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

GW School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Spring 2024

Photo: Kartik Bulusu

Class Policy on Collaboration

- **You may *not* *discuss Modules, Assignments, Quizzes and Exams* among yourselves.**
- **Each student is expected to work out the course deliverables *independently*.**
- **Under *no circumstances* may you look at another student's *Modules, Assignments, Quizzes and Exams*, or look for answers to *Modules, Assignments, Quizzes and Exams* anywhere other than in the text in the course website.**
- **You are encouraged to discuss the class material on Ed-discussion board or in-person with the instruction team.**
- **You may *not* discuss *Modules, Assignments, Quizzes and Exams* nor give out hints for the same on problems on the Ed-discussion board or with other students in-person.**

All violations will be treated as violations of the Code of Academic Integrity.

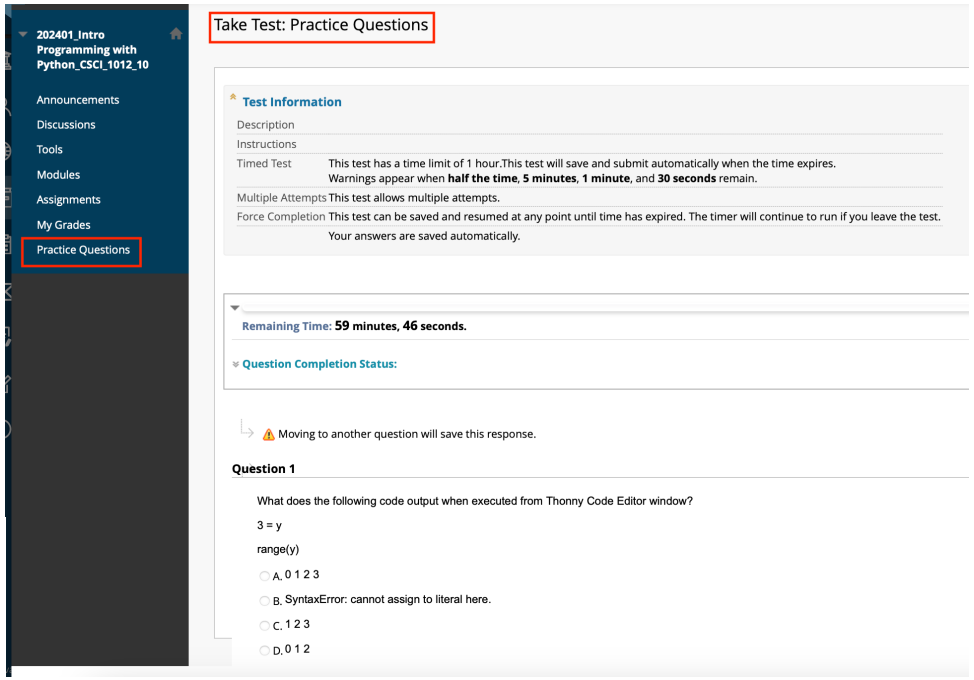
Announcements

- Exam on Monday, 4/15 during the lecture.
  - Paper exam
  - Closed everything (no notes, no Thonny)
  - Multiple choice/ fill in the blanks/ write the output – similar to quizzes
  - 25 questions
  - 100 points

- Mid-semester survey on the course webpage
  - Considered as a quiz worth 10 points.
  - Complete it by 11:59 PM on Saturday 03/30/2024

Scan the QR code below or click here to access the survey:

- Practice questions for you to review is created on Blackboard



- 48-hour extensions:
  - After the 4th extension your subsequent late submissions will not be considered toward final grade.

HWs
- Due dates
- Late work
- Extensions

Date	Topic(s)	Wednesday Lab Date	Friday Lab Date	Assignment(s)
Week 10 [03/25/2024]	Booleans & Built-ins	03/27/2024	03/29/2024	Unit 1 » Module 3 & Assignment 2 (Due **April 01, 2024** by **11:59 PM**)
Week 13 [04/15/2024]	**Examination**	04/17/2024	04/19/2024	Unit 2 » Module 0 & Module 1 (Due **April 22, 2024** by **11:59 PM**)

- **IMPORTANT:** Please attend the ONLY lab that you registered into.

