Inlämningsuppgift2

Issue date: 2022-09-26

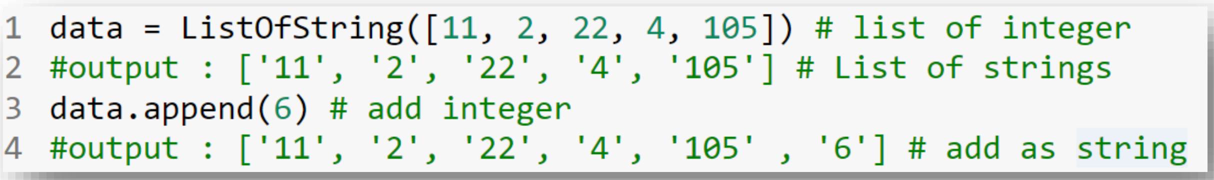
Return due date: 2022-10-08 kl 23:59

Create your own Lists:

At many moments in your Python coding trip, you may need to create own list of a datatype with modified behavior, new functionalities, or both.

In this project we are going to user python built-in data type list and other from collection library to create our own data types. Please answer all these questions, *25 points per each*!

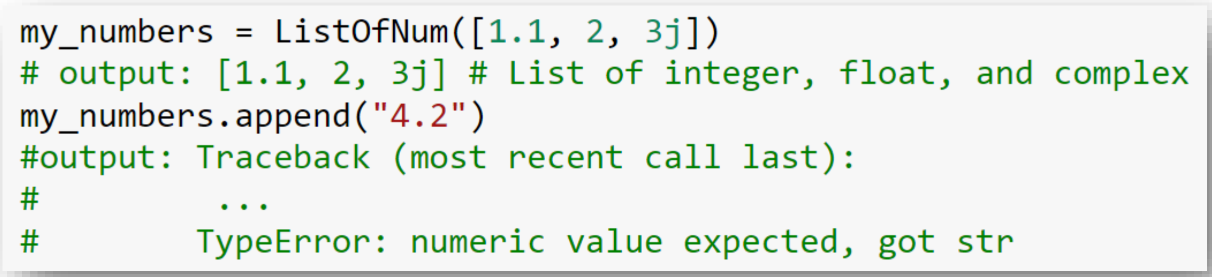
1. Create a class ListOfString() where you need a list to be automatically stores all its items as strings and inherit from List class. Your class converts all the input values into strings on the fly, so to achieve this you need to **modify** these methods and **demonstrate** how we can use each of them as example here:



```
data = ListOfString([11, 2, 22, 4, 105]) # list of integer
#output : ['11', '2', '22', '4', '105'] # List of strings
data.append(6) # add integer
#output : ['11', '2', '22', '4', '105' , '6'] # add as string
```

   a. __init__ initializes all the class's new instances(constructor).
   b. __setitem__() allows user to assign a new value to an existing item using the item's index, like in a_list[index] = item.
   c. extend () adds a series of items to the end of the list.
   d. insert () allows you to insert a new item at a given position in the underlying list using the item's index.
   e. append () adds a single new item at the end of the underlying list.
   f. Modify needed Dunder methods to Add concatenation '+' that add, add in place "+=" and with reflected (swapped) operands (a+b is as b+a). *(10 points of 25 points)*

2. Create Class ListOfNum() where you need a list that accepts numeric data only. Your ListOfNum instance should store ***only*** integer, float, and complex. Otherwise raise an error. So, to achieve this you need inherit from List again and to **modify** these methods and **demonstrate** how we can use each of them and **test** with non-numbers of values as this example here: *(5 points per each)*

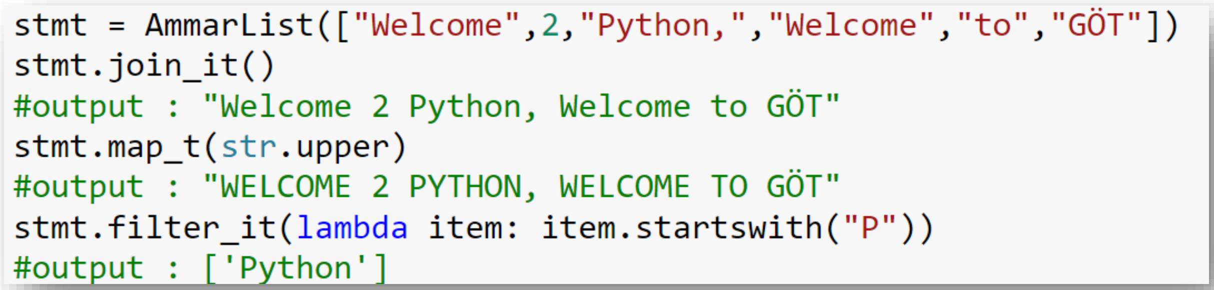


```
my_numbers = ListOfNum([1.1, 2, 3j])
# output: [1.1, 2, 3j] # List of integer, float, and complex
my_numbers.append("4.2")
#output: Traceback (most recent call last):
#         ...
#        TypeError: numeric value expected, got str
```

   a. __init__ initializes all the class's new instances(constructor).
   b. __setitem__() allows user to assign a new value to an existing item using the item's index, like in a_list[index] = item.
   c. extend () adds a series of items to the end of the list.
   d. insert () allows you to insert a new item at a given position in the underlying list using the item's index.
   e. append () adds a single new item at the end of the underlying list.

3. Now let's upgrade built in List datatype with new datatype called <your_name>List() example: AmmarList(). This datatype inherits from List datatype and add extra methods. **Write** your own class and **demonstrate the new** added methods with best performance to built-in datatype as next methods: *(5 points per each, d. 10 points)*

   a. .join_it() concatenates all the list's items in a single string.
   b. .map_it(action) yields new items that result from applying an action() callable to each item in the underlying list.
   c. .filter_it(predicate) yields all the items that return True when calling predicate() on them.
   d. .for_each_item(func) calls func() on every item in the underlying list to generate results!



```
stmt = AmmarList(["Welcome",2,"Python,","Welcome","to","GÖT"])
stmt.join_it()
#output : "Welcome 2 Python, Welcome to GÖT"
stmt.map_t(str.upper)
#output : "WELCOME 2 PYTHON, WELCOME TO GÖT"
stmt.filter_it(lambda item: item.startswith("P"))
#output : ['Python']
```

4. In first question we inherited from List class let’s here inherit from UserList a class in collection library, and name is ListOfUserString() *(10 points per each, c. 15 points)*
    a. let’s create the same methods (a.,b.,c.,d. and e.) from question#1
    b. Let’s use timeit using test_data = range(100_000) with extend() method.
    c. Compare the results? and explain why?

Points <70: IG

Points >70: G

Points >85: VG