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

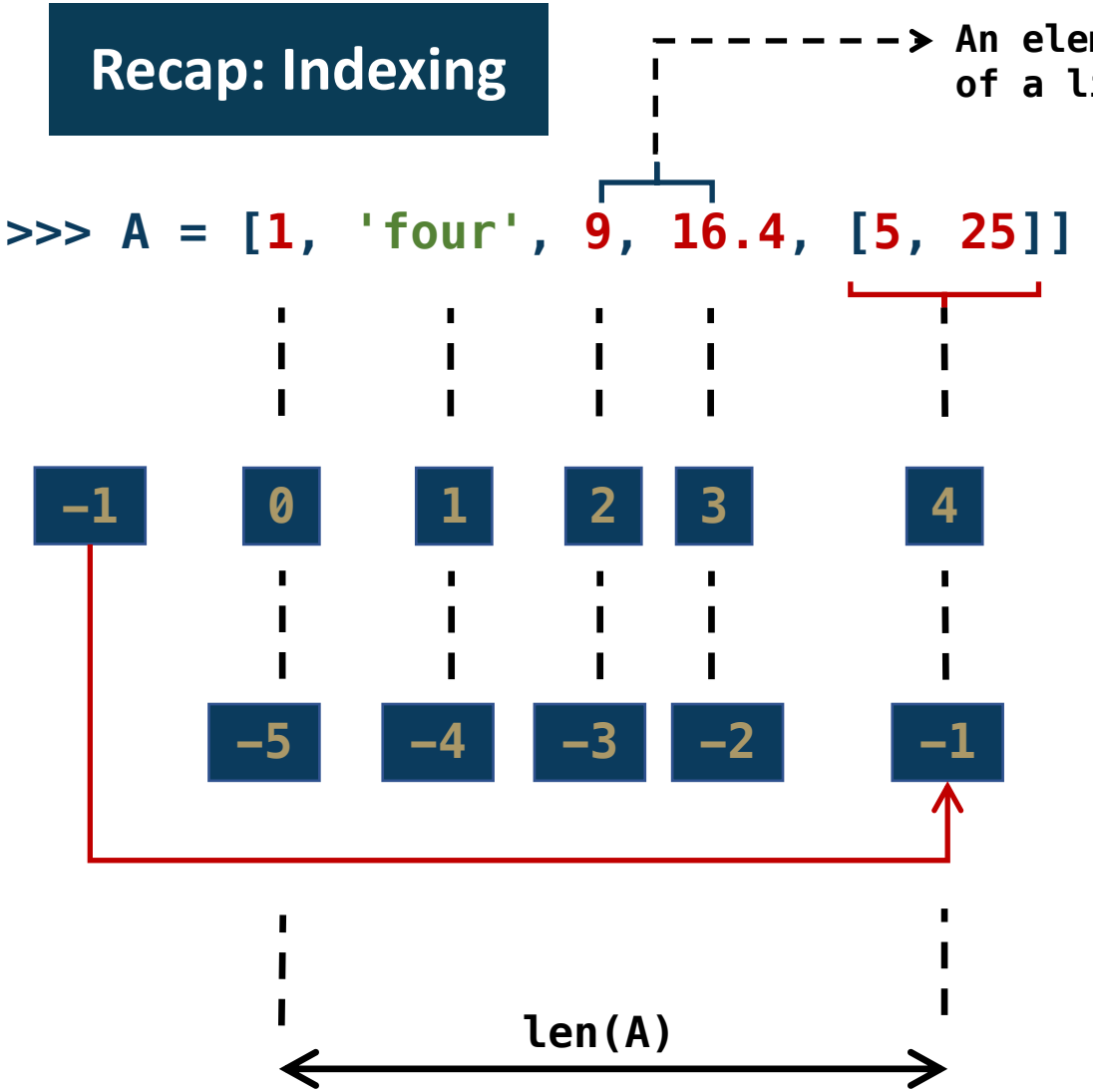
Photo: Kartik Bulusu

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.

