LGT3109
Introduction to Coding for Business with Python
(week 7)

Xiaoyu Wang

Dept. of LMS, The HK PolyU

Summary of Week 6-Strings

- String type : a sequence of characters
- Read/Convert
- Indexing strings []: forward and backward
- Looping through strings with for and while, and len()
- Slicing strings [2:4]
- Concatenating strings with +
- Repeating a string with *
- String comparison
- String is immutable

- String module in standard library
- Searching in strings: find()
- Replacing text: replace()
- Stripping white space: strip(), lstrip(), rstrip()
- String prefix and suffix: startswith(), endswith()
- Parsing and Extracting
- F-Strings: f'{rate}*{quantity}={rate*quantity}.'

- Debugging for “index out of range”

Summary of Week 6-Files

- A file can be treated as a sequence of lines
- Opening a file - file handle
  - file_handle=open(file_name,'r')
- Closing a file
  - file_handle.close()
- File structure - newline character
  - \n
- Writing files
  - file_handle=open(file_name,'w')
  - file_handle.write()

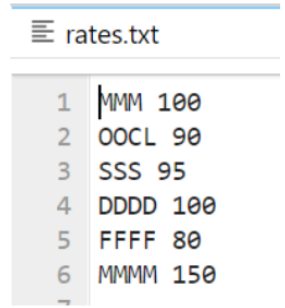
List

- Basic concepts
- List traverse
- List operations
- List functions
- List and strings
- References

Motivation Case

- Foxconn ships from Hong Kong to Los Angeles.
- Foxconn receives shipping rates from multiple carriers and store the rates in a text file (rates.txt).
- Carrier ID consists of letters: MMM, OOCL.
- Followed by a space and shipping rate.
- Foxconn wants to know the shipping rate for a list of carriers.

- **rates.txt**



- **Output**

```
Enter names of carriers: OOCL DDDD
['OOCL', 'DDDD']
Shipping rate of carrier OOCL is 90
Shipping rate of carrier DDDD is 100
```

Motivation Case

```
input_data = input('Enter names of carriers: ')

file = open('rates.txt', 'r')
for line in file:
    space_pos = line.find(' ')
    carrier_name = line[:space_pos]
    if carrier_name in input_data:
        rate = int(line[space_pos+1:])
        print('Shipping rate of carrier', carrier_name,'is', rate)

file.close()
```

Is this code correct?

Motivation Case

- Cannot distinguish between MMM and MMMM.
- Use list to solve this problem.

rates.txt

```
1 MMM 100
2 OOCL 90
3 SSS 95
4 DDDD 100
5 FFFF 80
6 MMMM 150
```

```
Enter names of carriers:  MMMM OOCL
Shipping rate of carrier MMM is 100
Shipping rate of carrier OOCL is 90
Shipping rate of carrier MMMM is 150
```

Wrong, because
"MMM" is in "MMMM OOCL"

List

- Basic concepts
- List traverse
- List operations
- List functions
- List and strings
- References

List Constant

- A list is a sequence of values:
  - ➢string, number, mixed, nested.
- Square brackets, elements are separated by commas.
- A list can be empty.

```
>>> print([1, 24, 76])
[1, 24, 76]
>>> print(['red', 'yellow',
'blue'])
['red', 'yellow', 'blue']
>>> print(['red', 24, 98.6])
['red', 24, 98.6]
>>> print([ 1, [5, 6], 7])
[1, [5, 6], 7]
>>> print([])
[]
```

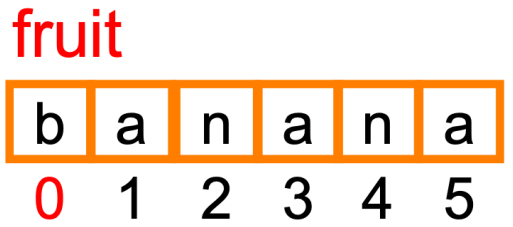
From String to List

- String: an ordered sequence of characters.

```
fruit = 'bababa'
print(fruit)
```

```
bababa
```

