**Лабораторна робота №2**

**Тема. Функції і методи рядків. Списки (list). Функції і методи списків. Одномірні масиви. Кортежі.**

**Мета роботи**. Отримати навички роботи з функціями та рядками, з списками, кортежами.

**Зміст**.

1. Вивчення відомостей про роботу з функціями та рядками, з списками, кортежами.

2. Виконання роботи.

3. Отримання результату.

**Ключові положення.**

Рядки в апострофах і в лапках - одне і те ж. Причина наявності двох варіантів в тому, щоб дозволити вставляти в літерали рядків символи лапок і апострофів, не використовуючи екранування. Екрановані послідовності дозволяють вставити символи, які складно ввести з клавіатури.

У таблиці 1 приведено список функцій та методів рядків.

|  |  |
| --- | --- |
| Функція або метод | Призначення |
| **S = ‘str’; S = “str”; S = ‘’‘str’‘’; S = “”“str”“”** | літерали рядків |
| **S = “s\np\ta\nbbb”** | екрановані послідовності |
| **S = r”C:\temp\new”** | Неформатовані рядки (пригнічують  екранування) |
| **S = b”byte”** | рядок байтів |
| **S1 + S2** | Конкатенація (додавання рядків) |
| **S1 \* 3** | повторення рядка |
| **S[i]** | Звернення за індексом |
| **S[i:j:step]** | витяг зрізу |
| **len**(S) | довжина рядка |
| **S.find**(str, [start],[end]) | Пошук підрядка в рядку. Повертає номер першого входження або -1 |
| **S.rfind**(str, [start],[end]) | Пошук підрядка в рядку. Повертає номер останнього входження або -1 |
| **S.index**(str, [start],[end]) | Пошук підрядка в рядку. повертає  номер першого входження або викликає ValueError |
| **S.rindex**(str, [start],[end]) | Пошук підрядка в рядку. Повертає номер останнього входження або викликає ValueError |
| **S.replace**(шаблон, заміна) | заміна шаблону |
| **S.split**(символ) | Розбиття рядка по разделителю |
| **S.isdigit**() | Чи полягає рядок з цифр |
| **S.isalpha**() | Чи полягає рядок з букв |
| **S.isalnum**() | Чи полягає рядок з цифр або букв |
| **S.islower**() | Чи полягає рядок із символів в нижньому регістрі |
| **S.isupper**() | Чи полягає рядок із символів у верхньому регістрі |
| **S.isspace**() | Чи має рядок символи, що не відображаються (пробіл, символ перекладу страниці ('\ f'), "новий рядок" ('\ n'), "переклад каретки" ('\ r'), "горизонтальна табуляція" ('\ t') і "вертикальна табуляція "  ('\ v')) |
| **S.istitle**() | Чи починаються слова в рядку з великої літери |
| **S.upper**() | Перетворення рядка до верхнього регістру |
| **S.lower**() | Перетворення рядка до нижнього регістру |
| **S.startswith**(str) | Чи починається рядок S з шаблону str |
| **S.endswith**(str) | Закінчується рядок S шаблоном str |
| **S.join**(список) | Збірка рядка зі списку з роздільником S |
| **ord**(символ) | Символ в його код ASCII |
| **chr**(число) | Код ASCII в символ |
| **S.capitalize**() | Змінює перший символ рядка в верхній регістр, а всі інші в нижній |
| **S.center**(width, [fill]) | Повертає відцентрований рядок, по краях якого стоїть символ fill (пробіл за замовчуванням) |
| **S.count**(str, [start],[end]) | Повертає кількість непересічних входжень підрядка в діапазоні [початок, кінець] (0 і довжина рядка за замовчуванням) |
| **S.expandtabs**([tabsize]) | Повертає копію рядка, в якій всі  символи табуляції замінюються одним або декількома пропусками, в залежності від поточного стовпця. Якщо TabSize НЕ  вказано, розмір табуляції вважається рівним 8 прогалин |
| **S.lstrip**([chars]) | Видалення символів пробілів на початку рядка |
| **S.rstrip**([chars]) | Видалення символів пробілів у кінці рядку |
| **S.strip**([chars]) | Видалення символів пробілів на початку і в кінці рядка |
| **S.partition**(шаблон) | Повертає кортеж, що містить частину перед першим шаблоном, сам шаблон, і частину після шаблону. Якщо шаблон не знайдений, повертається кортеж, що містить сам рядок, а потім дві порожніх рядки |
| **S.rpartition**(sep) | Повертає кортеж, що містить частину перед останнім шаблоном, сам шаблон, і частину після шаблону. Якщо шаблон не знайдений, повертається кортеж, що містить дві порожні рядки, а потім сам рядок |
| **S.swapcase**() | Перекладає символи нижнього регістра в верхній, а верхнього - в нижній |
| **S.title**() | Першу букву кожного слова переводить в верхній регістр, а всі інші в нижній |
| **S.zfill**(width) | Робить довжину рядку не меншою width, в разі потреби заповнюючи перші символи нулями |
| **S.ljust**(width, fillchar=” ”) | Робить довжину рядку не меншою width, в разі потреби заповнюючи останні символи символом fillchar |
| **S.rjust**(width, fillchar=” ”) | Робить довжину рядку не меншою width, в разі потреби заповнюючи перші символи символом fillchar |
| **S.format**(\*args, \*\*kwargs) | форматування рядка |