Recap: Indexing




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

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

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


```
>>> A[4] # & A[-1] returns the list within the list
[5, 25]


>>> print(A)
[1, 'four', 9, 16.4, [5, 25]]

>>> for i in A:
        print(i, end=' ‘)

1 four 9 16.4 [5, 25]
```

Mutability

- Can objects 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]
```

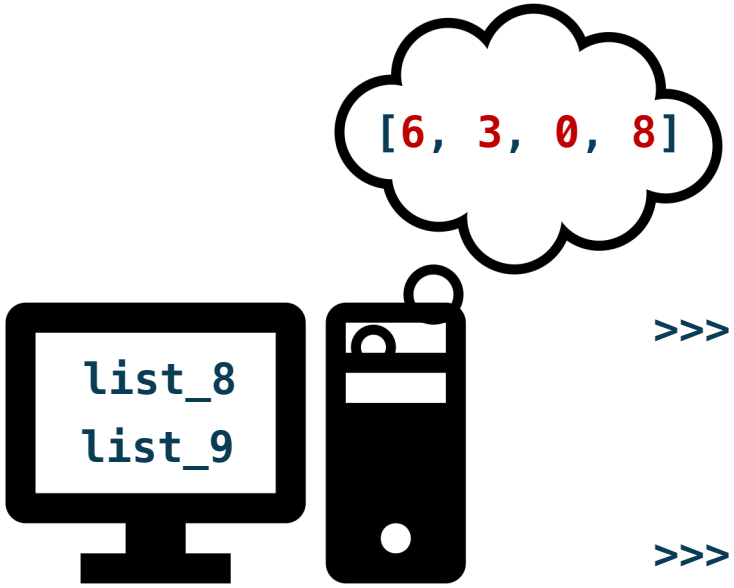
Lists are mutable

Sources:
Tuples, Lists, Aiiasing and Mutability By Ana Bell: https://youtu.be/RvRKT-jXvko
Guttag, John, Introduction to Computation and Programming Using Python: With Application to Understanding Data (The MIT Press) second edition

Sources:
Tuples, Lists, Aiiasing and Mutability By Ana Bell: https://youtu.be/RvRKT-jXvko

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]
```

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

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

`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**

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 be 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:
Conditionals: https://www2.seas.gwu.edu/~cs4all/1012/unit1/module1.1.html

Discuss the following codes

```
x = 5
y = 6
z = 7

if (x != y) and (x != z) and (y != z):
    print('x, y, z are all different')
```

