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: 1957 E 310
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: ~~1957 E 315~~ TOMP 309
Section-36: ~~PHIL 348~~ TOMP 306
Section-37: TOMP 107

GW School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Spring 2024

Photo: Kartik Bulusu

Ungraded In-class Concept Check #1

Some rules to observe:

- 5 minutes restriction
- You are limited to ONE response
- You can discuss with your colleagues in-class
- Watch real-time results

https://forms.gle/qYVdw3xDTbLPWsgQA

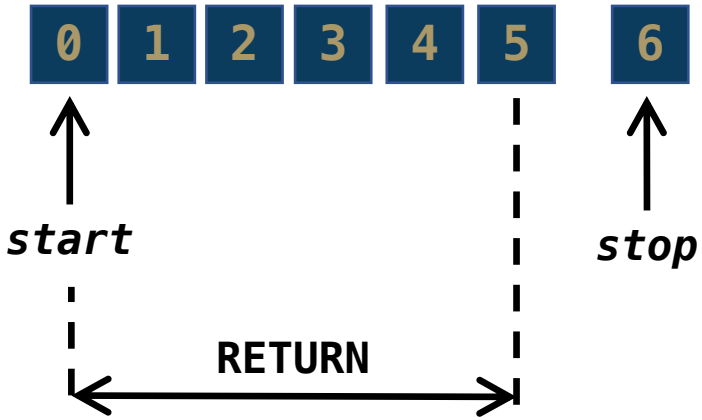
Recap: Built-in function range()

Python's range() function
returns a sequence of numbers
works only with integers

start:	at the value (default = 0)
step:	up or down at the increment value (default = 1)
stop:	at the value but not including it

range(stop)

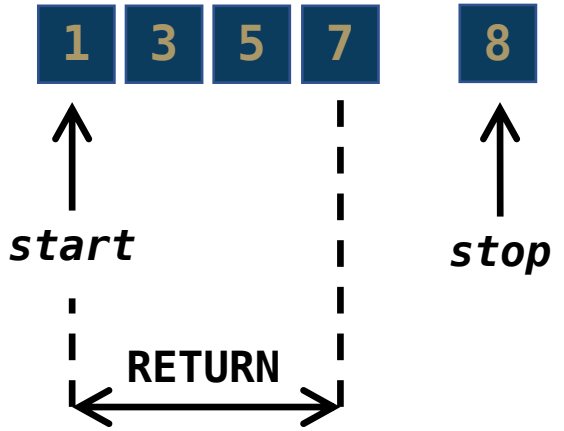
```
>>> range(6)
```



range(start, stop)

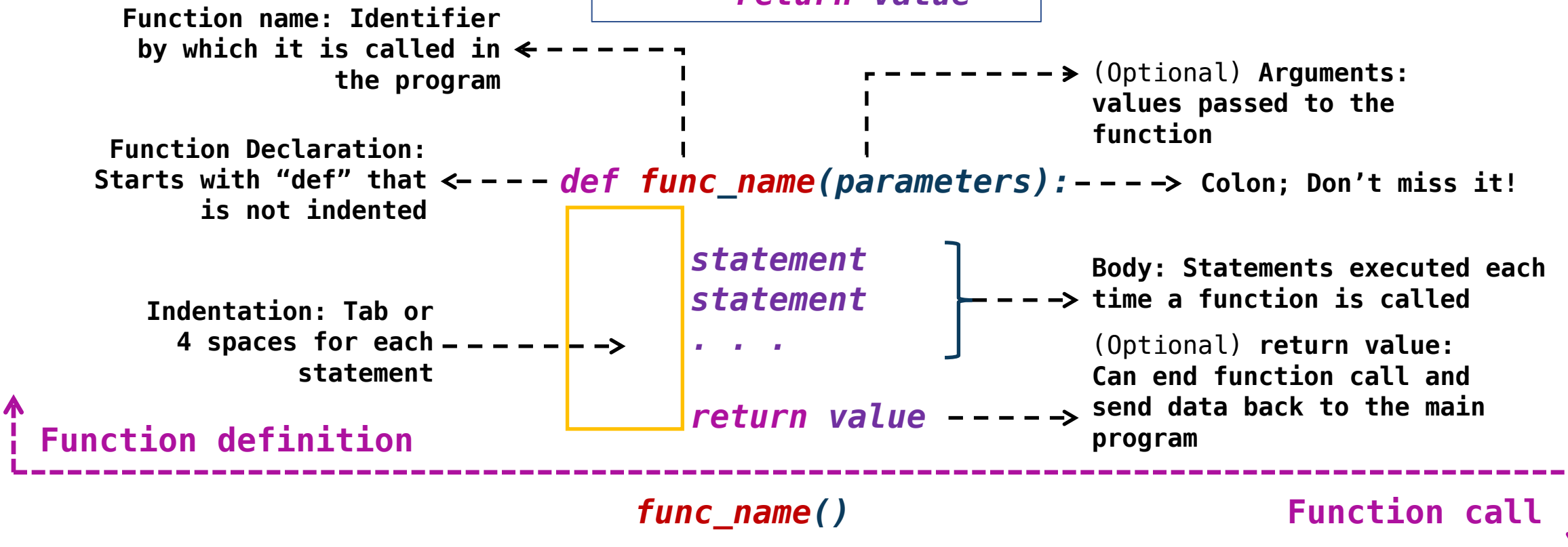
range(*start*, *stop*, *step*)

```
>>> range(1,8,2)
```



