МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп’ютерних наук

та інформаційних технологій

Звіт

до лабораторної роботи №7

з дисципліни «Комп’ютерна лінгвістика»

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

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

Виконала:

студентка групи ПРЛм-11

Неїжмак О.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**

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

**Присвоювання.**

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

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:

**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.

### 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 Таблиця 1:

| 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 |

Функції з Таблиці 1. можна поєднувати у різний спосіб. Наприклад, для одержання унікальних елементів s впорядкованих у зворотному порядку необхідно використовувати наступний вираз reversed(sorted(set(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).

**Стиль програмування**

Програмування це більше мистецтво ніж наука. Одна з найгрунтовніших книг по програмуванню, написана Дональдом Кнутом (Donald Knuth) так і називається The Art of Computer Programming. Приклади програм у більшості підручників написані таким чином, щоб не тільки комп’ютери але і люди змогли прочитати і зрозуміти текст програми. При програмуванні важливе значення має стиль програмування, який впливає на розуміння тексту програми і передбачає такі складові, як розміщення тексту програми, вибір процедурного чи декларативного стилю, використання змінних в циклах та багато інших.

### Стиль програм Python

При написанні програм необхідно вміло вибирати імена змінних, вставляти пробіли, додавати коментарі та робити інші кроки, для того щоб, текст програми був зрозумілий і зручний для читання не тільки автору але й іншим. Розробники мови програмування Python опублікували (http://www.python.org/dev/peps/pep-0008/) керівництво по стилю програм мовою Python. Основне поняття, представлене в цьому керівництві, це узгодженість, для забезпечення максимальної читабельності тексту програми. В лабораторній роботі коротко розглядаються основні рекомендації описані у керівництві.

При розміщенні тексту програми потрібно використовувати чотири пробіли перед вкладеною частиною програми. При записі тексту програми у файлі потрібно уникати використання табуляції для цих відступів, оскільки вони можуть бути помилково інтерпретовані деякими текстовими редакторами і не збережені. Рядок програми повинен бути меншим за 80 символів. При необхідності, та частина рядка в круглих, квадратних та фігурних дужках, може бути розділена на декілька рядків (зроблений перенос).

### Процедурний чи декларативний стиль

Одне і те саме завдання може бути виконане різними шляхами з використанням різних змінних та з різною ефективністю. Інший критерій, який впливає та розробку програм, це стиль програмування.

**Варіант – 10**

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].

>>> help(str)

>>> help(list)

>>> help(tuple)

>>> x='cooperation'

>>> y=4

>>> x.\_\_getitem\_\_(y)

'e'

>>> x[y]

'e'

1. 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*

*\_\_getitem\_\_(...)*

*x.\_\_getitem\_\_(y) <==> x[y]*

*\_\_len\_\_(...)*

*x.\_\_len\_\_() <==> len(x)*

*Не можна з кортежами:*

*\_\_imul\_\_(...)*

*x.\_\_imul\_\_(y) <==> x\*=y*

*\_\_delitem\_\_(...)*

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

*\_\_init\_\_(...)*

*x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature*

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

>>> a = 'caramba',

>>> a

('caramba',)

>>> a = ('caramba',)

>>> a

('caramba',)

1. 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.

>>> words = ['is', 'NLP', 'fun', '?']

>>> print words

['is', 'NLP', 'fun', '?']

>>> tmp=words[0]

>>> words[0]=words[1]

>>> words[1]=tmp

>>> words[3]='!'

>>> print words

['NLP', 'is', 'fun', '!']

>>> words = ['is', 'NLP', 'fun', '?']

(words[0], words[1], words[3])=(words[1], words[0], '!')

>>> print words

['NLP', 'is', 'fun', '!']

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

>>> help(cmp)

Help on built-in function cmp in module \_\_builtin\_\_:

cmp(...)

cmp(x, y) -> integer

Return negative if x<y, zero if x==y, positive if x>y.

>>> 'a'<'b'

True

>>> 'a'>'b'

False

>>> cmp('a','b')

-1

>>> cmp('a','a')

0

>>> cmp('b','a')

1

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

>>> def ngram(n, sent):

gram=[sent[i:i+n] for i in range(len(sent)-n+1)]

return gram

>>> sent = ['The', 'dog', 'gave', 'John', 'the', 'newspaper']

>>> ngram(1,sent)

[['The'], ['dog'], ['gave'], ['John'], ['the'], ['newspaper']]

>>> ngram(len(sent),sent)

[['The', 'dog', 'gave', 'John', 'the', 'newspaper']]

1. 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?

>>> 'Monthy' < 'Python'

True

>>> 'Z' <'a'

True

>>> 'Monthy' < 'Montague'

False

>>> ('Monthy',1 ) < ('Monthy' , 2)

True

1. 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.

>>> def probily (text):

"""Приймає як аргумент стрічку. Змінній а1 присвоює перетворену у список стрічку.

Змінній а2 присвоює об’єднаний пробілами у стрічку список.

В результаті повертає готову стрічку а2."""

a1 = text.split()

a2 = ' '.join(a1)

return а2

>>> probily(' I want to live in America. ')

'I want to live in America.'

1. 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.

>>> import re

>>> def resubdel(sent):

"""Приймає стрічку як аргумент. За допомогою функції re.sub видаляє зайві пробіли з початку, середини та кінця стрічки. Потім функція повертає змінну sent."""

sent=re.sub('^\s+','',sent)

sent=re.sub('\s+',' ',sent)

sent=re.sub('\s+$','',sent)

return sent

>>> resubdel(' I want to live in America. ')

'I want to live in America.'

1. 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.

*>>> def cmp\_len(word1,word2):*

*return cmp(len(word1),len(word2));*

*>>> def sortlen(input\_list):*

*while cmp\_len(input\_list[0],input\_list[1])==-1:*

*for i in range(len(input\_list)-1):*

*if cmp\_len(input\_list[i],input\_list[i+1])==-1:*

*tmp = input\_list[i]*

*input\_list[i]=input\_list[i+1];input\_list[i+1]=tmp*

*elif cmp\_len(input\_list[i],input\_list[i+1])==1:*

*tmp = input\_list[i]; input\_list[i]=input\_list[i+1]*

*input\_list[i+1]=tmp*

*>>> list = ['strong','deaf','cry','way']*

*>>> sortlen(list)*

*>>> list*

*['strong', 'deaf', 'cry', ' ay']*