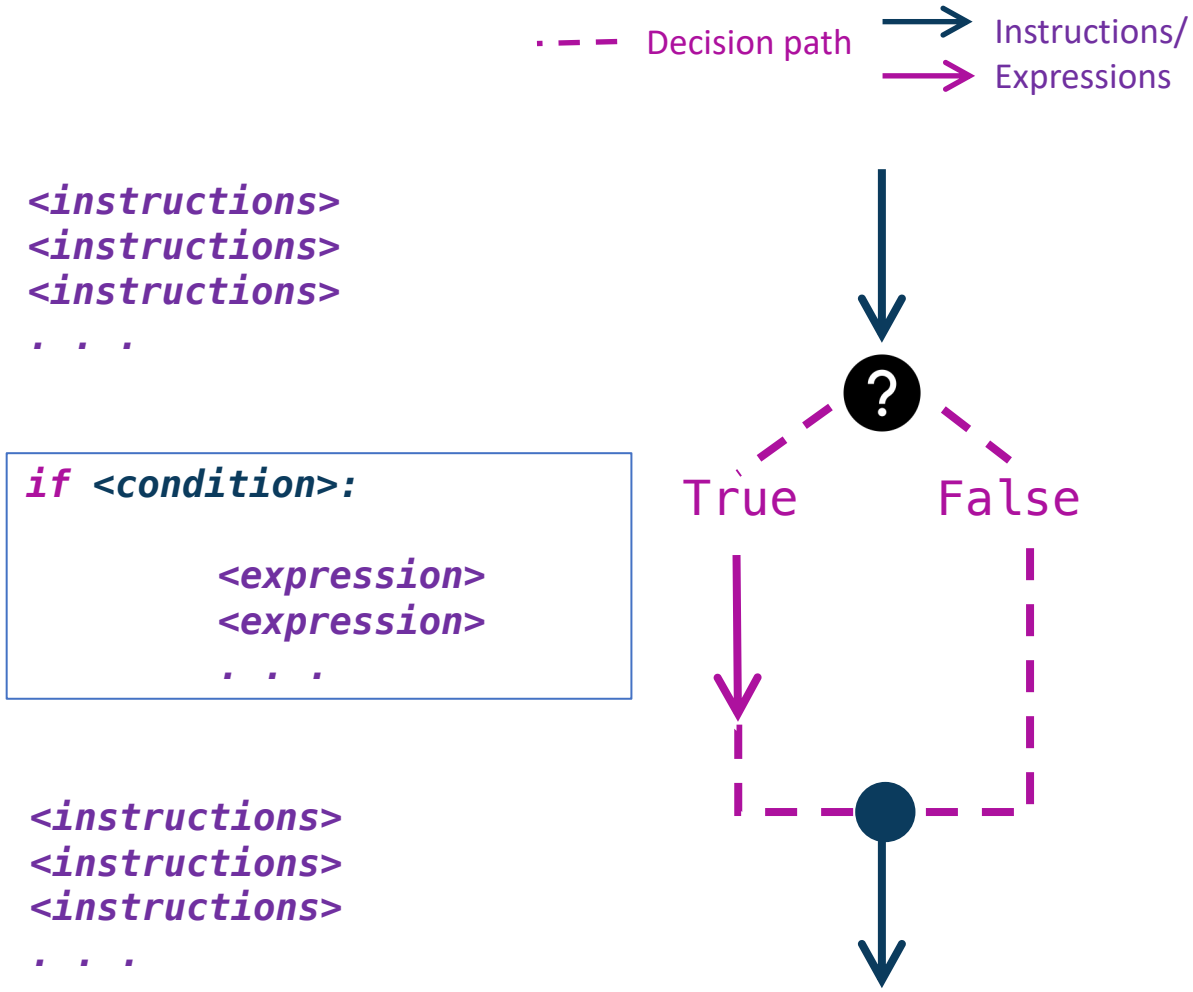
```
x = 8

if not ( (x == 5) or (x == 6) ):
    print('x is neither 5 nor 6')
```

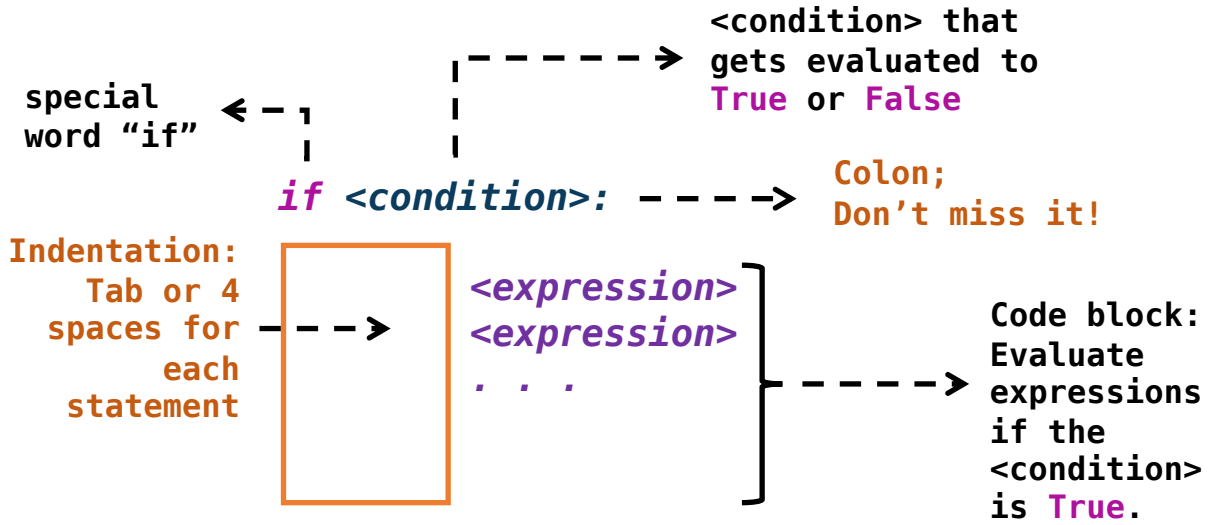
`a=1, b=1, c=3, d=4, e=5`

```
( (a <= b) and (c+d > e) and (d > 1) )

( (a > c) or ( (c+1 < e) and (c-b > a) ) )

not ( (b == d-c) and (a > b) or (c < d) )
```