- List: an ordered sequence of values (of any types).




```
fruit_list = ['banana', 'apple', 'papaya']
print(fruit_list)
```

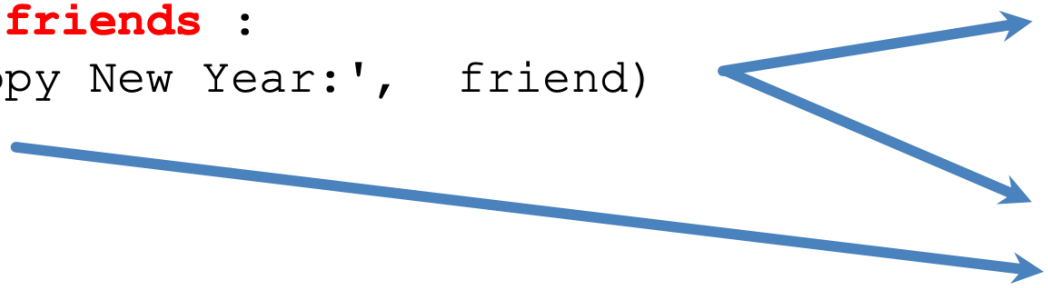
We Already Use Lists!

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Blastoff!')
```

```
5
4
3
2
1
Blastoff!
```

Lists and Definite Loops-Best Pals

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
    print('Happy New Year:',  friend)
print('Done!')
```

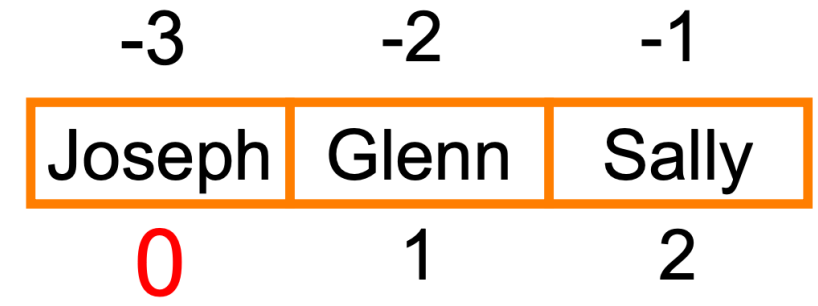


```
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Done!
```

```
z = ['Joseph', 'Glenn', 'Sally']
for x in z:
    print('Happy New Year:',  x)
print('Done!')
```

Looking Inside Lists

- Get at any single element in a list using an index.
- Similar rule as string index.
- Index an element that does not exist, you get an index error.




```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> print(friends[1])
Glenn
>>> print(friends[-1])
Sally
>>> print(friends[3])
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    print(friends[3])
IndexError: list index out of range
```

Lists are Mutable

- Strings are immutable.
  - ➢contents cannot be changed. Make new string to make change.
- Lists are mutable.
  - ➢Element of a list can be changed using the index operator.

```
>>> fruit = 'Banana'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not
support item assignment
>>> x = fruit.lower()
>>> print(x)
banana
>>> lotto = [2, 14, 26, 41, 63]
>>> print(lotto)
[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print(lotto)
[2, 14, 28, 41, 63]
```

A List is a Kind of Collection

- A collection allows us to put many values in a single variable.

➢ 'Joseph', 'Glenn', 'Sally' in one variable friends.

- A collection is nice because we can carry many values in one convenient package.

```
friends = [ 'Joseph', 'Glenn', 'Sally' ]

carryon = [ 'socks', 'shirt', 'perfume' ]
```

List

- Basic concepts
- List traverse
- List operations
- List functions
- List and strings
- References

How Long is a List?