Late Work

- **Late work is not accepted, with the following exceptions:**
  - Every student many 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.

Submission Tips

- Pay attention to file names

extra spaces

assignment1.zip ≠ ***assignment 1.zip***

caesar_shift.py ≠ ***caesar_shift .py***

extra spaces

- Pay attention to small details

```
x is 3 and j is 2, x + j = 5
x is 3 and j is 3, x + j = 6
x is 3 and j is 4, x + j = 7
x is 3 and j is 5, x + j = 8
x is 3 and j is 6, x + j = 9
x is 3 and j is 7, x + j = 10
x is 3 and j is 8, x + j = 11
```

≠

missing commas

```
x is 3 and j is  2 x + j 5
x is 3 and j is  3 x + j 6
x is 3 and j is  4 x + j 7
x is 3 and j is  5 x + j 8
x is 3 and j is  6 x + j 9
x is 3 and j is  7 x + j 10
x is 3 and j is  8 x + j 11
```

extra space *missing =*

- Don't add extra endline spaces

print("++++", end = "") →

```
++++
+  +
+  +
++++
```

→ ***What Auto Grader Sees…***

'++++\n+ +\n+ +\n+ +\n++++'

≠

print("++++", end = " ") →

```
++++
+  +
+  +
++++
```

→ '++++ \n+ +\n+ +\n+ +\n++++ '

extra spaces

Boolean

int	Represent integers including 0 (whole numbers)
float	Represent real numbers (with decimals)
bool	Represent Boolean values `True` and `False`
NoneType	Special and has one value, `None`

```
>>> type(True)
<type 'bool'>

>>> type(False)
<type 'bool'>
```

Boolean is used in comparing objects

```
>>> True
True

>>> False
False

>>> false
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'false' is not defined
```

= *Assignment*

```
>>> a = 4
>>> b = 5
>>>
```

== *Test for equality*

```
>>> a == b
>>> False
>>>
```

!= *Test for inequality*

```
>>> a != b
>>> True
>>>
```

Test

```
>>> a > b
>>> False
>>>
```

Test

```
>>> a < b
>>> True
>>>
```

How can we add “tests” to our code ?

Sources:
Branching and Iterations By Ana Bell: https://youtu.be/0jljZRnHwOI
Guttag, John, Introduction to Computation and Programming Using Python: With Application to Understanding Data (The MIT Press) second edition

Assume `i` and `j` are variables of `int`, `float` and `string` type.

We can test using logic operators

`i > j`	Greater than
`i >= j`	Greater than or equal to
`i < j`	Less than
`i <= j`	Less than or equal to
`i == j`	Equality
`i != j`	Inequality

```
if <condition>:
        <expression>
        <expression>
        . . .
```

`True`, if i and j are same.

`True`, if i and j are not the same.

Note:
= is an assignment
== is a test

Comparisons evaluate to a Boolean

- `True`
- `False`

You are allowed to compare

- `int` with `int`
- `float` with `float`
- `int` and `float`
- `string` with `string`

But not a number with strings.

String comparisons are lexicographical
- **Follows what comes first in the alphabet**

Source:
Python Booleans: https://youtu.be/9OK32jb_TdI

Casting as Boolean objects

```
>>> bool(8)
True
>>>

>>> bool(3.141)
True
>>>
```

```
>>> bool("Python")
True
>>>

>>> bool(" ") # Space
True
>>>
```

```
>>> bool(0)
False
>>>
```

```
>>> bool("") # Empty String
False
>>>

>>> bool([]) # Empty list
False
>>>
```

trivial → False
Non-trial → True

Casting Boolean objects as other object types

```
>>> str(True)
'True'
>>>
```

```
>>> str(False)
'False'
>>>
```

```
>>> int(True)
1
>>>
```

```
>>> int(False)
0
>>>
```

```
>>> 4 + True
5
>>>
```

```
>>> 4 * False
0
>>>
```

0 ←→ False
1 ←→ True

Sources:
Branching and Iterations By Ana Bell: https://youtu.be/0jljZRnHwOI
Guttag, John, Introduction to Computation and Programming Using Python: With Application to Understanding Data (The MIT Press) second edition

Can we test logic operators on Boolean values ?