Demo

Skeleton of the control flow - branching using `if-construct`





Sources:
Conditionals: https://www2.seas.gwu.edu/~cs4all/1012/unit1/module1.1.html

Discuss the following codes

```
s = 0
for i in range(6):
    s = s + i

if s < 15:
    print('Less than 15')

print('Done')
```

```
earnings = [-5, 2, 3, -9, 12, 4, -30]
total = 0
for k in earnings:
    if k >= 0:
        total += k

print('Total profit =', total)
```

```
n = 10
for i in range(1, n+1):
    if i % 2 == 0:
        print(i, 'is even')
```

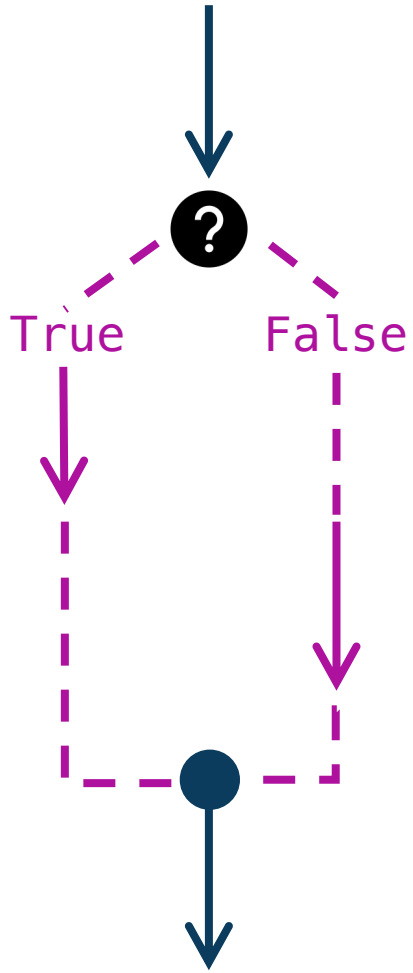
Demo

Skeleton of the control flow - branching using `if-else-construct`

Decision path

Instructions/ Expressions

```
<instructions>
<instructions>
. . .

if <condition>:
        <expression>
        <expression>
        . . .
else:
        <expression>
        <expression>
        . . .

<instructions>
<instructions>
. . .
```



special word "if"

<condition> that gets evaluated to True or False

`if <condition>:` → Colon; Don't miss it!

Indentation: Tab or 4 spaces for each statement

`<expression>`
`<expression>`
`. . .`

Code block: Evaluate expressions if the <condition> is True.

`else:` → Colon; Don't miss it!

Indentation: Tab or 4 spaces for each statement

`<expression>`
`<expression>`
`. . .`

Code block: Evaluate expressions if the preceding <condition> is False.

School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

GW

```
A = [-5, 2, 4, -9, 12, 13, -30, -21, -20]
B = [2, -9, 11, 16, 13]
```

```
2 in A also found in B
-9 in A also found in B
13 in A also found in B
```

Sources:
Conditionals: https://www2.seas.gwu.edu/~cs4all/1012/unit1/module1.1.html

