**cd C:\Users\hrli1\Desktop\**

**Jupyter notebook**  opens jupyter notebook

A sequence is an ordered arrangement of items. To represent a sequence in Python, we actually have several types of objects like list, arrays, tuples

**Elements inside a list can be mixed type**

**[]** creates a list

**[1, 2, 1, 2]** creates a list of 1 2 1 2

**list(“hello”)** creates a list of ['h', 'e', 'l', 'l', 'o']

**first\_list = [1,2,3,4,5]**

**second\_list = [6, 7, 8, 9]**

**3 in first\_list** checks if 3 is in the first list and returns a value True or False

**3 not in first\_list** checks if 3 is not in the first list and returns a value True or False

**third\_list = first\_list + second\_list** will create a new list with values [1, 2, 3, 4, 5, 6, 7, 8, 9]

**third\_list[2]** will gives 3

**third\_list[0:5]** will gives [1, 2, 3, 4, 5]

**third\_list[1::3]** will gives [2, 5, 8]

**third\_list[:5]** will gives [1,2,3,4, 5]

**print(len(third\_list))** will give the length of the list

**print(max(third\_list))** will give the maximum of the list

**print(min(third\_list))** will give the minimum of the list

**my\_list = [1,2,2,3,3,4,3]**

**my\_list.index(3)** will locate the first 3 which is 3, remember 0, 1, 2, 3.

**my\_list.count(2)** will count how many 2s in the list, this will give 2

**“Eggs”.count(“g”)** will count how many gs are in the string Eggs and will give 2

List is the first object we come across that is mutable, which means we can change the data inside list. Unlike int or float, we can only create a new int and assign a value to it.

**x = []** create an empty list named x,

**print(x)** this will []

**x.append(5)** this will append a into x

**x** now this will gives [5]

**x[0] = 10** this will assigne 10 to the first data in x which was 5

we can keep adding values in the list by **x.append(7)** which will give x [5,7] or we can do **x.extend([12,13])** to extend the list

**x[0:2] = [15,16]** will change the first and second elements in the list to be 15 and 16

**del x[1:3]** will delete the from the second count three elements in the list and will give x =[15]

**x = [1,2,3,4,5] y = [10,12] x[1:5:2] = y** gives x = [1,10,3,12,5] which means starting from the second elements to the last one in x replace the value by y and the step size is 2

**x.clear()** will clear the list which returns an empty list

**x.insert(0,12)** will assign 12 to index 0 and will return x = [12]

**x.insert(0,15)** will assign 15 to index 0 and will return x = [15, 12]

**x.insert(1,500)** will assign 500 to index 1 and will return x = [15, 500, 12]

**last\_value = x.pop()** this will take the last value from the list and assign it to the variable last\_value **and clear that value in the list**

you can also give pop an index to pop, e.g. **last\_value = x.pop(0)** this will take the first value from the list and assign it to the variable last\_value **and clear that value in the list**

**x = [500, 600, 700, 800, 50, 500]**

**x.remove(500)** this will only remove the first 500 in the list

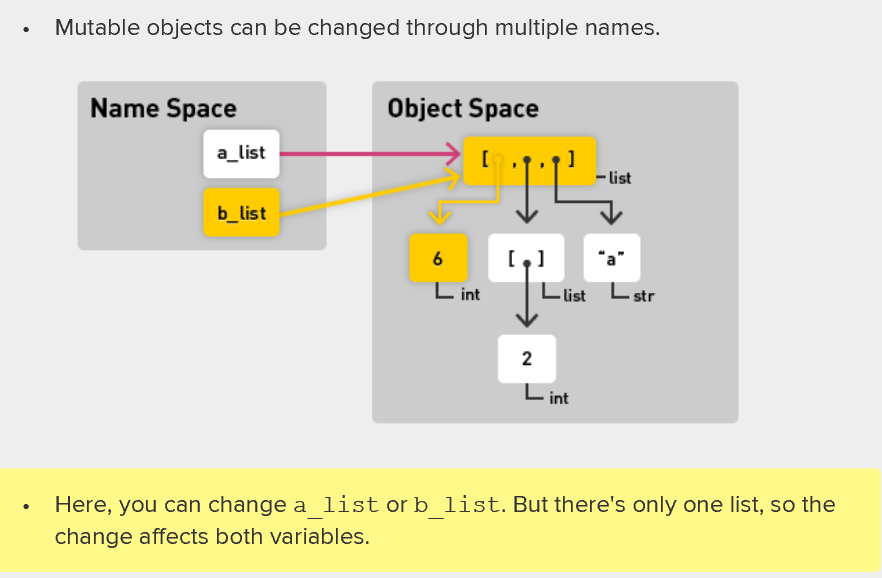
**x.sort()** this will arrange the list in order from small to large

