List Operators

Tuesday, March 1, 2022 3:13 PM

Sometimes we need to store a collection of data ⇒ Use lists
General form of list is :

my_list = [val1 , val2 , val3 , ... , val N]

• List elements can be of any type and can be of more than 1 type.

e.g: my_list = ['hello', True, 8.12 , 'lol']

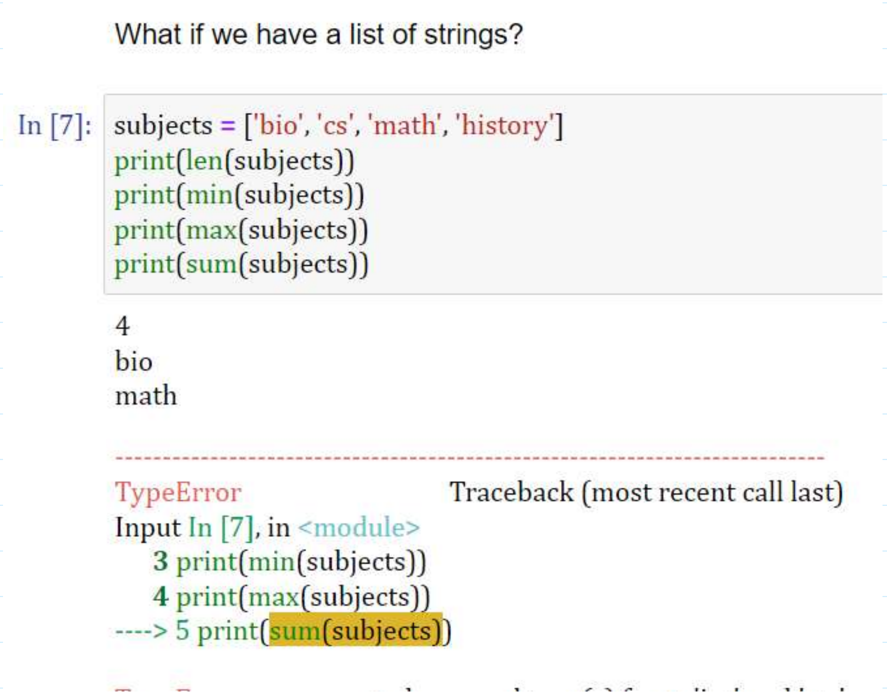
* List Operations (Indexing and Slicing)

+) A list can be indexed just like a string.
+) A list can be sliced just like a string.

```
>>> grades = [80, 90, 70, 45, 98, 57 ]
>>> grades [0: 2]                  >>> grades [ : : -2 ]
    [ 80 , 90 ]                        [ 57 , 45 , 90 ]
```

* Built-in functions in Python can be applied to lists

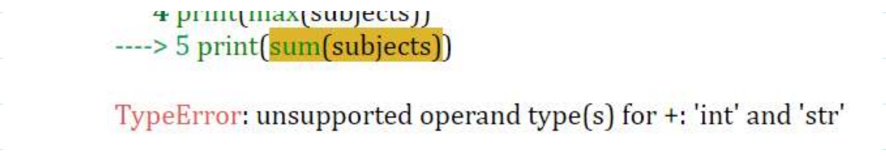
+) len (list) : returns the number of elements in a string
+) min (list) : returns the smallest element
+) max (list) : returns the biggest element (compare ASCII)
+) sum (list) : returns the sum of elements of lists (must be numeric)


What if we have a list of strings?

```
In [7]: subjects = ['bio', 'cs', 'math', 'history']
        print(len(subjects))
        print(min(subjects))
        print(max(subjects))
        print(sum(subjects))

4
bio
math

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [7], in <module>
      3 print(min(subjects))
      4 print(max(subjects))
----> 5 print(sum(subjects))
```



```
      4 print(max(subjects))
----> 5 print(sum(subjects))

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

• Nested Lists

$$\text{nested\_list} = [\ \underbrace{\text{list1},\ \text{list2},\ \dots,\ \text{listN}}_{[\text{val1},\ \text{val2},\ \dots,\ \text{valN}]}\ ]$$

To access a nested item, first select a sublist, then treat as a regularly list