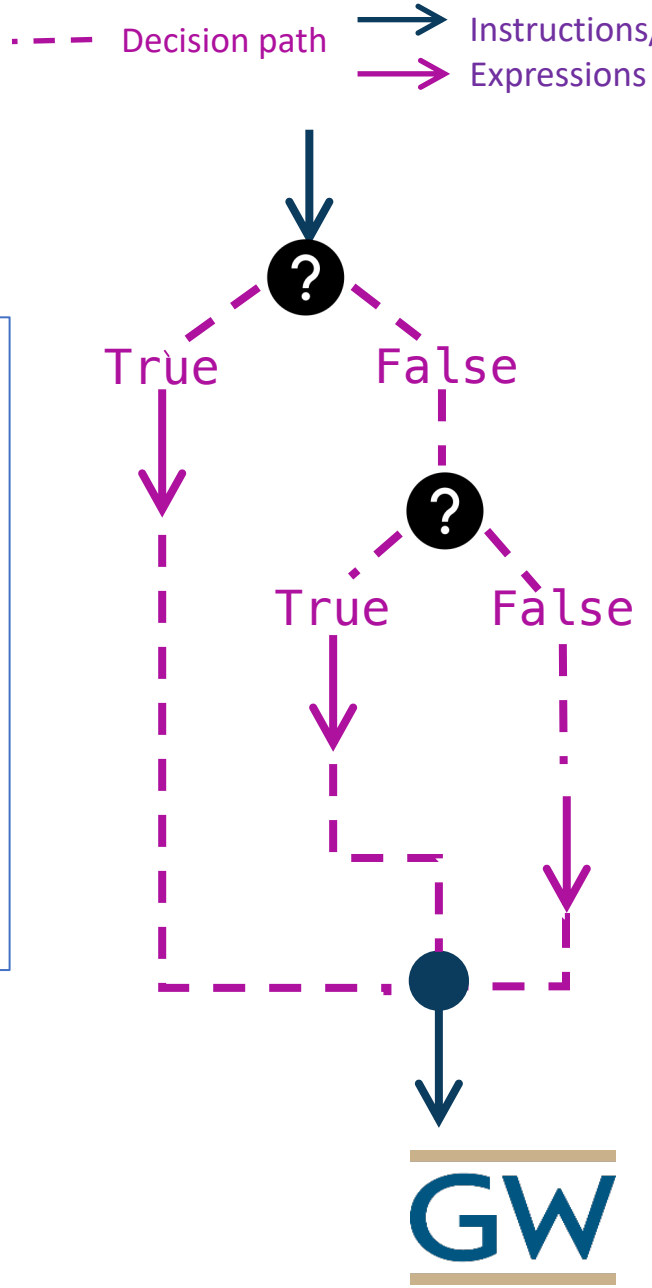
Discuss the following codes

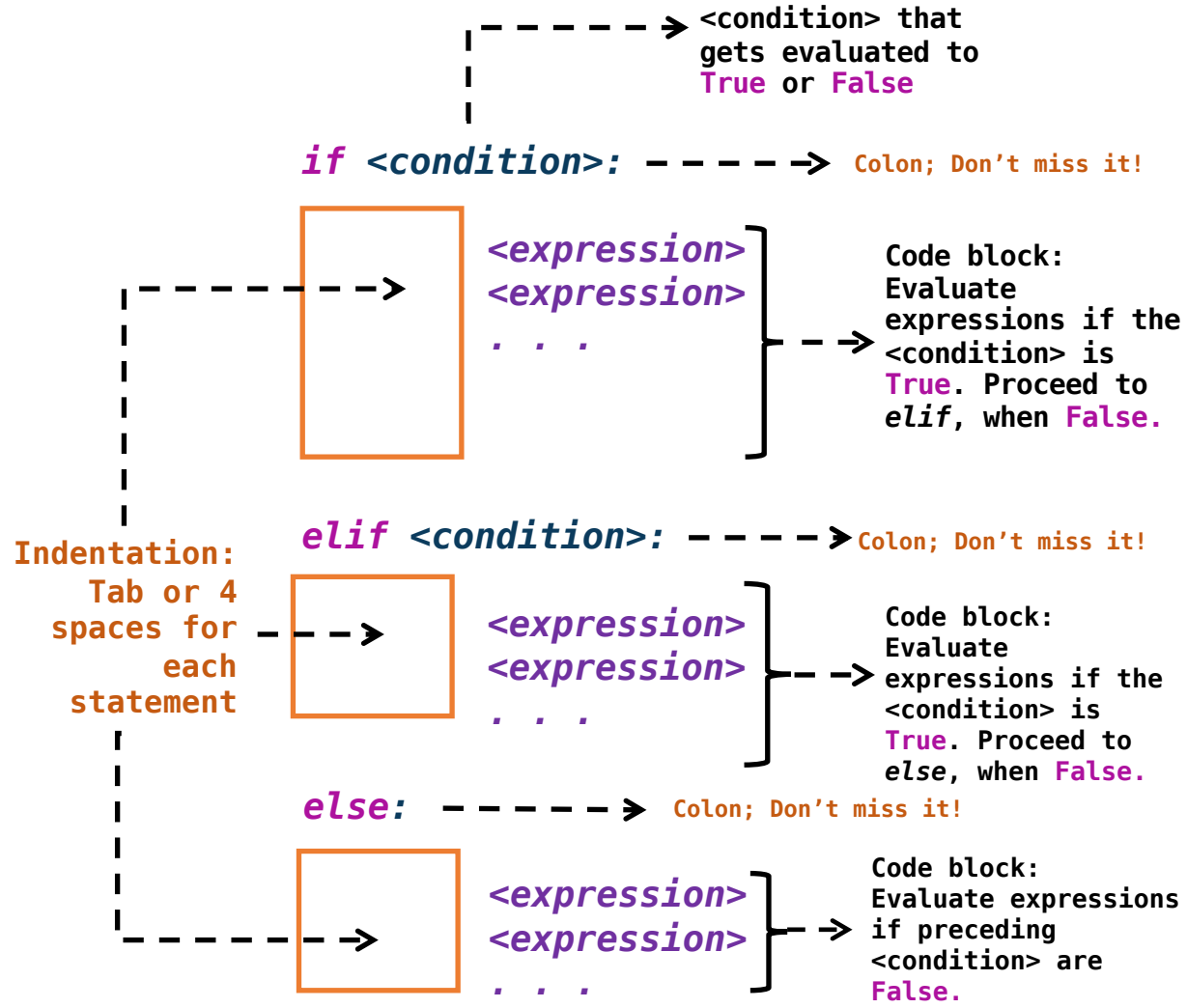
Demo

```
# The list of numbers:
A = [-5, 2, 4, -9, 12, 13, -30]

# Receive what the user types in (as a string):
user_str = input('Enter an integer: ')

# Convert string to integer:
k = int(user_str)

# Check whether in the list:
if k in A:
    print(k,'is in the list')
else:
    print(k,'is not in the list')
```