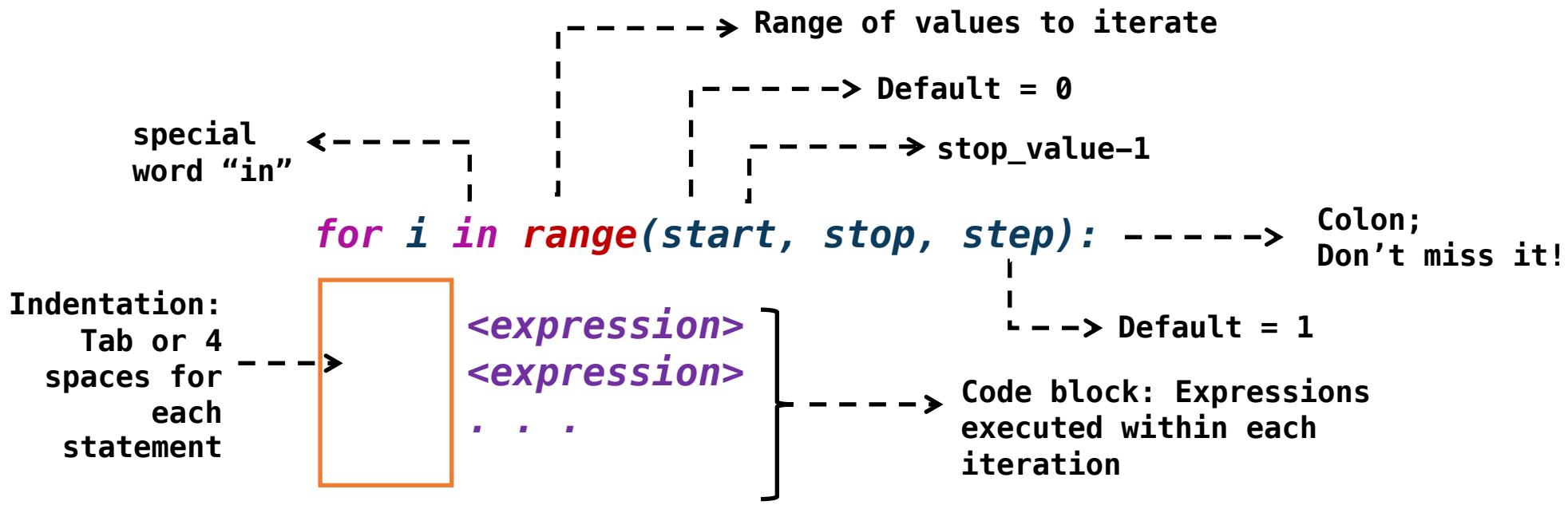
Recap: Syntax and Skeleton of a user-defined `function`

```
def name(arguments):

        statement
        statement
        . . .

        return value
```



Recap: Skeleton of the for-loop

```
for i in range(start, stop, step):

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



Recap: Skeleton of the `python program` (so far)

```
def name(arguments):

        statement
        statement
        . . .

        return value
```

```
for i in range(start, stop, step):

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

Source: https://www2.seas.gwu.edu/~cs4all/1012/unit0/module0.3.html

```
def print_big_N():
    print('*   *')
    print('**  *')
    print('* * *')
    print('*  **')
    print('*   *\n')

def print_big_O():
    print('*****')
    print('*   *')
    print('*   *')
    print('*   *')
    print('*****\n')


print_big_N()
for i in range(6):
    print_big_O()
```

Computers calculate stuff

Icon Sources:

Floppy Disk by Diardha Gumi: https://thenounproject.com/browse/icons/term/floppy-disk/
harddisk by Start Up Graphic Design: https://thenounproject.com/browse/icons/term/harddisk/
calculations by Xinh Studio: https://thenounproject.com/browse/icons/term/calculations/
pi by Tom Fricker: https://thenounproject.com/browse/icons/term/pi/
calculations in physics by Olena Panasovska: https://thenounproject.com/browse/icons/term/calculations-in-physics/
waves by Joey Hiller: https://thenounproject.com/browse/icons/term/waves/
mockup chart by Dawid Sobolewski: https://thenounproject.com/browse/icons/term/mockup-chart/
mockup chart by Dawid Sobolewski: https://thenounproject.com/browse/icons/term/mockup-chart/

Computers need some language to communicate

English ?

Primitive constructs

- Words
  - Alphabets
- Sentences with words

Python ?

Primitive constructs

Numbers, strings

Basic math operators

+

-

*

/

> = <

Python programs

Sources:
What is computation? By Ana Bell: https://youtu.be/nykOeWgQcHM
Bilbo Baggins: https://en.wikipedia.org/wiki/Bilbo_Baggins
Bilbo Baggins: https://heroes-and-villain.fandom.com/wiki/Bilbo_Baggins

- Sequences of definitions and commands
- Everything in `Python` is an object
  - number 9, is an object
  - range() is an object
- Manipulate objects
- Objects:
  - have a `type()`, that determine what `python` program can do with them

Analogy

`type`

`human`

- `Python` has two kinds of objects
  - scalar: very basic and other objects can be made with them
  - non-scalar: have an internal structure

Source:
What is computation? By Ana Bell: https://youtu.be/nykOeWgQcHM

Our focus is on scalar objects

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(9)
<class 'int'>

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

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

Integer operators

i+j	Sum of integers
i-j	Difference
i*j	Product
i/j	Division
i%j	Remainder of i/j
ij**	i raised to the power j
i//j	Integer division

→ `i and j are integers.` **Result is an integer.**

→ `i and j are integers.` **Results in decimal. (float)**

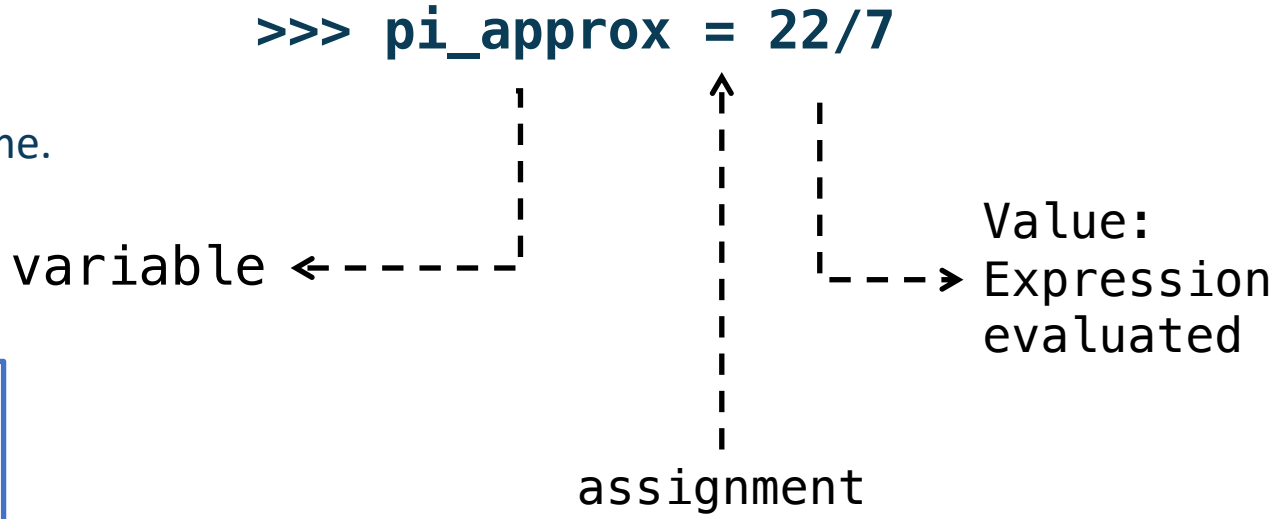
→ `i and j are integers.` **Result is an integer.**

Source:
What is computation? By Ana Bell: https://youtu.be/nykOeWgQcHM

Integer variables

Give a name to the values of the expressions

Equal sign is an assignment of a value to a variable name.



