

## ПРАКТИЧНА РАБОТА 1

### РОБОТА ЗІ СПИСКАМИ ТА КОРТЕЖАМИ В PYTHON

Список - це упорядкований набір об'єктів, що зберігаються в одній змінній. На відміну від масивів в інших мовах, у списків немає ніяких обмежень на тип змінних, тому в них можуть зберігатися різні об'єкти, в тому числі і інші колекції.


Списки:

- впорядкований **змінюваний** набір об'єктів;
- підтримують індекси і зрізи;
- підтримують ітерацію;
- мають вбудовані функції і методи.

Кортеж - по суті це незмінний список, який можна хешувати, а значить використовувати в якості ключа в словниках.

Кортежі:

- впорядкований **незмінний** набір об'єктів;
- схожі на списки з поправкою на незмінюваність;
- хешуємі.

Якщо не встановлено Python 3.6 та среда розробки IDE (наприклад Anaconda, PyCharm Community Edition), в браузері перейдіть на сайт <https://edube.org/sandbox>, вибравши в налаштуванні через  емулятор Python (рис.1).

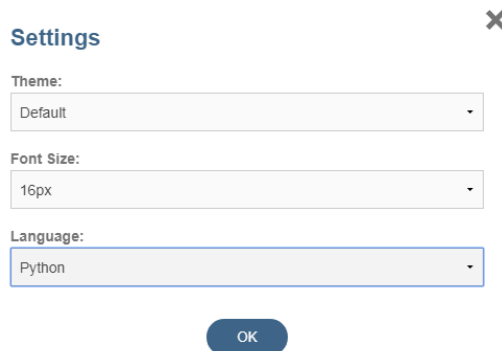


Рисунок 1 –Налаштування в sandbox

Для введення значень з клавіатури використовуйте функцію `input()` з параметрами. Наприклад: `n=int(input("Enter chislo"))`.

## ЧАСТИНА 1. РОБОТА ЗІ СПИСКАМИ

Ознайомтесь з методами списків в документації <https://docs.python.org/3/tutorial/datastructures.html> або `help(list)`.

### 1. Створення списків.

Списки можна створювати за допомогою квадратних дужок ( `[]` ), або за допомогою функції `list()`.

`empty_list1 = []` створення списку за допомогою `[]`

<code>print(empty_list1)</code>	<code>[]</code>
<code>print(type(empty_list1))</code>	<code>&lt;class 'list'&gt;</code>
<code>empty_list2 = list()</code>	створення списку за допомогою літерала <code>list</code>
<code>print(empty_list2)</code>	<code>[]</code>
<code>list3 = [1,2,'three', 'four']</code>	створення списку з заздалегідь завданим набором різних даних
<code>print(list3)</code>	<code>[1, 2, 'three', 'four']</code>
<code>list4 = [5]*10</code>	створення списку з однакових значень за допомогою множення
<code>print(list4)</code>	<code>[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]</code>
<code>print(len(list4))</code>	10 (визначення довжини списку)

## 2. Створення копії списків.

Очистіть вікно редактора та створіть копії списків першим способом, коли копіюється ім'я масиву, а не його вміст.

Перший спосіб:

<code>a=[1,2,3,4]</code>	
<code>b=a</code>	В цьому випадку копіює ім'я масиву, а не його вміст. Фактично два імені (a і b) ідентифікують одне й те місце розташування в пам'яті комп'ютера.
<code>print("a=", a)</code>	<code>a= [1, 2, 3, 4]</code>
<code>print("b=", b)</code>	<code>b= [1, 2, 3, 4]</code>
<code>a[1]=5</code>	Зміна a впливає b
<code>print("Posle izmenenija a[1]")</code>	<code>Posle izmenenija a[1]</code>
<code>print("a=", a)</code>	<code>a= [1, 5, 3, 4]</code>
<code>print("b=", b)</code>	<code>b= [1, 5, 3, 4]</code>
<code>b[3]=10</code>	і навпаки
<code>print("Posle izmenenija b[3]")</code>	<code>Posle izmenenija b[3]</code>
<code>print("a=", a)</code>	<code>a= [1, 5, 3, 10]</code>
<code>print("b=", b)</code>	<code>b= [1, 5, 3, 10]</code>

Очистіть вікно редактора та створіть копії списків другим способом, коли копіюється вміст списку, а не його ім'я.

