.Б.

Перевірив:

Дупак Б.П.

Львів 2015

**Мета робота**

Вивчити основи структурного програмування мовою Python, повторити та закріпити матеріал, пройдений у попередніх лабораторних роботах, покращити загальні навички програмування.

**Короткі теоретичні відомості**

**Присвоювання**

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

Розглянемо приклад: 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 показано, як присвоювання нового значення елементу списку не приводить до зміни його інших елементів: Потрібно чітко розрізняти модифікацію об’єкту через посилання на об’єкт і перезаписування посилання на об’єкт.

**Порівняння**

Python підтримує два способи порівняння. Оператор is перевіряє об’єкти на ідентичність. Створивши список з декількох копій одного і того самого об’єкту не складно переконатися, що елементи цього списку не тільки ідентичні, згідно ==, але і є одним і тим самим об’єктом.

Використовуючи функцію id() можна легко зрозуміти відмінності між об’єктами і побачити що елементи списку не є ідентичними - списки мають різні ідентифікатори.

**Умовні твердження (висловлювання)**

В частині умов if твердження, не пусті стрічки вважаються «true», а пусті стрічки чи списки вважаються «false» і не обробляються. Тому, не потрібно використовувати додаткову перевірку if len(element) > 0: в умові. Для пояснення відмінностей між використанням  if...elif та використання декількох if тверджень розглянемо наступний приклад: якщо  if вираз твердження задовольняється то, Python ніколи не буде робити спроби перевірити чи справджується elif вираз. На екран «2» ніколи не буде виведено. Навпаки, якщо замінити  elif на if, на екран буде виведено і  «1» і «2». elif вираз потенційно є більш інформативний ніж простий if вираз; коли він справджується то це означає не тільки те що умова задовольняється але і те, що умова if виразу не справджується.

Вбудовані функції all() та any() можуть бути застосовані до списку або іншої послідовності для перевірки, чи всі, або будь-який з елементів задовольняють умову: Операції над послідовностями різних типів Елементи послідовності s можна обробляти почергово (здійснювати над ними ітерації, ітерувати) різними способами, основні з яких, наведено у таблиці 1. Способи ітерування елементів послідовностей

| **Вираз Python** | **Пояснення** |
| --- | --- |
| for item in s | Проітерувати елементи s |
| for item in sorted(s) | Проітерувати впорядковані елементи s |
| for item in set(s) | Проітерувати унікальні елементи s |
| for item in reversed(s) | Проітерувати зворотньо впорядковані елементи s |
| for item in set(s).difference(t) | Проітерувати елементи s, які не входять в t |
| for item in random.shuffle(s) | Проітерувати випадково впорядковані елементи s |

Послідовності різних типів можна перетворювати між собою. Наприклад, tuple(s) – перетворення послідовності будь-якого типу в кортеж, list(s) – перетворення послідовності будь-якого типу в список. Для перетворення списку стрічок в єдину стрічку потрібно використовувати функцію join(), наприклад, ':'.join(words). Деякі інші об’єкти, такі як  FreqDist, також можуть бути перетворені в послідовність, використовуючи list(), а також їх можна й ітерувати. Генерація виразів Використання list comprehensions дозволяє отримувати компактний та зручний для читання текст програми. Наприклад, наступний фрагмент програми для токенізації та нормалізації тексту:

|  |  |
| --- | --- |
|  | **>>> text = '''"When I use a word," Humpty Dumpty said in rather a scornful tone,**  **... "it means just what I choose it to mean - neither more nor less."'''**  **>>> [w.lower() for w in nltk.word\_tokenize(text)]**  **['"', 'when', 'i', 'use', 'a', 'word', ',', '"', 'humpty', 'dumpty', 'said', ...]** |

