I. Змінні та пам’ять.

1. Створити змінні, що посилаються на два цілих числа, що однакові за значенням, де значення належить проміжку від **-5** до **256.** Перевірте, чи будуть ці змінні рівні тільки за значенням або ж ще будуть посилатися на один і той самий об’єкт в пам’яті**?** Наведіть код та дайте текстову відповідь нижче**.**

In [22]:

```
x = 27
y = 27
print(id(x))
print(id(y))
print(id(x) == id(y))
```

```
2881939926128
2881939926128
True
```

Python під час запуску створює об'єкти для чисел від **-5** до **256,** в подальшому змінні з відповідними значеннями лише посилаються на ці об'єкти**.**

1. За допомогою якої функції можна перевірити належність змінної до вказаного типу даних **(**напр**.** чи змінна **True** посилається на значення булевого та цілочисленого типу**)?**

In [23]:

```
print(isinstance(True, bool))
print(isinstance(False, int))
```

```
True
True
```

II. Цілі числа та числа з рухомою комою

1. Створити дві змінні, що посилаються на будь-які ціле число та число з рухомою комою та продемонструвати такі арифметичні операції**:** додавання, віднімання, ділення, множення, ділення без залишку, ділення по модулю, приведення до ступеню**.** Всі результати операцій вивести на екран**.**

In [24]:

```
pi = 3.14
n = 3
print(pi + n)
print(pi - n)
print(pi * n)
print(pi / n)
print(pi % n)
print(pi // n)
print(pi ** n)
```

```
6.140000000000001
0.14000000000000012
9.42
1.0466666666666666
0.14000000000000012
1.0
30.959144000000002
```

1. Використовуючи змінні з вправи **3,** продемонструйте механізм явного перетворення типів, де числа з

рухомою комою перетворюються на цілі числа. Також визначте змінну, що посилається на значення булевого типу і спробуйте явно привести її до цілого числа.

In [25]:

```
print(int(pi))
boolean = True
print(int(boolean))
```

```
3
1
```

III. Рядки (String).

1. Створити пустий рядок двома різними способами.

In [26]:

```
str = ""
str2 = ''
```

1. Створити рядок з апострофом. Зробити його сирим. Вивести обидва рядка на екран.

In [27]:

```
apostrophe = "\'"
apostrophe2 = r"\'"
print(apostrophe)
print(apostrophe2)
```

```
'
\'
```

1. Створити змінну, що буде посилатися на Ваше прізвище латинкою. Створити форматований рядок, який буде мати вигляд **"My surname is __"**, де на місці нижніх підкреслень буде Ваше прізвище зі змінної.

In [28]:

```
surname = "Kolomiichuk"
print(f"My surname is {surname}")
```

```
My surname is Kolomiichuk
```

1. Маючи рядок **"My dog is crazy."** перетворити його на список **["my", "dog", "is", "crazy"]**

In [29]:

```
line = "My dog is crazy."
split = line[:-1].lower().split()
print(split)
```

```
['my', 'dog', 'is', 'crazy']
```

IV. Робота зі списками.

1. Створити список двома різними за синтаксисом способами. За допомогою вбудованої функції обчисліть довжину одного з них.

In [30]:

```
listA = list("Kolomiichuk")
```

```
listB = [1,2,3,4,5]
print(listA.__len__())
```

```
11
```

1. Створіть два списка та за допомогою спеціального методу додайте другий з них в якості останнього елемента першого.

In [31]:

```
listX = [1, 2, 3, 4, 5]
listY = [i for i in range(7, 10)]
listX.append(listY)
print(listX)
```

```
[1, 2, 3, 4, 5, [7, 8, 9]]
```

1. Створіть список, де елементами цього списку також є списки. Отримай перший елемент з останнього рядка та виведи значення на екран.

In [32]:

```
listList = [listX, listY, listB]
print(listList[-1][0])
```

```
1
```

1. Створіть список з десяти елементів різного типу. Отримайте всі елементи, окрім двох перших та двох останніх та збережіть їх в новій змінній.

