

CSCi 1012 [Section 10]



Introduction to Programming with Python

Prof. Kartik Bulusu, CS Dept.

Course start date January 17, 2024

Lecture location 1957 E street Room 213

Lecture times Monday, 3:45 PM to 5:00 PM

Wednesday-lab

3:45 PM to 5:00 PM

Section-30: MON 352

Section-31: SEH 4040

Section-34: TOMP 310

Section-35: TOMP 204

Friday-lab

3:45 PM to 5:00 PM

Section-32: SEH 4040

Section-33: TOMP 309

Section-36: TOMP 306

Section-37: TOMP 107



School of Engineering
& Applied Science

Spring 2024 THE GEORGE WASHINGTON UNIVERSITY

Photo: Kartik Bulusu

Recap: int- and float-objects and their operations

int	Represent integers including 0 (whole numbers)
float	Represent real numbers (with decimals)

```
>>> type(9)
<class 'int'>
```

```
>>> type(9.0)
<class 'float'>
```

i+j	Sum of integers or floats	→	i and j are integers. Result is an integer.	i or j are floats. Result is a float.
i-j	Difference			
i*j	Product			
i/j	Division	---	i and j are integers or floats. Results in a float.	
i%j	Remainder of i/j	→	i and j are integers. Result is an integer.	i or j are floats. Result is a float.
i**j	i raised to the power j			
i//j	Integer division			

Recap: String operations

```
>>> my_FirstName = "Kartik"  
>>> my_LastName = 'Bulusu'
```

Concatenation of Strings

Puts the strings together

```
>>> my_FullName2 = my_FirstName + ' ' + my_LastName
```

*-operator on Strings

Repeats the string

```
>>> greeting = my_FullName2 + ('Hello')*3
```

A new object type: List

Compound data type

- Collection of data
- Made up of other data types
 - **int**
 - **float**
 - **strings**
 - **Other data types (Boolean etc)**

→ **List**

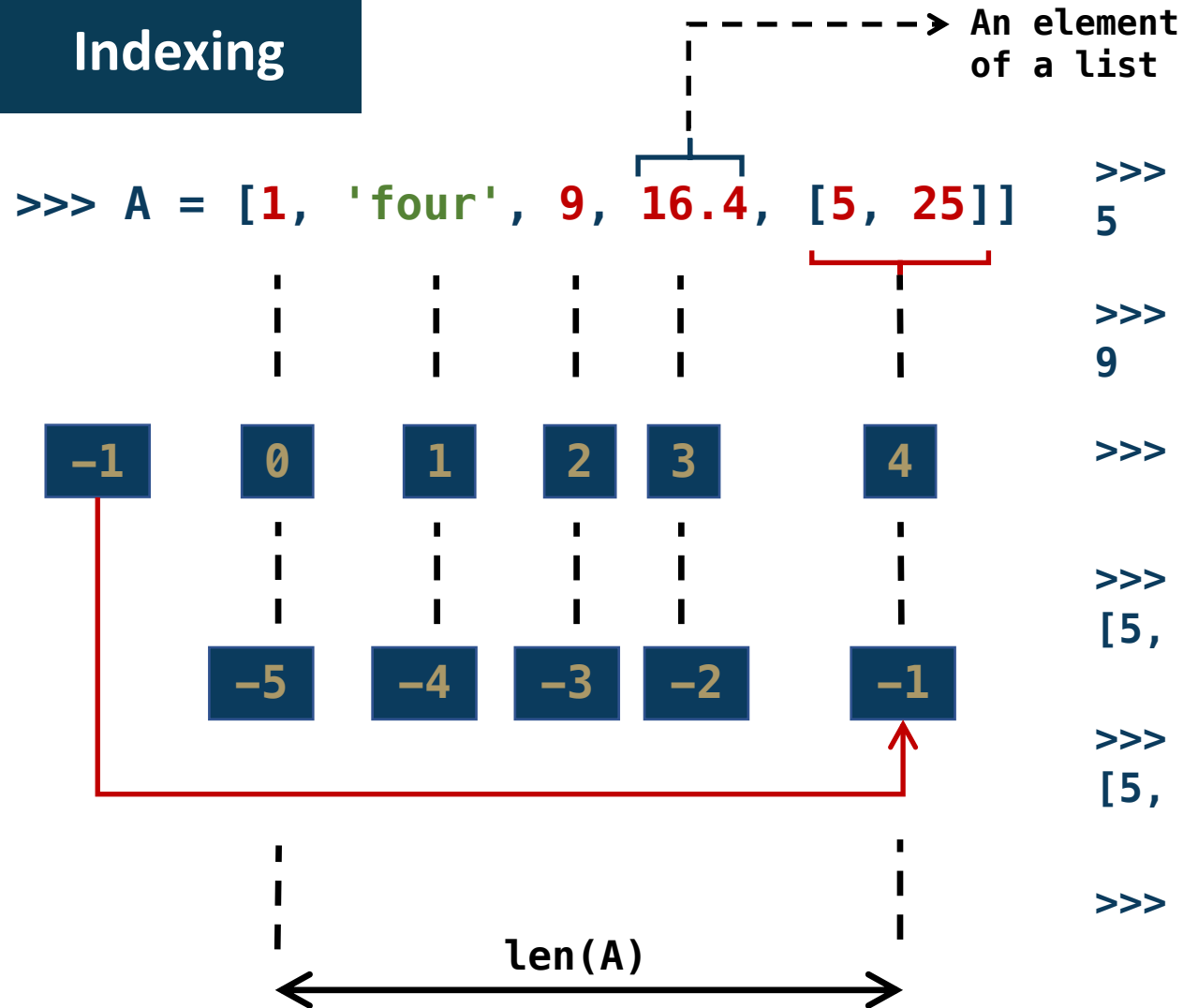
```
>>> A = [1, 'four', 9, 16.4, [5, 25]]

>>> B = [] # empty list
```