- Like string, the len() function returns the number of elements in the list.
- len() tells us the number of elements of any set or sequence (such as a string...)

```
>>> greet = 'Hello Bob'
>>> print(len(greet))
9
>>> x = [ 1, 2, 'joe', 99]
>>> print(len(x))
4
>>> y = ['spam', 1, ['Brie', 'Roquefort', 'Pol le Veq'], [1, 2, 3]]
>>> print(len(y))
4
```

Using the range Function

- The range function returns a sequence of numbers that range from zero to one less than the parameter.

- The sequence of numbers generated by range() is like a list.

```
>>> print(list(range(4)))
[0, 1, 2, 3]
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print(len(friends))
3
>>> print(list(range(len(friends))))
[0, 1, 2]
```

Simplify the for-loop

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Blastoff!')
```

```
for i in range(5):
    print(5-i)
print('Blastoff!')
```

5
4
3
2
1
Blastoff!

Traversing a list

- The most common way to traverse is with a for loop and in.

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
    print('Happy New Year:',  friend)
print('Done!')
```

- An alternative way to traverse is with a for loop and index variable.

```
friends = ['Joseph', 'Glenn', 'Sally']
for i in range(len(friends)):
    friend[i]=friend[i].lower()
    print('Happy New Year:',  friend[i])
print('Done!')
```

len(friends) is 3
range(len(friends)) is range(3) which generates a sequence of numbers of 0,1,2

List

- Basic concepts
- List traverse
- List operations
- List functions
- List and strings
- References

Concatenating List

- We can create a new list by adding (concatenating) two existing lists together.

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

Repeating Lists

- We can create a new list by repeating a list a given number of times.

```
>>> a = [1, 2]
>>> b = a*3
>>> print(b)
[1, 2, 1, 2, 1, 2]
```

Is Something in a List?

- Like String, Python provides two operators, in and not in, that let you check if an item is in a list.
- Logical operators that return True or False.
- They do change modify the list.

```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
>>>
```

Slice List

- Attention: just like in strings, the second number is up to but not included!

```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41,12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

List

- Basic concepts
- List traverse
- List operations
- List functions
- List and strings
- References

Functions for List

- Like strings, Python also provides built-in module for list.

```
>>> x = list()
>>> type(x)
<type 'list'>
>>> dir(x)
['append', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']
>>>
```

Adding Element

- We can create an empty list using list() or [ ].
- We can add element (one and only one element) using append().
- The list stays in order and new elements are added at the end of the list.

```
>>> stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print(stuff)
['book', 99]
>>> stuff.append('cookie')
>>> print(stuff)
['book', 99, 'cookie']
```

Adding Element with Index

- If you want to insert an object with a specific indexed place, use insert(index , object).

```
>>> t = ['a','b','c']
>>> print(t)
['a', 'b', 'c']
>>> t.insert(3,'cc')
>>> print(t)
['a', 'b', 'c', 'cc']
>>> t.insert(4,'dd')
>>> print(t)
['a', 'b', 'c', 'cc', 'dd']
>>> t.insert(-5,'aa')
>>> print(t)
['aa', 'a', 'b', 'c', 'cc', 'dd']
```

Adding List

- extend() adds another list to the end of a list.

```
>>> t1 = ['a', 'b', 'c']
>>> t2 = ['d', 'e']
>>> t1.extend(t2)
>>> print(t1)
['a', 'b', 'c', 'd', 'e']
>>>
```

Sorting List

- A list can be sorted (i.e., change its order) using sort().

➢ sort() works in ascending order by default.

➢ If reverse parameter is set to True, the list is sorted in descending order.

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> friends.sort()
>>> print(friends)
['Glenn', 'Joseph', 'Sally']
>>> print(friends[1])
Joseph
>>> friends.sort(reverse=True)
>>> print(friends)
['Sally', 'Joseph', 'Glenn']
```

Deleting Element

- Know the index of the element to delete, use pop.
- returns the deleted element.
- If you do not need the deleted element, use del.
- Know the element to remove (not index), use remove.
- To remove more than one element, use del with a slice index.

```
>>> t = ['a', 'b', 'c']
>>> x = t.pop(1)
>>> print(t)
['a', 'c']

>>> t = ['a', 'b', 'c']
>>> del t[1]
>>> print(t)
['a', 'c']

>>> t = ['a','b','b','c']
>>> t.remove('b')
>>> print(t)
['a', 'b', 'c']

>>> t = ['a', 'b', 'c', 'd',
'e', 'f']
>>> del t[1:5]
>>> print(t)
['a', 'f']
```

Finding Index of a Value

- If you want to know the index of an object in a list, use index.