In [33]:

```
listK = [1, True, "Kolomiichuk", 3.14, [1, 2, 3], (1, 2, 3), [[1,2], False], {1, 2, 3}, {1: "one", 2: "two", 3: "three"}, None]
print(listK[2:-2])
```

```
['Kolomiichuk', 3.14, [1, 2, 3], (1, 2, 3), [[1, 2], False], {1, 2, 3}]
```

V. Робота з кортежами.

1. Створити кортеж з один елементом.

In [34]:

```
tupleA = (0,)
```

1. Порівняйте список та кортеж. Назвіть схожості та відмінності, випадки використання.

Списки - **mutable**, кортежі - **immutable**. Списки "працюють" повільніше, але мають більше можливостей для роботи з даними. Кортежі моуть бути ключами словника.

1. Створіть кортеж з **11**ти елементів чисел з рухомою комою та отримайте кожен парний за індексом елемент в зворотньому порядку. Наприклад, маючи **(1.2, 2.3, 3.3, 4.3, 5.3, 6.3, 7.3, 8.3, 9.3, 0.3)**, отримати **(0.3, 8.3, 6.3, 4.3, 2.3)**. Результат збережіть в нову змінну та виведіть на екран.

In [35]:

```
floats =(1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10.10, 11.11)
floats2 = floats[::2][::-1]
print(floats2)
```

```
(11.11, 9.9, 7.7, 5.5, 3.3, 1.1)
```

VI. Множини (Set).

1. Створити множину без елементів. Після цього за допомогою методу додайте кілька різних елементів до множини. Чи множини є змінним типом даних?

In [36]:

```
setA = set()
setA.add(1)
setA.add("true")
setA.add(False)
setA.add(4)
setA.add(1)
setA.add("true")
print(setA)
```

```
{False, 1, 'true', 4}
```

Множини - це змінний тип даних.

1. Створити множину, маючи список **my_list = [1, 1, 2, 67, 67, 8, 9]**. Пояснити, чому "зникли" деякі елементи.

In [37]:

```
my_list = [1, 1, 2, 67, 67, 8, 9]
my_set = set(my_list)
print(my_set)
```

```
{1, 2, 67, 8, 9}
```

Зникли елементи з однаковим значенням та, відповідно, однаковими хешами.

1. Створіть дві множини. Продемонстуйте над ними операції: об'єднання, різниці, пересічі та симетричної різниці. Використовуйте методи, що не змінюють множини, а створюють нові.

In [38]:

```
setB = {1, 2, 3, 4, 5}
print(setA.union(setB))
print(setA.intersection(setB))
print(setA.difference(setB))
print(setA.symmetric_difference(setB))
```

```
{False, 1, 2, 3, 4, 5, 'true'}
{1, 4}
{False, 'true'}
{False, 2, 3, 5, 'true'}
```

VII. Словники (Dictionary).

1. Створіть пустий словник. До нього додайте чотири пари елементів такі, щоб їхні ключі були різних типів. Чи може список бути ключем? Чому?

In [39]:

```
dictionary = dict()
dictionary["one"] = 1
dictionary[2] = "two"
dictionary[(34, False)] = [1, 2, 3]
```

```
dictionary[2.22] = {1, 2, 3}
print(dictionary)
```

```
{'one': 1, 2: 'two', (34, False): [1, 2, 3], 2.22: {1, 2, 3}}
```

Список не може бути ключем, оскільки ключі повинні бути **immutable**. У разі використання списків та їх зміни, поведінка словника буде некоректною через фактичну зміну ключа, але не даних в хеш-таблиці.

1. Створіть словник, де значенням в одній з пар теж буде словник, який теж має вкладений словник. Виведіть на екран значення, що міститься в словнику, що знаходиться на найнижчому рівні ієрархії вкладеності (найбільш внутрішній).

In [40]:

```
dictdictdict = {"one": 1, 2: {"three" : 3, "four" : {5: 5, 6: 6}}}
print((dictdictdict[2]["four"][5], dictdictdict[2]["four"][6]))
```

```
(5, 6)
```

Вітаю! Ви велика(ий) молодець, що впоралась(вся). Похваліть себе та побалуйте чимось приємним. Я Вами пишаюся.