```
>>>  nested_list [0][3]
     val4
```

Modifying Lists

Tuesday, March 1, 2022 3:23 PM

The other types we've learned so far are immutable

Lists are mutable

```
>>> s = "I love cats"
    s[0] = "u"
    Error ✗
```

```
>>> grades = [80, 90, 70, 100]
>>> grades[2] = 100
>>> print(grades)
[80, 90, 100, 100]
```

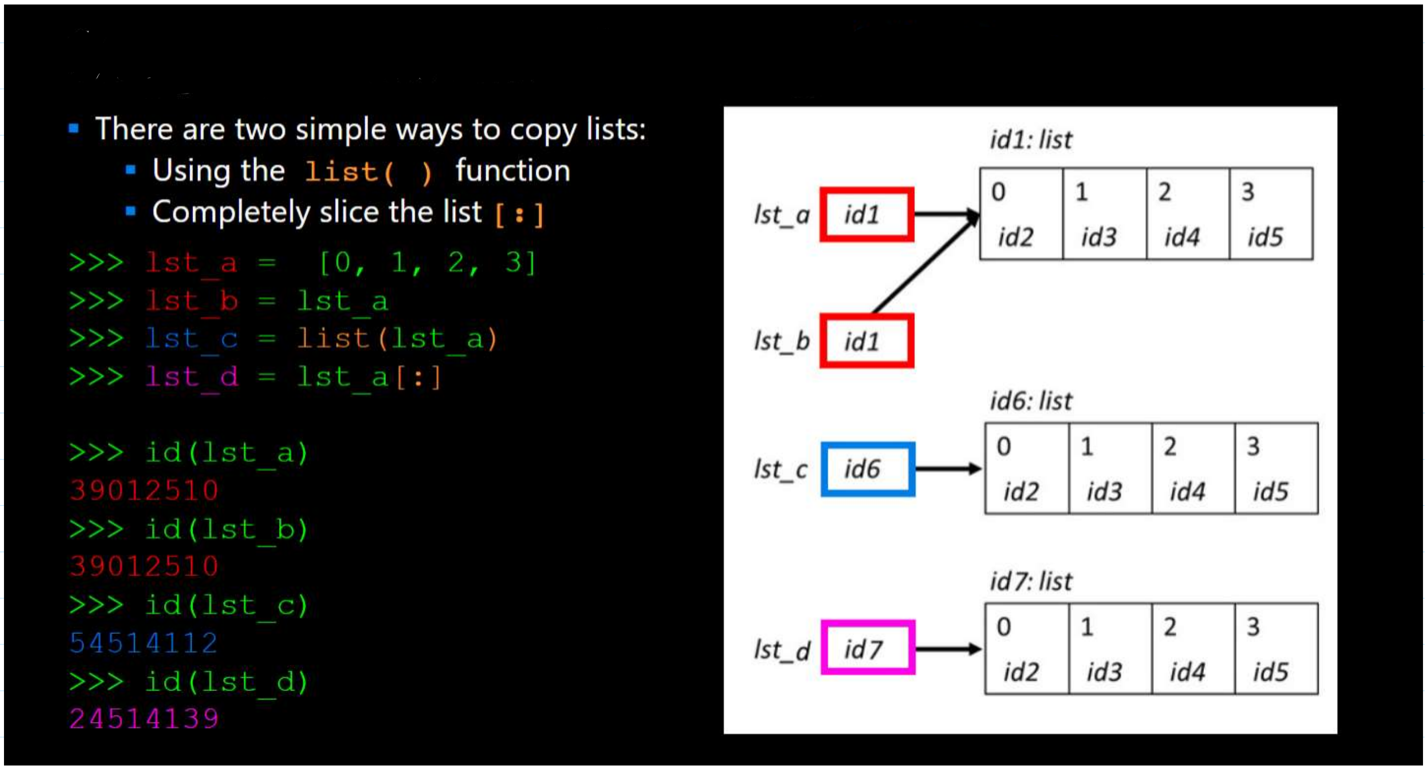
* Aliasing !!! (nicknames)

When 2 variables refer to the same object, they are aliases

⇒ When we modify 1 variable, we are also modifying the other one.

⇒ Do not assign list1 = list2

Copy lists by using list() or [:]



- There are two simple ways to copy lists:
  - Using the `list( )` function
  - Completely slice the list `[:]`