Assume `a` and `b` are variables with Boolean values i.e., `True, False`

Example:

```
>>> a = True
>>> b = True

>>> not a
False

>>> a and b
True

>>> a or b
True
```

`not, and, or` are key words that can used on Boolean variables

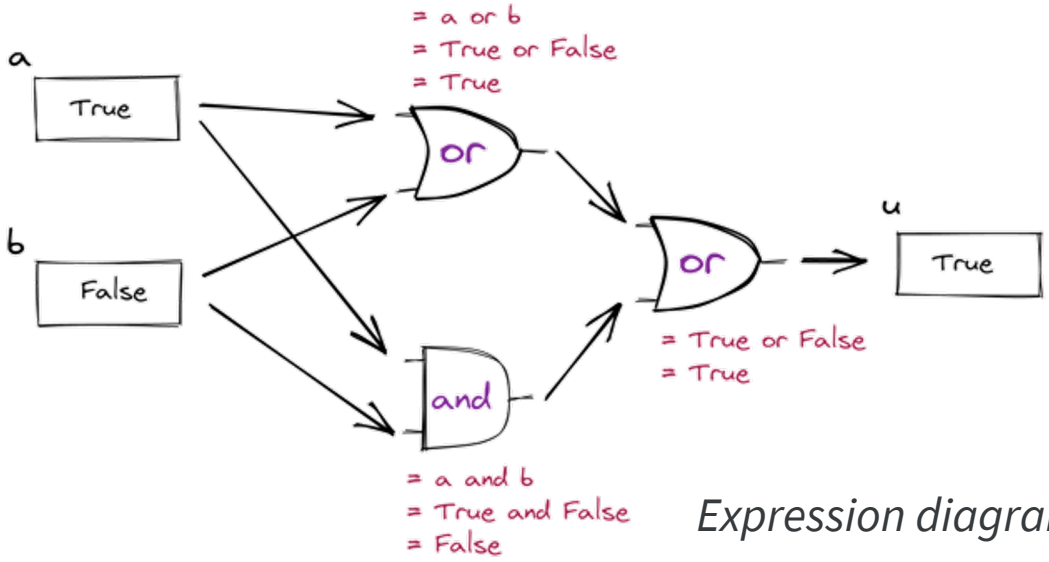
`not a` → `True` if `a` is `False`; `False` if `a` is `True`
`a and b` → `True` if both are `True`
`a or b` → `True` if either or both are `True`

a	b	a and b	a or b
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Sources:
https://www2.seas.gwu.edu/~cs4all/1012/unit1/module1.3.html