Key ideas to take home today

- Indexing
- Mutability
- Iterating over a list (**for** loop)
- List operations
- How we can tie it all together

Indexing




```
>>> len(A) # number of list elements
5
```

```
>>> A[2] # the third element in the list
9
```

```
>>> A[5] → 
```

```
>>> A[4] # returns the list within the list
[5, 25]
```

```
>>> A[-1] # Also returns the list within the list
[5, 25]
```

```
>>> A[-6] → 
```

```
>>> A[2]+1
>>> i=3; A[i-2]
```

Mutability

- Objects that can be changed after they are created
- Difference between Strings and Lists

Consider a string

```
>>> s = "Kartik"
>>> s[0]
'K'
>>> s[1]
'a'
>>> s[2]
'r'
>>> s[3]
't'
>>> s[4]
'i'
>>> s[5]
'k'
```

Consider replacing s[0] with "C"

```
>>> s[0] = 'C'
```



Strings are immutable

Consider a list

```
>>> A = [1, 4, 9, 16, 25]
```

Consider replacing A[0] with 32

```
>>> A[0] = 32
```

```
>>> A
[32, 4, 9, 16, 25]
```

Lists are mutable

Demo

for loops using range()

```
for i in range(start, stop, step):
    <expression>
    <expression>
    . . .
```

for loops using a list, L

```
for i in L:
    <expression>
    <expression>
    . . .
```

Sum of the elements in a list, L:

L = [1, 4, 9, 16, 25]

```
total = 0
for i in range(len(L)):
    total += L[i]

print(total)
```

:	:	:	:	:
:	:	:	:	:
:	:	:	:	:
0	1	2	3	4

```
total = 0
for i in L:
    total += i

print(total)
```

Note some similarities:
 # range(p) goes from 0 to p-1
 # index goes from 0 to len(L)-1
 # You are indexing into the element values