Skeleton of the control flow - branching using `if-elif-else-construct`

```
<instructions>
. . .

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

elif <condition>:
        <expression>
        <expression>
        . . .

else:
        <expression>
        <expression>
        . . .

<instructions>
. . .
```





Sources:
Conditionals: https://www2.seas.gwu.edu/~cs4all/1012/unit1/module1.1.html

Discuss the following codes

```
import random

trials = 100
total = 0

for i in range(trials):
    x = random.uniform(5, 10)
    print(x)
    total += x

print('mean =', total/trials)
```

Instead try writing a Python program using `if-elif-else`-construct to

1. Input an integer between 5 and 20
2. Generate random numbers between 5 and 10 if the integer is within that range
3. Generate random numbers between 11 and 20 if the integer is within that range
4. Calculate the mean of the random numbers
5. Return a message if the value is outside of the range between 5 and 20

Demo

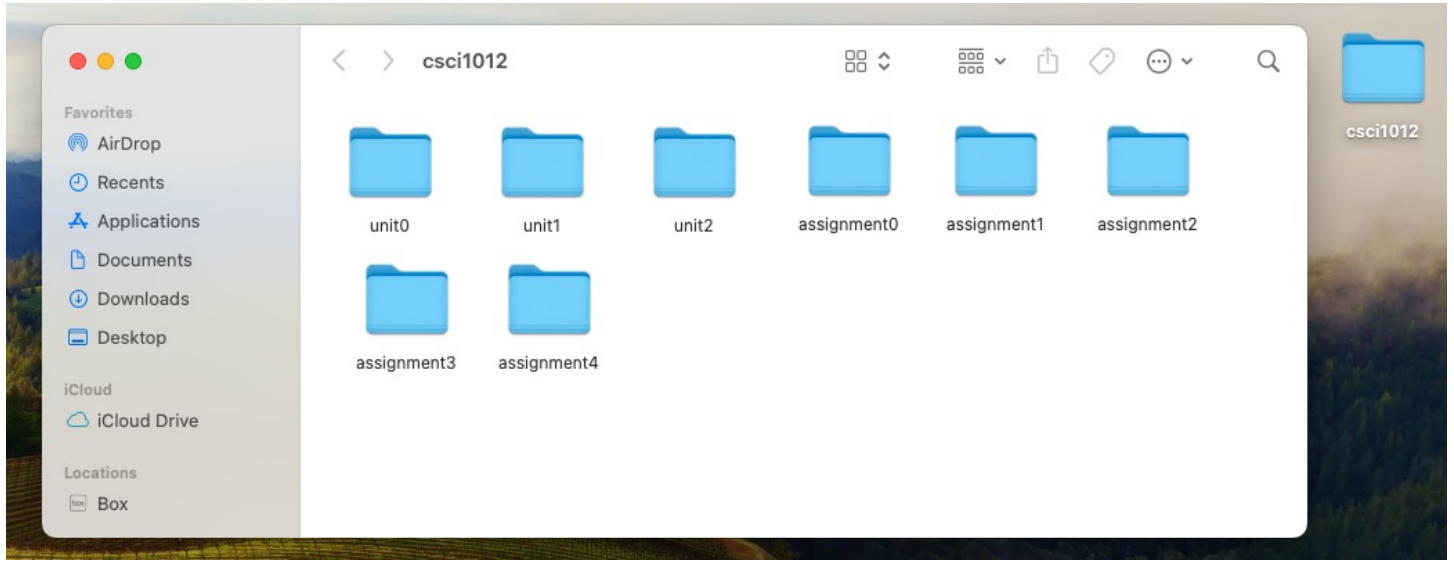
File-folder-structure

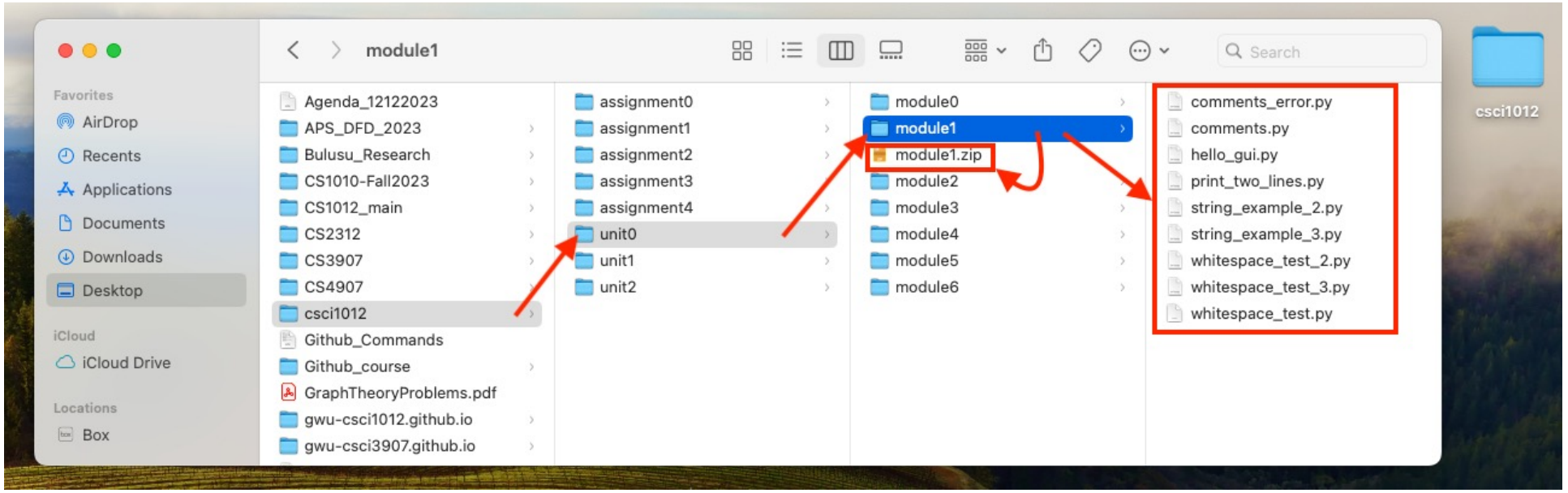
`module0.zip` (correct)

`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 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**)
Week 7 [03/04/2024]	Conditionals, **Make up Quiz**	03/06/2024	03/08/2024	Unit 1 » Module 1 (Due **March 11, 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.

See you all in the Wednesday and Friday Labs!