Другий спосіб:

<code>a=[1,2,3,4]</code>	
<code>b=a[:]</code>	Копіювання вмісту списку, а не його імені.
<code>print("a=", a)</code>	<code>a= [1, 2, 3, 4]</code>
<code>print("b=", b)</code>	<code>b= [1, 2, 3, 4]</code>
<code>a[1]=5</code>	Зміна a не впливає на b
<code>print("Posle izmenenija a[1]")</code>	<code>Posle izmenenija a[1]</code>
<code>print("a=", a)</code>	<code>a= [1, 5, 3, 4]</code>
<code>print("b=", b)</code>	<code>b= [1, 2, 3, 4]</code>
<code>b[3]=10</code>	і навпаки

print("Posle izmenenija b[3]")	Posle izmenenija b[3]
print("a=", a)	a= [1, 5, 3, 4]
print("b=", b)	b= [1, 2, 3, 10]

### 3. Додавання і вилучення елементів списку

Очистіть вікно редактора та виконайте додавання і вилучення елементів списку.

Списки, на відміну від рядків, є змінною структурою даних, а значить ми можемо додавати елементи в існуючий список.

Додавання елемента до списку здійснюється за допомогою методу `append()`.

list = []	
list.append( 3 )	
list.append( "hello" )	
print (list)	[3, 'hello']

Якщо необхідно розширити список іншим списком, використовується метод `extend`, який додає його в кінець вашого списку.

list.extend(['Yes', 5, 3])	
print(list)	[3, 'hello', 'Yes', 5, 3]

Також можна використовувати перевантажений оператор `+`, який додає в кінець списку.

list+= [None]	
print(list)	[3, 'hello', 'Yes', 5, 3, None]

Для видалення елемента зі списку, в разі, якщо ви знаєте його значення, використовуйте метод `remove (x)`, при цьому буде видалено перше посилання на даний елемент.

list.remove(3)	
print (list)	['hello', 'Yes', 5, 3, None]

Якщо видалення за індексом, використовуйте метод `del`.

del list[2]	
print(list)	['hello', 'Yes', 3, None]

### 4. Індекси і зрізи

Очистіть вікно редактора.

Щоб звернутися до конкретного елементу списку, ми використовуємо той же механізм, що і для рядків. Нумерація елементів починається з нуля. Можливо використовувати від'ємні індекси в зрізах. Уважно вивчить результати приведених команд.

range_list = [0,1,2,3,4,5,6,7,8,9]	
print(range_list [0])	0
print(range_list [-1])	9
range_list[10]	IndexError: list index out of range
print(range_list[1:3])	[1, 2]
print(range_list[1:-1])	[1, 2, 3, 4, 5, 6, 7, 8]
print(range_list[-1:1])	[]

<code>print(range_list[3:])</code>	<code>[3, 4, 5, 6, 7, 8, 9]</code>
<code>print(range_list[:5])</code>	<code>[0, 1, 2, 3, 4]</code>
<code>print(range_list[::2])</code>	<code>[0, 2, 4, 6, 8]</code>
<code>print(range_list[::-1])</code>	<code>[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]</code>
<code>print(range_list[1::2])</code>	<code>[1, 3, 5, 7, 9]</code>
<code>print(range_list[5:1:-1])</code>	<code>[5, 4, 3, 2]</code>

Майте на увазі, що при отриманні зрізу ми отримуємо новий список.

```
print(range_list[:] is range_list)    False
```

Раніше описана команда `del` може видаляти більше, ніж просто елемент списку за один раз. Також можна видалити зрізи. В цьому випадку зріз не створює ніякого нового списку!

```
del range_list[1:3]
print(range_list)                [0, 3, 4, 5, 6, 7, 8, 9]
```

Можливо також вилучення всіх елементів відразу. Список стає порожнім.

```
del range_list[:]
print(range_list)                []
```

Видалення списку, а не його вмісту.

```
del range_list
print(range_list)                NameError: name 'range_list' is not defined
```

## 5. Оператор `in`

Очистіть вікно редактора.

Python пропонує нам два дуже потужних оператора, здатних переглядати список, щоб перевірити, чи зберігається конкретне значення в списку чи ні.

Перший з них (`in`) перевіряє, чи зберігається цей елемент (його лівий аргумент) десь всередині списку (правий аргумент) - оператор повертає `True` в цьому випадку.

Другий (`not in`) перевіряє, чи відсутній даний елемент (його лівий аргумент) в списку. Оператор повертає `True` в цьому випадку.

```
list = [0, 12, 5, 9, 10]
print(15 in list)                False
print(15 not in list)            True
print(12 in list)                True
```

## 6. Ітерація

Очистіть вікно редактора.