Combining Boolean operators

a and b → True if both are True
a or b → True if either or both are True



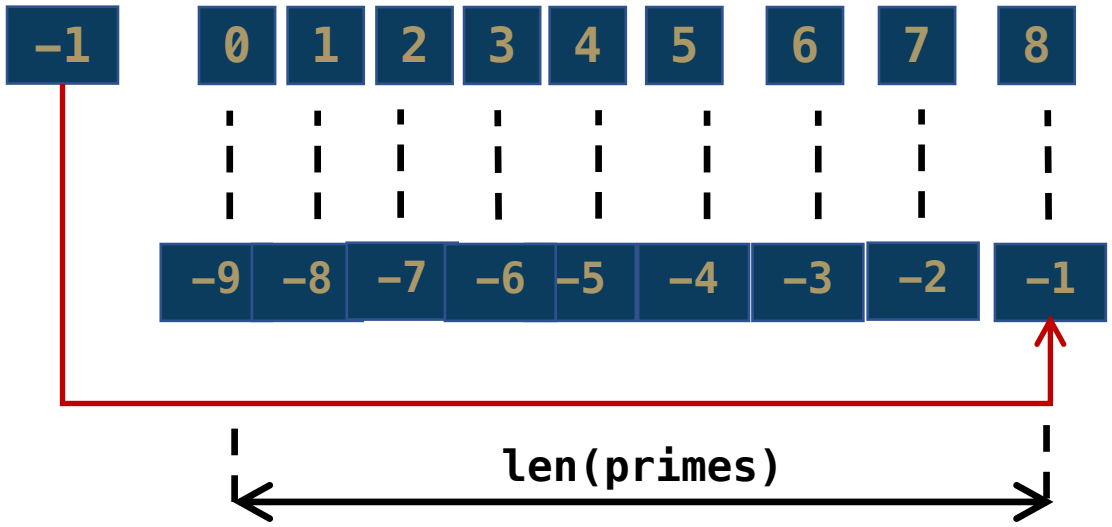
Expression diagram

a	b	a and b	a or b	(a and b) or (a or b)
True	True	**True**	**True**	True **or** True → **True**
True	False	**False**	**True**	False **or** True → **True**
False	True	**False**	**True**	False **or** True → **True**
False	False	**False**	**False**	False **or** False → **False**

Slicing Lists

Retrieve list-elements with a range of values

```
>>> primes = [2, 3, 5, 7, 9, 11, 13, 17, 19]
```



start *stop*

```
>>> primes[2:5]
[5, 7, 9]
```

start *stop* *step*

```
>>> primes[0:7:2]
[2, 5, 9, 13]
```

start *stop* *step*

```
>>> primes[8:2:-2]
[19, 13, 9]
```

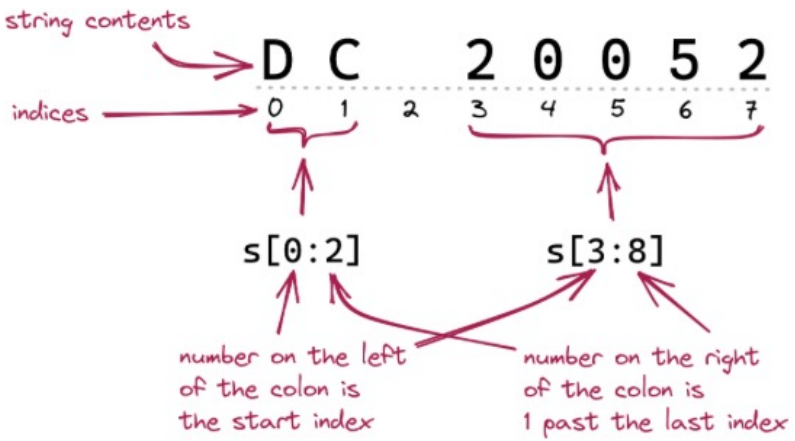
start: at the index value
step: up or down at the increment value (default = 1)
stop: at the index value but not including it

Sources:
https://www2.seas.gwu.edu/~cs4all/1012/unit1/module1.3.html

Slicing Strings

Retrieve string-elements with a range of values

```
s = 'DC 20052'
state = s[0:2]
zipcode = s[3:8]
print(state, zipcode)
```