Наприклад:

**Конкатенація (додавання)**>>> S1 = 'spam'  
>>> S2 = 'eggs'  
>>> print(S1 + S2)  
'spameggs'

Наприклад:

**Дублювання рядка**

>>> print('spam' \* 3)  
spamspamspam

Наприклад:

Доступ до індексу  
>>> S = 'spam'  
>>> S[0]  
's'  
>>> S[2]  
'a'  
>>> S[-2]  
'a'

Як видно з прикладу, в Python є можливість доступу по негативному індексу, при цьому відлік йде від кінця рядка.

Витяг зрізу. Оператор вилучення зрізу: [X: Y]. X - початок зрізу, а Y - закінчення; символ з номером Y в зріз не входить. За замовчуванням перший індекс дорівнює 0, а другий - довжині рядка.

Наприклад:

>>> s = 'spameggs'  
>>> s[3:5]  
'me'  
>>> s[2:-2]  
'ameg'  
>>> s[:6]  
'spameg'  
>>> s[1:]  
'pameggs'  
>>> s[:]  
'spameggs‘

Можна задати крок з яким треба витягувати зріз.

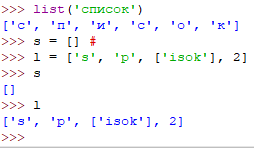
Наприклад:

>>> s[::-1]  
'sggemaps'  
>>> s[3:5:-1]  
''  
>>> s[2::2]  
'aeg'

Списки в Python - впорядковані змінювані колекції об'єктів довільних типів

(Майже як масив, але типи можуть відрізнятися).

Щоб використовувати списки, їх потрібно створити. Створити список можна декількома способами. Наприклад, можна обробити будь-який ітеріруемий об'єкт (наприклад, рядок) вбудованою функцією **list:**



Як видно з прикладу, список може містити будь-яку кількість будь-яких об'єктів (у тому числі і вкладені списки), чи не містити нічого.

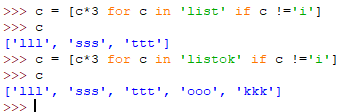
І ще один спосіб створити список - це генератори списків. Генератор списків – спосіб побудувати новий список, застосовуючи вираз до кожного елементу послідовності.

Генератори списків дуже схожі на цикл for.