- **Assignment binds the variable name to the value**
- Value is stored in the computer memory
- Invoke the variable name to retrieve the value
- **Variable refers to the value in the memory**

Ungraded In-class Concept Check #2

Some rules to observe:

- 5 minutes restriction
- Unlimited responses
- You can discuss with your colleagues in-class
- Watch real-time results

https://forms.gle/WU9pMeV682T9nZo56

Source:
https://www2.seas.gwu.edu/~cs4all/1012/unit0/module0.4.html

Changing bindings

Variable names using new assignment statements

```
i = 5
j = 6
k = i * j
print(k)
m = i / j
print(m)
n = i // j
print(n)
```

Binding does not change until you tell the computer to do it

```
i = 21
      j = 6
      m = i / j
```

```
i = 21
      j = 6
      m = i // j
```

Value may still be stored in memory but lost the handle (variable) for it

PEMDAS

Python follows the PEMDAS order of mathematical expressions involving more than one operation.

PEMDAS:
P -- Parentheses
E -- Exponents
M-- Multiplication
D -- Division
A -- Addition
S -- Subtraction

```
i = 5
j = 6

k1 = (i*j) - ((i+1) * (j-1))
print(k1)

k2 = i*j - (i+1)*(j-1)
print(k2)

k3 = i*j - i+1*j-1

print(k3)
```

Demo

Source:
https://www2.seas.gwu.edu/~cs4all/1012/unit0/module0.4.html

Example: Sum of integers with tracing

```
s = 0

for i in range(0, 5):
        s = s + i

print(s)

# Short cut: s += i
```

iteration #	*value of i*	*value of s*
1	0	0
2	1	1
3	2	3
4	3	6
5	4	10

Other short cuts:

```
s -= i          # Same as s = s - i
p *= 2          # Same as p = p * 2
d /= 2          # Same as d = d / 2
```

Source:
https://www2.seas.gwu.edu/~cs4all/1012/unit0/module0.4.html

Example: Odd numbers with pseudocode and tracing

Pseudocode

- Program–like outline (not the real program)
- Meant to put the workflow on paper prior to programming
- Expresses ideas in plain English

```
N = 5
for i ranging from 1 to N:
        Calculate the i–th odd number
        Print it
```

```
N = 5
for i in range(1, N+1):
        k = (2*i – 1)
        print(k)
```

iteration #	*value of i*	*value of s*
1	1	1
2	2	3
3	3	5
4	4	7
5	5	9

Source:
https://www2.seas.gwu.edu/~cs4all/1012/unit0/module0.4.html

Example: For any given n, compute $1+2+2^2+2^3+...+2^n$ - Nested `for` loops

```
k = 4
p = 1

for i in range(1, k+1):
        p = p * 2
print(p)
```

iteration #	*value of i*	*value of p*
		1
1	1	2
2	2	4
3	3	8
4	4	16

```
n = 3
s = 1
for i in range(1, n+1):
        p = 1
        for i in range(1, k+1):
                p = p * 2
        s = s + p
print(s)
```