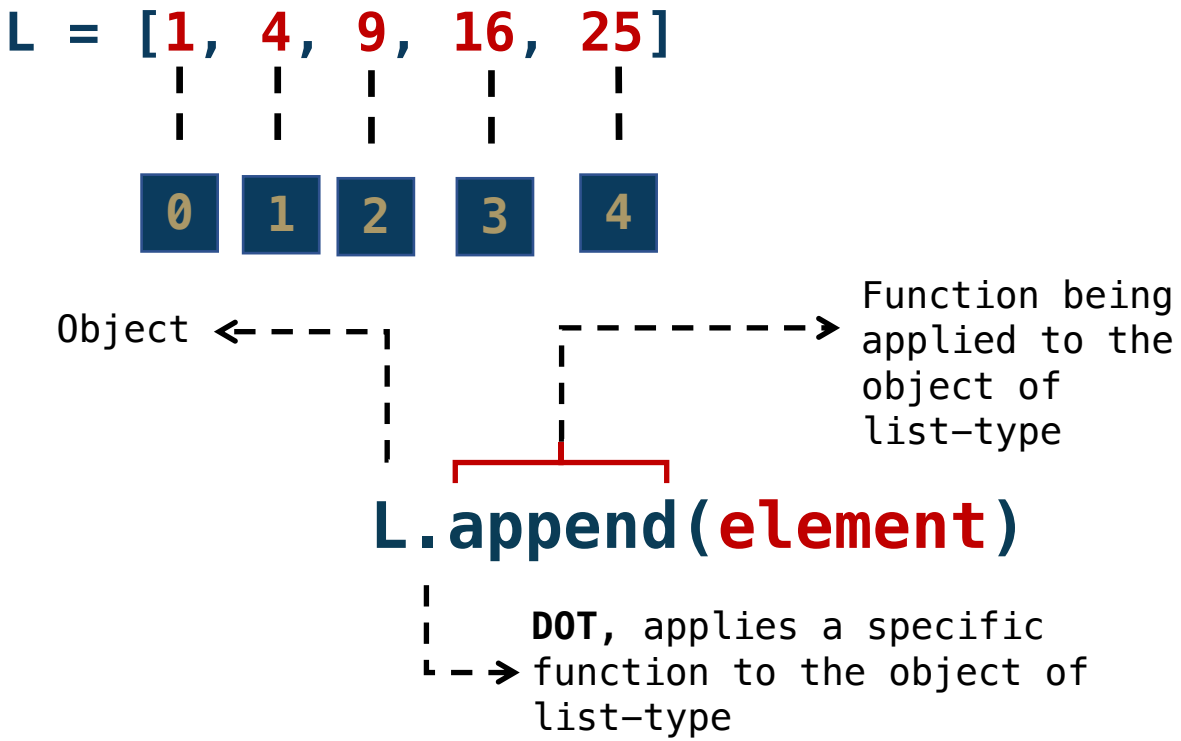

```
def extract_zip(s):
    start = len(s) - 5
    end = len(s)
    return s[start:end]

example1 = 'DC 20052'
example2 = 'District of Columbia, 20052'
example3 = '20052'
print(extract_zip(example1))
print(extract_zip(example2))
print(extract_zip(example3))
```

```
def find_common_prefix(w1, w2):
    for k in range(len(w1)):
        if w1[k] != w2[k]:
            break
    return w1[0:k]

print(find_common_prefix('river', 'rivet'))
```

Recall: Built-in function with lists




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

Built-in function with Strings

```
A = ['to','infinity','and','beyond','and', 'even','further']
s = 'infinity'

# Convert to uppercase:
print(s.upper())

# Count occurrences of the char 'i' in s:
print(s.count('i'))

# Locate which index 'f' first occurs in s:
print(s.find('f'))

# Occurrences of 'and' in list A:
print(A.count('and'))

# Occurrences of 'i' in 2nd string in list A:
print(A[1].count('i'))

if A[3].startswith('be'):
    print('starts with be')

data = '42'
print(data.isnumeric())
```

Try the following on the shell window:

```
>>> dir(str)
>>> dir(list)
>>> dir(bool)
>>> dir(complex)
```