```
>>> lst_a =  [0, 1, 2, 3]
>>> lst_b = lst_a
>>> lst_c = list(lst_a)
>>> lst_d = lst_a[:]

>>> id(lst_a)
39012510
>>> id(lst_b)
39012510
>>> id(lst_c)
54514112
>>> id(lst_d)
24514139
```

List Methods

Tuesday, March 1, 2022 3:41 PM

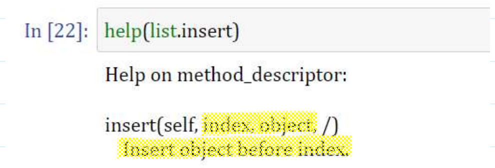
① Adding items to a list

+) To add an object to bottom, use list method append

```
>>> colors = ['blue', 'green']
>>> colors.append('brown')
>>> colors
['blue', 'green', 'brown']
```

What if we append a list? ⟹ Add sublist

+) To insert an object to some position, use list method insert



```
In [22]: help(list.insert)

Help on method_descriptor:

insert(self, index, object, /)
    Insert object before index.
```

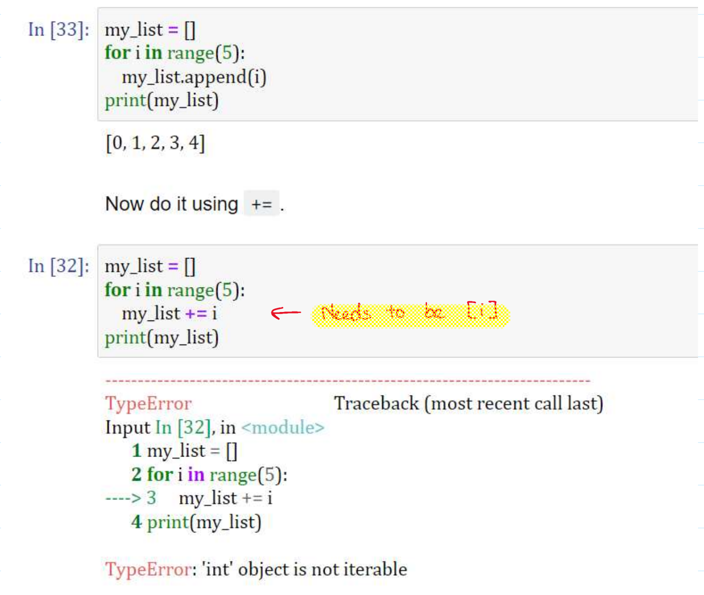
+) To add a list, use list method extend

```
>>> colors = ['blue', 'red']
>>> colors.extend(['red', 'yellow'])
>>> colors
['blue', 'red', 'red', 'yellow']
```

+) You can add an element to a list using append, or +

+) Like strings, you cannot subtract & can only multiply by int.

NOTE: You can only use + between lists.



```
In [33]: my_list = []
         for i in range(5):
             my_list.append(i)
         print(my_list)

[0, 1, 2, 3, 4]
```

Now do it using `+=`.