Списки як і рядки підтримують протокол ітерації. Значить, можемо використовувати цикл `for` для того, щоб ітеруватися за елементами списку. Зверніть увагу, що ітерації відбуваються саме за елементами списку, а не за індексами, як у багатьох мовах.

```
list = [0, 10, 20, 30]
sum = 0
for e_list in list:
```

```

sum += e_list
print(e_list)

0
10
20
30

print('sum={}'.format(sum))    sum=60

```

Щоб виводити не тільки елемент списку, а також його індекс, існує вбудована функція `enumerate`, яка повертає індекс і поточний елемент.

```

list = [0, 10, 20, 30]
for i, e_list in enumerate(list):
    print('#{} {}'.format(i, e_list))

#0 0
#1 10
#2 20
#3 30

```

## 7. Методи списків

<code>list.append(x)</code>	Додає елемент в кінець списку
<code>list.extend(L)</code>	Розширює існуючий список за рахунок додавання всіх елементів зі списку <i>L</i> .
<code>list.insert(i, x)</code>	Вставити елемент <i>x</i> в позицію <i>i</i> . Перший аргумент - індекс елемента після якого буде вставлений елемент <i>x</i> .
<code>list.remove(x)</code>	Видаляє перше входження елемента <i>x</i> зі списку.
<code>list.pop([i])</code>	Видаляє елемент з позиції <i>i</i> і повертає його. Якщо використовувати метод без аргументу, то буде видалений останній елемент зі списку.
<code>list.clear()</code>	Видаляє всі елементи зі списку.
<code>list.index(x[, start[, end]])</code>	Повертає індекс елемента.
<code>list.count(x)</code>	Повертає кількість входжень елемента <i>x</i> в список.
<code>list.sort(key=None, reverse=False)</code>	Сортує елементи в списку по зростанню. Для сортування в зворотному порядку використовуйте прапор <i>reverse = True</i> . Додаткові можливості відкриває параметр <i>key</i> , за більш детальною інформацією зверніться до документації.
<code>list.reverse()</code>	Змінює порядок розташування елементів у списку на зворотний.
<code>list.copy()</code>	Повертає копію списку.

## ЧАСТИНА 2. РОБОТА З КОРТЕЖАМИ

Ознайомтесь з методами списків в документації <https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences> або `help(tuple)`.

### 1. Створення кортежів.

Кортежі можна створювати за допомогою круглих скобок ( `()` ), або за допомогою функції `tuple()`.

<code>empty_tuple1= ()</code>	створення кортежу за допомогою <code>()</code>
<code>print(empty_tuple1)</code>	<code>()</code>
<code>print(type(empty_tuple1))</code>	<code>&lt;class 'tuple'&gt;</code>
<code>empty_tuple2= tuple()</code>	створення кортежу за допомогою літерала <code>tuple</code>
<code>print(empty_tuple2)</code>	<code>()</code>
<code>tuple3 = (1,2,'three', 'four')</code>	створення кортежу з заздалегідь заданим набором різних даних
<code>print(tuple3)</code>	<code>(1, 2, 'three', 'four')</code>
<code>tables = (int, str, tuple)</code>	
<code>print(tables)</code>	<code>(&lt;class 'int'&gt;, &lt;class 'str'&gt;, &lt;class 'tuple'&gt;)</code>
<code>print(type(tables))</code>	<code>&lt;class 'tuple'&gt;</code>
<code>x,y,z=(1,2),(3,4),(5,6)</code>	
<code>print(x)</code>	<code>(1, 2)</code>

Будьте уважніше при визначенні кортежу з одного елемента, не забувайте писати кому, тому що якщо ви забудете про неї, то Python вважатиме вашу змінну типом `int`.

<code>one_element_tuple = (1,)</code>	
<code>guess_what = (1)</code>	
<code>print(type(one_element_tuple))</code>	<code>&lt;class 'tuple'&gt;</code>
<code>print(type(guess_what))</code>	<code>&lt;class 'int'&gt;</code>

Треба зазначити, що кортежі в пам'яті займають менший об'єм в порівнянні зі списками. Це дає приріст продуктивності, який пов'язаний з тим, що кортежі працюють швидше.

<code>lst = [10, 20, 30]</code>	
<code>tpl = (10, 20, 30)</code>	
<code>print (lst.__sizeof__())</code>	<code>64</code>
<code>print (tpl.__sizeof__())</code>	<code>48</code>

Також важлива особливість кортежів - у них є функція `hash`, і тому вони можуть використовуватися в якості ключів в словниках.