Наприклад:

>>> c = [c \* 3 for c in 'list']  
>>> c  
['lll', 'iii', 'sss', 'ttt']

Наприклад:



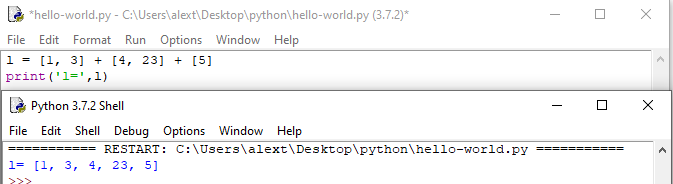
Для списків доступні основні вбудовані функції, а також методи списків.

Табл.1

|  |  |
| --- | --- |
| Метод | Що робить |
| **list.append**(x) | Додає елемент в кінець списку |
| **list.extend**(L) | Розширює список list, додаючи в кінець все елементи списку L |
| **list.insert**(i, x) | Вставляє на i-ий елемент значення x |
| **list.remove**(x) | Видаляє перший елемент у списку, який має значення x. ValueError, якщо такого елемента не існує. |
| **list.pop**([i]) | Видаляє i-ий елемент і повертає його. Якщо індекс не вказано,  видаляється останній елемент |
| **list.index**(x, [start [, end]]) | Повертає положення першого елемента зі значенням x (при цьому пошук ведеться від start до end) |
| **list.count**(x) | Повертає кількість елементів зі значенням x |
| **list.sort**([key=функція]) | Сортує список на основі функції |
| **list.reverse**() | Розгортає список |
| **list.copy**() | Поверхнева копія списку |
| **list.clear**() | Очищає список |

Списки можна додавати за допомогою знака **«+»**

Наприклад:

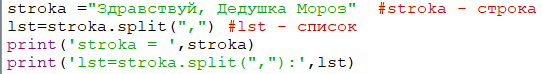


Створення списку за допомогою функції **Split ().**

Використовуючи функцію split в Python можна отримати з рядка список.

stroka = "Привіт, країна"

lst = stroka.split (",")

[](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_split_prog.png?attredirects=0)

Результат:

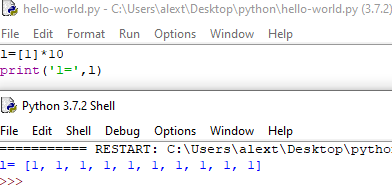
[https://sites.google.com/site/moiboarkin/_/rsrc/1547547471226/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_split_res.png](https://sites.google.com/site/moiboarkin/laboratornye-raboty/5-kurs/laboratornye-raboty-po-python/l-r-6/ex_split_res.png?attredirects=0)

**Генератори списків.**

1. **Спосіб –** складення однакових списків замінюється множенням. Список з 10 елементів заповнених одиницями

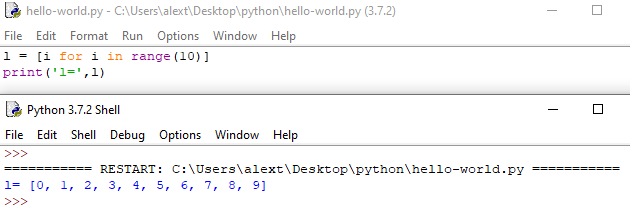
l=[l]\*10

print(‘l=’,l)



1. **Спосіб**

l = [i for i in range(10)]



**Модуль random** надає функції для генерації випадкових чисел, букв, випадкового вибору елементів послідовності.

**random.randint (A, B)** - випадкове ціле число N, A ≤ N ≤ B.

**random.random ()** - випадкове число від 0 до 1.

Випадкові числа в списку:

10 чисел, генерованих випадковим чином в діапазоні (10,80)

from random import randint

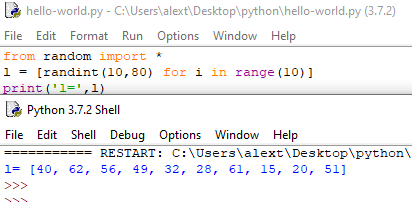
l = [randint (10,80) for x in range (10)]

10 чисел, генерованих випадковим чином в діапазоні (0,1)

l = [random () for i in range (10)].