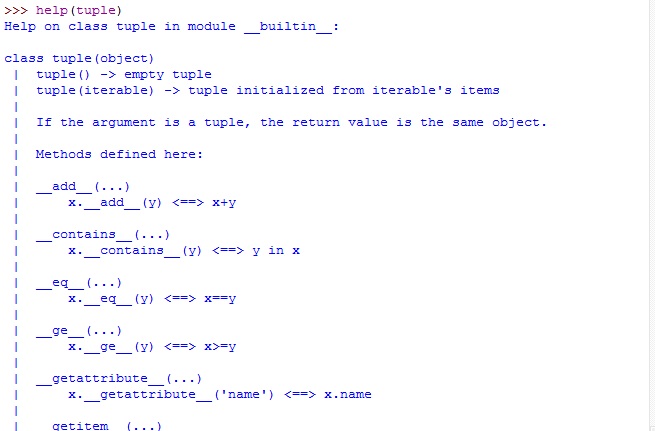
Функції, як основа структурного програмування Функції забезпечують ефективний спосіб збереження та повторного використання частин програм. Наприклад, припустимо, що існує необхідність часто читати тексти з HTML файлу. Це потребує виконання певної послідовності кроків: відкриття файлу, читання, нормалізація, видалення HTML розмітки. Можна зібрати ці кроки у функцію get\_text().

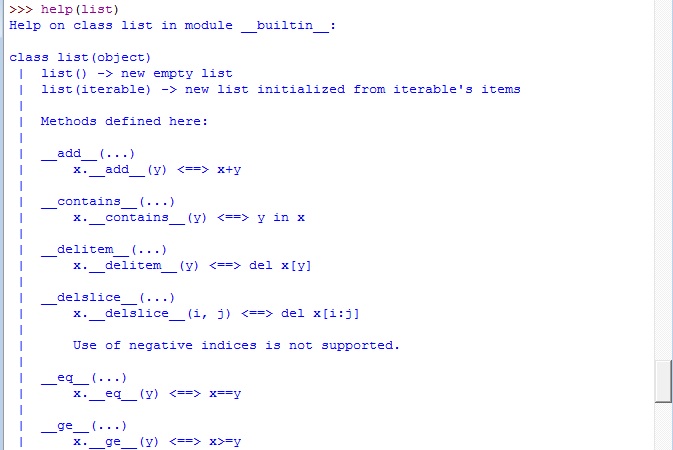
|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **import re**  **def get\_text(file):**  **"""Read text from a file, normalizing whitespace and stripping HTML markup."""**  **text = open(file).read()**  **text = re.sub('\s+', ' ', text)**  **text = re.sub(r'<.\*?>', ' ', text)**  **return text** | |
|  |

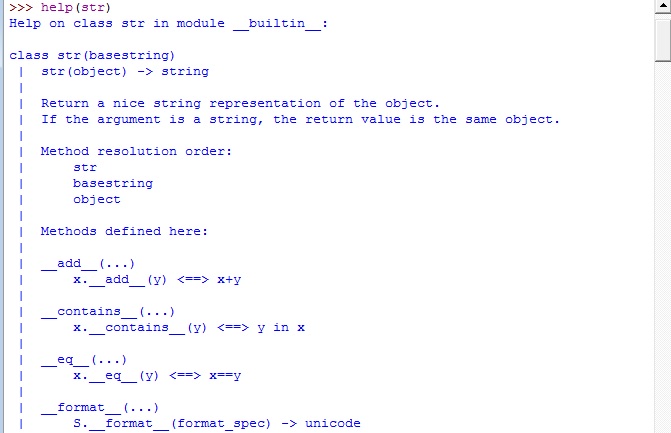
Тепер, в будь-який момент, можна отримати текст з HTML файлу викликавши функцію get\_text() вказавши назву файлу, як її аргумент. Функція повертає стрічку, значення, якої можна присвоїти будь-якій змінній, наприклад: contents = get\_text("test.html"). При створенні функції важливе значення має вибір назви функції, що дозволяє зробити програму більш читабельною. У випадку попереднього прикладу, кожен раз коли програма повинна читати текст, немає потреби засмічувати текст основної програми чотирма рядками , потрібно тільки викликати функцію get\_text (). Назва цієї функції має певну «семантичну інтерпретацію» і допомагає відразу зрозуміти що програма робить.Тепер, в будь-який момент, можна отримати текст з HTML файлу викликавши функцію get\_text() вказавши назву файлу, як її аргумент.

**Виконання індивідуального завдання**  
20 варінат

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].







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.