- Returns the first index of the object, and Value Error if not in the list.

```
>>> t = ['a','b','c']
>>> t.index('a')
0
>>> t.index('b')
1
>>> t.index('d')
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    t.index('d')
ValueError: 'd' is not in list
```

Average in a Loop

```
#initialize the variables:
count = 0
sum = 0
print('Before', count, sum)
for value in [9, 41, 12, 3, 74, 15] :
#for each entry, update the variables:
    count = count + 1
    sum = sum + value
    print(count, sum, value)
#output the variables:
print('After', count, sum, sum / count)
```

```
Before 0 0
1 9 9
2 50 41
3 62 12
4 65 3
5 139 74
6 154 15
After 6 154 25.666
```

Built-in Functions and Lists

- Remember the loops in Lecture 5?
- There are built-in functions that works for list.

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print(len(nums))
6
>>> print(max(nums))
74
>>> print(min(nums))
3
>>> print(sum(nums))
154
>>> print(sum(nums)/len(nums))
25.6
```

List

- Basic concepts
- List traverse
- List operations
- List functions
- List and strings
- References

Best Friends: Lists and Strings

- split of a string breaks the string into parts based on a delimiter and produces a list of strings.

- Access a particular word or loop through all the words.

```
>>> abc = 'With three words'
>>> stuff = abc.split()
>>> print(stuff)
['With', 'three', 'words']
>>> print(len(stuff))
3
>>> print(stuff[0])
With
```

```
>>> print(stuff)
['With', 'three', 'words']
>>> for w in stuff :
...     print(w)
...
With
Three
Words
>>>
```

Delimiter

- Use of split() of a string.
- If delimiter is not specified, multiple spaces are treated as one delimiter.
- You can specify what delimiter character to use for splitting.

```
line = 'Professor  Xiaoyu Wang'
print(line.split())
```

```
['Professor', 'Xiaoyu', 'Wang']
```

```
line = 'Professor  Xiaoyu Wang'
print(line.split(' '))
```

```
['Professor', '', 'Xiaoyu', 'Wang']
```

```
line = 'Professor,Xiaoyu Wang'
print(line.split())
```

```
['Professor,Xiaoyu', 'Wang']
```

```
line = 'Professor,Xiaoyu Wang'
print(line.split(','))
```

```
['Professor', 'Xiaoyu Wang']
```

Parsing Lines Using if Statement and Split()

'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'

if-statement is to find "interesting lines"

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if not line.startswith('From ') :
        continue
    words = line.split()
    print(words[2])
```

split method is to find interesting part of the lines

Sat
Fri
Fri
Fri
...

```
>>> line = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
>>> words = line.split()
>>> print(words)
['From', 'stephen.marquard@uct.ac.za', 'Sat', 'Jan', '5', '09:14:16', '2008']
>>>
```

The Double Split Pattern

- Sometimes we split a line, and then grab one piece and split that piece again, using different delimiter.

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008

words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

The Double Split Pattern

- Sometimes we split a line, and then grab one piece and split that piece again, using different delimiter.

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008

words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

```
stephen.marquard@uct.ac.za
```

The Double Split Pattern

- Sometimes we split a line, and then grab one piece and split that piece again, using different delimiter.