**x.reverse()** will arrange the list in order from large to small

It is tempting to think of lists as containing objects, like a mailbox contains letters, but the analogy is misleading. **Instead, a list has references to its items not containing the objects. These are pointers to the correct memory locations. Similar to the wat that a variable points to a specific object.**

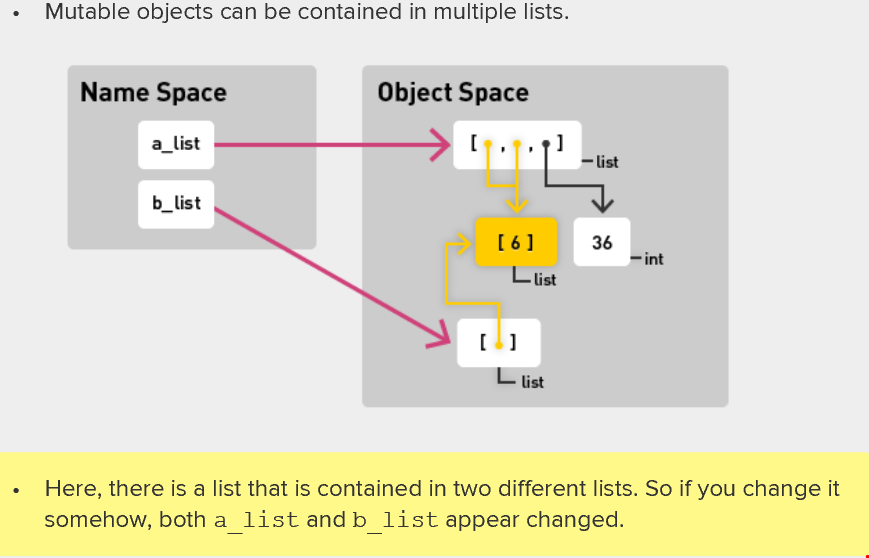
Warning One:

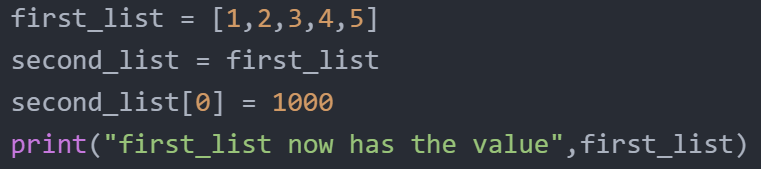
If A\_list and B\_list point to the same list, and you change an element in A\_list, the corresponding one in B\_list will also be changed, this is due to the mutable!



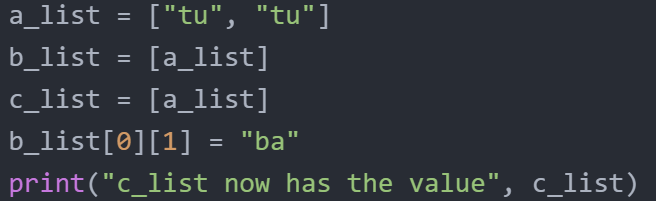
Warning Two:

Mutable objects can be contained in multiple lists

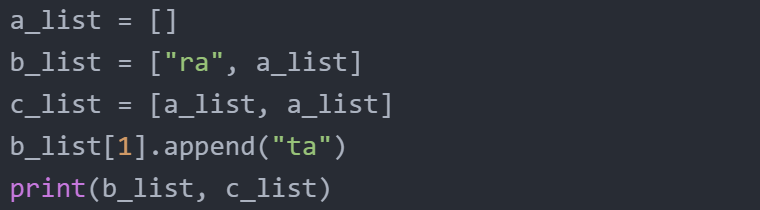




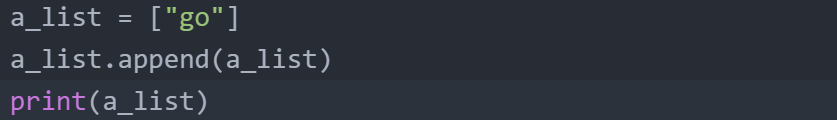




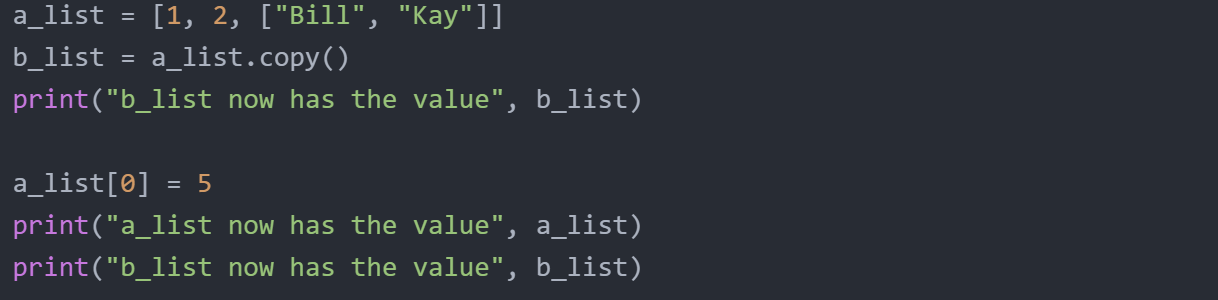


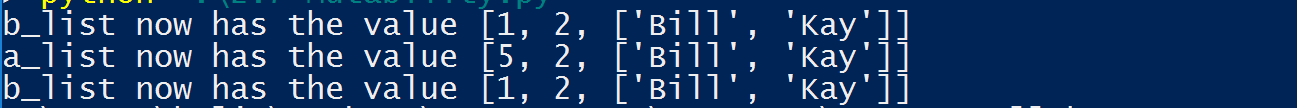


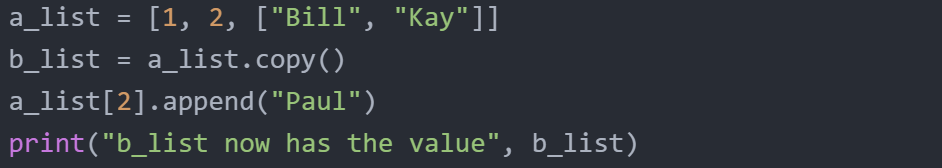




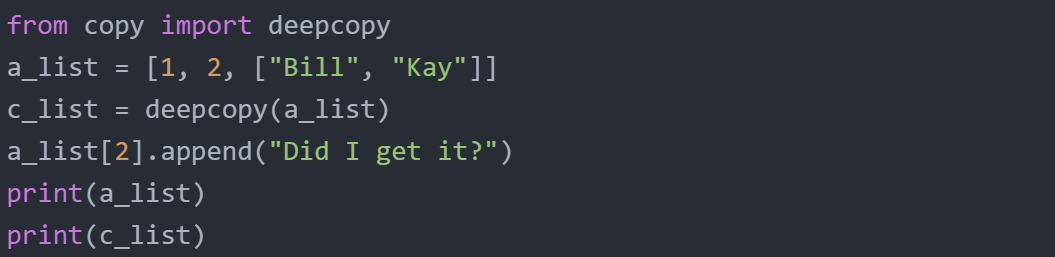














Tuples

Tuples are like list but immutable.

**X = (1,2,3)** defines a tuple with value 1, 2, 3

**Max(x)** maximum of a tuples

**Len(x)** the length of a type

**X[0]** the index , the first element in the tuple

**X[0::2]** returns (1,3)

**Y = X + (10,11,12)** create a new tuple Y with value (1,2,3,10,11,12)

We can also change a list into a tuple using **a\_list = [1,2,3] tuple(a\_list)** or similarly **list(x)**

**X = “goodbye” ; X[0:4] ; list(x)** gives [‘g’, ’o’, ’o’, ’d’, ’b’, ’y’, ‘e’]

**low, high = 10, 20** this creates two variables named low and high with value 10 and 20 respectively

**print(low, high)** will gives 10 and 20

Ranges

Create a range using **x = range (10**). This means a range from 0 to 10. If we use the list command like **list(x)** we will get [0,1,2,3,4,5,6,7,8,9]

**start = 0, stop = 10, step =1**

**print(list(range(start, stop, step)))**

THIS WILL GIVE [0,1,2,3,4,5,6,7,8,9]