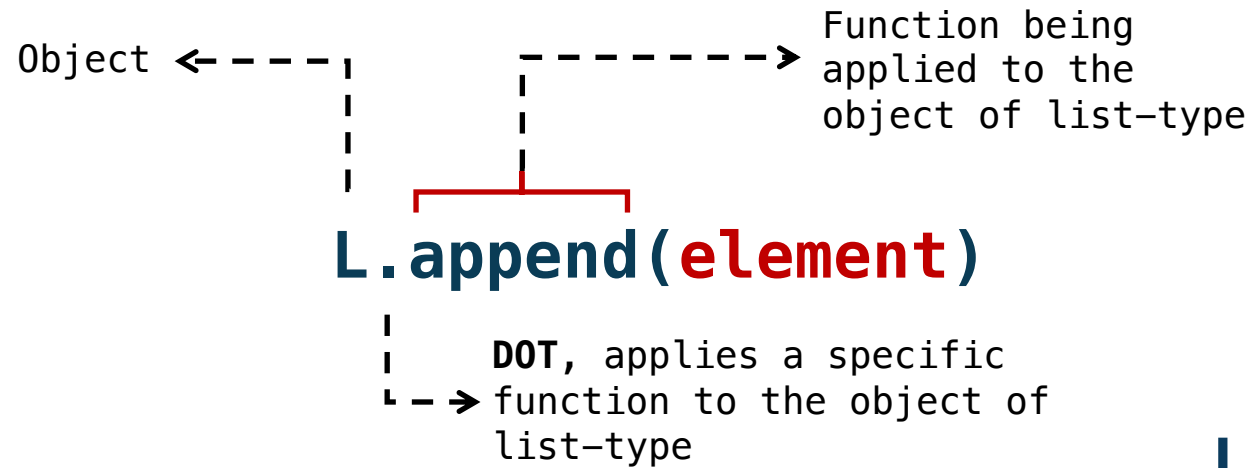
Demo

It's a lot cleaner code!
 # You are picking out the
 # element values themselves

List operations: append

Append

- Puts a list element at the end of the list
- Mutates the list



```
>>> L.append(36)
>>> print(L)
[1, 4, 9, 16, 25, 36]
```

Demo

`L = [1, 4, 9, 16, 25]`

0	1	2	3	4

`L = [1, 4, 9, 16, 25, 36]`

0	1	2	3	4	5

Sources:

Tuples, Lists, Aliasing and Mutability By Ana Bell: <https://youtu.be/RvRKT-iXvko>

Guttag, John, Introduction to Computation and Programming Using Python: With Application to Understanding Data (The MIT Press) second edition

Tie some concepts up

1	4	9	16	25
1	3	5	7	11

2	7	14	23	36

0	1	2	3	4
---	---	---	---	---

```
A = [1, 4, 9, 16, 25]
B = [1, 3, 5, 7, 11]

C = []
for i in range(5):
    element_total = A[i] + B[i]
    C.append(element_total)

print(C)
```

Demo

```
# A list of strings:
A = ['cats', 'and', 'dogs']
s = ''
for w in A:
    s += w
print(s)
```

Demo

```
# Some real numbers:
C = [1.1, 2.22, 3.333, 4.4444]
total = 0
for x in C:
    total = total + x
print('Average =', total/4)
```

Demo

List operations:

- Concatenation with + operator
- extend

```
>>> list_1 = [1, 2, 3]
>>> list_2 = [4, 5, 6]
```

Concatenation of Lists with + operator

Puts the lists together

```
>>> list_3 = list_1 + list_2
>>> list_3
[1, 2, 3, 4, 5, 6]
```

list_1.extend(another_list)

Mutates a list by making it longer by another list

```
>>> list_1.extend([7, 8])
>>> list_1
[1, 2, 3, 7, 8]
```

Demo

List operations:

Removing elements in a list

- `del(list_4[index])`
- `list_4.remove(element)`
- `list_4.pop()`

`del`

Deletes the element being indexed

```
>>> del(list_4[2])
>>> list_4
[1, 2, 7, 8, 7]
```

`.remove()`

Removes the first occurrence of a specific element
If you assigned a variable it will be returned as 'None'.

```
>>> list_4.remove(7)
>>> list_4
[1, 2, 8, 7]
```

`.pop()`

Removes element at the end of the list and returns
the removed element

```
>>> list_4.pop()
7
>>> list_4
[1, 2, 8]
```

```
>>> list_4 = [1, 2, 3, 7, 8, 7]
              | | | | |
              | | | | |
              | | | | |
              0 1 2 3 4 5
```

```
>>> list_4 = [1, 2, 7, 8, 7]
              | | | | |
              | | | | |
              | | | | |
              0 1 2 3 4
```

```
>>> list_4 = [1, 2, 8, 7]
              | | | |
              | | | |
              | | | |
              0 1 2 3
```

```
>>> list_4 = [1, 2, 8]
              | | |
              | | |
              | | |
              0 1 2
```



List operations:

Converting

- strings to lists and
- lists to strings

Consider a string

```
>>> s = '3 <= 4'
```

`list(s)`

Returns a list with every character including spaces

```
>>> list(s)
['3', ' ', '<', '=', ' ', '4']
```

```
# A way to extract the characters in a string into a list:
s = 'abcdef'
B = list(s)
print(B)
```

`.split()`

- Splits a string on a character as parameter
- Default to spaces

```
>>> s.split('=')
['3 <', ' 4' ]
```

Consider a list

```
>>> list_5 = ['g', 'r', 'e', 'a', 't']
```

`''.join()`

- Converts a list of characters into a string
- Can add a character between the quotes

```
>>> ''.join(list_5)
'great'
```

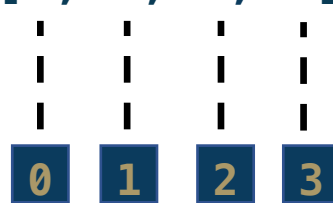
```
>>> '-'.join(list_5)
'g-r-e-a-t'
```

Demo

Lists are mutable!