```
In [32]: my_list = []
         for i in range(5):
             my_list += i        ← Needs to be [i]
         print(my_list)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [32], in <module>
      1 my_list = []
      2 for i in range(5):
----> 3     my_list += i
      4 print(my_list)

TypeError: 'int' object is not iterable
```

② Removing items from a list

+) To remove an object from a list, use list method remove

```
>>> colors = ['blue', 'yellow', 'pink']
>>> c = colors.remove('pink')
>>> print(c)
    pink
>>> print(colors)
    ['blue', 'yellow']
```

← NOTE :
- Unlike strings, for which when we use methods, we are only "picking copies" from original string.
  → str = str.lowercase() to modify
- For lists, the list is modified as soon as we call method.
  → list_name.remove('lol') to modify
  - If : list_name = list_name.method('red')
    → $\boxed{\text{None}}$

+) Remove an object if it is in the list

```
colors = ['blue', 'green', 'pink']

if 'red' in colors :
    colors.remove('red')
```

If we don't use if, there will be error.

+) Remove the indexed item from the list. If no index, remove last item.

```
>>> colors = ['hi', 'im', 'minh']
>>> c = colors.pop()
>>> print(c)
    minh
>>> print(colors)
    ['hi', 'im']
```

list still modified. As long as method is called, list is modified.

③ Organizing items

+) list_name.order() → smallest to largest
+) list_name.reverse() → reverse

④ Getting information from Lists

+) list_name.count(object)
+) list_name.index(object) → Returns index. Error if not found.

Method	Description	Example

list.append(object)	Append object to end of list	`>>> colours = ['blue', 'yellow']`
		`>>> colours.append('brown')`
		`>>> colours`
		`['blue', 'yellow', 'brown']`

list.extend(list)	Append the items in the list parameter to the list	`>>> colours = ['blue', 'yellow']`
		`>>> colours.extend(['pink','green'])`
		`>>> colours`
		`['blue', 'yellow', 'brown', 'pink', 'green']`

list.insert(int, object)	Insert object at the given index, moving items to make room	`>>> grades = [95, 65, 75, 85]`
		`>>> grades.insert(3, 80)`
		`>>> grades`
		`[95, 65, 75, 80, 85]`

Method	Description	Example

list.remove(object)	Remove the first occurrence of the object; error if not there	`>>> colours = ['blue', 'yellow']`
		`>>> colours.remove('blue')`
		`>>> colours`
		`['yellow']`

list.pop([index])	Remove the item at the end of the list; optional index to remove from anywhere	`>>> colours = ['blue', 'yellow', 'pink']`
		`>>> colours.pop()`
		`'pink'`
		`>>> colours`
		`['blue', 'yellow']`
		`>>> colours.pop(0)`
		`'blue'`
		`>>> colours`
		`['yellow']`

Method	Description	Example

list.reverse()	Reverse the list	`>>> colours = ['blue', 'yellow', 'pink']`
		`>>> colours.reverse()`
		`>>> colours`
		`['pink', 'yellow', 'blue']`

list.sort()	Sort the list from smallest to largest (also sorts list of strings alphabetically)	`>>> grades = [95, 65, 75, 85]`
		`>>> grades.sort()`
		`>>> grades`
		`[65, 75, 85, 95]`

list.count(object)	Return the number of times object occurs in list	`>>> letters = ['a', 'a', 'b', 'c']`
		`>>> letters.count('a')`
		`2`

list.index(object)	Return the index of the first occurrence of object; error if not there	`>>> letters = ['a', 'a', 'b', 'c']`
		`>>> letters.index('a')`
		`0`

Note : Cannot count objects in nested lists.

Looping through lists

Thursday, March 3, 2022 3:13 PM

⊛ Nested lists Require Nested loops !

```
aps106_grades = [['Midterm 1', 60],
                 ['Midterm 2', 90],
                 ['Exam', 100]]
```

```
for row in aps106_grades:
    for column in row:
        print(column)
```

$\Rightarrow$

```
Midterm 1
60
Midterm 2
90
Exam
100
```