<code>print(hash(tuple()))</code>	<code>3527539</code>
-----------------------------------	----------------------

Все, що говорилося про списках, відноситься і до кортежів, за винятком того, що кортеж - це список, доступний тільки для читання. Тому всі функції і методи, що не міняють вміст, можна застосовувати і до кортежів.

## 2. Доступ до елементів кортежу.

Очистіть вікно редактора.

Доступ до елементів кортежу здійснюється також, як до елементів списку – через індекс. Але змінювати елементи кортежу не можна!

<code>a = ( 1 , 2 , 3 , 4 , 5 )</code>	
<code>print(a[0])</code>	<code>1</code>
<code>print(a[1:3])</code>	<code>(2, 3)</code>

```
a[1] = 3
```

```
TypeError: 'tuple' object does not support  
item assignment
```

Але незважаючи на те, що самі кортежі незмінні, об'єкти всередині них можуть бути змінними. Наприклад, якщо у нас кортеж містить список, ми можемо додавати елементи в цей список. Наприклад:

```
b = ([], [1, 2])
```

```
print(b)
```

```
([], [1, 2])
```

```
b[0].append(0)
```

```
b[1].append(3)
```

```
print(b)
```

```
([0], [1, 2, 3])
```

### 3. Видалення кортежів

Видалити окремі елементи з кортежу неможливо, але можна видалити кортеж цілком.

```
a = ( 1 , 2 , 3 , 4 , 5 )
```

```
del a[0]
```

```
TypeError: 'tuple' object doesn't support item  
deletion
```

```
del a
```

```
print (a)
```

```
NameError: name 'a' is not defined
```

### 4. Перетворення кортежу в список і навпаки

На базі кортежу можна створити список. Вірно і зворотне твердження: для перетворення списку в кортеж досить передати його в якості аргументу функції `tuple()`.

```
lst = [ 1 , 2 , 3 , 4 , 5 ]
```

```
print ( type (lst))
```

```
<class 'list' >
```

```
print (lst)
```

```
[1, 2, 3, 4, 5]
```

```
tpl = tuple(lst)
```

```
print ( type (tpl))
```

```
<class 'tuple'>
```

```
print (tpl)
```

```
(1, 2, 3, 4, 5)
```

Обернена операція.

```
tpl = ( 2 , 4 , 6 , 8 , 10 )
```

```
print ( type (tpl))
```

```
<class 'tuple'>
```

```
print (tpl)
```

```
(2, 4, 6, 8, 10)
```

```
lst = list (tpl)
```

```
print ( type (lst))
```

```
<class 'list'>
```

```
print (lst)
```

```
[2, 4, 6, 8, 10]
```

### Завдання для самостійного виконання.

1.	Визначити індекси елементів списку, значення яких не менше заданого мінімуму і не більше заданого максимуму. Досліджуваний масив (список в Python) заповнюється випадковими числами в діапазоні від 0 до 99 (включно) і складається з 50 елементів. Мінімум і максимум задається користувачем. Результат надати у вигляді нового списку.
----	--

2.	Видаляти зі списку, що складається з 20 чисел в діапазоні від 0 до 100, всі числа, які більше <code>min</code> і менше <code>max</code> . Мінімум і максимум задається користувачем. При цьому видаляемі числа зберігаються в іншому списку.
3.	У списку, що складається з позитивних і негативних цілих чисел, знайти перший, третій і шостий позитивні елементи і обчислити їх добуток.
4.	Дан список-масив, заповнений випадковим чином нулями і одиницями. Знайти найдовшу безперервну послідовність одиниць і визначити індекси першого і останнього елементів в ній.
5.	Створіть список, в який занесіть марки автомобілів на парковці. Підрахунок кількість автомобілів якоїсь марки. Відсортувати список.
6.	Дані відомості про місячну температуру повітря. Визначити день з найменшою або найбільшою температурою. Знайти дні з однаковою температурою і зберегти їх в іншому списку.
7.	Додавати або вилучати число (не додавати число до списку, якщо воно там вже є). Перевірити, чи упорядкований список за спаданням.
8.	Знайти слова в списку, що повторюються, та вилучити їх.
9.	Дан список позитивних і негативних дійсних чисел. Отримати з цього списку інший список, що складається тільки з позитивних елементів першого, що стоять на парних місцях.
10.	Записати в масив <code>u(20)</code> куби парних чисел і квадрати непарних чисел, вказаних у цілочисельному масиві <code>n(20)</code> . В одержаному масиві поміняти місцями максимальний і мінімальний елементи.