What are the side effects?

```
>>> list_8 = [6, 3, 0, 8]
```



Aliasing

```
>>> list_9 = list_8
```

```
>>> list_8[2] = 7
>>> list_8
[6, 3, 7, 8]
```

```
>>> list_9
[6, 3, 7, 8]
```

Aliasing can be avoided by Cloning

```
>>> list_9 = list_8[:] #list_8.copy()
>>> list_9
[6, 3, 0, 8]
```

```
>>> list_8[2] = 7
>>> list_8
[6, 3, 7, 8]
```

```
>>> list_9
[6, 3, 0, 8]
```

Demo



Key thing to remember is that variables of lists may be affected by the changes or mutations

List operations:

Sorting

- `sorted()`
- `list_6.sort()`
- `list_6.reverse()`

```
>>> list_6 = [5, 3, 0, 9]
              |  |  |  |
              |  |  |  |
              |  |  |  |
              0  1  2  3
```

`sorted()`

- Returns a non-mutated sorted list
- Can assign a variable

```
>>> list_7 = sorted(list_6)
>>> list_7

[0, 3, 5, 9]
```

`list_6.sort()`

Mutates the list into a sorted list

```
>>> list_6.sort()

[0, 3, 5, 9]
```

`list_6.reverse()`

Mutates the list into a reversed list

```
>>> list_6.reverse()

[9, 5, 3, 0]
```

Many many more!