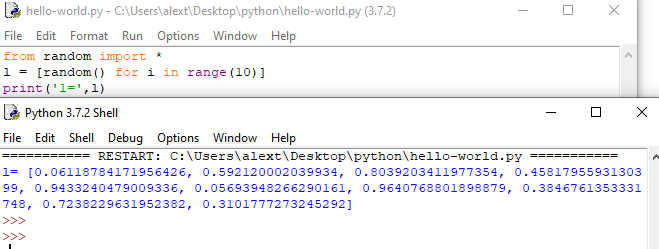
Наприклад:

Десять чисел генерованих випадковим чином у інтервалі 10-80.

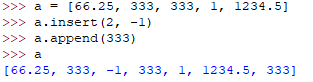


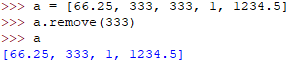
Наприклад:

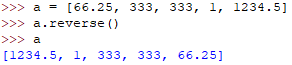
Десять чисел генерованих випадковим чином у інтервалі 0-1.

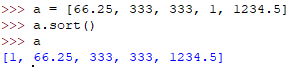


Наприклад:









**Введення списку (масиву) в мові Python.**

Для введення елементів списку використовується цикл for і команда range ():

for i in range (N):

x [i] = int (input ())

Простіший варіант введення списку:

x = [int (input ()) for i in range (N)]

Вивід цілого списку (масиву):

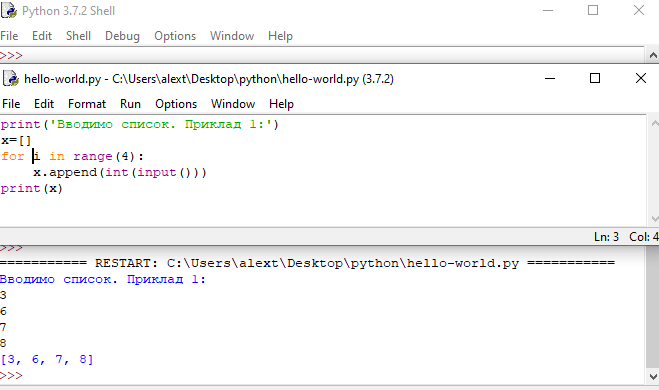
print (L)

Поелементний вивід списку :

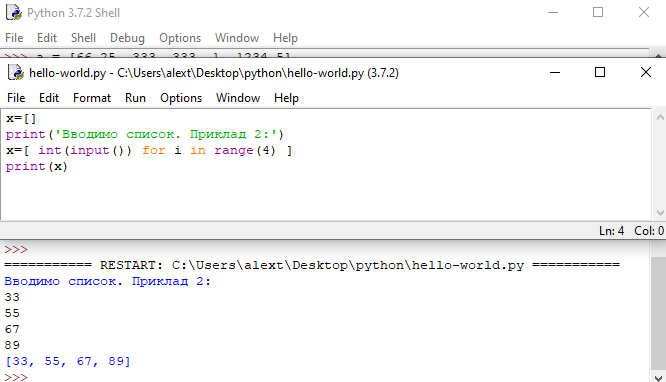
for i in range(N):

  print ( L[i], end = " " )

Наприклад:



Наприклад:



**Взяття елемента за індексом**

>>> a = [1, 3, 8, 7]  
>>> a[0]  
1  
>>> a[3]  
7  
>>> a[4]  
Traceback (most recent call last):  
File "<stdin>", line 1, in <module>  
IndexError: list index out of range

Як і в багатьох інших мовах, нумерація елементів починається з нуля. При спробі доступу до неіснуючого індексу виникає виняток IndexError. В даному прикладі змінна a була списком, однак взяти елемент за індексом можна і у інших типів: рядків, кортежів.