+

=

Demo

Ungraded Concept Check

https://forms.gle/rPPGwpTd1tKTbw2x6

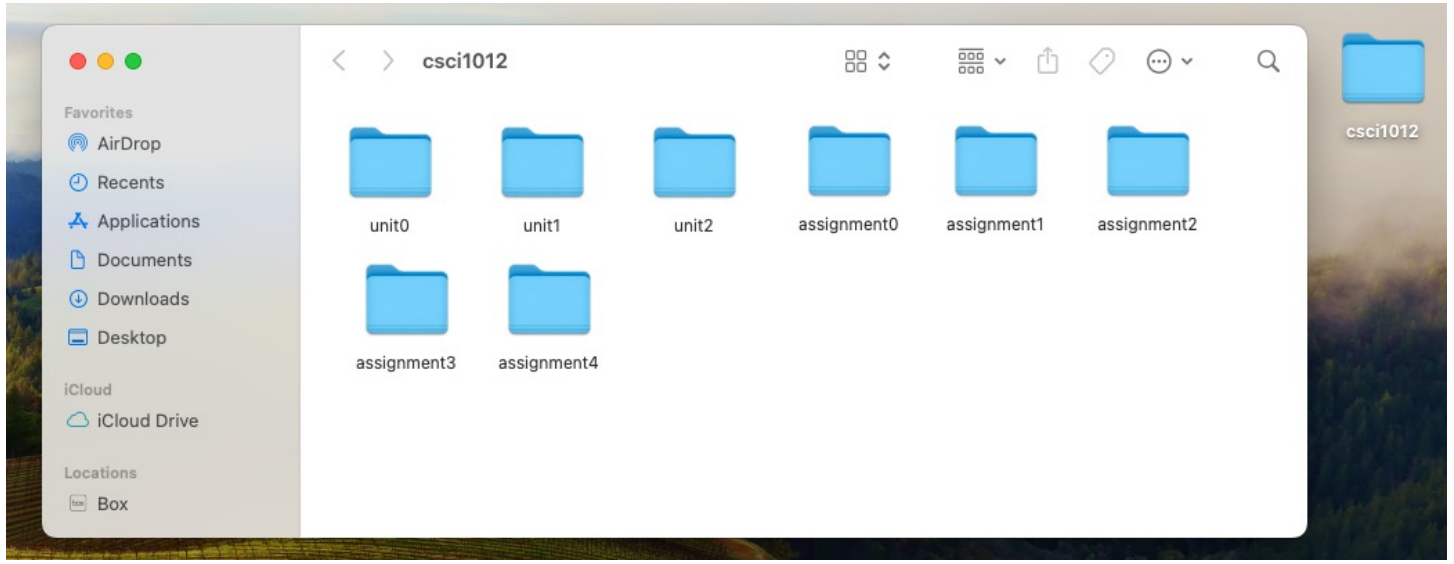
File-folder-structure

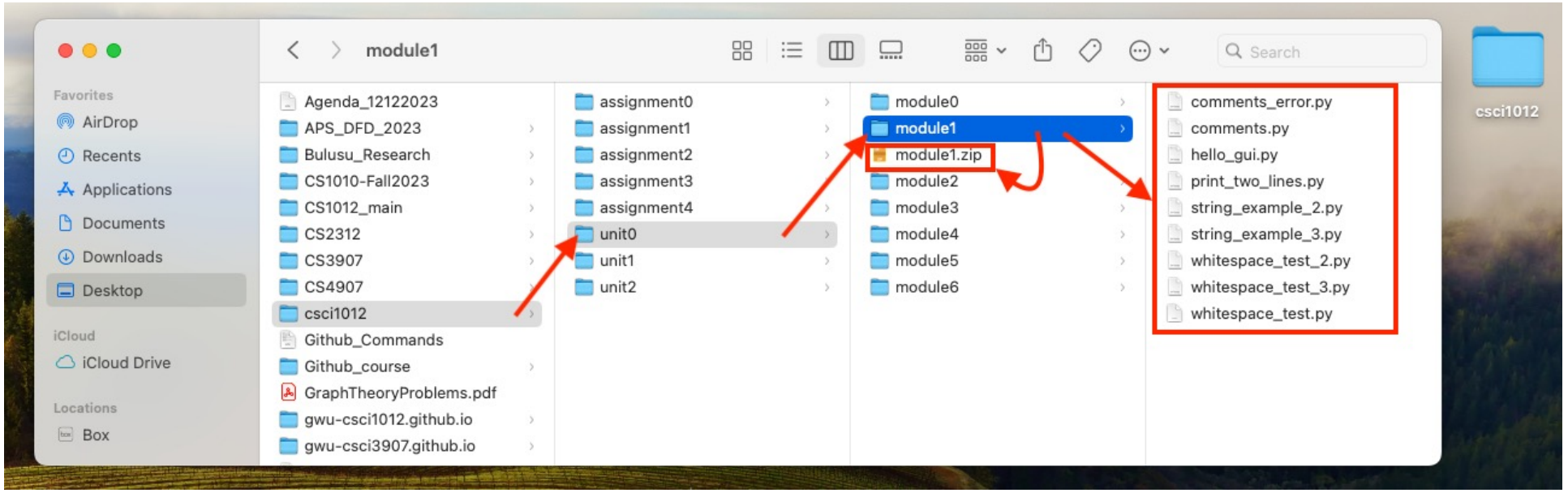
`module0.zip` (correct)

`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 2 [01/29/2024]	Looping: `for` Loops	01/31/2024	02/02/2024	Unit 0 » Module 3 (Due **February 05, 2024** by **11:59 PM**)
Week 3 [02/05/2024]	Integers	02/07/2024	02/09/2024	Unit 0 » Module 4 (Due **February 12, 2024** by **11:59 PM**) & Assignment 0 (Due **February 16, 2024** by **11:59 PM**)

- **Office hours location change:** Friday 10:00 AM to 2:00 PM is SEH B1280
- **CSCI 1012.36 (CRN: 94171)** - Moved to TOMP 306
- **CSCI 1012.33 (CRN: 94168)** - Moved to TOMP 309
- **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!