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
```

The Double Split Pattern

- Sometimes we split a line, and then grab one piece and split that piece again, using different delimiter.

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'
```

Finding Shipping Rates from a File

```
input_data = input('Enter names of carriers: ')
carrier_names = input_data.split()
print(carrier_names)

file = open('rates.txt', 'r')
for line in file:
    space_pos = line.find(' ')
    carrier_name = line[:space_pos]
    if carrier_name in carrier_names:
        rate = int(line[space_pos+1:])
        print('Shipping rate of carrier', carrier_name,'is', rate)

file.close()
```

- Parse: transform input string to a list of carrier names
- For each line of the file:
  - Parse: If the line starts with a carrier_name in the input list
    - Extract the shipping rate from the line, and print the rate

rates.txt

```
1 MMM 100
2 OOCL 90
3 SSS 95
4 DDDD 100
5 FFFF 80
6 MMMM 150
```

```
Enter names of carriers:  MMMM OOCL
['MMMM', 'OOCL']
Shipping rate of carrier OOCL is 90
Shipping rate of carrier MMMM is 150
```

How to Apply Double Split Pattern?

```
for line in file:
    words = line.split()
    carrier_name = words[0]
    if carrier_name in carrier_names:
        rate = int(words[1])
        print('Shipping rate of carrier'
```

Finding Shipping Rates from a File

```
input_data = input('Enter names of carriers: ')
carrier_names = input_data.split()
print(carrier_names)

file = open('rates.txt', 'r')
rates = []
for line in file:
    space_pos = line.find(' ')
    carrier_name = line[:space_pos]
    if carrier_name in carrier_names:
        rate = int(line[space_pos+1:])
        print(f'Shipping rate of carrier {carrier_name} is {rate}.')
        rates.append(rate)
print('Average rate:', sum(rates)/len(rates))
print('Maximum rate:', max(rates))
print('Minimum rate:', min(rates))
file.close()
```

```
Enter names of carriers: OOCL MMM
['OOCL', 'MMM']
Shipping rate of carrier MMM is 100.
Shipping rate of carrier OOCL is 90.
Average rate: 95.0
Maximum rate: 100
Minimum rate: 90
```

- append
- max
- min
- sum
- len

Join a list

- join is an inverse of split.
- join connects the list of strings and uses the delimiter to concatenate them.

```
t=['lgt', '3109', 'tutorial', '5', 'score']
delimiter = ' '
print(delimiter.join(t))
```

```
lgt 3109 tutorial 5 score
```

```
t = ['lgt', '3109', 'tutorial', '5', 'score']
delimiter = '_'
print(delimiter.join(t))
print(delimiter.join(t)+'.txt')
```

```
lgt_3109_tutorial_5_score
lgt_3109_tutorial_5_score.txt
```

List

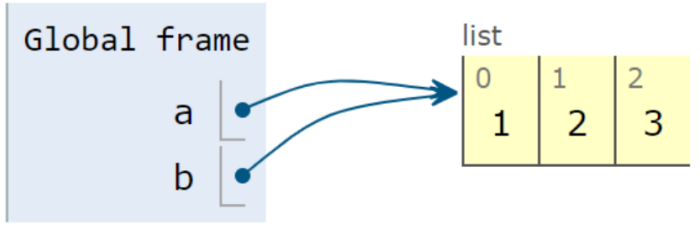
- Basic concepts
- List traverse
- List operations
- List functions
- List and strings
- References

Aliasing

- A variable refers to an object which has a value.
- The association of a variable with an object is called a reference.
- An object with more than one reference: the object is aliased.
- For aliased object (if mutable), changes with one alias affect the other.