<https://docs.python.org/3/tutorial/datastructures.html>

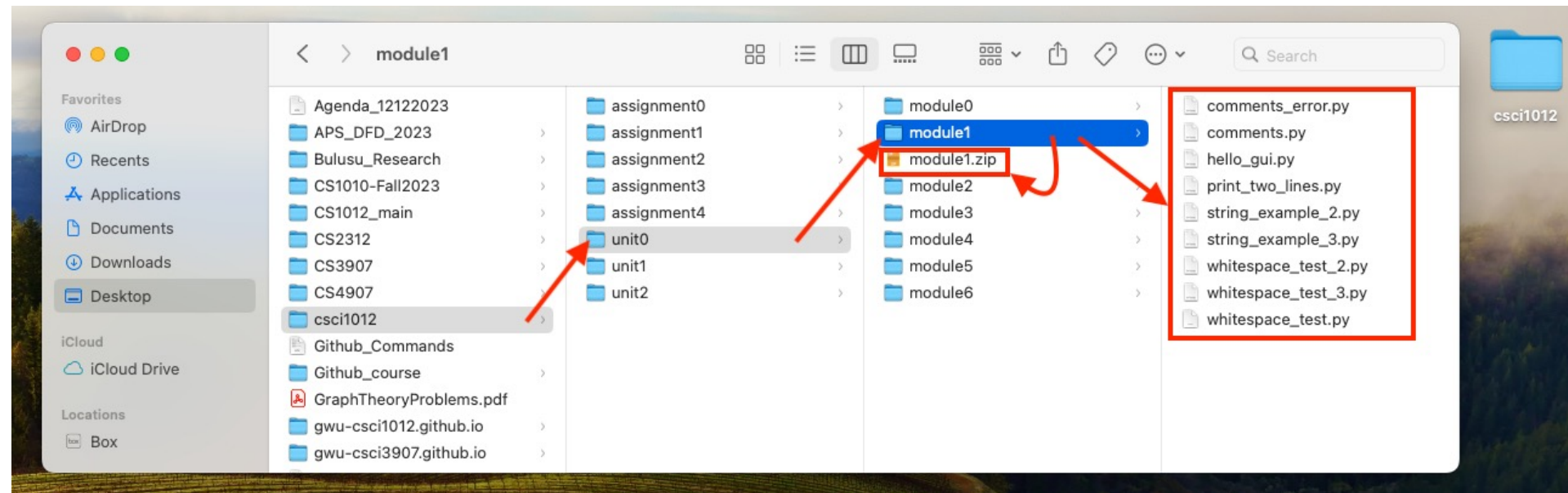
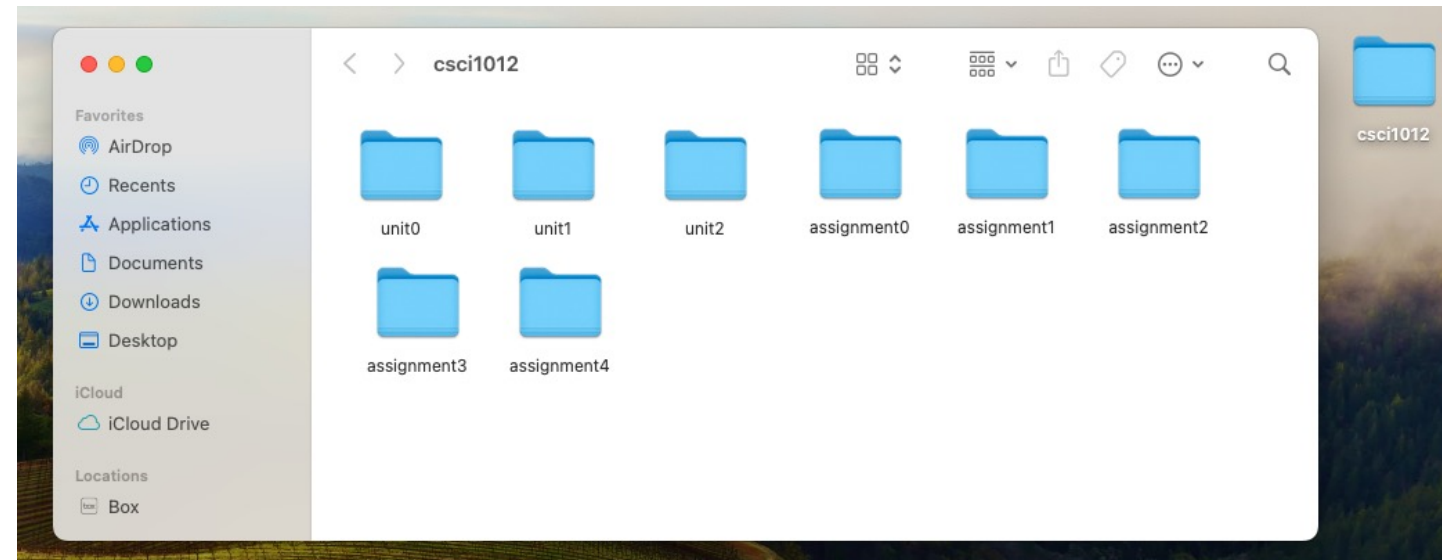
File-folder-structure

`module0.zip` (correct)

`Module0.zip` (wrong: starts with uppercase)

`module 0.zip` (wrong: space before 0)

`module0.docx` (wrong: not a zip).



HWs

- Due dates
- Late work
- Extensions

Date	Topic(s)	Wednesday Lab Date	Friday Lab Date	Assignment(s)
Week 5 President's Day - No Class!	Floating Points	02/21/2024	02/23/2024	Unit 0 » Module 6 (Due February 26, 2024 by 11:59 PM)
Week 6 [02/26/2024]	Lists	02/28/2024	03/01/2024	Assignment 1 (Due March 01, 2024 by 11:59 PM) & Unit 1 » Module 0 (Due March 04, 2024 by 11:59 PM)

- **CSCI 1012.30 (CRN: 94165)** - Moved to MONROE 352
- **Office hours location change:** Friday 10:00 AM to 2:00 PM is SEH B1280
- **IMPORTANT:** Please attend the ONLY lab that you registered into.

Late Work

- **Late work is not accepted, with the following exceptions:**
 - Every student may turn in as many as four (in total, not each) assignments or modules 48 hours after the deadline with no penalty. Requesting an extension is not necessary.
- **Extensions will be granted should there arise circumstances beyond your control** that impede your ability to complete coursework.
 - Notify your professor as soon as feasible in these cases.
 - Examples of such circumstances include (but are not limited to) illness, death in the family, and loss of housing. To ensure fairness toward all students, we will request documentation of such circumstances.

See you all in the Wednesday and Friday Labs!