**start = 0, stop = 10, step =5**

**print(list(range(start, stop, step)))**

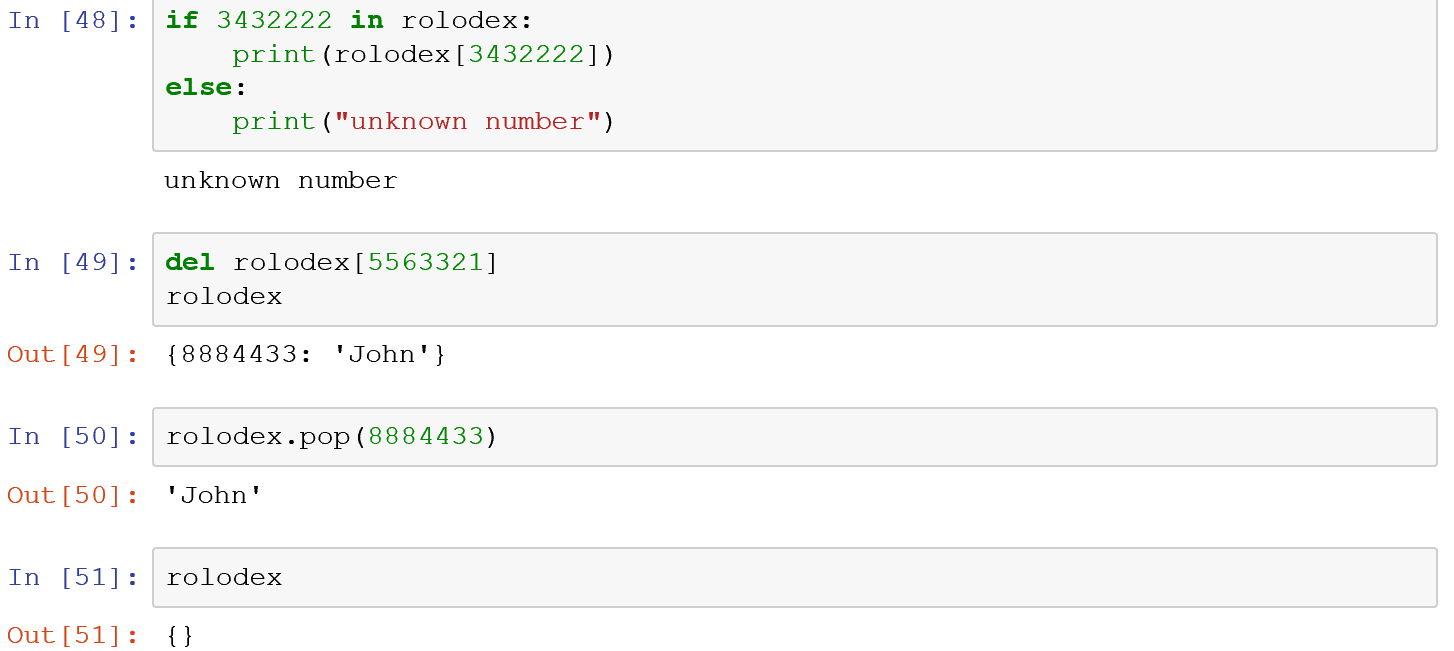
THIS WILL GIVE [0,5]

**list(range(start, stop, step))**; **5 in x** will return true.

A dictionary is NOT a SEQUENCE !!!!!

For a dictionary, the values can be at any type but the keys have to be immutable!

Dictionary uses hash table. (data structure)





Questions:­

1. When we say that python is strongly typed, do we mean that we can’t do function or operator overloading in python like in C++?

2. In C++, we think object as a piece of memory, is it the same thing here in python? We just do not need to declare the memory when we define a variable? It is true that the fundamental difference between different python types is just different type allocate different amount of memory? If that’s true, what about a list. It is an type that dynamically allocate memory based on the elements?

3. >>> sev\_f = float(7)

>>> sev\_i = int(7)

>>> sev\_f ==sev\_i

True

3. The video about loops is kind of indicating that a variable can be used inside and outside a loop, is it true for Python? Does Python distinguish local variables and global variables?

4. I am also confused by 3.6 lists and mutability. I was trying to understand the concept but I just didn’t get it. I tried the following codes:

a)

>>> a = 1 b = 2 c = 3 d = 4

>>> x = [a,b,c,d] y = [a,b,c,d]

>>> a = 5

>>> x

[1,2,3,4]

>>> x[0] = 9

>>> y

[1, 2, 3, 4]

If a list is only the reference, why when we change the assigned value to the object it pointing to, the element of the list is unchanged? Also, I thought if I change the value in X (e.g x[0] = 9), Y will not change, and it indeed didn’t change, but in the video, it seems it should be changed. Could you help me understand this concept?

b) >>> a = 1 b = 2 c = 3 d = 4

>>> x = [a,b,c,d] X=x Y=x

>>> x = [5,b,c,d]

>>> X

[1 2 3 4]

This time, it still give unchanged X, why?

What exactly does mutable mean?