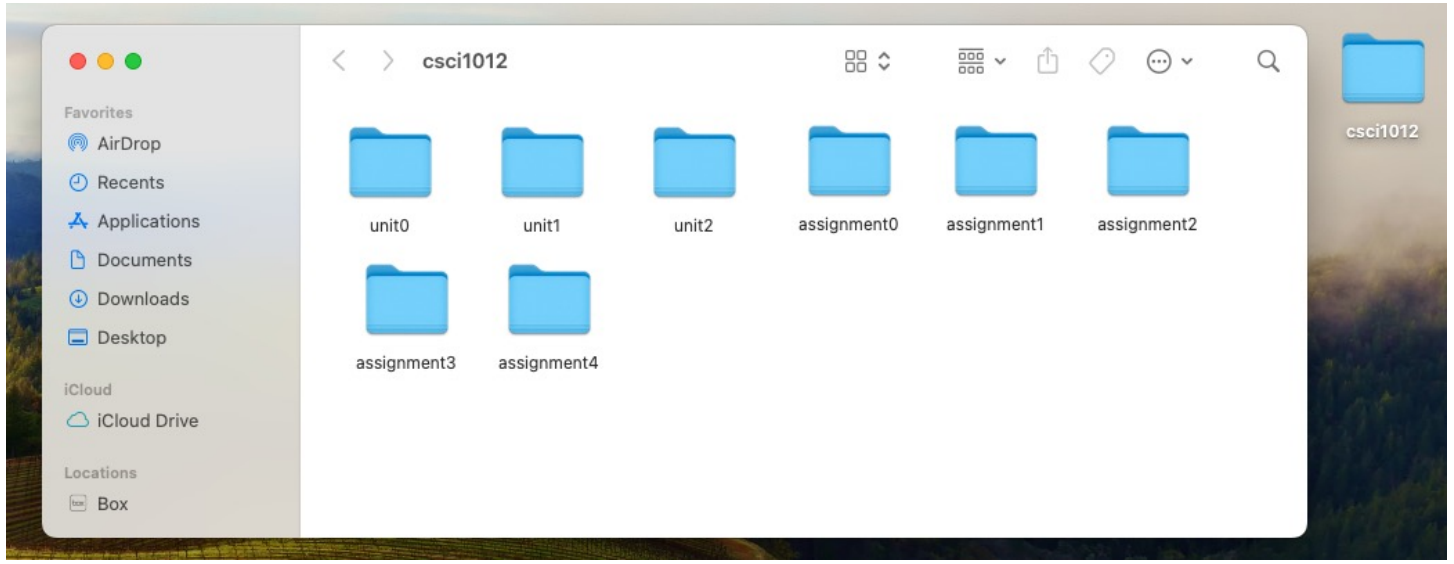
File-folder-structure

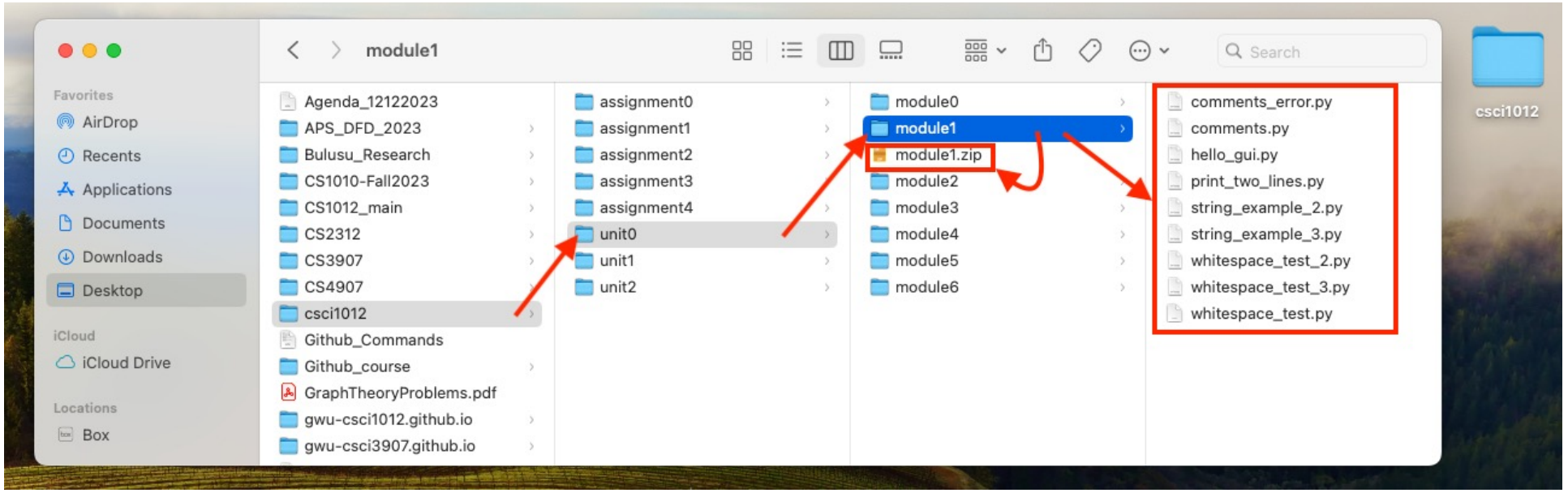
`module0.zip` (correct)

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

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

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





See you all in the Wednesday and Friday Labs!