Спільні:

|  \_\_add\_\_(...)

|  x.\_\_add\_\_(y) <==> x+y

|  \_\_contains\_\_(...)

|  x.\_\_contains\_\_(y) <==> y in x

|  \_\_eq\_\_(...)

|  x.\_\_eq\_\_(y) <==> x==y

Не можна з кортежами:

|  \_\_imul\_\_(...)

|  x.\_\_imul\_\_(y) <==> x\*=y

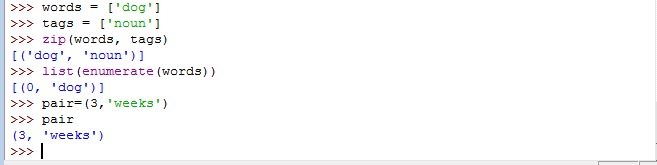
|  \_\_delitem\_\_(...)

|   x.\_\_delitem\_\_(y) <==> del x[y]

|  \_\_iadd\_\_(...)

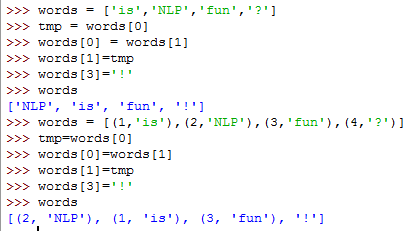
|  x.\_\_iadd\_\_(y)<==>x+=y

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

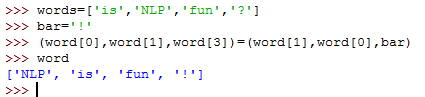


4.Створити список words = ['is', 'NLP', 'fun', '?']. Використовуючи операції присвоювання подібні до words[1] = words[2] та тимчасову змінну tmp перетворити цей список в список ['NLP', 'is', 'fun', '!']. Здійснити аналогічні перетворення використовуючи присвоювання в кортежах.

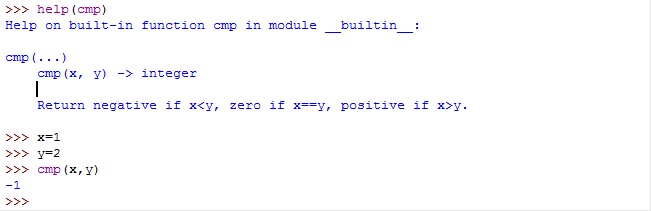
Список



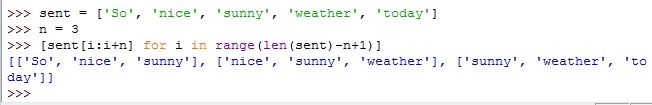
Кортеж



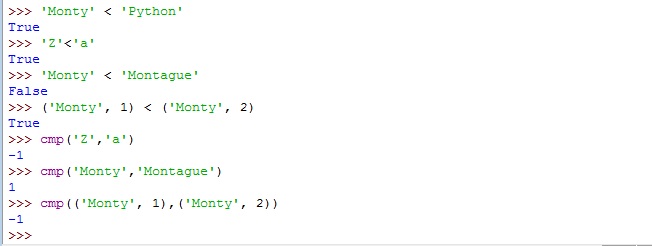
5.Прочитати про вбудовану функцію здійснення порівнянь cmp, набравши help(cmp). Продемонструвати чим поведінка цієї функції відрізняється від поведінки операторів порівняння.



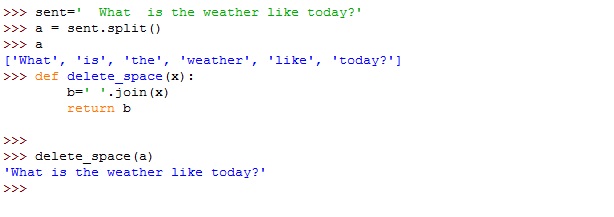
6.Написати програму для коректного виділення в тексті n-грамів з врахуванням граничних випадків: n = 1, та n = len(sent)?



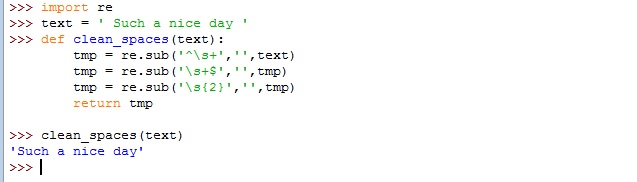
7.Використати оператори нерівності для порівняння стрічок, наприклад. 'Monty' < 'Python'. Що станеться, якщо виконати 'Z' < 'a'? Порівняти стрічки,як мають однаковий префікс, наприклад 'Monty' < 'Montague'. Спробувати порівняти структуровані об’єкти ,наприклад. ('Monty', 1) < ('Monty', 2). Чи отримали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



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.



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



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.