В Python також підтримуються негативні індекси, при цьому нумерація йде з кінця, наприклад:

>>> a = [1, 3, 8, 7]  
>>> a[-1]  
7  
>>> a[-4]

1  
>>> a[-5]  
Traceback (most recent call last):  
File "<stdin>", line 1, in <module>  
IndexError: list index out of range

**Зрізи**

В Python, крім індексів, існують ще й зрізи.

item [START: STOP: STEP] - бере зріз від номера START, до STOP (не включаючи його), з кроком STEP.

За замовчуванням START = 0, STOP = довжина об'єкта, STEP = 1. Відповідно, якісь (а можливо, і всі) параметри можуть бути опущені.

Наприклад:

>>> a = [1, 3, 8, 7]  
>>> a[:]  
[1, 3, 8, 7]  
>>> a[1:]  
[3, 8, 7]  
>>> a[:3]  
[1, 3, 8]  
>>> a[::2]  
[1, 8]

Усі ці параметри можуть бути негативними.

Наприклад:

>>> a = [1, 3, 8, 7]  
>>> a[::-1]  
[7, 8, 3, 1]  
>>> a[:-2]  
[1, 3]  
>>> a[-2::-1]  
[8, 3, 1]  
>>> a[1:4:-1]  
[]

В останньому прикладі вийшов порожній список, так як START <STOP, а STEP негативний. Те ж саме відбудеться, якщо діапазон значень виявиться за межами об'єкта:

Наприклад:

>>> a = [1, 3, 8, 7]  
>>> a[10:20]  
[]

Також за допомогою зрізів можна не тільки отримувати елементи, але і додавати і видаляти елементи (зрозуміло, тільки для змінних послідовностей).

Наприклад:

>>> a = [1, 3, 8, 7]  
>>> a[1:3] = [0, 0, 0]  
>>> a  
[1, 0, 0, 0, 7]  
>>> del a[:-3]  
>>> a  
[0, 0, 7]

**Кортеж**

Кортеж - не змінний!! Інформація захищена від змін. Має менший розмір. Використовують як словник.

Наприклад:

>>> a = (1, 2, 3, 4, 5, 6)  
>>> b = [1, 2, 3, 4, 5, 6]  
>>> a.\_\_sizeof\_\_()  
36  
>>> b.\_\_sizeof\_\_()  
44

Наприклад:

Зробимо пустий кортеж.

>>> a = tuple() *# За допомогою вбудованої функції tuple()*>>> a  
()  
>>> a = () *# За допомогою літерала кортежу*>>> a  
()  
>>>

Наприклад:

Зробимо кортеж з одного елементу:  
>>> a = ('s', )  
>>> a  
('s',)

Або

>>> a = 's',  
>>> a  
('s',)

Ну і створити кортеж з ітеріруемого об'єкта можна за допомогою все тієї ж горезвісної

**функції tuple ()**

Наприклад:

>>> a = tuple('hello, world!')  
>>> a  
('h', 'e', 'l', 'l', 'o', ',', ' ', 'w', 'o', 'r', 'l', 'd', '!')

**Операції з кортежами**

Всі операції над списками, що не змінюють список (додавання, множення на число, методи index () і count () і деякі інші операції). Можна також по-різному змінювати елементи місцями і так далі.

**Лабораторне завдання**

1. Перевірити, чи буде рядок читатися однаково справа наліво і зліва направо (тобто чи є він паліндромом). Рядок **«а роза упала на лапу азора»**