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> b is a
False
```

```
>>> a = [1, 2, 3]
>>> b = a
>>> b is a
True
```



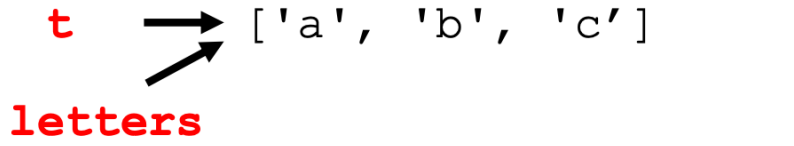

```
>>> b[0] = 17
>>> print(a)
[17, 2, 3]
```

List Arguments

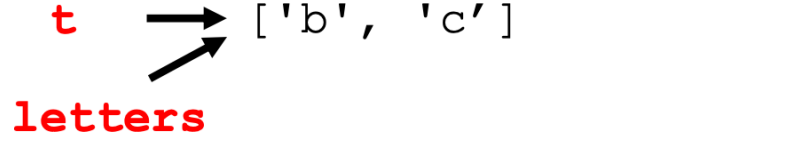
- Pass a list to a function, the function gets a reference to the list.
- Changes will be made to the object that the parameter refers to.

```
def delete_head(t):
    del t[0]
>>> letters = ['a', 'b', 'c']
>>> delete_head(letters)
>>> print(letters)
['b', 'c']
```

Before *del t[0]* in delete_head:



After *del t[0]* in delete_head:

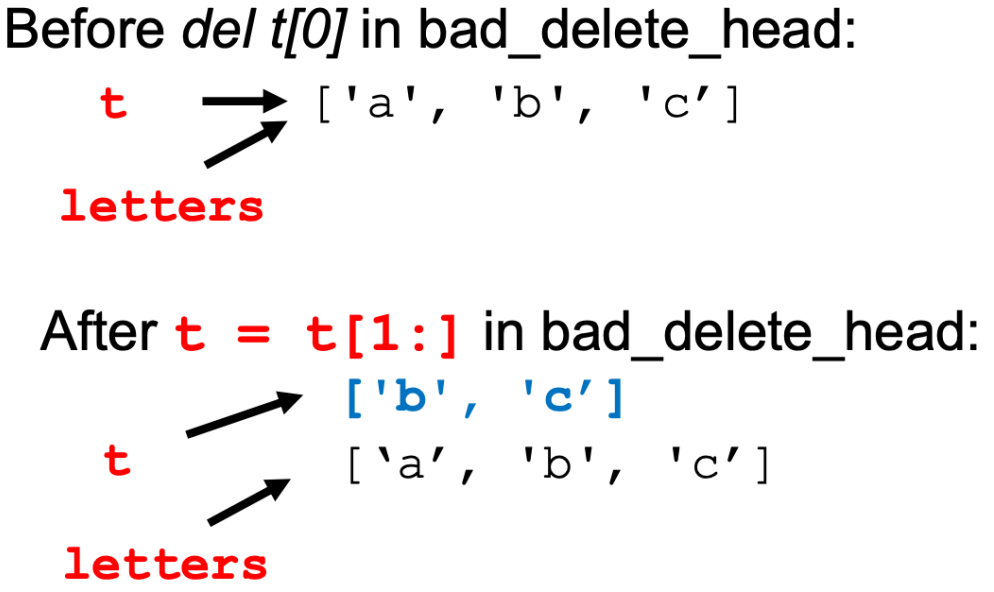


List Arguments

- In a function, it is important to distinguish between operations that modify objects and operations that create new objects.

```
def bad_delete_head(t):
    t = t[1:]

>>> letters = ['a', 'b', 'c']
>>> bad_delete_head(letters)
>>> print(letters)
['a', 'b', 'c']
```



Attention

- Most list functions modify the argument and return None.

➢List is mutable

- String functions return a new string and leave the original alone.

➢String is immutable

```
astring = astring.strip() #correct
alist = alist.sort()# WRONG!
t = t.append(x)          # WRONG!
```

Acknowledgement

- Acknowledgements / Contributions

- These slides are Copyright 2010-Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

- Initial Development: Charles Severance, University of Michigan School of Information
- Further Development: Zhou Xu, Hong Kong Polytechnic University
- Continuous development: Xiaoyu Wang, Hong Kong Polytechnic University