1. Спочатку введемо рядок командою: s = input ('Введіть рядок').
2. Потім визначимо логічну змінну flag і дамо їй значення 1: flag = 1.
3. Для початку в введеному рядку потрібно видалити пробіли. Для цього скористаємося циклічної конструкцією for, яка виконається стільки раз, скільки має рядок символів (довжину рядка). Довжину рядка визначимо функцією len (s).
4. У тілі циклу будемо перевіряти таку умову: s [i]! = ''. Даний логічний вираз буде істинним в тому випадку, якщо i-ий елемент рядку не буде дорівнює пробілу, тоді виконається команда після двокрапки: string + = s [i]. До рядку string, який був оголошений на початку програми, буде додаватися посимвольний рядок s, але вже без пробілів.
5. Для перевірки рядка на "паліндром" скористаємося циклічної конструкцією for.
6. Довжину половини рядка знайдемо діленням остачі на 2. Якщо кількість символів непарна, то символ, що стоїть в середині не враховується, тому що його порівнювана пара - він сам.
7. Кількість повторів циклу дорівнює довжині половини рядка. Довжину рядка визначимо функцією len (s), де аргумент введена нами рядок s. Знаючи довжину рядка, можна обчислити кількість повторів циклу. Для цього цілочисленно поділимо довжину рядка на 2: len (s) // 2.
8. Для завдання діапазону для циклу використовуємо функцію range (), в якій аргументом буде половина довжини рядка: range (len (s // 2)).

for i in range (len (s // 2)).

1. Якщо символ з індексом i не дорівнює "симетричному" символу з кінця рядка (який знаходиться шляхом індексації з кінця)

if s [i]! = s [-1-i], то змінній flag присвоюється значення 0 і відбувається вихід з циклу командою break.

1. Далі, за допомогою умовної конструкції if-else в залежності від значення flag або - 0, або -1 виводиться повідомлення, що рядок паліндром, або ні.

2.Перетворення рядка до верхнього регістру**.**

1. Перетворення рядка до нижнього регістру.
2. Переведіть першу букву кожного слова в верхній регістр, а всі інші в нижній.
3. У рядку замінити букву (а) буквою (о). Підрахувати кількість замін. Підрахувати, скільки символів в рядку.

№ Варіант - остання цифра у списку групи.

|  |  |
| --- | --- |
| **№** | **Рядок** |
| **1** | любите отдыхать на природе |
| **2** | палатках не для вас |
| **3** | заполненные отели на курортах утомляют |
| **4** | украинский стартап предлагает новый формат отдыха |
| **5** | речь идет о геодезических куполах |
| **6** | альтернатива гостиничным номерам |
| **7** | идея стартапа родилась несколько лет назад |
| **8** | компактные сферические домики на природе |
| **9** | купольные конструкции на склоне горы в окружении гор |
| **0** | невероятная близость к природе |

1. З масиву **Х** довжиною 10+№ (де №-остання цифра студента у списку групи), серед елементів якого є позитивні та негативні та нуль, сформуйте новий масив **Y**, узявши тільки елементи з **Х** які більше по модулю заданого числа **М**. Виведіть на екран число **М** та масиви заданий та отриманий.
2. У масиві цілих чисел, довжиною 10+№ (де №-остання цифра студента у списку групи), усі негативні елементи замініть позитивними. Виведіть заданий та отриманий масиви на екран.
3. Дано одномірний масив, який має 10+№ цілих елементів. Введіть масив з клавіатури. Знайдіть максимальний елемент. Виведіть на екран масив у зворотньому порядку.
4. Нехай журнал по предмету «Інформаційні технології» представлено у вигляді списку: my\_len = [['IT-31’, ['Акулова Алена', 'Бабушкіна Ксенія ', .......]], [' IT-32 ', [... ..]], [' IT-33 ', [....]]]. Виведіть список студентів конкретної групи построчно у вигляді:

<Назва групи>

<ПІБ>

<ПІБ>

1. Через функцію range розрахувати суму непарних чисел діапазону від а до с, початкове та кінцеве значення ввести через вікно. Вивести суму на екран.
2. Знайдіть добуток цифр у тризначному числі, яке введено користувачем.
3. Дані натуральні числа від 35 до 87. Програма має вивести числа, які при діленні на 7 дають остачу 1,2,5 .
4. У масиві у порядку зменшення є купюри [1000, 500, 200, 100, 50, 20, 10, 5, 2, 1]. Реалізовати видачу суми, введеної з екрану мінімальною кількістю купюр.
5. Дано два масива. Зробити третій масив з двох цих масивів. Розташування у масиві у порядку зростання.